Hillforts Primer

An Analysis of the Atlas of Hillforts of Britain and Ireland

Part 1

Mike Middleton, March 2022
https://orcid.org/0000-0001-5813-6347

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
HTML: Read only topographic

- Introduction
- Mountains & Hills
- Admin Data
- Location Data

Part 2: Management & Landscape

Colab Notebook: Live code
HTML: Read only
HTML: Read only topographic

Part 3: Boundary & Dating

Colab Notebook: Live code
HTML: Read only
HTML: Read only topographic

Part 4: Investigations & Interior

Colab Notebook: Live code
HTML: Read only
HTML: Read only topographic

Part 5: Entrance, Enclosing & Annex

Colab Notebook: Live code
HTML: Read only
HTML: Read only topographic

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

Colab Notebook: Live code
HTML: Read only
HTML: Read only topographic

Introduction

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

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

Document Structure

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

Run All.

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

Source Data

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

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

Lock, G. and Ralston, I. 2017. Atlas of Hillforts of Britain and Ireland. [ONLINE] Available at: https://hillforts.arch.ox.ac.uk
Rest services:
https://maps.arch.ox.ac.uk/server/rest/services/hillforts/Atlas_of_Hillforts/MapServer
Licence: https://creativecommons.org/licenses/by-sa/4.0/
Help: https://hillforts.arch.ox.ac.uk/assets/help.pdf
Data Structure: https://maps.arch.ox.ac.uk/assets/data.html
Hillforts: Britain, Ireland and the Nearer Continent (Sample):
https://www.archaeopress.com/ArchaeopressShop/DMS/A72C523E8B6742ED97BA86470E747C6sample.pdf

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

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

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

Acknowledgments

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

Bypass Code Setup

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

Go to Mountains & Hills.

Python Modules and Code Setup

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

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

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

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

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

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

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

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

import os
import collections
import math
import random
import PIL
import urllib
random.seed(42) # A random seed is used to ensure that the random numbers created c

from slugify import slugify

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

```
Python: 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]
Scikit-Learn: 1.2.2
pandas: 1.5.3
numpy: 1.23.5
matplotlib: 3.7.1
seaborn: 0.12.2
scipy: 1.10.1
```

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

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

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

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

Ref: https://matplotlib.org/

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

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

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

Plot Figures and Maps functions

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

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

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/h

    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. 2022, Hillforts Primer", size='small', color='grey'
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", size
```

In [ ]:
```
def add_annotation_l_xy(ax):
    ax.annotate("Middleton, M. 2022, Hillforts Primer", size='small', color='grey'
    ax.annotate("Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk", size
```

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

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

```
    add_annotation_plot(ax)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

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

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

    return n_bins
```

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

In [ ]:
```
def plot_continuous(data, x_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
def plot_data_range(data, feature, o="v"):
    fig = plt.figure(figsize=(12,8))
```

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

    bp = boxplot_stats(data, whis=[2.2, 97.8])

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

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

In [ ]:
```
def location_XY_plot():
    plt.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, frin
    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='Silve
    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['Densit
    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, -26
    y_values = [7019842, 6944572, 6850593, 6779602, 6735058, 6710127, 6684152, 666
    plt.plot(x_values, y_values, 'k', ls='-', lw=15, alpha=0.25, label = '≥ 21 Ha
    add_to_legend = Line2D([0], [0], color='k', lw=15, alpha=0.25, label = '≥ 21 H
    return add_to_legend
```

In [ ]:
```
def add_21Ha_fringe():
    x_values = [-367969,-126771,29679,-42657,-248650,-304545,-423647,-584307,-3679
    y_values = [7019842,6847138,6671658,6596650,6554366,6611780,6662041,6752378,70
    plt.plot(x_values, y_values, 'k', ls=':', lw=5, alpha=0.45, label = '≥ 21 Ha F
    add_to_legend = Line2D([0], [0], color='k', ls=':', lw=5, alpha=0.45, label =
    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=Fals
    # plots selected data over the grey dots. yes_no controlls filtering the data
    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_ty
    plt.colorbar(label=data_type)
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

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

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

In [ ]:
```
def add_cluster_split_lines(plt, ax, extra=None):
    x_min = -550000
    if extra == 'ireland':
        x_min = -1200000
    plt.vlines(x=[-500000], ymin=7070000, ymax=9000000, colors='r', ls='-', lw=3)
    plt.hlines(y=[7070000], xmin=x_min, xmax=200000, colors='r', ls='-', lw=3)
    ax.annotate("N/S split", color='k', xy=(50000, 7090000), xycoords='data')
    ax.annotate("E/W split", color='k', xy=(-480000, 8660000), xycoords='data')
    ax.annotate("Irish Sea split", color='k', xy=(-1150000, 7510000), xycoords='da
    plot_line((-800000,6400000), (-550000,7070000))
    plot_line((-550000,7070000), (-666000,7440000))
    plot_line((-666000,7440000),(-900000,7500000))
```

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

In [ ]:
```
def plot_wales(wales_data):
    plt.subplots(figsize=(7.5, 7.5))
```

```
    plt.scatter(wales_data['Main_X'], wales_data['Main_Y'])
    plt.xlim(150000,400000)
    plt.ylim(150000,400000)
    bespoke_plot(plt, 'Wales - Main_X/Main_Y')
```

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

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

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

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

In [ ]:
```
def plot_ireland(roi_and_ni_data):
    plt.subplots(figsize=(8.0, 10.0))
    plt.scatter(roi_and_ni_data['Main_X'], roi_and_ni_data['Main_Y'])
    plt.xlim(400000,800000)
    plt.ylim(500000,1000000)
    bespoke_plot(plt, 'Republic of Ireland and Northern Ireland Combined - Main_X/M
```

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

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

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

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

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

In [ ]:
```
def density_transformed(transformed_location_numeric_data_short):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], transformed_
    plt.colorbar(label='Density Transformed')
    title = 'Density Transformed'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

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

    rect = patches.Rectangle((x_lo, y_lo), x_hi-x_lo, y_hi-y_lo, linewidth=w2, edg
    rect2 = patches.Rectangle((x_q1, y_q1), x_q3-x_q1, y_q3-y_q1, linewidth=w1, ed

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

In [ ]:
```
def plot_density_boxplot(transformed_location_numeric_data_short):
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], transformed_
    rect, rect2, vline, xline = get_rectangles(location_X_data, location_Y_data, 3
    ax.add_patch(rect)
    ax.add_patch(rect2)
    plt.vlines(x=[vline[1]], ymin=vline[0], ymax=vline[2], colors='purple', ls='--
    plt.hlines(y=[xline[1]], xmin=xline[0], xmax=xline[2], colors='purple', ls='--
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of Boxplot (All Data)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_northern_density(show_eildon, north):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(7000000,8700000)
    plt.scatter(north['Location_X'], north['Location_Y'], c=north['Density_trans']
    if show_eildon:
        plt.plot(-301491, 7476601,'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), xycoords='
```

```
        plt.plot(-366753,7357536,'ko')
        ax.annotate('SC0889: Hayknowes', xy=(-366753,7357536), xycoords='data', ha
        plt.plot(-177825,7447736,'ko')
        ax.annotate('EN0580: Howick Hill Camp', xy=(-177825,7447736), xycoords='da
        plt.plot(-487909,7558812,'ko')
        ax.annotate('SC1458: Quinloch Muir', xy=(-487909,7558812), xycoords='data'
        plt.plot(-276998,7682350,'ko')
        ax.annotate('SC3102: Red Head', xy=(-276998,7682350), xycoords='data', ha=
    add_annotation_plot(plt)
    plt.colorbar(label='Density Transformed')
    title = 'Density Transformed (North)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_northern_density_transformed(show_eildon, north_top5):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(7000000,8700000)
    plt.scatter(north_top5['Location_X'], north_top5['Location_Y'], c=north_top5['
    if show_eildon:
        plt.plot(-301491, 7476601,'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), xycoords='
    add_annotation_plot(plt)
    plt.colorbar(label='Density Transformed - top 5%')
    title = 'Density Transformed - top 5% (North)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_density_transformed_north_capped(show_eildon, north_clip):
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(7000000,8700000)
    plt.scatter(north_clip['Location_X'], north_clip['Location_Y'], c=north_clip['
    if show_eildon:
        plt.plot(-301491, 7476601,'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), xycoords='
        plt.plot(-609918,7575512 ,'ko')
        ax.annotate('SC2466: Dunadd', xy=(-609918,7575512), xycoords='data', ha='l
        plt.plot(-487909,7558812,'ko')
        ax.annotate('SC1458: Quinloch Muir', xy=(-487909,7558812), xycoords='data'
        plt.plot(-449967,7321023,'ko')
        ax.annotate('SC0244: Drummore Castle', xy=(-449967,7321023), xycoords='dat
        plt.plot(-276998,7682350,'ko')
        ax.annotate('SC3102: Red Head', xy=(-276998,7682350), xycoords='data', ha=
        plt.plot(-177825,7447736,'ko')
        ax.annotate('EN0580: Howick Hill Camp', xy=(-177825,7447736), xycoords='da
    add_annotation_plot(plt)
    plt.colorbar(label='Density Transformed - Capped')
    title = 'Density Transformed - Capped (North)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_density_transformed_north_boxplot(north, location_X_north_data, location_
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticklabel_format(style='plain')
```

```
    plt.xlim(-1200000,220000)
    plt.ylim(7000000,8700000)
    plt.scatter(north['Location_X'], north['Location_Y'], c=north['Density_trans']
    rect, rect2, vline, xline = get_rectangles(location_X_north_data, location_Y_n
    ax.add_patch(rect)
    ax.add_patch(rect2)
    plt.vlines(x=[vline[1]], ymin=vline[0], ymax=vline[2], colors='purple', ls='--
    plt.hlines(y=[xline[1]], xmin=xline[0], xmax=xline[2], colors='purple', ls='--
    show_eildon = False
    if show_eildon:
        plt.plot(-301491, 7476601,'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), xycoords='
    add_annotation_plot(plt)
    plt.colorbar(label='Density Transformed')
    title = 'Density Transformed overlayed by the extent of Boxplot (N)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:

```
def plot_north_west_density(show_eildon, north_west, location_Y_north_w_data):
    fig, ax = plt.subplots(figsize=((4.75 * 1.5)+2.0, (7.92 * 1.5)))
    show_background(plt, ax, 'nw+')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,-480000)
    plt.ylim(7000000,8200000)
    plt.scatter(north_west['Location_X'], north_west['Location_Y'], c=north_west['
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_north_w_data, locatio
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    if show_eildon:
        plt.plot(-609918,7575512 ,'ko')
        ax.annotate('SC2466: Dunadd', xy=(-609918,7575512), xycoords='data', ha='l
    add_annotation_plot(plt)
    plt.colorbar(label='Density Transformed')
    title = 'Density Transformed (Northwest)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:

```
def plot_north_east_density(show_eildon,north_east, location_Y_north_e_data):
    fig, ax = plt.subplots(figsize=((3.43*1.5)+2.0, (11.22*1.5)))
    show_background(plt, ax, 'ne')
    plt.ticklabel_format(style='plain')
    plt.xlim(-520000, 0)
    plt.ylim(7000000,8700000)
    plt.scatter(north_east['Location_X'], north_east['Location_Y'], c=north_east['
    rect1, rect2, vline1, xline1 = get_rectangles(location_X_north_e_data, locatio
    ax.add_patch(rect1)
    ax.add_patch(rect2)
    plt.vlines(x=[vline1[1]], ymin=vline1[0], ymax=vline1[2], colors='purple', ls=
    plt.hlines(y=[xline1[1]], xmin=xline1[0], xmax=xline1[2], colors='purple', ls=
    if show_eildon:
        plt.plot(-301491, 7476601,'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), xycoords='
        plt.plot(-247628,7490081,'ko')
        ax.annotate('EN4137: Cornhill', xy=(-247628,7490081), xycoords='data', ha=
    add_annotation_plot(plt)
    plt.colorbar(label='Density Transformed')
    title = 'Density Transformed (Northeast)'
    plt.title(get_print_title(title))
```

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

In [ ]:
```
def plot_norhtern_density_boxplots(show_eildon, north, location_X_cluster_north_wes
    fig, ax = plt.subplots(figsize=(9.37+2.0, 11.33))
    show_background(plt, ax, 'n')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(7000000,8700000)
    plt.scatter(north['Location_X'], north['Location_Y'], c=north['Density_trans']
    plt.axvline(x=-500000, color='r', linestyle='-', lw=3)
    ax.annotate("E/W split", color='k', xy=(-480000, 7010000), xycoords='data')
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_north_west, l
    #rect3, rect4, vline2, xline2 = get_rectangles(location_X_north_w_data, locatio
    rect5, rect6, vline3, xline3 = get_rectangles(location_X_north_e_data, locatio
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    ax.add_patch(rect5)
    ax.add_patch(rect6)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', ls=
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', ls=
    show_eildon = False
    if show_eildon:
        plt.plot(-301491, 7476601,'ko') # SC3327: Eildon Hill North
        ax.annotate('SC3327: Eildon Hill North', xy=(-301491, 7476601), xycoords='
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed (Northeast & Norht West)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_clusters(cluster_north_ireland, cluster_south_ireland, cluster_south, clus
    fig, ax = plt.subplots(figsize=(14.2 * 0.66, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(cluster_north_ireland['Location_X'], cluster_north_ireland['Locati
    plt.scatter(cluster_south_ireland['Location_X'], cluster_south_ireland['Locati
    plt.scatter(cluster_south['Location_X'], cluster_south['Location_Y'], c='blue'
    plt.scatter(cluster_north_east['Location_X'], cluster_north_east['Location_Y']
    plt.scatter(cluster_north_west['Location_X'], cluster_north_west['Location_Y']
    add_cluster_split_lines(plt, ax, 'ireland')
    title = 'Location Cluster Data Packages'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_north_west_dnesity_minus(show_eildon, cluster_north_west, location_X_clus
    fig, ax = plt.subplots(figsize=((2.772 * 1.75)+2.0, (7.26 * 1.75)))
    show_background(plt, ax, 'nw-')
    plt.ticklabel_format(style='plain')
    plt.xlim(-900000,-480000)
    plt.ylim(7100000,8200000)
    plt.scatter(cluster_north_west['Location_X'], cluster_north_west['Location_Y']
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_north_west, l
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    plt.plot(-609918,7575512 ,'ko')
    if show_eildon:
```

```
        ax.annotate('SC2466: Dunadd', xy=(-609918,7575512), xycoords='data', ha='l
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Trans & Boxplot (NW minus Ireland)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_southern_density(show_gaer_fach, south):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(6400000,7100000)
    plt.scatter(south['Location_X'], south['Location_Y'], c=south['Density'], cmap
    if show_gaer_fach:
        plt.plot(-383564,6803609,'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564,6803609), xyco
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_south_top_5(show_gaer_fach, south_top5):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(6400000,7100000)
    plt.scatter(south_top5['Location_X'], south_top5['Location_Y'], c=south_top5['
    if show_gaer_fach:
        plt.plot(-383564,6803609,'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564,6803609), xyco
    plt.colorbar(label='Density - top 5%')
    add_annotation_plot(plt)
    title = 'Density - top 5% (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_south_denstiy_transformed(show_gaer_fach, south):
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(6400000,7100000)
    plt.scatter(south['Location_X'], south['Location_Y'], c=south['Density_trans']
    if show_gaer_fach:
        plt.plot(-383564,6803609,'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564,6803609), xyco
        plt.plot(-267643,6699776,'ko')
        ax.annotate('EN4166: Freezing Hill', xy=(-267643,6699776), xycoords='data'
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_southern_density_boxplot(show_gaer_fach, south,location_X_south_data, loca
    fig, ax = plt.subplots(figsize=(9.47+2.0, 5.0))
    show_background(plt, ax, 's+')
    plt.ticklabel_format(style='plain')
    plt.xlim(-1200000,220000)
    plt.ylim(6400000,7100000)
    plt.scatter(south['Location_X'], south['Location_Y'], c=south['Density_trans']
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_south_data, location_Y
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    if show_gaer_fach:
        plt.plot(-383564,6803609,'ko')
        ax.annotate('WA1308: Gaer Fach, Merthyr Cynog', xy=(-383564,6803609), xycoo
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed and Boxplot (South)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_south_density_transformed_boxplot(show_gaer_fach, cluster_south, location_
    fig, ax = plt.subplots(figsize=((6.73*1.5)+2.0, (4.62*1.5)))
    show_background(plt, ax, 's')
    plt.ticklabel_format(style='plain')
    plt.xlim(-800000,220000)
    plt.ylim(6400000,7100000)
    plt.scatter(cluster_south['Location_X'], cluster_south['Location_Y'], c=cluster
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, locatio
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    if show_gaer_fach:
        plt.plot(-370211,6775546,'ko')
        ax.annotate('WA1336: Nant Tarthwynni East', xy=(-370211,6775546), xycoords=
        plt.plot(-510962,6777200,'ko')
        ax.annotate('WA2240: Hafod Camp', xy=(-510962,6777200), xycoords='data', ha
    plt.colorbar(label='Density Transformed')
    add_annotation_plot(plt)
    title = 'Density Transformed and Boxplot (South minus Ireland)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_ns_boxplots(transformed_location_numeric_data_short, location_X_south_data
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], transformed_
    plt.axhline(y=7070000, color='r', linestyle='-', lw=3)
    ax.annotate("N/S split", color='k', xy=(50000, 7090000), xycoords='data')
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_south_data, location_Y
    rect5, rect6, vline3, xline3 = get_rectangles(location_X_north_data, location_Y
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    ax.add_patch(rect5)
    ax.add_patch(rect6)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', ls=
```

```
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', ls=
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of Boxplots: (N & S)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_ne_nw_s_boxplots(transformed_location_numeric_data_short, location_X_clus
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], transformed_
    add_cluster_split_lines(plt, ax)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, locati
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    rect7, rect8, vline2, xline2 = get_rectangles(location_X_cluster_north_west, l
    rect9, rect10, vline3, xline3 = get_rectangles(location_X_north_e_data, locati
    ax.add_patch(rect7)
    ax.add_patch(rect8)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    ax.add_patch(rect9)
    ax.add_patch(rect10)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', ls=
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', ls=
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of Boxplots: NE, NW & South'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_density_ireland(show_dooncarton, ireland_density_data):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'i')
    plt.scatter(ireland_density_data['Location_X'], ireland_density_data['Location_
    plt.xlim(-1200000,-600000)
    plt.ylim(6650000,7450000)
    if show_dooncarton:
        plt.plot(-1095290,7223461,'ko')
        ax.annotate('IR1821 Dooncarton ', xy=(-1095290,7223461), xycoords='data', h
        plt.plot(-1077483,6894257,'ko')
        ax.annotate('IR1279: Doon West', xy=(-1077483,6894257), xycoords='data', ha
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_density_ireland_top_5(show_dooncarton, ireland_density_top5):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'i')
    plt.scatter(ireland_density_top5['Location_X'], ireland_density_top5['Location_
    plt.xlim(-1200000,-600000)
    plt.ylim(6650000,7450000)
    if show_dooncarton:
        plt.plot(-1095290,7223461,'ko')
        ax.annotate('IR1821 Dooncarton ', xy=(-1095290,7223461), xycoords='data', h
        plt.plot(-1077483,6894257,'ko')
```

```
        ax.annotate('IR1279: Doon West', xy=(-1077483,6894257), xycoords='data', ha
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland - top 5%'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_density_north_ireland(show_dooncarton,north_cluster):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 5.0))
    show_background(plt, ax, 'in')
    plt.scatter(north_cluster['Location_X'], north_cluster['Location_Y'], c=north_
    plt.xlim(-1200000,-600000)
    plt.ylim(7050000,7450000)
    if show_dooncarton:
        plt.plot(-1095290,7223461,'ko')
        ax.annotate('IR1821 Dooncarton ', xy=(-1095290,7223461), xycoords='data', h
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland - North'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_density_south_ireland(show_dooncarton,south_cluster):
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 5.0))
    show_background(plt, ax, 'is')
    plt.scatter(south_cluster['Location_X'], south_cluster['Location_Y'], c=south_
    plt.xlim(-1200000,-600000)
    plt.ylim(6650000,7080000)
    if show_dooncarton:
        plt.plot(-1077483,6894257,'ko')
        ax.annotate('IR1279: Doon West', xy=(-1077483,6894257), xycoords='data', ha
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density Ireland - South'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_ireland_one_boxplot(ireland_density_data, location_X_ireland_data, locati
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'i')
    plt.scatter(ireland_density_data['Location_X'], ireland_density_data['Location_
    rect1, rect2, vline, xline = get_rectangles(location_X_ireland_data, location_Y
    ax.add_patch(rect1)
    ax.add_patch(rect2)
    plt.vlines(x=[vline[1]], ymin=vline[0], ymax=vline[2], colors='purple', ls='--
    plt.hlines(y=[xline[1]], xmin=xline[0], xmax=xline[2], colors='purple', ls='--
    plt.xlim(-1200000,-600000)
    plt.ylim(6650000,7450000)
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density overlayed by the extent of Boxplot (Ireland)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_ireland_boxplots(ireland_density_data, location_X_south_data_ireland, loca
    fig, ax = plt.subplots(figsize=(8.0 + 2.0, 10.0))
    show_background(plt, ax, 'i')
    plt.scatter(ireland_density_data['Location_X'], ireland_density_data['Location_
    plt.axhline(y=7060000, color='r', linestyle='-', lw=3)
    ax.annotate("N/S split", color='k', xy=(-680000, 7040000), xycoords='data')
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_south_data_ireland, l
    rect5, rect6, vline3, xline3 = get_rectangles(location_X_north_data_ireland, l
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    ax.add_patch(rect5)
    ax.add_patch(rect6)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', ls=
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', ls=
    plt.xlim(-1200000,-600000)
    plt.ylim(6650000,7450000)
    plt.ticklabel_format(style='plain')
    plt.colorbar(label='Density')
    add_annotation_plot(plt)
    title = 'Density overlayed by the extent of Boxplots - Ireland (N & S)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_five_clusters(transformed_location_numeric_data_short,location_X_cluster_
    fig, ax = plt.subplots(figsize=((14.2 * 0.66)+2.0, 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(transformed_location_numeric_data_short['Location_X'], transformed_
    plt.axhline(y=7070000, color='r', linestyle='-', lw=3)
    rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, locati
    ax.add_patch(rect3)
    ax.add_patch(rect4)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    add_cluster_split_lines(plt, ax, 'ireland')
    rect7, rect8, vline2, xline2 = get_rectangles(location_X_cluster_north_west, l
    rect9, rect10, vline3, xline3 = get_rectangles(location_X_north_e_data, locati
    ax.add_patch(rect7)
    ax.add_patch(rect8)
    plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple', ls=
    plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple', ls=
    ax.add_patch(rect9)
    ax.add_patch(rect10)
    plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple', ls=
    plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple', ls=
    rect11, rect12, vline4, xline4 = get_rectangles(location_X_south_data_ireland,
    rect13, rect14, vline5, xline5 = get_rectangles(location_X_north_data_ireland,
    ax.add_patch(rect11)
    ax.add_patch(rect12)
    plt.vlines(x=[vline4[1]], ymin=vline4[0], ymax=vline4[2], colors='purple', ls=
    plt.hlines(y=[xline4[1]], xmin=xline4[0], xmax=xline4[2], colors='purple', ls=
    ax.add_patch(rect13)
    ax.add_patch(rect14)
    plt.vlines(x=[vline5[1]], ymin=vline5[0], ymax=vline5[2], colors='purple', ls=
    plt.hlines(y=[xline5[1]], xmin=xline5[0], xmax=xline5[2], colors='purple', ls=
    plt.colorbar(label='Density Transformed')
    title = 'Density overlayed by the extent of the five main cluster Boxplots'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

In [ ]:
```
def plot_adjusted_density(ireland_density_data,cluster_south,cluster_north_west,no
                          location_X_cluster_south, location_Y_cluster_south,
                          location_X_cluster_north_west, location_Y_cluster_north_west,
                          location_X_north_e_data, location_Y_north_e_data,
                          location_X_south_data_ireland, location_Y_south_data_ireland,
                          location_X_north_data_ireland, location_Y_north_data_ireland)
    fig, ax = plt.subplots(figsize=((14.2 * 0.66), 23.0 * 0.66))
    show_background(plt, ax)
    location_XY_plot()
    plt.scatter(ireland_density_data['Location_X'], ireland_density_data['Location_
    plt.scatter(cluster_south['Location_X'], cluster_south['Location_Y'], c=cluste
    plt.scatter(cluster_north_west['Location_X'], cluster_north_west['Location_Y']
    plt.scatter(north_east['Location_X'], north_east['Location_Y'], c=north_east['
    plt.axhline(y=7070000, color='r', linestyle='-', lw=3)
    if show_boxplots:
        rect3, rect4, vline2, xline2 = get_rectangles(location_X_cluster_south, lo
        ax.add_patch(rect3)
        ax.add_patch(rect4)
        plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple',
        plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple',
    add_cluster_split_lines(plt, ax, 'ireland')
    if show_boxplots:
        rect7, rect8, vline2, xline2 = get_rectangles(location_X_cluster_north_west
        rect9, rect10, vline3, xline3 = get_rectangles(location_X_north_e_data, lo
        ax.add_patch(rect7)
        ax.add_patch(rect8)
        plt.vlines(x=[vline2[1]], ymin=vline2[0], ymax=vline2[2], colors='purple',
        plt.hlines(y=[xline2[1]], xmin=xline2[0], xmax=xline2[2], colors='purple',
        ax.add_patch(rect9)
        ax.add_patch(rect10)
        plt.vlines(x=[vline3[1]], ymin=vline3[0], ymax=vline3[2], colors='purple',
        plt.hlines(y=[xline3[1]], xmin=xline3[0], xmax=xline3[2], colors='purple',
        rect11, rect12, vline4, xline4 = get_rectangles(location_X_south_data_irela
        rect13, rect14, vline5, xline5 = get_rectangles(location_X_north_data_irela
        ax.add_patch(rect11)
        ax.add_patch(rect12)
        plt.vlines(x=[vline4[1]], ymin=vline4[0], ymax=vline4[2], colors='purple',
        plt.hlines(y=[xline4[1]], xmin=xline4[0], xmax=xline4[2], colors='purple',
        ax.add_patch(rect13)
        ax.add_patch(rect14)
        plt.vlines(x=[vline5[1]], ymin=vline5[0], ymax=vline5[2], colors='purple',
        plt.hlines(y=[xline5[1]], xmin=xline5[0], xmax=xline5[2], colors='purple',
    title = 'The five main density clusters (Density adjusted by region)'
    plt.title(get_print_title(title))
    save_fig(title)
    plt.show()
```

Review Data Functions

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

In [ ]:
```
def test_numeric(data):
    temp_data = data.copy()
    columns = data.columns
    out_cols = ['Feature','Entries', 'Numeric', 'Non-Numeric', 'Null']
    feat, ent, num, non, nul = [],[],[],[],[]
    for col in columns:
```

```
        if temp_data[col].dtype == 'object':
            feat.append(col)
            temp_data[col+'_num'] = temp_data[col].str.isnumeric()
            entries = temp_data[col].notnull().sum()
            true_count = temp_data[col+'_num'][temp_data[col+'_num'] == True].sum(
            null_count = temp_data[col].isna().sum()
            ent.append(entries)
            num.append(true_count)
            non.append(entries-true_count)
            nul.append(null_count)
        else:
            print(f'{col} {temp_data[col].dtype}')
    summary = pd.DataFrame(list(zip(feat, ent, num, non, nul)))
    summary.columns = out_cols
    return summary
```

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

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

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

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

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

Null Value Functions

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

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

```
    new_data[feature] = data[feature].fillna(-1)
    return new_data
```

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

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

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

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

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

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

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

Reprocessing Functions

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

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

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

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

Save Image Functions

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

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

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

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

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

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

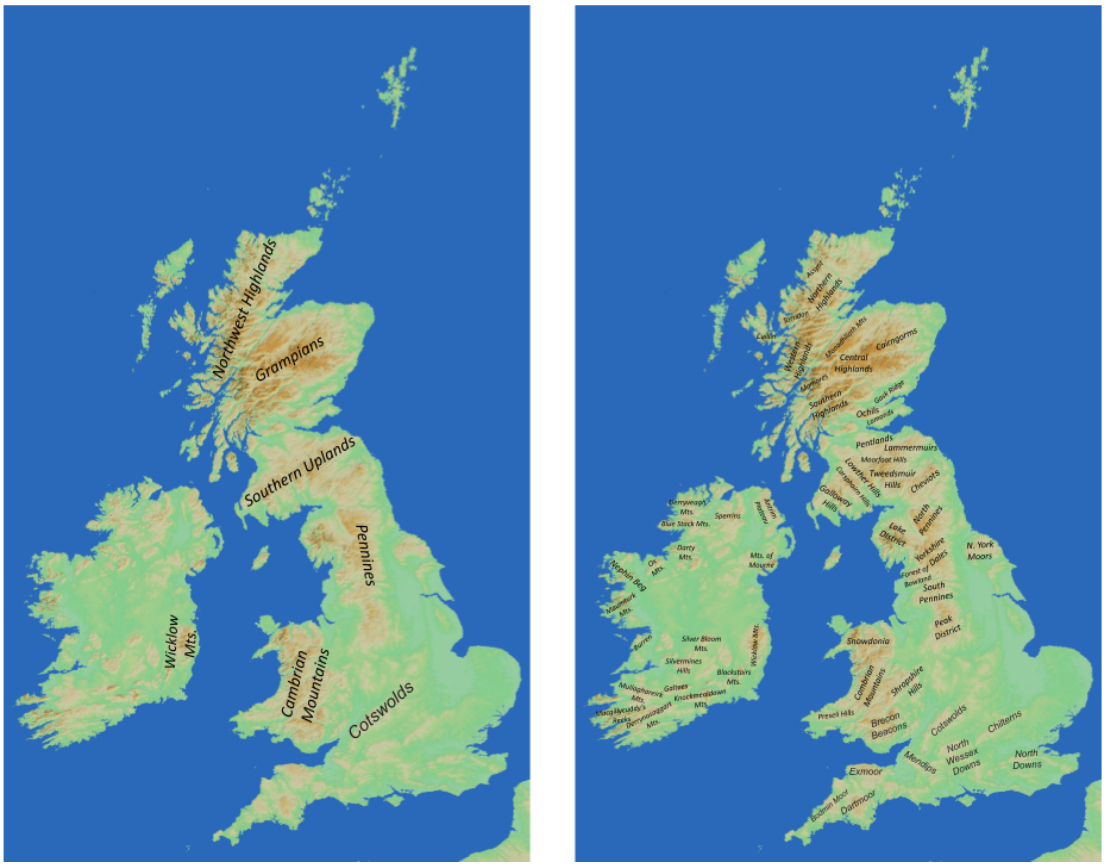
In [ ]:
```
def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    global fig_no
    global IMAGES_PATH
```

```
    if save_images:
        #IMAGES_PATH = r'/content/drive/My Drive/Colab Notebooks/Hillforts_Primer_1
        fig_no+=1
        fig_no_txt = format_figno(fig_no)
        file_name = file_name = get_file_name(f'{part}_{fig_no_txt}')
        file_name = f'hillforts_primer_{file_name}{topo_txt}.{fig_extension}'
        fig_list.loc[len(fig_list)] = [fig_no, file_name, get_print_title(fig_id)]
        path = os.path.join(IMAGES_PATH, file_name)
        print("Saving figure", file_name)
        plt.tight_layout()
        plt.savefig(path, format=fig_extension, dpi=resolution,  bbox_inches='tight
    else:
        pass
```

Mountains & Hills

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

CopernicusDEM:
https://image.discomap.eea.europa.eu/arcgis/rest/services/GioLandPublic/DEM/MapServer
World Hillshade:
https://services.arcgisonline.com/arcgis/rest/services/Elevation/World_Hillshade/MapServer/0



The Hills and Mountains of Britain and Ireland
Mike Middleton *CC BY-SA 3.0*

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/ma
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-93-2b53084ab660>:2: DtypeWarning: Columns (10,12,68,83,84,85,86,16
5,183) have mixed types. Specify dtype option on import or set low_memory=False.
  hillforts_data = pd.read_csv(hillforts_csv, index_col=False)
```

Out[ ]:

	OBJECTID	Main_Atlas_Number	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site
0	1	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbur
1	2	2	EN	England	EN0002 Bach Camp, Herefordshire	Bac

Download Function

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

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

Review Data

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

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

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

Bias

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

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

Missing Data

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

This study will use two values to replace null data:

- In numeric data, -1;
- in text data, ‘NA’.

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

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

Rows and Features (columns)

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

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

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

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

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

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

Name and Number

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

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

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...

Admin Data

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

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

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

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

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

Out[ ]:

	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_Na
0	1	EN	England	EN0001 Aconbury Camp, Herefordshire	Aconbury Camp	Aconb Beac
1	2	EN	England	EN0002 Bach Camp, Herefordshire	Bach Camp	N
2	3	EN	England	EN0003 Backbury Camp, Herefordshire	Backbury Camp	Ethelbe Ca
3	4	EN	England	EN0004 Brandon Camp, Herefordshire	Brandon Camp	N
4	5	EN	England	EN0005 British Camp, Herefordshire	British Camp	Herefordsh Beac

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

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

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

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

In [ ]:
```
admin_data_numeric_review = test_numeric(admin_data)
admin_data_numeric_review
```

```
OBJECTID int64
```

Out[ ]:

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

Admin Numeric Data

OBJECTID

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

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

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

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

Out[ ]:

	OBJECTID
0	1
1	2
2	3
3	4
4	5

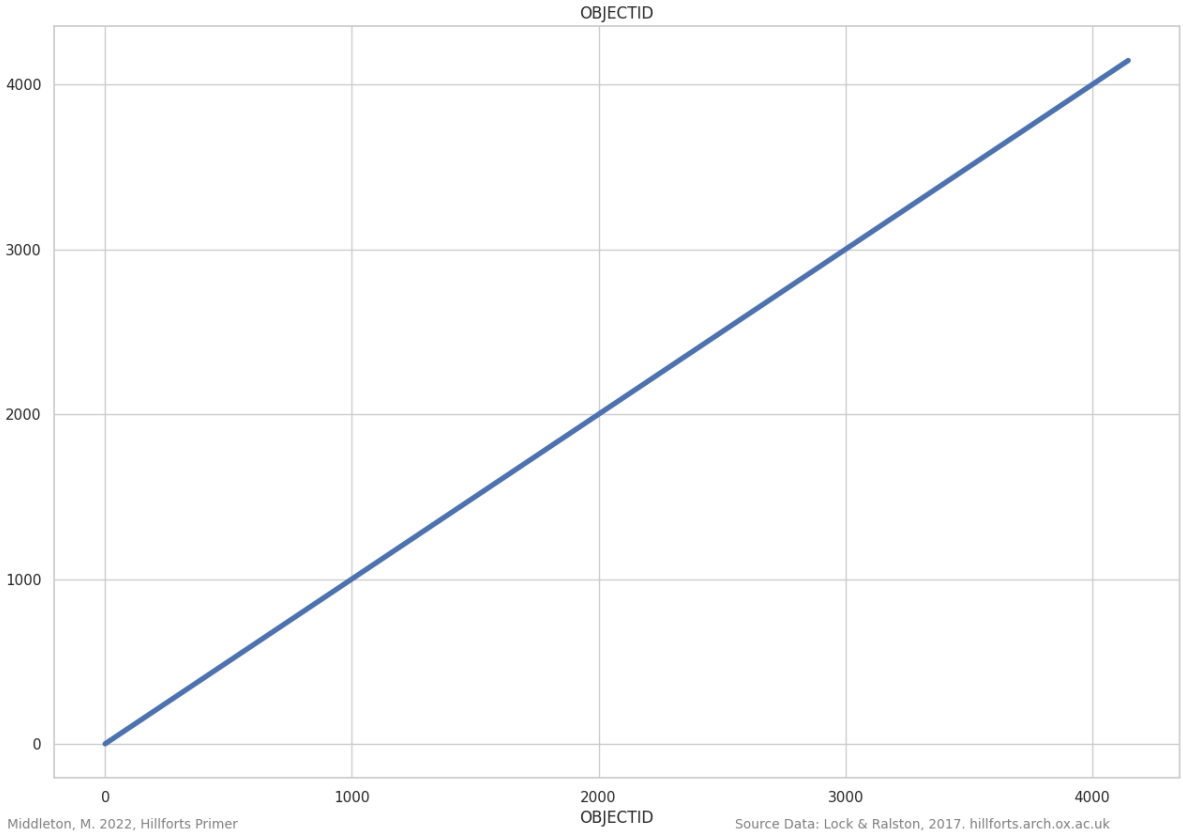
In [ ]:
```
admin_numeric_data['OBJECTID'].describe()
```

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

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

```
0 duplicates.
```

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



Admin Text Data

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

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

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

Out[ ]:

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

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

Out[ ]:

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

Main Title Name

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

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

Out[ ]:

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

Main Site Name

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

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

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

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

Out[ ]:

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

Main Alt Name

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

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

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

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

Out[ ]:

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

Main Summary

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

In [ ]:
```
sample_summary = admin_text_data['Main_Summary'][admin_text_data['Main_Title_Name'
sample_summary
```

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

Admin Text Data - Resolve Null Values

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

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

```
Main_Alt_Name 0
```

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

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

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

Admin Encodable Data

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

In [ ]:
```
admin_encodable_features = [
 'Main_Country_Code',
 'Main_Country',
 'Main_HER',
 'Main_HER_PRN',
 'Main_HER_ID',
```

```
 'Main_NMR_Mapsheet',
 'Main_NMR_ID',
 'Main_SM',
 'Main_Boundary']

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

Out[ ]:

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Ma
0	EN	England	Herefordshire	MHE413	910	SO 5
1	EN	England	Herefordshire	MHE52	344	SO 5
2	EN	England	Herefordshire	MHE411	908	SO 5
3	EN	England	Herefordshire	MHE818	1639	SO 4
4	EN	England	Herefordshire	MHE435	932	SO

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

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

Out[ ]:

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

Main Country Code

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

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

In [ ]:
```
pd.unique(admin_encodable_data['Main_Country_Code'])
```

Out[ ]:
```
array(['EN', 'WA', 'SC', 'NI', 'IR', 'IOM'], dtype=object)
```

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

Main Country Data Packages

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

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

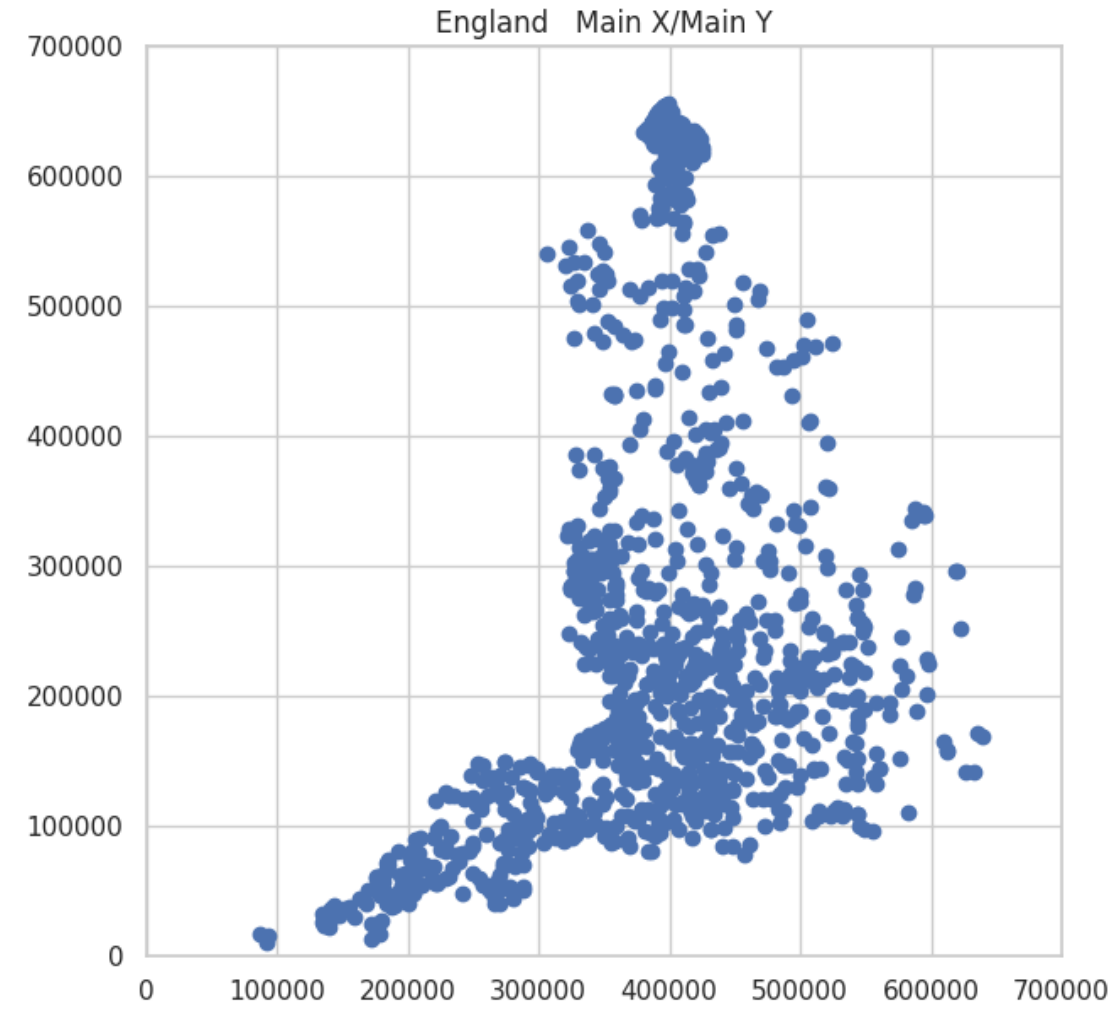
England

England contains 1225 records. The distribution of these records can be seen below, plotted against the Ordnance Survey National Grid.
See: Main_X / Main_Y
See: Main_Coordinate_System

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

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

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



Notes on Downloading Data

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

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

Wales

Wales contains 690 records. These are plotted against the Ordnance Survey National Grid.
See: Main_X / Main_Y
See: Main_Coordinate_System

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

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

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

See: Notes on Downloading Data

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

Scotland

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

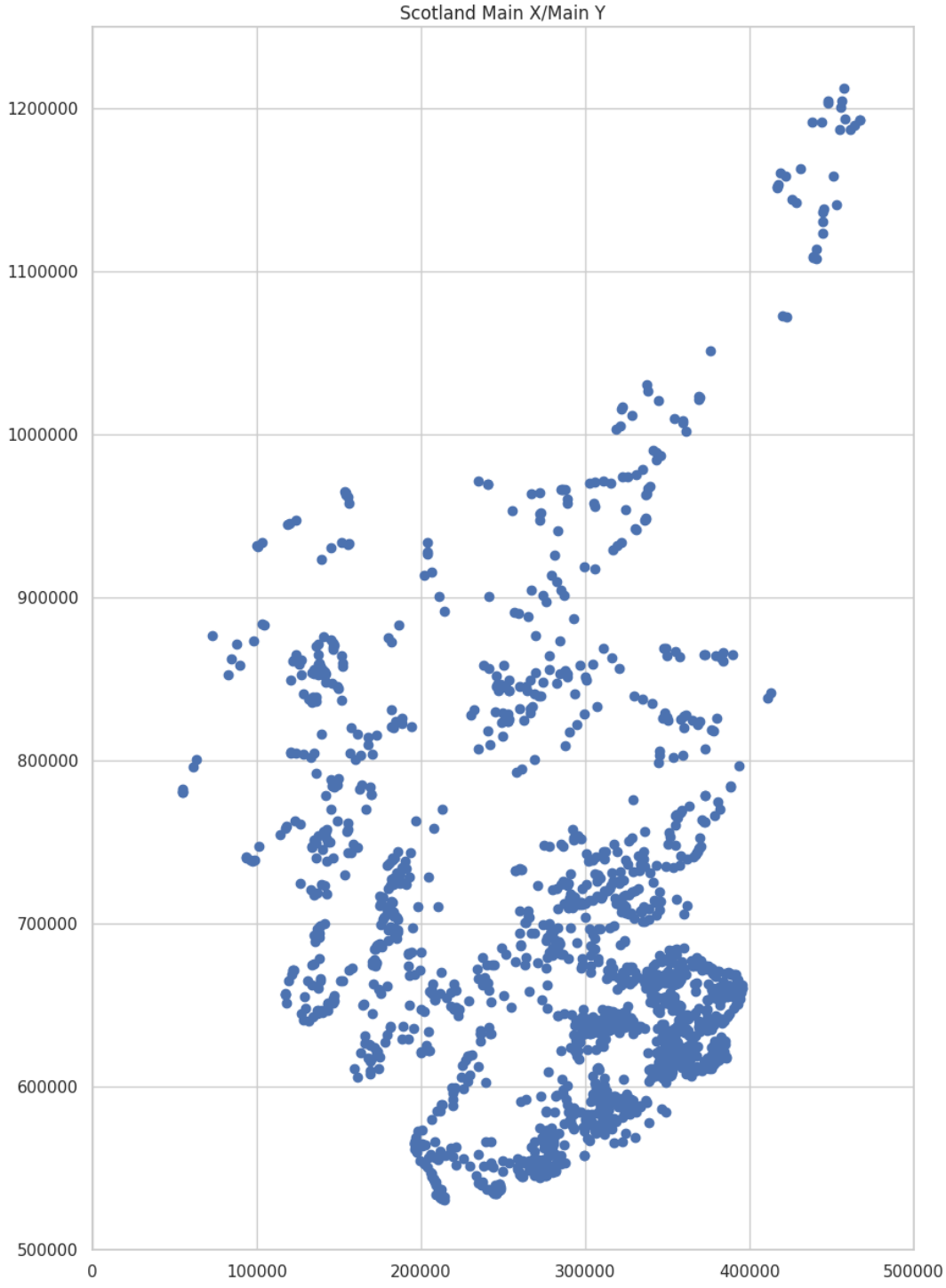
See: Main_X / Main_Y

See: Main_Coordinate_System

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

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

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



See: Notes on Downloading Data

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

Northern Ireland

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

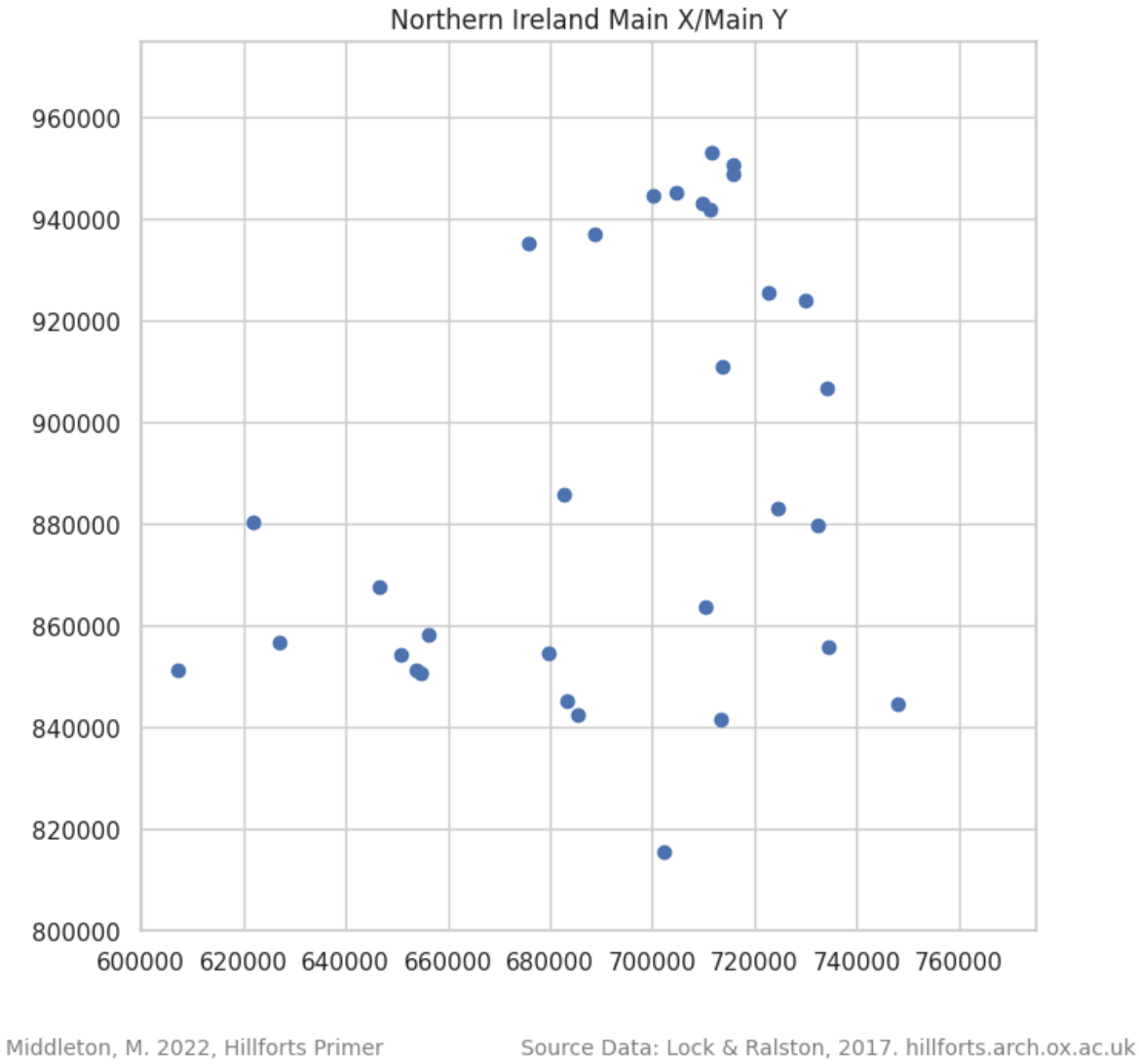
See: Main_X / Main_Y

See: Main_Coordinate_System

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

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

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



See: Notes on Downloading Data

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

Republic of Ireland

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

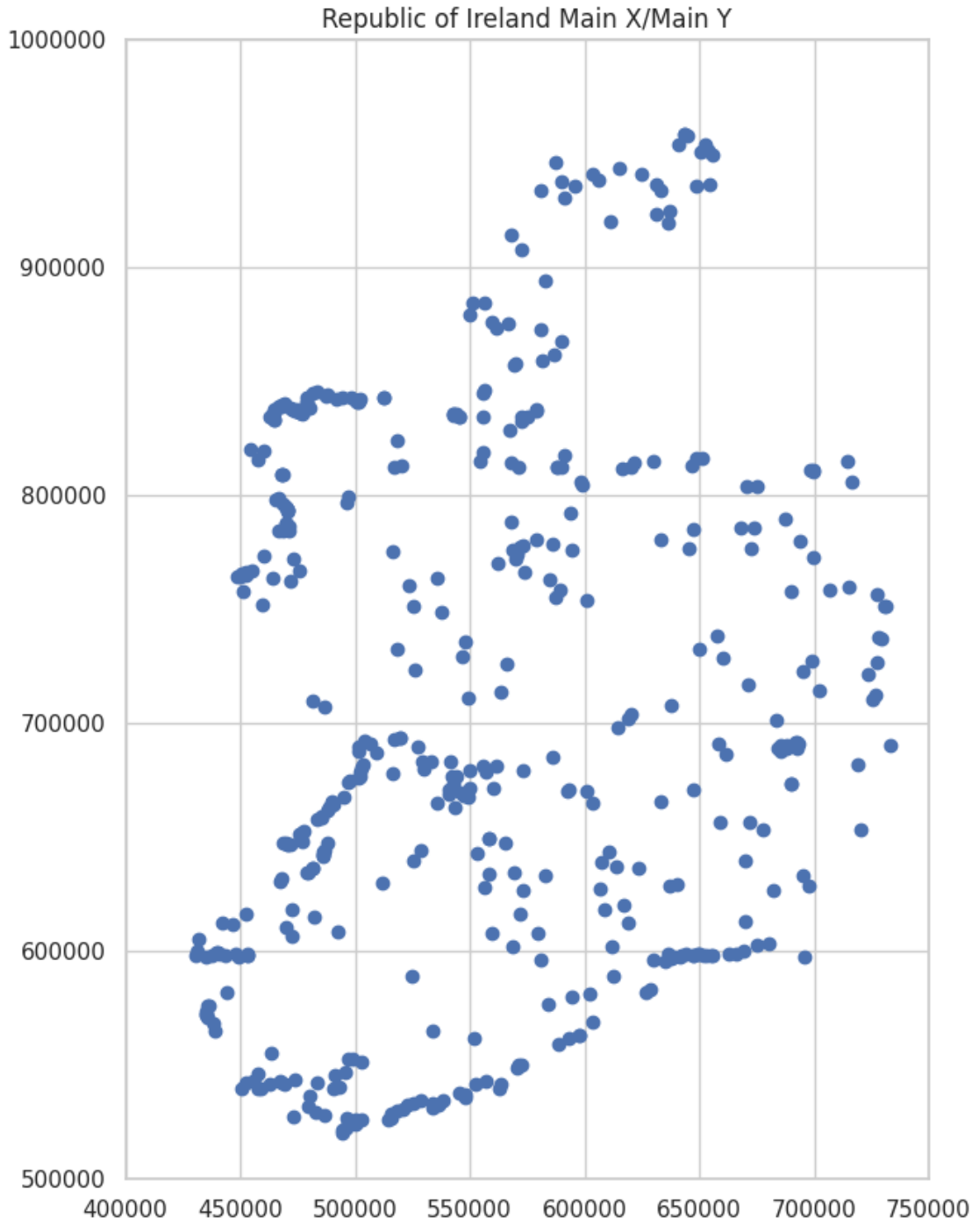
See: Main_X / Main_Y

See: Main_Coordinate_System

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

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

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



See: Notes on Downloading Data

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

Isle Of Man

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

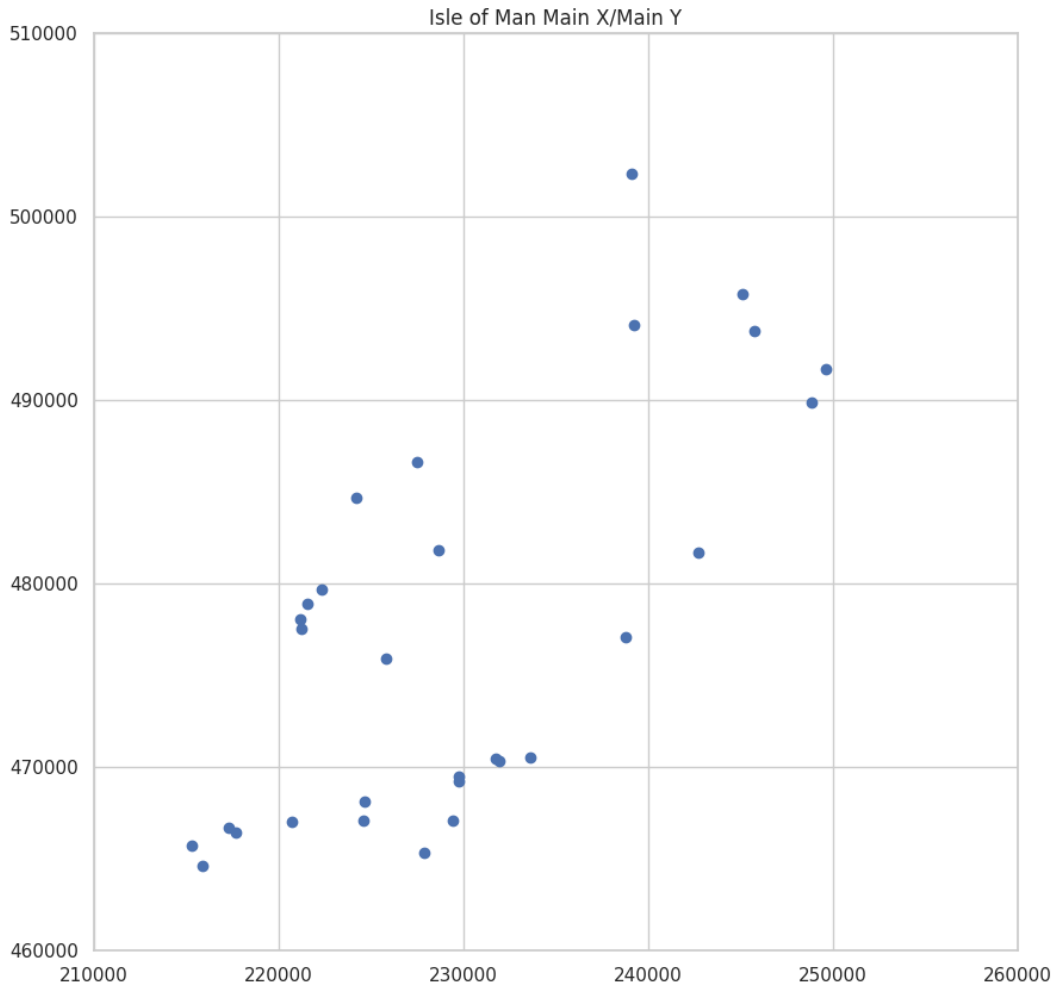
See: Main_X / Main_Y

See: Main_Coordinate_System

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

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

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



See: Notes on Downloading Data

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

Republic of Ireland and Northern Ireland Combined

All 507 records on the island or Ireland.

In [ ]:
```
roi_and_ni_data = hillforts_data[hillforts_data['Main_Country'].isin(['Republic of
```

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

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

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

See: Notes on Downloading Data

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

England, Wales, Scotland and Man Combined

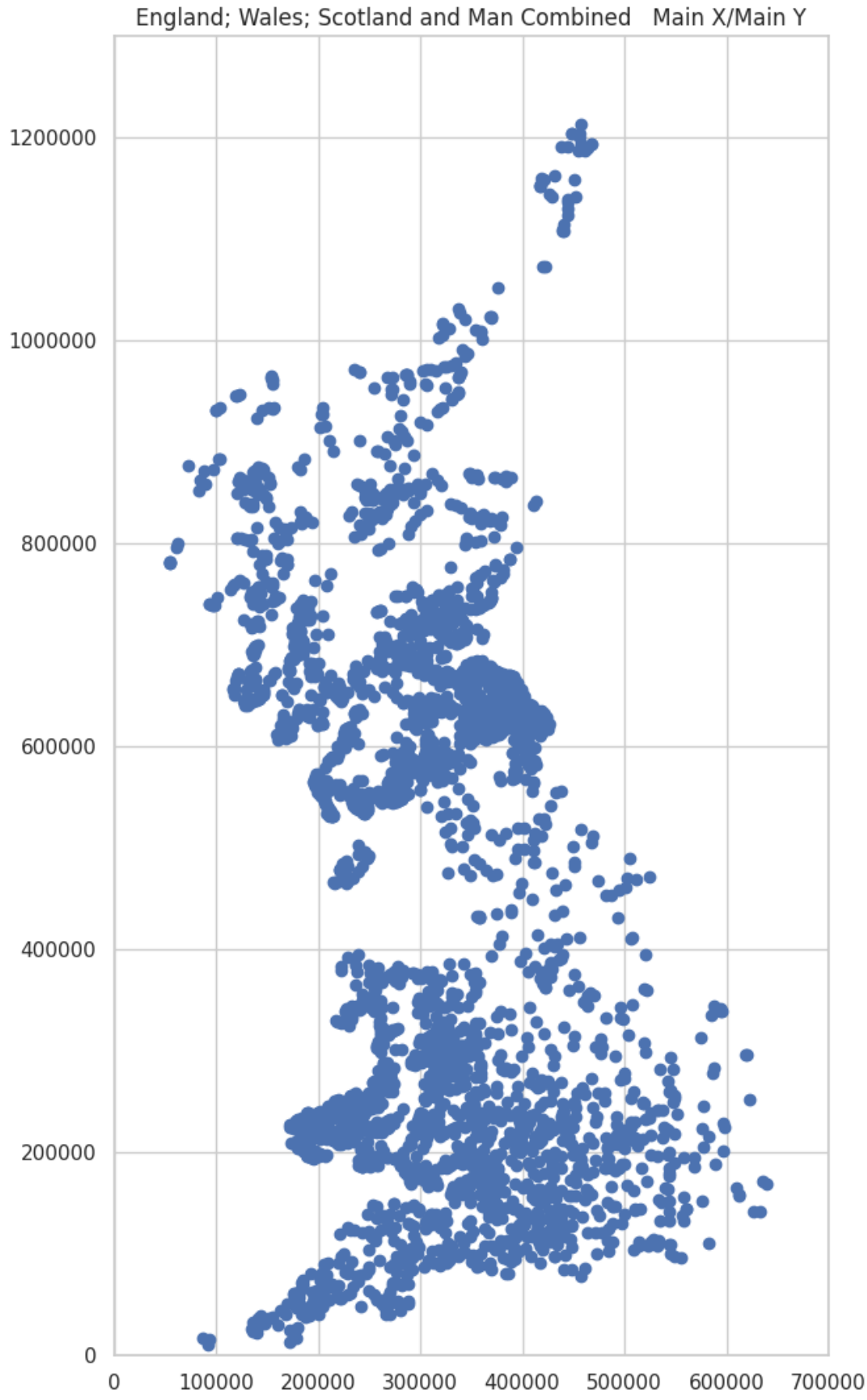
All Atlas data excluding the island or Ireland. 3640 records.

In [ ]:
```
eng_wal_sco_man_data = hillforts_data[hillforts_data['Main_Country'].isin(['Englan
```

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

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

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



See: Notes on Downloading Data

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

Main Country

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

In [ ]:
```
pd.unique(admin_encodable_data['Main_Country'])
```

Out[ ]:
```
array(['England', 'Wales', 'Scotland', 'Northern Ireland',
       'Republic of Ireland', 'Isle of Man'], dtype=object)
```

Main HER

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

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

In [ ]:
```
admin_encodable_data[admin_encodable_data['Main_HER'].isna()]
```

Out[ ]:

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_M
1141	SC	Scotland	NaN	No record found	NaN	NS
1375	SC	Scotland	NaN	No record found	NaN	NS 5
1401	SC	Scotland	NaN	No record found	NaN	NS 6
1405	SC	Scotland	NaN	No record found	NaN	NS
1420	SC	Scotland	NaN	No record found	NaN	NS 7
1423	SC	Scotland	NaN	No record found	NaN	NS
4145	SC	Scotland	NaN	NaN	NaN	NH

In [ ]:
```
Main_HER_not_null = admin_encodable_data[~admin_encodable_data['Main_HER'].isna()]
Main_HER_not_null['Main_HER'].describe()
```

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

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

Out[ ]:
```
Archaeological Survey of Ireland SMR Database                          475
Scottish Borders                                                       408
The West of Scotland Archaeology Service                               326
Dyfed                                                                  302
Dumfries & Galloway                                                    286
Northumberland                                                         270
Highland HER                                                           208
Clwyd Powys                                                            184
Glamorgan Gwent                                                        108
Gwynedd                                                                 96
East Lothian Council                                                    89
Devon                                                                   86
Cornwall and Scilly                                                     80
Perth and Kinross Heritage Trust                                        76
Shropshire                                                              63
Hampshire                                                               50
Wiltshire and Swindon                                                   49
Gloucestershire                                                         47
Fife Council                                                            44
Aberdeenshire Historic Environment Record                               42
Somerset                                                                38
Herefordshire                                                           38
Oxfordshire                                                             37
Dorset                                                                  37
Angus SMR per Aberdeenshire Council                                     35
Shetland Amenity Trust                                                  33
Northern Ireland Sites and Monuments Record                             32
Isle of Man                                                             30
Comhairle nan Eilean Siar - Western Isles Sites and Monuments Record    30
Stirling                                                                30
Name: Main_HER, dtype: int64
```

Main HER PRN

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

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

In [ ]:
```
null_Main_HER_PRN = admin_encodable_data[admin_encodable_data['Main_HER_PRN'].isna
```

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

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

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

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

In [ ]:
```
Main_HER_not_null = admin_encodable_data[~admin_encodable_data['Main_HER_PRN'].isn
Main_HER_not_null['Main_HER_PRN'].describe()
```

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

Main HER ID

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

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

Out[ ]:

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Ma
0	EN	England	Herefordshire	MHE413	910	SO 5
1	EN	England	Herefordshire	MHE52	344	SO 5
2	EN	England	Herefordshire	MHE411	908	SO 5
3	EN	England	Herefordshire	MHE818	1639	SO 4
4	EN	England	Herefordshire	MHE435	932	SO

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

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

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

Out[ ]:

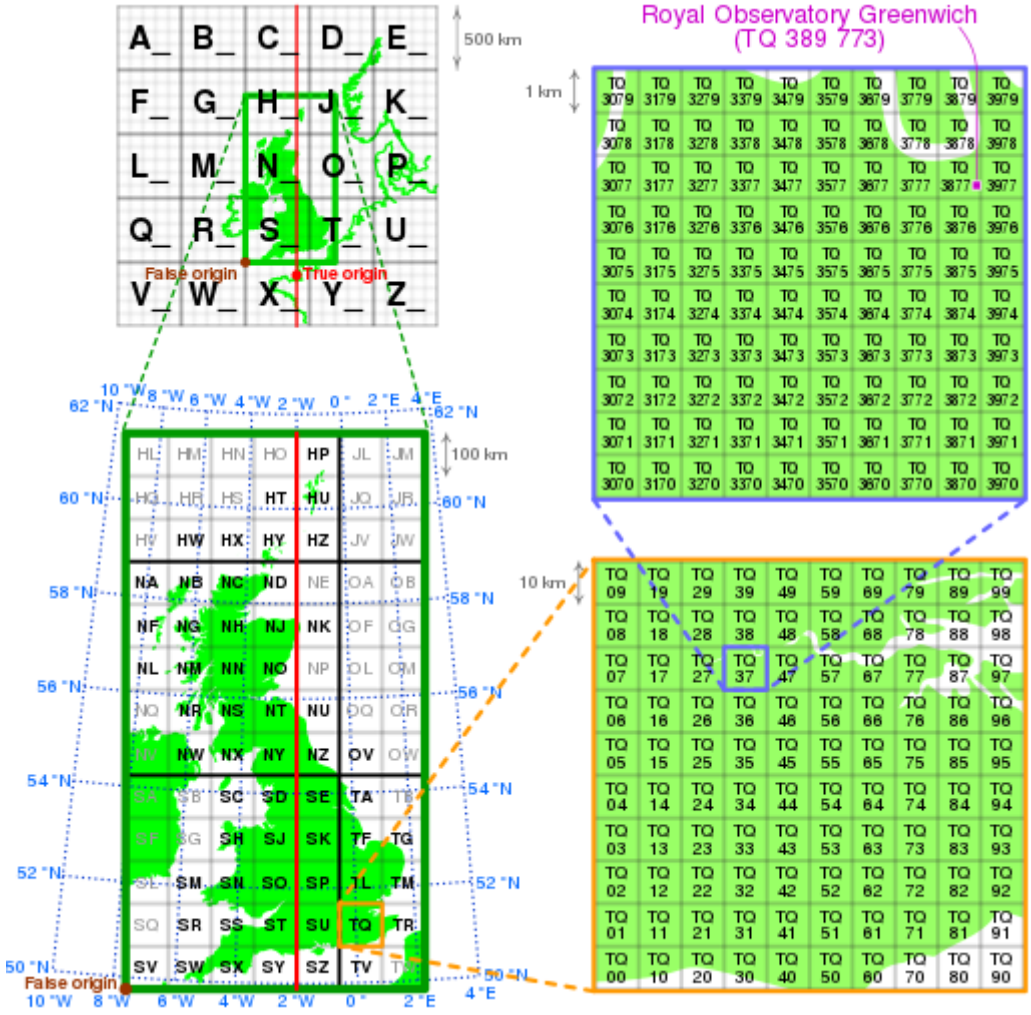
	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt_I
751	752	EN	England	EN0773 Borough Hill 1, Northamptonshire	Borough Hill 1	Borouc
752	753	EN	England	EN0774 Borough Hill 2, Northamptonshire	Borough Hill 2	Boroug Nor

Main NMR Mapsheet

The National Monuments Record (NMR) Mapsheets contains concatenated information relating to the Ordnance Survey mapsheet and the NMR site number. It also contains null

values. These will be resolved in Admin Encodable Data - Resolve Null Values and Admin Encodable Data - Main NMR Mapsheet Replaced.

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



The Ordnance Survey National Grid
cmglee, Strebe, MansLaughter, Alexrk2 from naturalearthdata, Pethrus and nandhp, *CC BY-SA 3.0*, *via Wikimedia Commons*

The first record below is on the 10km Ordnance Survey sheet: SO 53;
The NMR 5km Recordsheet is: SO 53 SW (southwest);
The site number on sheet SO 53 SW is one.

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

Out[ ]:

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_Ma
0	EN	England	Herefordshire	MHE413	910	SO 5
1	EN	England	Herefordshire	MHE52	344	SO 5
2	EN	England	Herefordshire	MHE411	908	SO 5
3	EN	England	Herefordshire	MHE818	1639	SO 4
4	EN	England	Herefordshire	MHE435	932	SO

Main NMR ID

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

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

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

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

Out[ ]:

	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt
1483	1484	SC	Scotland	SC1559 The Tappoch, Torwood, Falkirk	The Tappoch, Torwood	
3553	3554	EN	England	EN3750 Belle Tout, East Sussex	Belle Tout	

Main SM

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

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

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

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

Out[ ]:

	OBJECTID	Main_Country_Code	Main_Country	Main_Title_Name	Main_Site_Name	Main_Alt
3522	3523	SC	Scotland	SC3716 Dunsapie, City of Edinburgh	Dunsapie	Edir Holyroc Dunsap I
3523	3524	SC	Scotland	SC3717 Salisbury Crags, Edinburgh, City of Edi...	Salisbury Crags, Edinburgh	Edir Holyroc Salisbury
3525	3526	SC	Scotland	SC3719 Arthur's Seat, City of Edinburgh	Arthur's Seat	Edir Holyroc Arthur C
3526	3527	SC	Scotland	SC3720 Samson's Ribs, Arthur's Seat, City of E...	Samson's Ribs, Arthur's Seat	Edir Holyroc Arthur S

Main Boundary

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

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

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

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

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

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

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

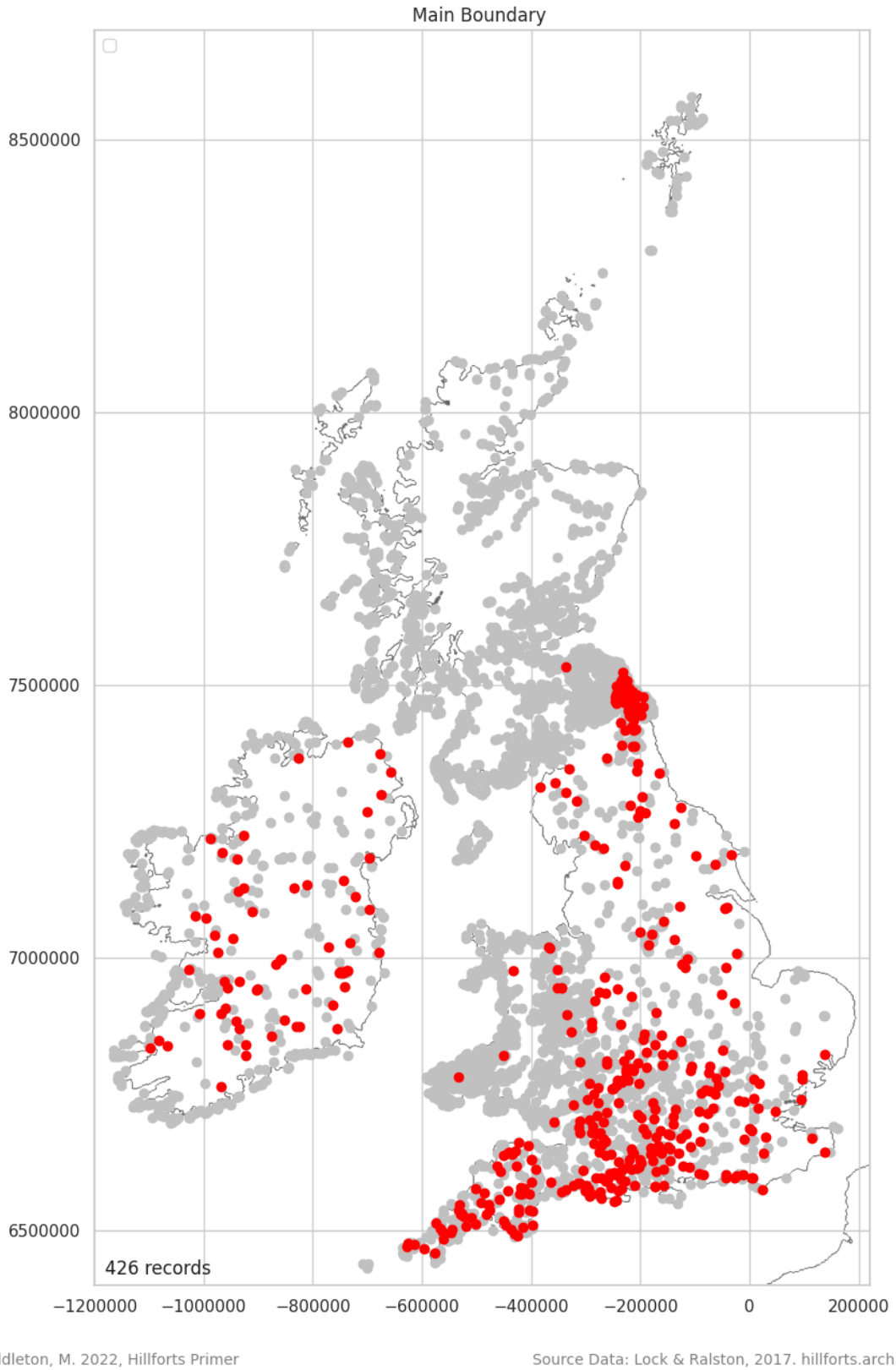
Main Boundary Mapped

The map below shows that there is a recording bias in this feature. There is only one record in Scotland and very few in Wales and Northern Ireland. The majority are in England and

even here, the data is patchy as can be seen by the void in the data, south of the River Severn, toward the western end of the Mendips.

In [ ]:
```
location_data = load_location(hillforts_data)
location_admin_data = pd.merge(location_data, admin_encodable_data, left_index=Tru
```

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




```
10.27%
```

Admin Encodable Data - Resolve Null Values

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

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

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

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

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

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

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

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

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

In [ ]:
```
admin_encodable_data = update_cat_list_for_NA(admin_encodable_data, cat_update_lis
admin_encodable_data = update_num_list_for_minus_one(admin_encodable_data, num_upd
```

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

Out[ ]:

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

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

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

Out[ ]:

	Main_Country_Code	Main_Country	Main_HER	Main_HER_PRN	Main_HER_ID	Main_NMR_M
1141	SC	Scotland	NA	No record found	-1	NS
1375	SC	Scotland	NA	No record found	-1	NS 5
1401	SC	Scotland	NA	No record found	-1	NS 6
1405	SC	Scotland	NA	No record found	-1	NS
1420	SC	Scotland	NA	No record found	-1	NS 7
1423	SC	Scotland	NA	No record found	-1	NS
4145	SC	Scotland	NA	NA	-1	NH

In [ ]:
```

```

Review Admin Data Split

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

In [ ]:
```
review_data_split(admin_data, admin_numeric_data, admin_text_data, admin_encodable_
```

```
Data split good.
```

Restructure Admin Encodable Features Containing Mixed Data Formats

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

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

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

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

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

In [ ]:
```
admin_encodable_data_plus = admin_encodable_data.copy()
num_update_list = []
txt_update_list = []
for col in reformat_list:
    admin_encodable_data_plus = split_text_num(admin_encodable_data_plus, col)
    admin_encodable_data_plus = update_cat_list_for_NA(admin_encodable_data_plus,
    admin_encodable_data_plus = update_num_list_for_minus_one(admin_encodable_data_
```

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

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

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

Out[ ]:

	Main_Country_Code	Main_Country	Main_HER	Main_NMR_Mapsheet	Main_Boundary	Main_
0	EN	England	Herefordshire	SO 53 SW 1	No	
1	EN	England	Herefordshire	SO 56 SW 3	No	
2	EN	England	Herefordshire	SO 53 NE 2	No	
3	EN	England	Herefordshire	SO 47 SW 2	No	
4	EN	England	Herefordshire	SO 74 SE 3	Yes	
5	EN	England	Herefordshire	NA	No	

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

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

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

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

In [ ]:
```
pd.DataFrame(Main_HER_not_null[Main_HER_not_null['Main_HER_ID'].str.contains('/', r
```

Out[ ]:

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

Restructured data for record 497.

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

Out[ ]:

	Main_HER_ID_num	Main_HER_ID_txt
497	10208	MNA

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

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

In [ ]:
```
pd.DataFrame(Main_HER_not_null[Main_HER_not_null['Main_HER_ID'].str.contains(',', r
```

Out[ ]:

	Main_HER_ID
113	2303627, 2402293, 2303627, 2744876, 2303627
116	2303474, 2399905, 2303474, 2399906, 2303474

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

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

Out[ ]:

	Main_HER_ID_num	Main_HER_ID_txt
113	2303627	NA

Main NMR Mapsheet Replaced

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

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

In [ ]:
```
admin_encodable_data_plus[['Main_NMR_Mapsheet']].head(6)
```

Out[ ]:

	Main_NMR_Mapsheet
0	SO 53 SW 1
1	SO 56 SW 3
2	SO 53 NE 2
3	SO 47 SW 2
4	SO 74 SE 3
5	NA

In [ ]:
```
admin_encodable_data_plus[['pt1','pt2']] = admin_encodable_data_plus['Main_NMR_Maps
admin_encodable_data_plus[['pt2','pt3']] = admin_encodable_data_plus['pt2'].str.sp
admin_encodable_data_plus[['pt3','Site_no']] = admin_encodable_data_plus['pt3'].st
admin_encodable_data_plus['OS_Mapsheet'] = admin_encodable_data_plus['pt1'] + admi
admin_encodable_data_plus = admin_encodable_data_plus.drop(['pt1','pt2', 'pt3', 'M
admin_encodable_data_plus = update_cat_list_for_NA(admin_encodable_data_plus, ['OS_
admin_encodable_data_plus = update_num_list_for_minus_one(admin_encodable_data_plus
admin_encodable_data_plus[['OS_Mapsheet', 'Site_no']].head(6)
```

```
<ipython-input-183-d5d0d1a396b6>:1: FutureWarning: In a future version of pandas a
ll arguments of StringMethods.split except for the argument 'pat' will be keyword-
only.
  admin_encodable_data_plus[['pt1','pt2']] = admin_encodable_data_plus['Main_NMR_M
apsheet'].str.split(" ", 1, expand=True)
<ipython-input-183-d5d0d1a396b6>:2: FutureWarning: In a future version of pandas a
ll arguments of StringMethods.split except for the argument 'pat' will be keyword-
only.
  admin_encodable_data_plus[['pt2','pt3']] = admin_encodable_data_plus['pt2'].str.
split(" ", 1, expand=True)
<ipython-input-183-d5d0d1a396b6>:3: FutureWarning: In a future version of pandas a
ll arguments of StringMethods.split except for the argument 'pat' will be keyword-
only.
  admin_encodable_data_plus[['pt3','Site_no']] = admin_encodable_data_plus['pt3'].
str.split(" ", 1, expand=True)
```

Out[ ]:

	OS_Mapsheet	Site_no
0	SO53SW	1
1	SO56SW	3
2	SO53NE	2
3	SO47SW	2
4	SO74SE	3
5	NA	-1

Drop Duplicate Admin Features

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

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

Ref: https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb

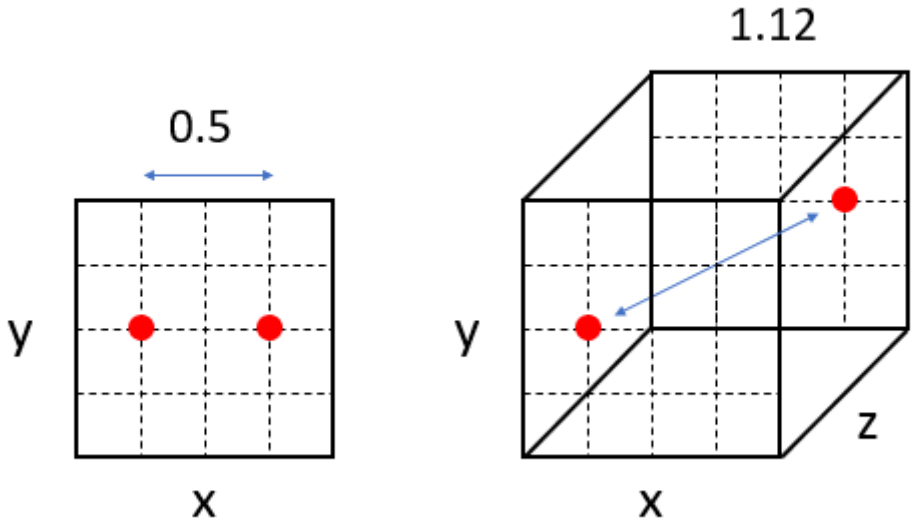


Figure showing how adding a dimension can increase the distance between data points and clusters
Mike Middleton *CC BY-SA 3.0*

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

Out[ ]:

	Main_Country_Code	Main_HER	Main_Boundary	Main_HER_PRN_num	Main_HER_PRN_txt	M
0	EN	Herefordshire	No	413	MHE	
1	EN	Herefordshire	No	52	MHE	
2	EN	Herefordshire	No	411	MHE	
3	EN	Herefordshire	No	818	MHE	
4	EN	Herefordshire	Yes	435	MHE	
5	EN	Herefordshire	No	21869	MHE	

Admin Data Packages

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

In [ ]:
```
admin_data_list = [admin_numeric_data, admin_text_data, admin_encodable_data_plus]
```

Download Admin Data

The download will only function is selected by the user. See: User Settings.

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

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

Location Data

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

In [ ]:
```
location_features = [
 'Main_Coordinate_System',
 'Main_X',
 'Main_Y',
 'Location_NGR',
 'Location_X',
 'Location_Y',
 'Location_Longitude',
 'Location_Latitude',
 'Location_Current_County',
 'Location_Historic_County',
 'Location_Current_Parish']

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

Out[ ]:

	Main_Coordinate_System	Main_X	Main_Y	Location_NGR	Location_X	Location_Y	Location_Lon
0	OSGB36	350350	233050	SO 503330	-303295	6798973	-2.
1	OSGB36	354700	260200	SO 547602	-296646	6843289	-2.
2	OSGB36	358700	238900	SO 587389	-289837	6808611	-2.
3	OSGB36	340000	272400	SO 400724	-320850	6862993	-2.
4	OSGB36	376000	240000	SO 760400	-261765	6810587	-2.

Only 'Location_NGR' contains null values.

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

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

Location Numeric Data

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

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

In [ ]:
```
location_numeric_features = [
 'Main_X',
 'Main_Y',
 'Location_X',
 'Location_Y',
 'Location_Longitude',
 'Location_Latitude']

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

Out[ ]:

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

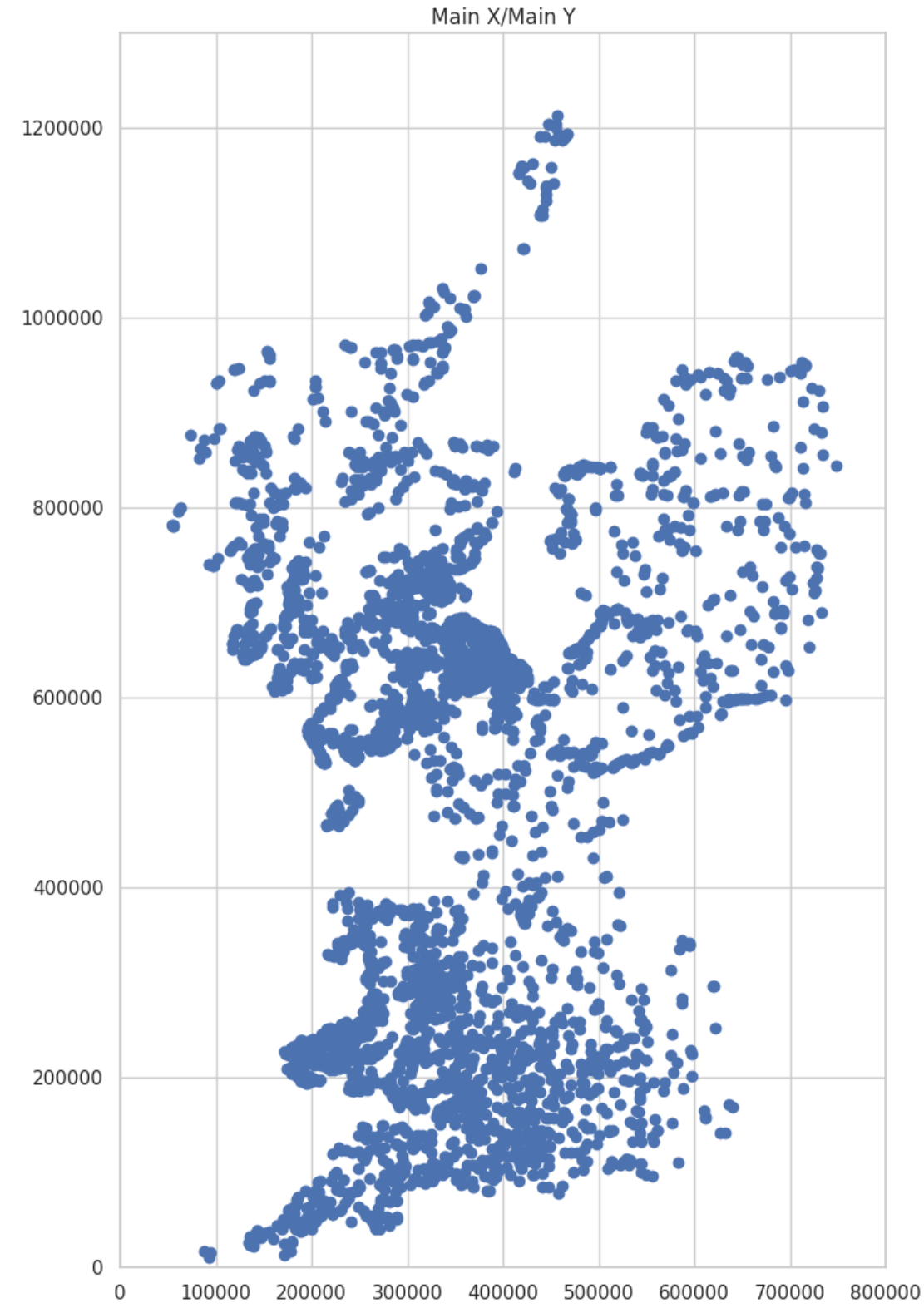
Main_X / Main_Y Mapped

'Main_X' and 'Main_Y' store the coordinates for each record based on the local grid. See: Location - Main Coordinate System. This means that when plotting the atlas data, using

'Main_X' and 'Main_Y', the Irish and UK grids plot over one another.

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

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



See: Location - Main Coordinate System)

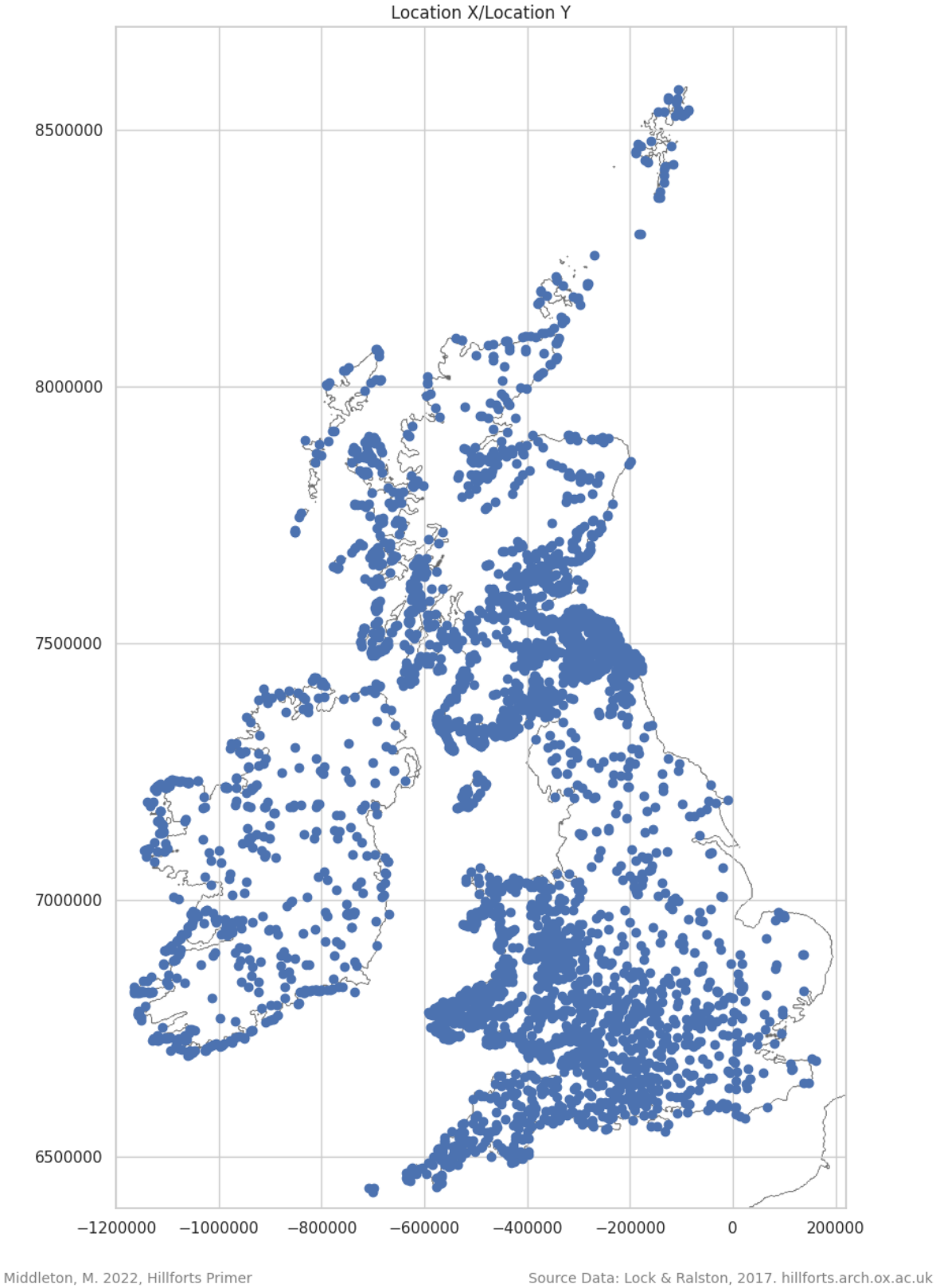
Location_X / Location_Y Mapped

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

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

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

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



Location_X Data Plotted - Easting

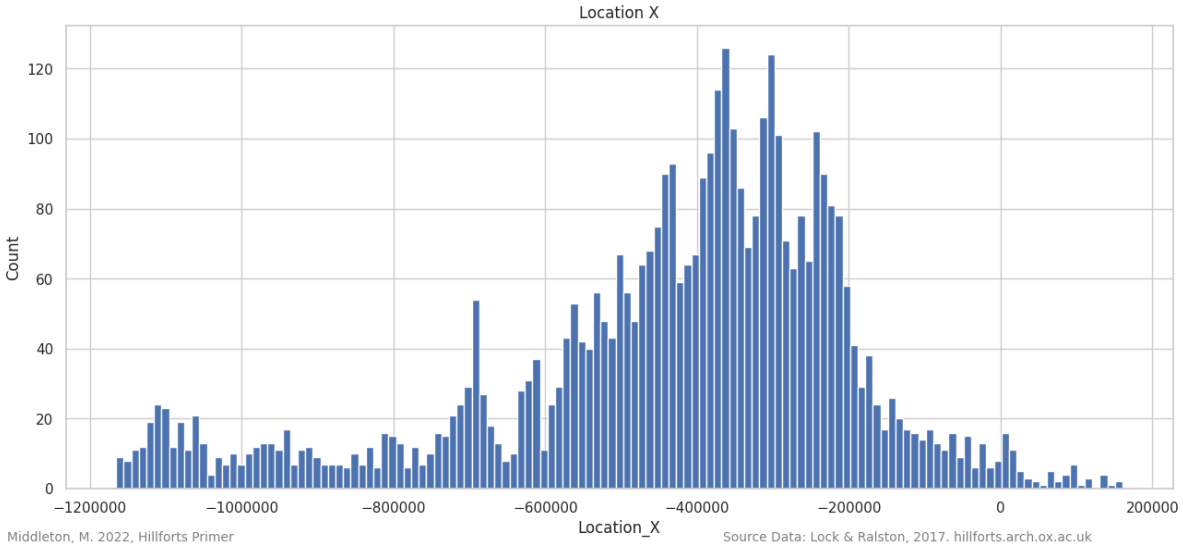
Within the 'Location_X' plot there is a single large peak and a number of smaller peaks, rising out of the main peak. The data is plotted using 10km 'bins' (counts per 10km). These will be examined in more detail below.

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

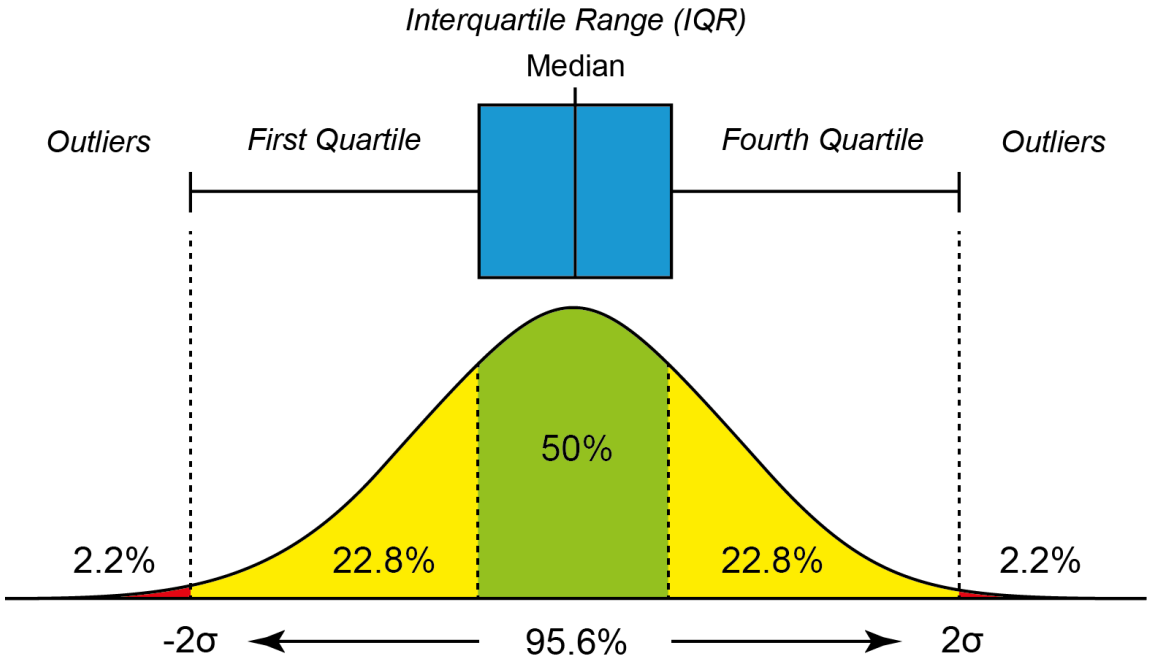
Out[ ]:
```
count    4.147000e+03
mean    -4.387439e+05
std      2.493718e+05
min     -1.165487e+06
25%     -5.398565e+05
50%     -3.839160e+05
75%     -2.779725e+05
max      1.606020e+05
Name: Location_X, dtype: float64
```

In [ ]:
```
plot_histogram(location_numeric_data['Location_X'], 'Location_X', 'Location_X', 10(
```

```
10000
```



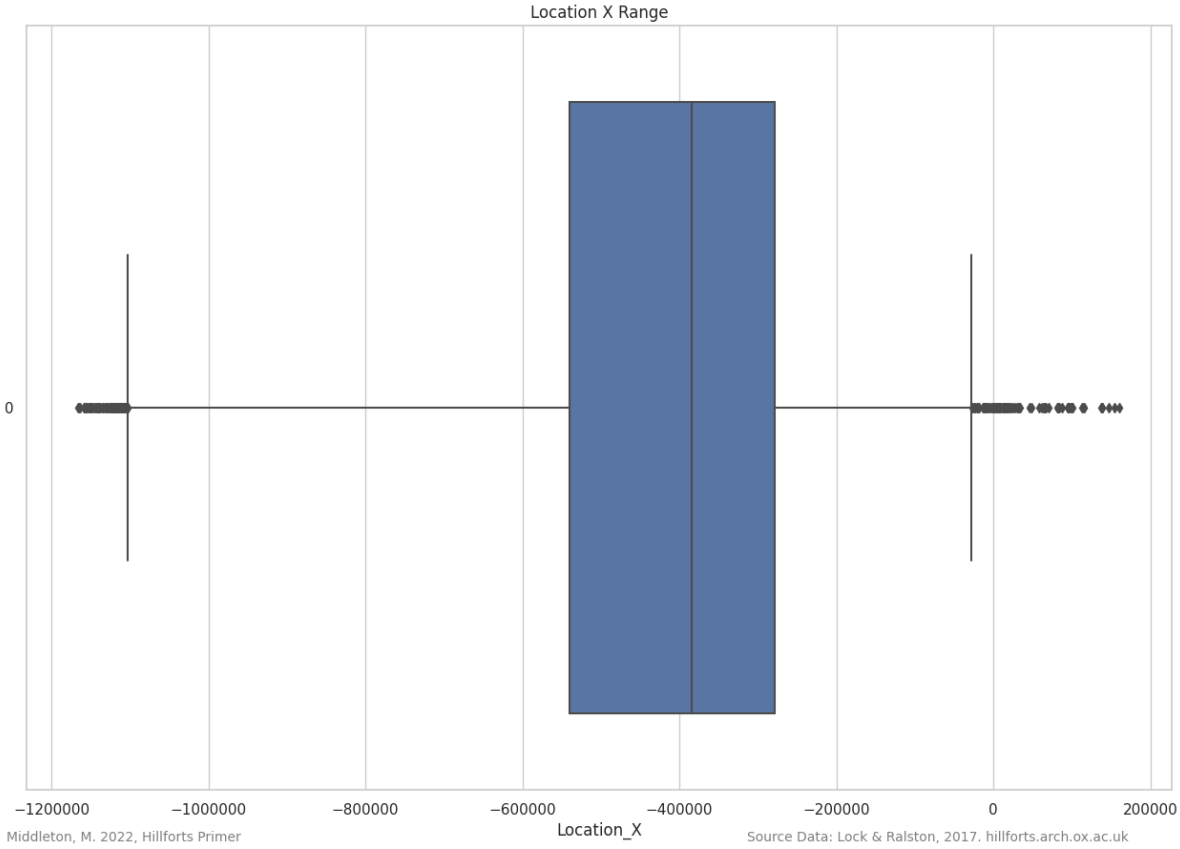
A boxplot shows the distribution of data. The central blue box shows the interquartile range (IQR), within which sits quarters two & three of the data (the mid 50%). The whiskers, ending in the black vertical lines, indicate the extent of all but 4.4% of the remaining data or the equivalent of two standard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



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

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

In [ ]:
```
location_X_data = plot_data_range(location_numeric_data['Location_X'], 'Location_X
```



Location_Y Data Plotted - Northing

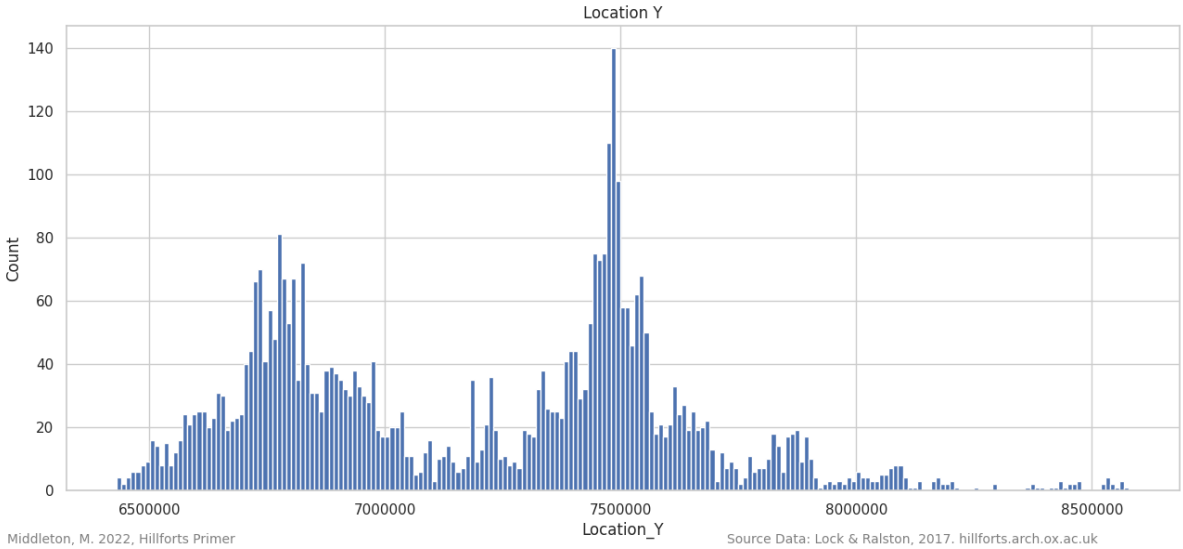
Within the 'Location_Y' data, there are two large peaks. These will be explored in more detail in Location - Add Density Feature - North & South Density. Again, the data is plotted using 10km 'bins'.

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

Out[ ]:
```
count    4.147000e+03
mean     7.194890e+06
std      4.145320e+05
min      6.430971e+06
25%      6.804457e+06
50%      7.243520e+06
75%      7.494886e+06
max      8.578090e+06
Name: Location_Y, dtype: float64
```

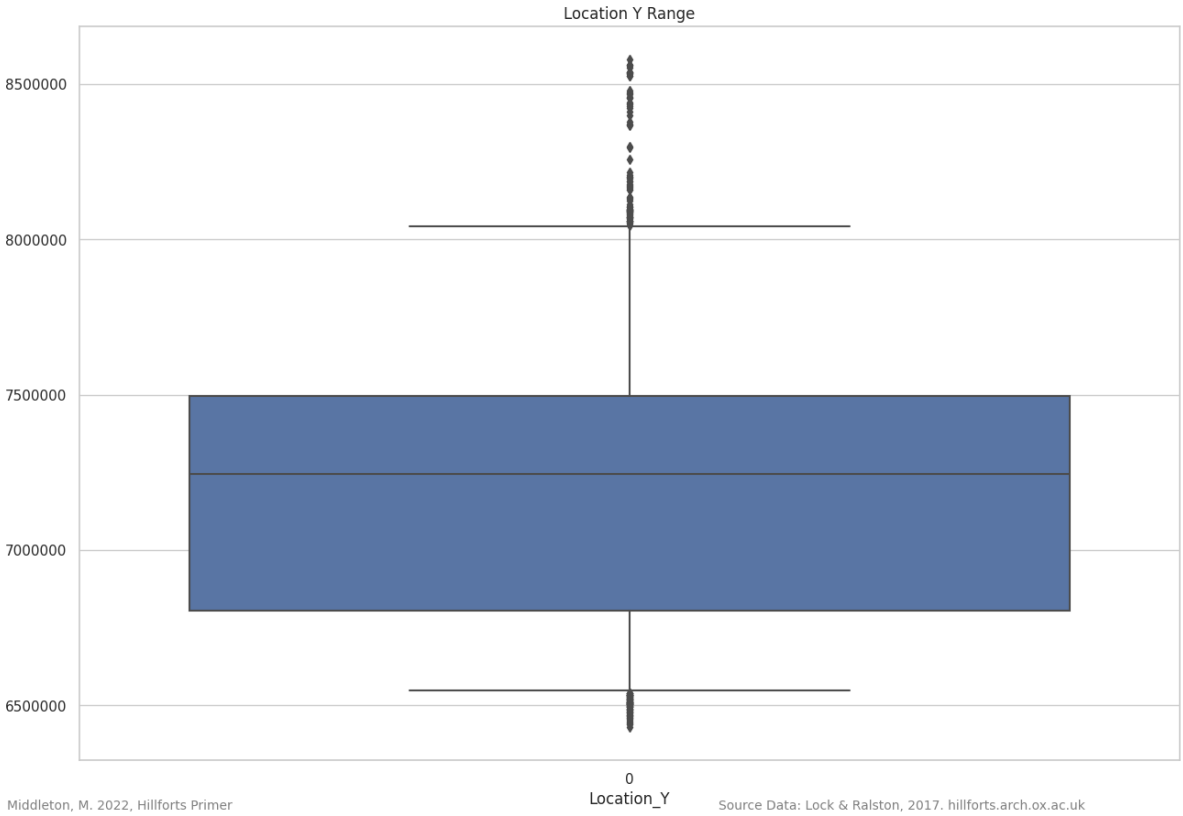
In [ ]:
```
plot_histogram(location_numeric_data['Location_Y'], 'Location_Y', 'Location_Y', 10(
```

```
10000
```



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

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

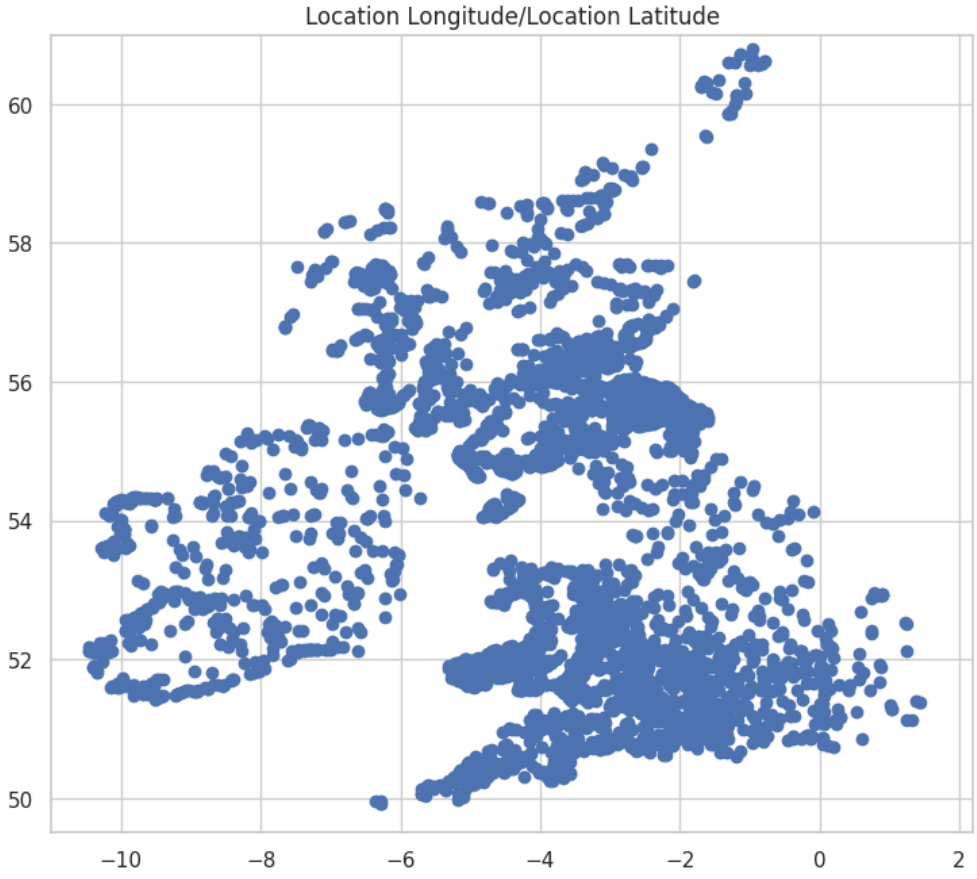


Longitude / Latitude Mapped

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

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

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



Middleton, M. 2022, Hillforts Primer

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

Location Text Data

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

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

Location Encodable Data

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

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

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

Out[ ]:

	Main_Coordinate_System	Location_NGR	Location_Current_County	Location_Historic_County	Loc
0	OSGB36	SO 503330	Herefordshire	Herefordshire	
1	OSGB36	SO 547602	Herefordshire	Herefordshire	
2	OSGB36	SO 587389	Herefordshire	Herefordshire	
3	OSGB36	SO 400724	Herefordshire	Herefordshire	
4	OSGB36	SO 760400	Herefordshire	Herefordshire	

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

Out[ ]:

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

Main Coordinate System

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

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

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

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

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

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

Location_NGR

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

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

300000, 200000 = SO

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

503, 330 = 503330

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

SO 503330 = 350300, 233000

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

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

Out[ ]:

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

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

Location Current County

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

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

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

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

In [ ]:
```
location_encodable_data['Location_Current_County'].value_counts()[:25]
```

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

Location Historic County

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

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

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

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

In [ ]:
```
location_encodable_data['Location_Historic_County'].value_counts()[:25]
```

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

Location Current Parish

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

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

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

There are 20 null values.

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

Out[ ]:
```
20
```

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

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

Out[ ]:
```
Cavers                                21
Kildalton And Oa                      20
Kirknewton                            19
Coldingham                            19
Kilninian And Kilmore                 18
Peebles                               18
Rerrick                               17
Chatton                               16
Lauder                                15
Ingram                                15
Garvald And Bara                      15
Kirkmaiden                            14
Ford                                  14
Small Isles                           13
Snizort                               13
Jedburgh                              12
Kilmore And Kilbride                  12
Horncliffe                            12
Hownam                                12
Broughton, Glenholm And Kilbucho      12
Campbeltown                           11
Doddington                            11
Whithorn                              11
Farr                                  11
Duirinish                             11
Name: Location_Current_Parish, dtype: int64
```

Review Location Data Split

In [ ]:
```
review_data_split(location_data, location_numeric_data, location_text_data, locati
```

```
Data split good.
```

Drop Features From Location Numeric Data

Restructured Location Numeric Data

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

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

Out[ ]:

	Location_X	Location_Y
0	-303295	6798973
1	-296646	6843289
2	-289837	6808611
3	-320850	6862993
4	-261765	6810587

Drop Features From Location Encodable Data

Restructured Location Encodable Data

As 'Location_NGR' is a duplicate of the data in 'Main_X' and 'Main _Y' and the study is using 'Location_X' and 'Location_Y', it will be deleted. See: Location_NGR.

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

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

location_encodable_data_short = location_encodable_data[location_encodable_features
location_encodable_data_short.head()
```

Out[ ]:

	Location_Current_County	Location_Historic_County	Location_Current_Parish
0	Herefordshire	Herefordshire	Aconbury
1	Herefordshire	Herefordshire	Kimbolton
2	Herefordshire	Herefordshire	Dormington
3	Herefordshire	Herefordshire	Adforton
4	Herefordshire	Herefordshire	Colwall

Resolve Null Values in Location Current Parish

Test for 'NA' in 'Location_Current_Parish'.

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

```
Location_Current_Parish 0
```

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

In [ ]:
```
location_encodable_data_short = update_cat_list_for_NA(location_encodable_data_sho
```

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

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

Out[ ]:

	Feature	Entries	Numeric	Non-Numeric	Null
0	Location_Current_County	4147	0	4147	0
1	Location_Historic_County	4147	0	4147	0
2	Location_Current_Parish	4147	0	4147	0

Add Density Feature

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

Ref: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html
Ref: https://matplotlib.org/stable/tutorials/colors/colormaps.html

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

Out[ ]:

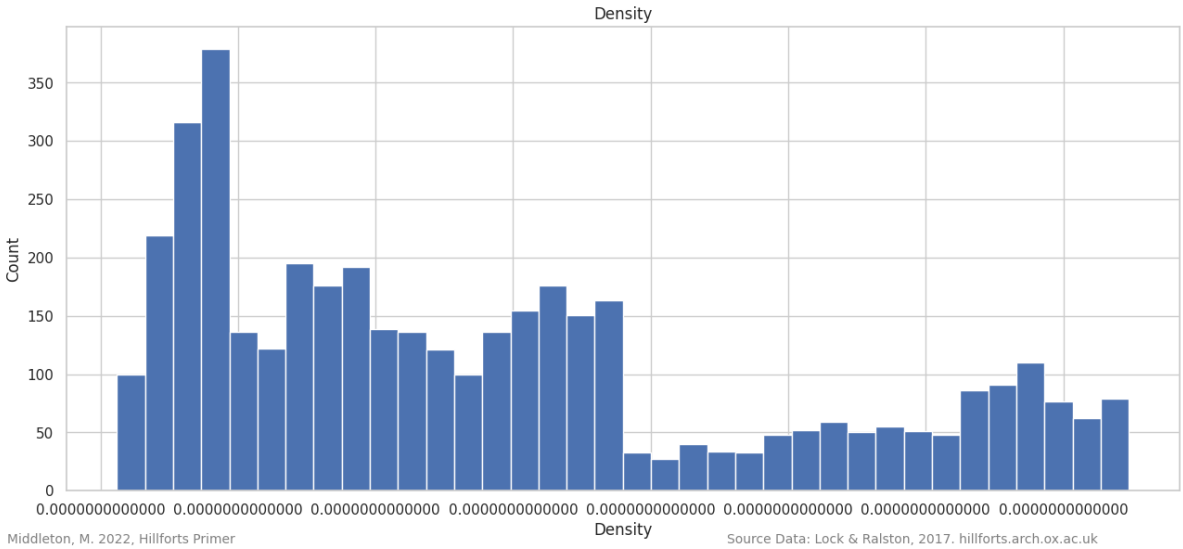
	Location_X	Location_Y	Density
0	-303295	6798973	1.632859e-12
1	-296646	6843289	1.540172e-12
2	-289837	6808611	1.547729e-12
3	-320850	6862993	1.670548e-12
4	-261765	6810587	1.369981e-12

Density Data Plotted

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

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

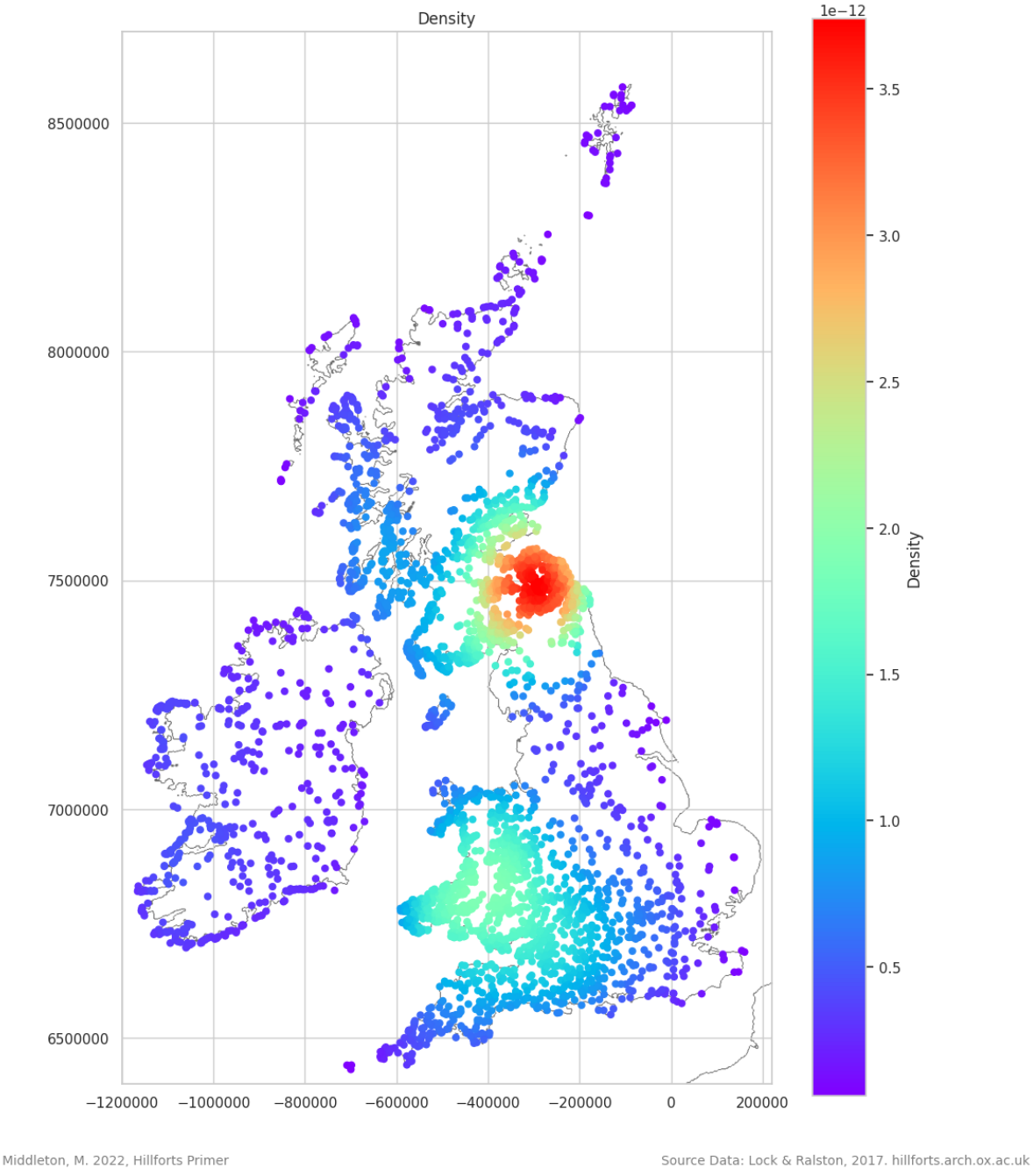
None



Density Data Mapped

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

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

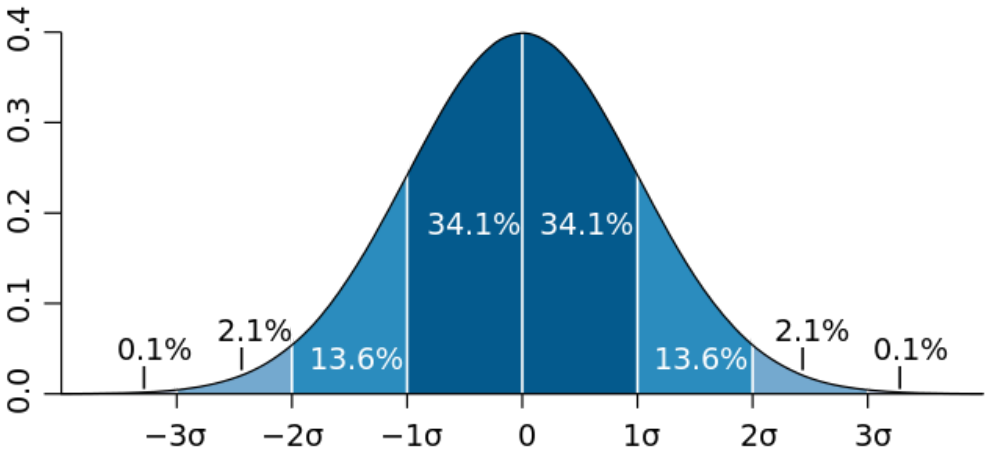


See: Location Data: Density Data Transformed Mapped.
See: Location Data: Ireland Density Data Mapped.

Density Data Transformed

In Density Data Plotted it was noted that the distribution of the density values was crushed to the left of the histogram and relatively flat across the remainder of the chart. Ideally, we

want the data to be distributed as close as is possible to a normal curve, with most of the data toward the centre of the histogram.



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

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

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

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

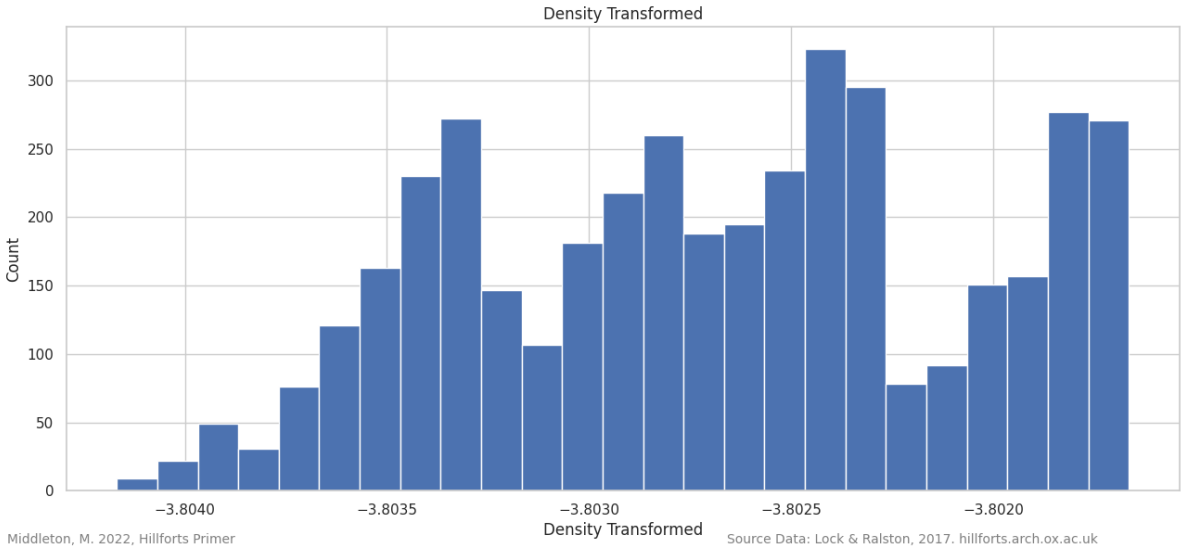
Out[ ]:

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

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

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

None



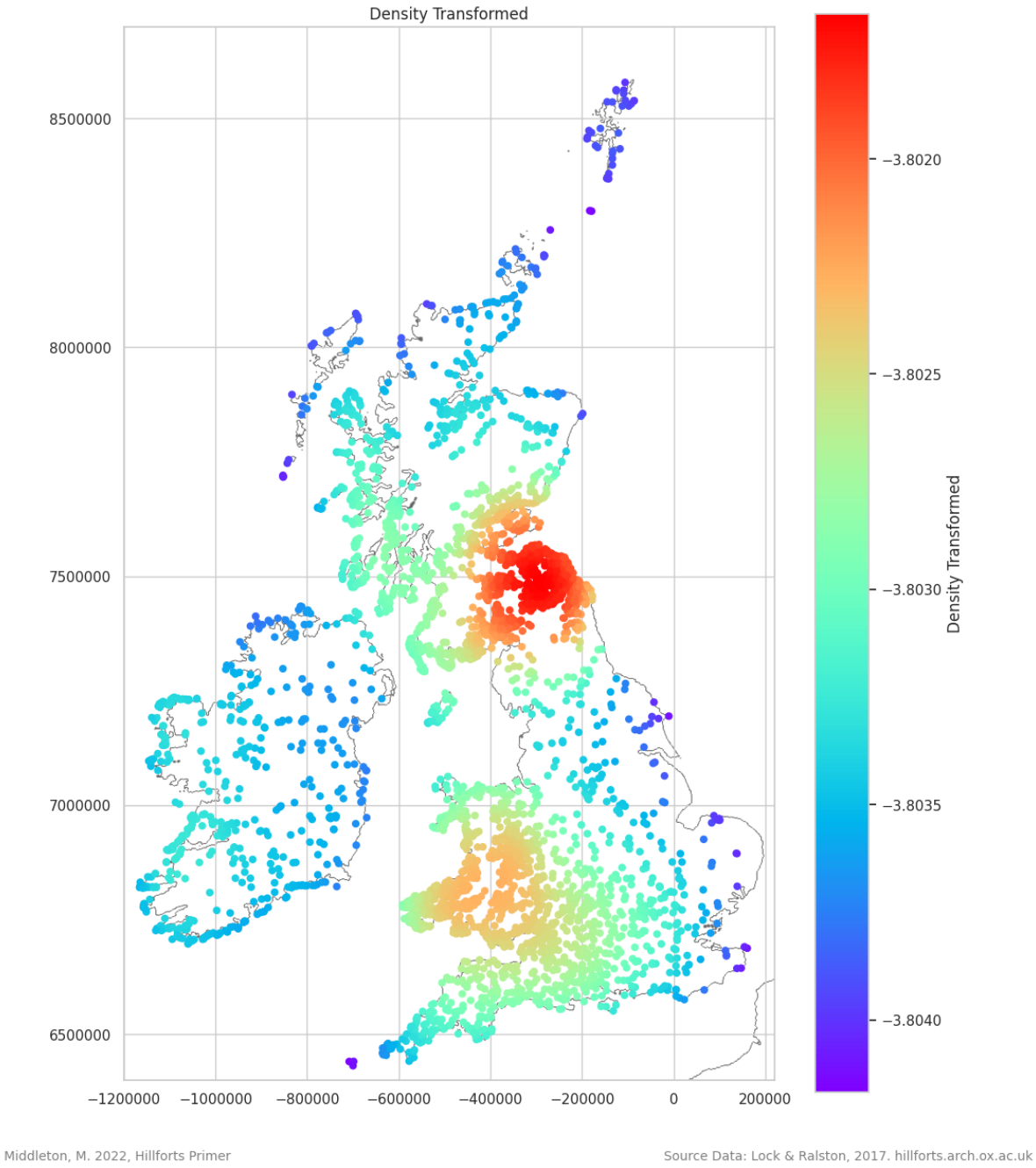
In [ ]:
```
best_lambda
```

Out[ ]:
```
0.2627814121099985
```

Density Data Transformed Mapped

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

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



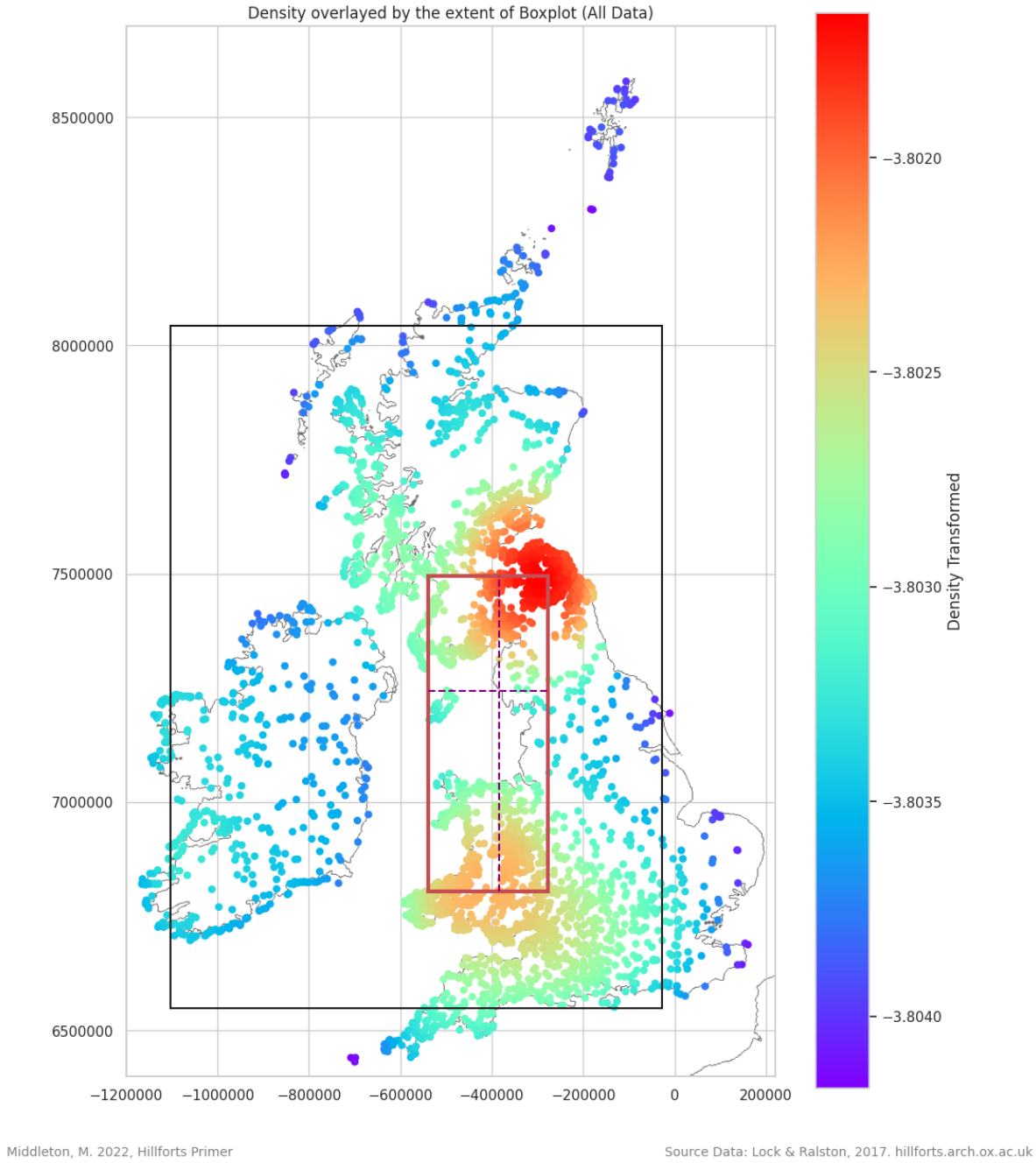
See: Location Data: Density Data Mapped.
See: Location Data: Ireland Density Data Mapped.

Density Map showing Extent of Boxplot

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

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

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



North / South Density Split

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

In [ ]:
```
split_location = 7070000
south = transformed_location_numeric_data_short[transformed_location_numeric_data_
north = transformed_location_numeric_data_short[transformed_location_numeric_data_
```

Northern Density

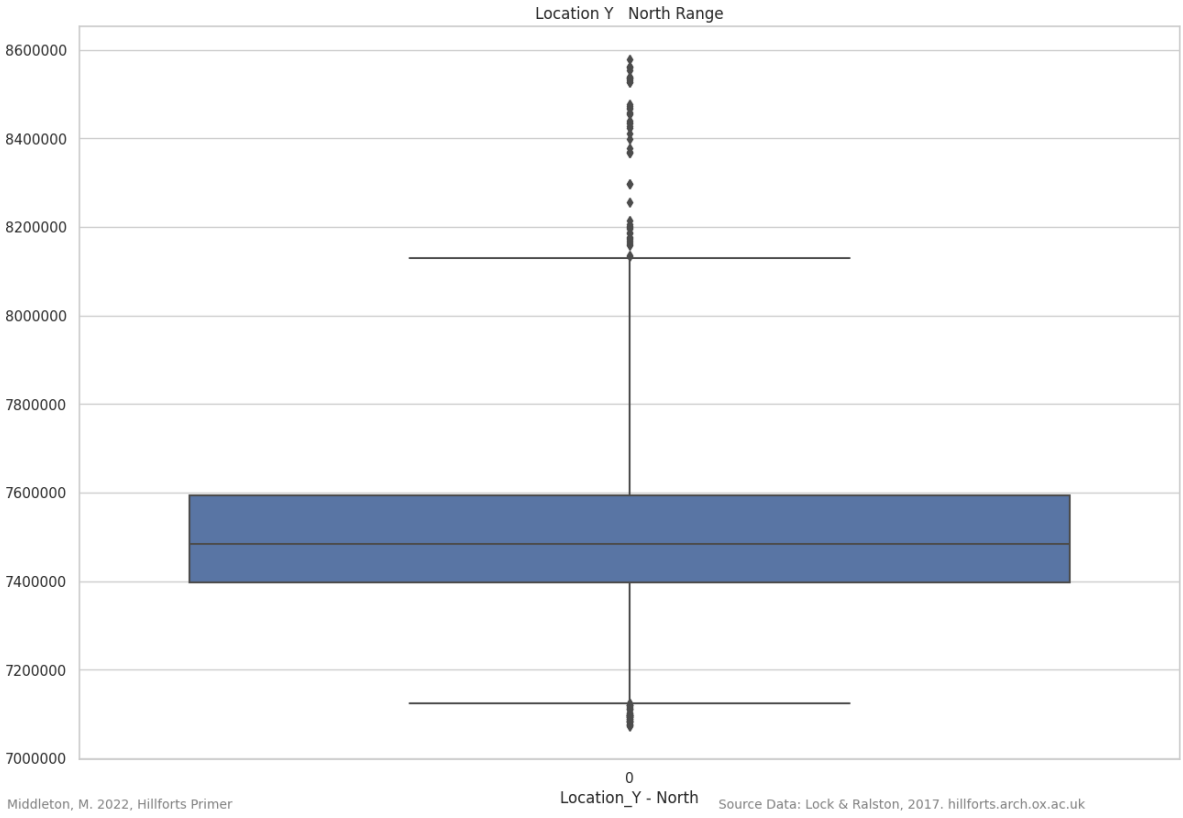
Northern Location Data Plotted

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

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

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

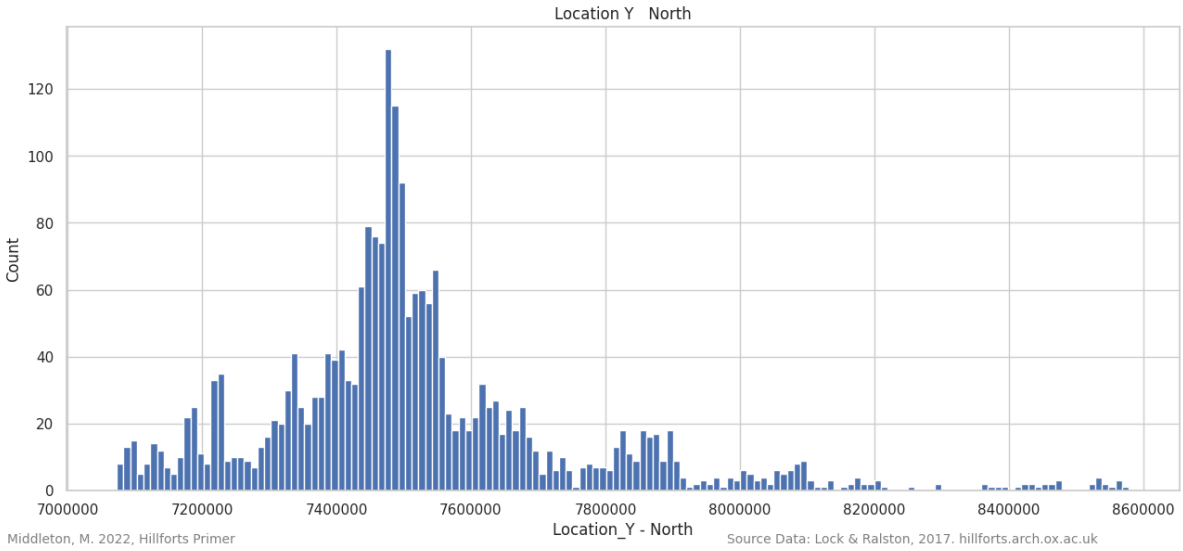
In [ ]:
```
location_Y_north_data = plot_data_range(north['Location_Y'], 'Location_Y - North')
```



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

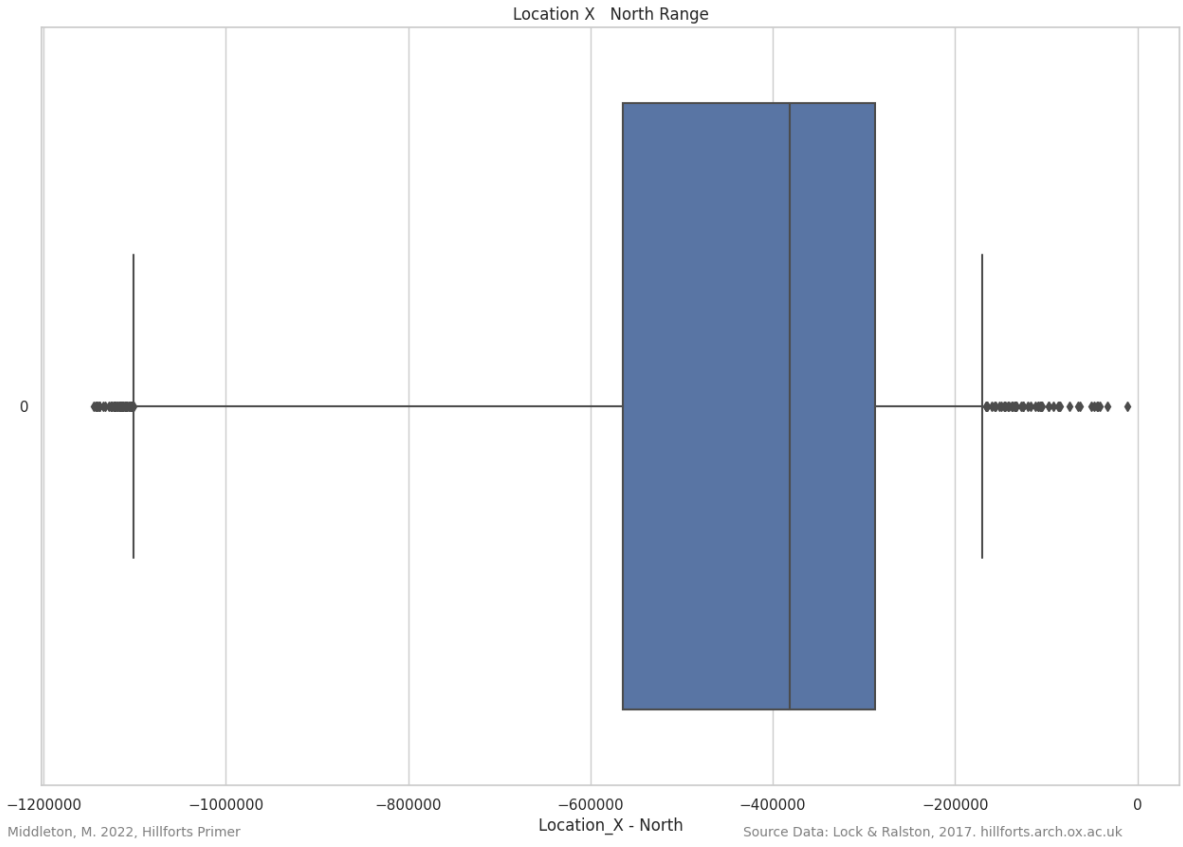
In [ ]:
```
plot_histogram(north['Location_Y'], 'Location_Y - North', 'Location_Y - North', 10
```

```
10000
```



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

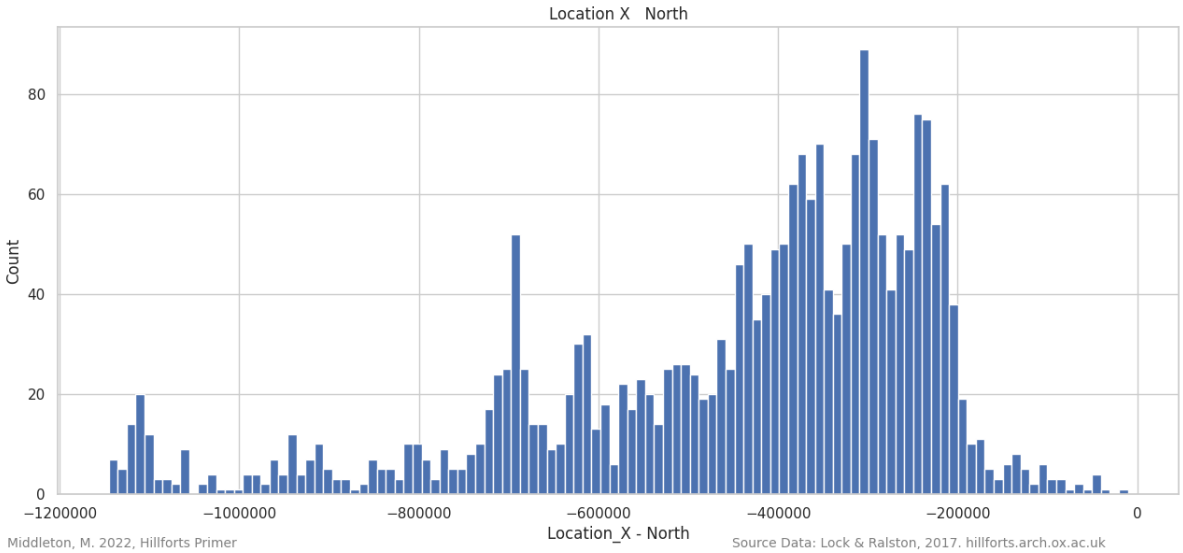
In [ ]:
```
location_X_north_data = plot_data_range(north['Location_X'], 'Location_X - North',
```



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

In [ ]:
```
plot_histogram(north['Location_X'], 'Location_X - North', 'Location_X - North', 10(
```

```
10000
```



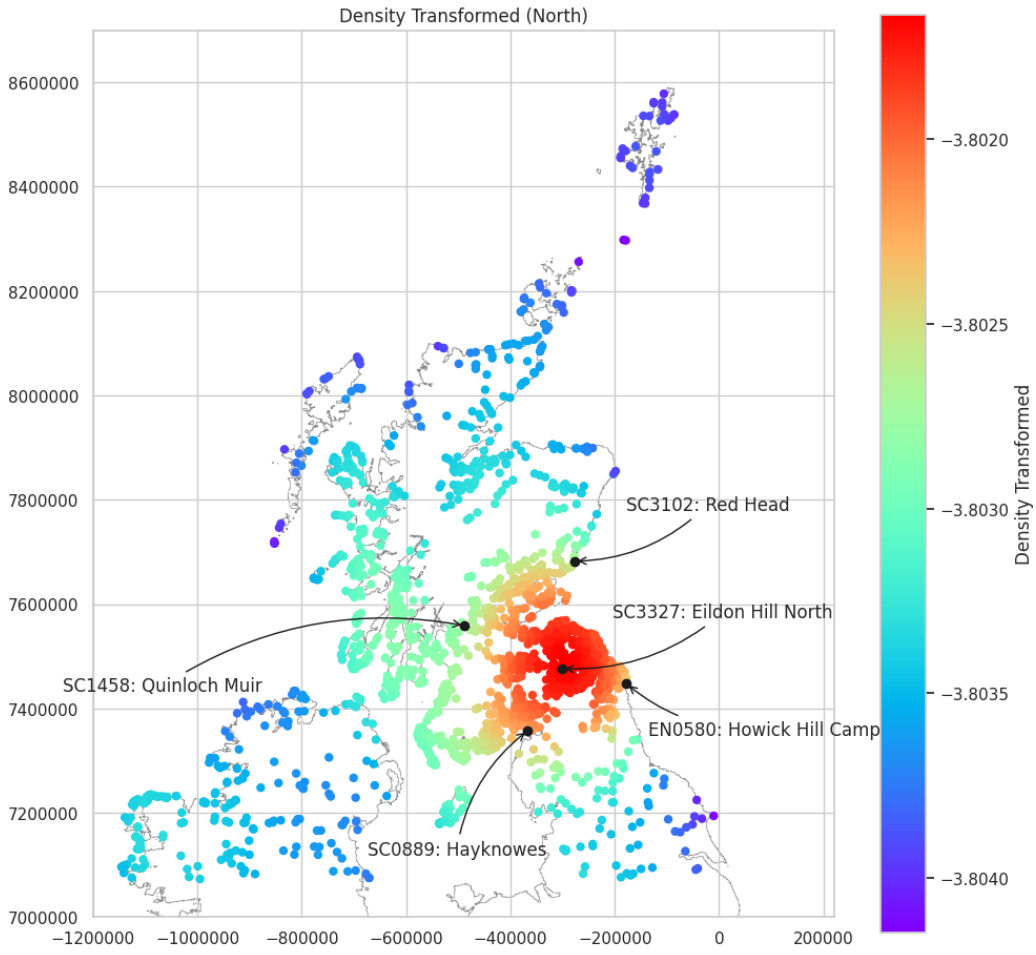
Northern Data Density Mapped (Transformed)

Data north of northing 7,070,000.

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

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

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



Middleton, M. 2022, Hillforts Primer

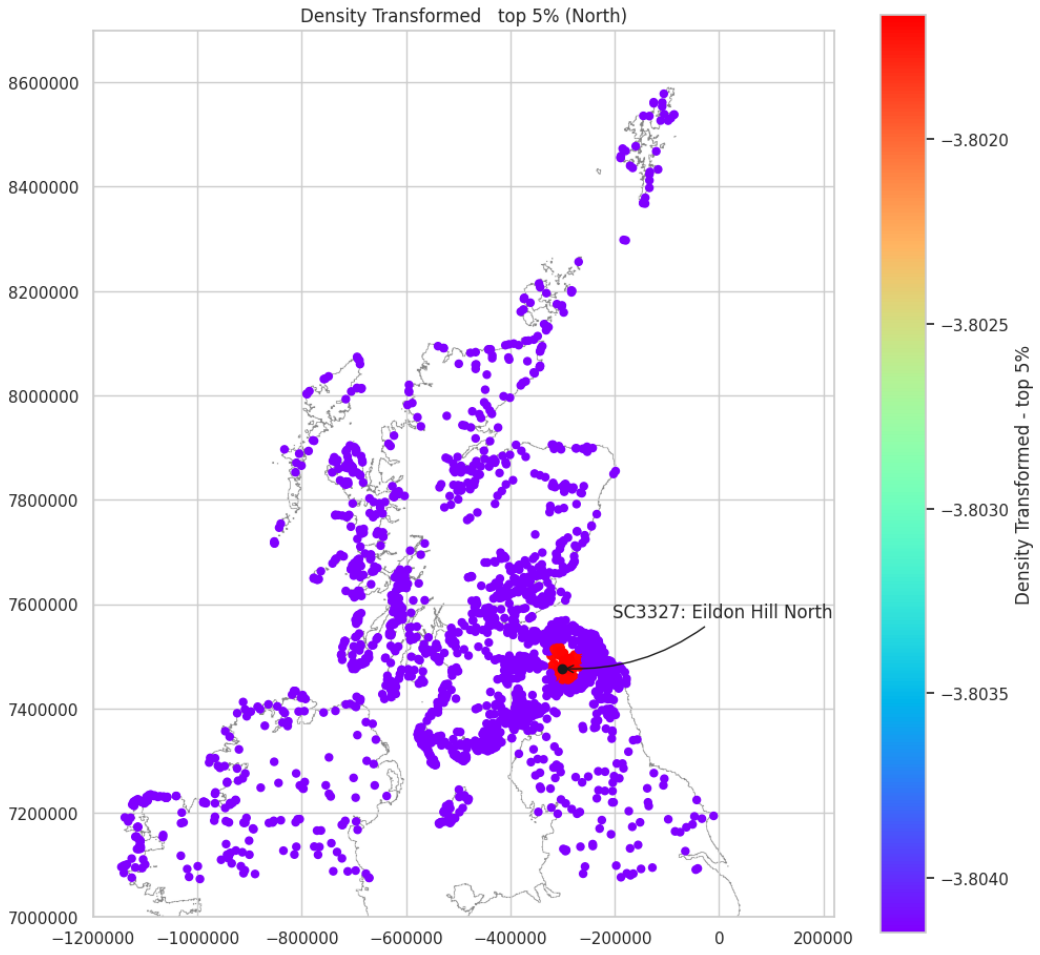
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Northern Data Density Mapped (top 5%)

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

In [ ]:
```
north_top5 = north.copy()
north_top5['Density_trans'].where(north_top5['Density_trans'] > north_top5['Densit
```

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



Northern Data Density Mapped (Capped)

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

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

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

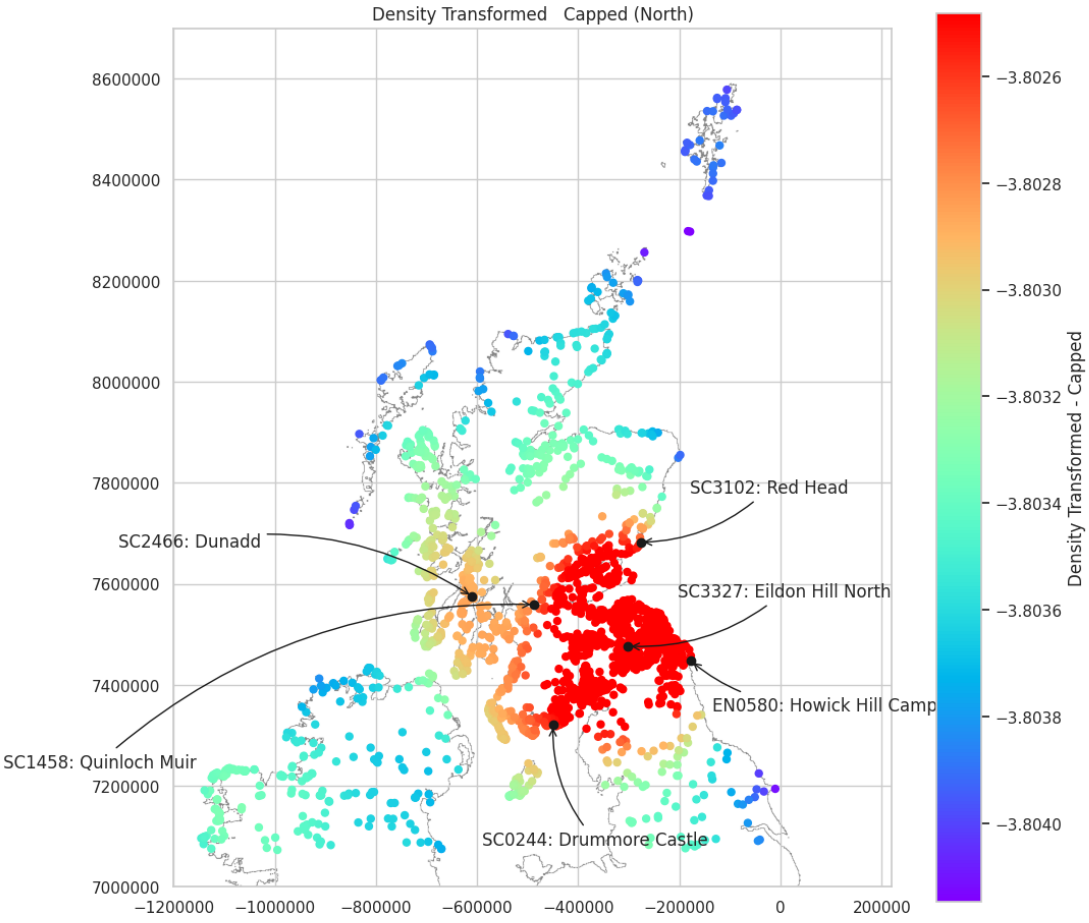
In [ ]:
```
north['Density_trans'].quantile(0.5)
```

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

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

In [ ]:
```
north_clip = north.copy()
north_clip['Density_trans'].where(north_clip['Density_trans'] < north['Density_tra
```

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



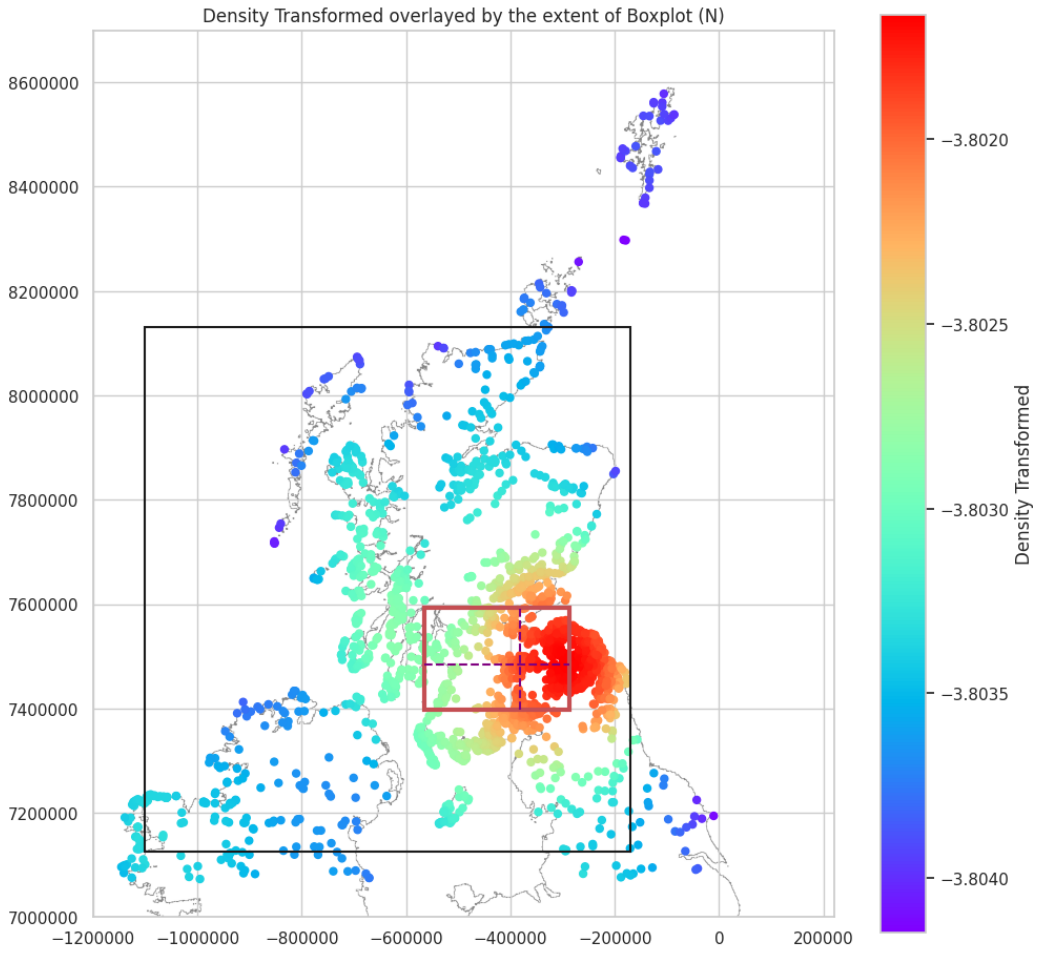
Middleton, M. 2022, Hillforts Primer

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

Density Map showing Extent of Boxplots (North)

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

In [ ]:
```
plot_density_transformed_north_boxplot(north, location_X_north_data, location_Y_no
```



Northern Data Split East-West

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

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

There are 716 records in the Northwestern data.

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

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

There are 1598 records in the Northeastern data.

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

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

Cluster Data Packages

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

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

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

In [ ]:
```
cluster_data = hillforts_data[['Location_X','Location_Y', 'Main_Country_Code']].co
cluster_data['Cluster'] = 'NA'
```

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

In [ ]:
```
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != 'NI', 'I', inpl
cluster_data['Cluster'].where(cluster_data['Main_Country_Code'] != 'IR', 'I', inpl
```

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

In [ ]:
```
cluster_data['Cluster'] = np.where(
   (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] >= 7060000) , 'N
   )
cluster_north_ireland = cluster_data[cluster_data['Cluster'] == 'North Irealnd'].c
```

In [ ]:
```
cluster_data['Cluster'] = np.where(
   (cluster_data['Cluster'] == 'I') & (cluster_data['Location_Y'] < 7060000) , 'So
   )
cluster_south_ireland = cluster_data[cluster_data['Cluster'] == 'South Irealnd'].c
```

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

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

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

In [ ]:
```
cluster_data['Cluster'] = np.where(
   (cluster_data['Cluster'] == 'NA') & (cluster_data['Location_Y'] >= 7070000) & (
   )
cluster_north_west = cluster_data[cluster_data['Cluster'] == 'Northwest'].copy()
```

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

In [ ]:
```
cluster_location_packages = [cluster_north_ireland, cluster_south_ireland, cluster_
```

Northwest Density

The density values are refreshed in the clipped Northwestern data.

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

There are 716 forts in the Northwest data package.

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

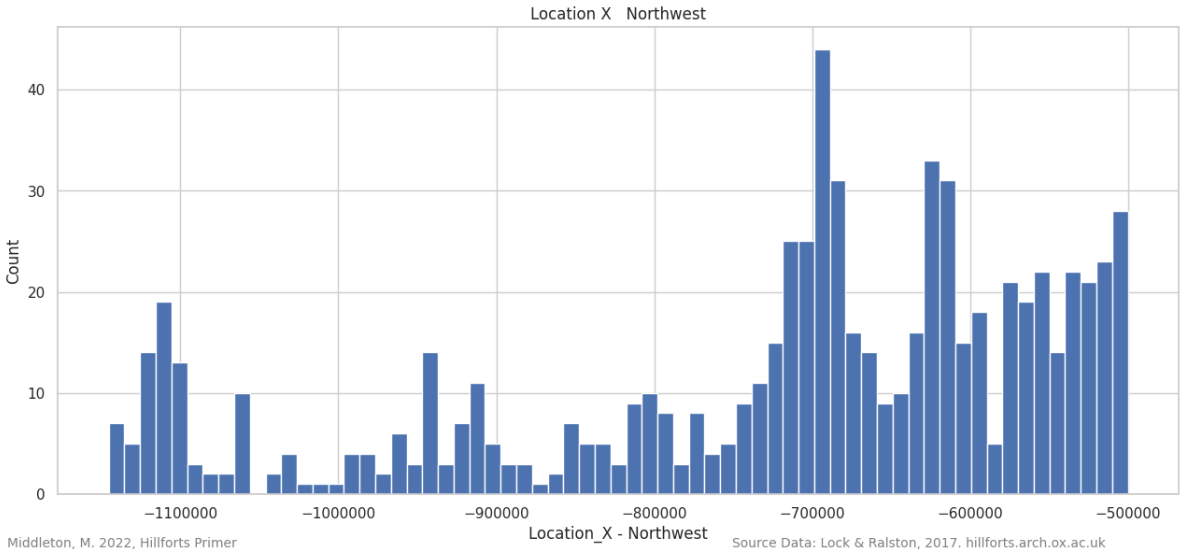
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 716 entries, 0 to 715
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Location_X     716 non-null    int64  
 1   Location_Y     716 non-null    int64  
 2   Density    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

Northwest Data Plotted

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

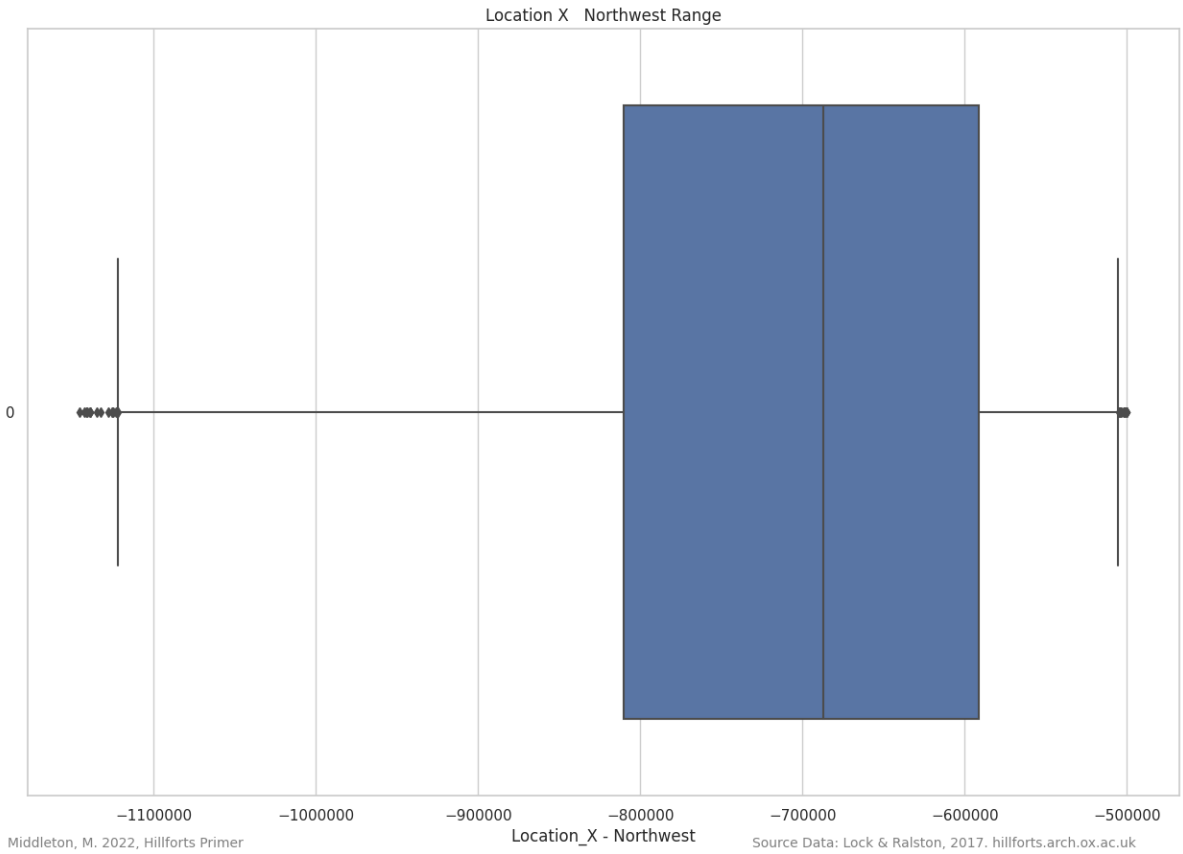
In [ ]:
```
plot_histogram(north_west['Location_X'], 'Location_X - Northwest', 'Location_X - No
```

```
10000
```



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

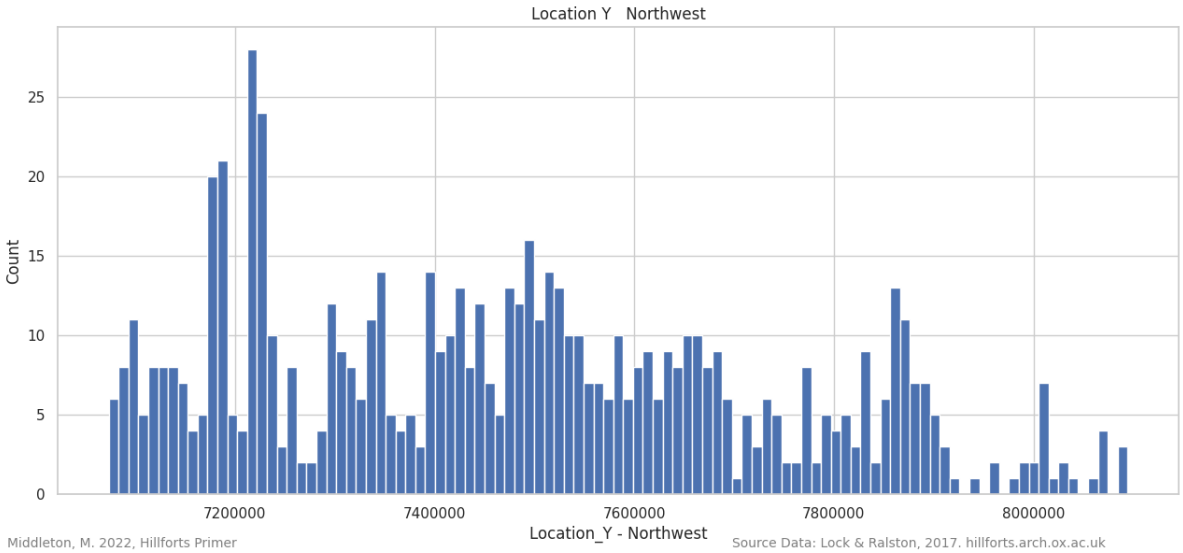
In [ ]:
```
location_X_north_w_data = plot_data_range(north_west['Location_X'], 'Location_X - I
```



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

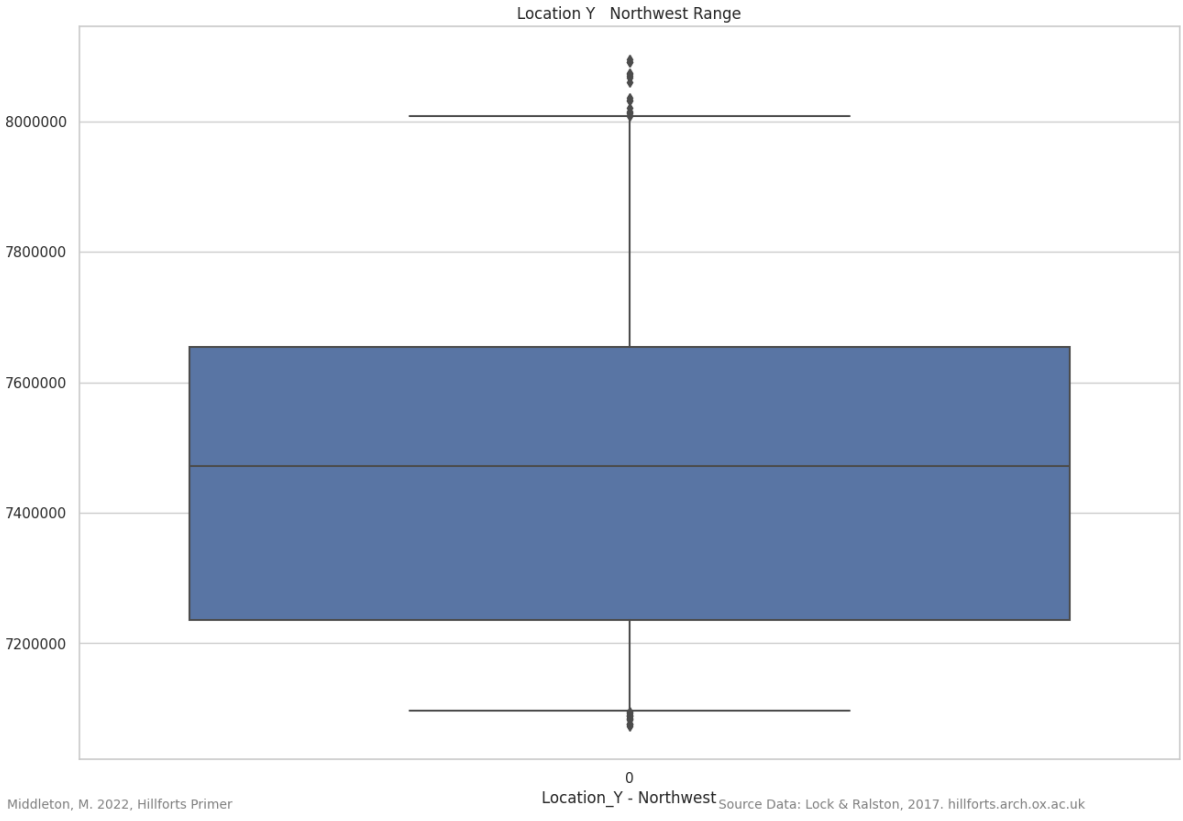
In [ ]:
```
plot_histogram(north_west['Location_Y'], 'Location_Y - Northwest', 'Location_Y - N
```

```
10000
```



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

In [ ]:
```
location_Y_north_w_data = plot_data_range(north_west['Location_Y'], 'Location_Y - N
```



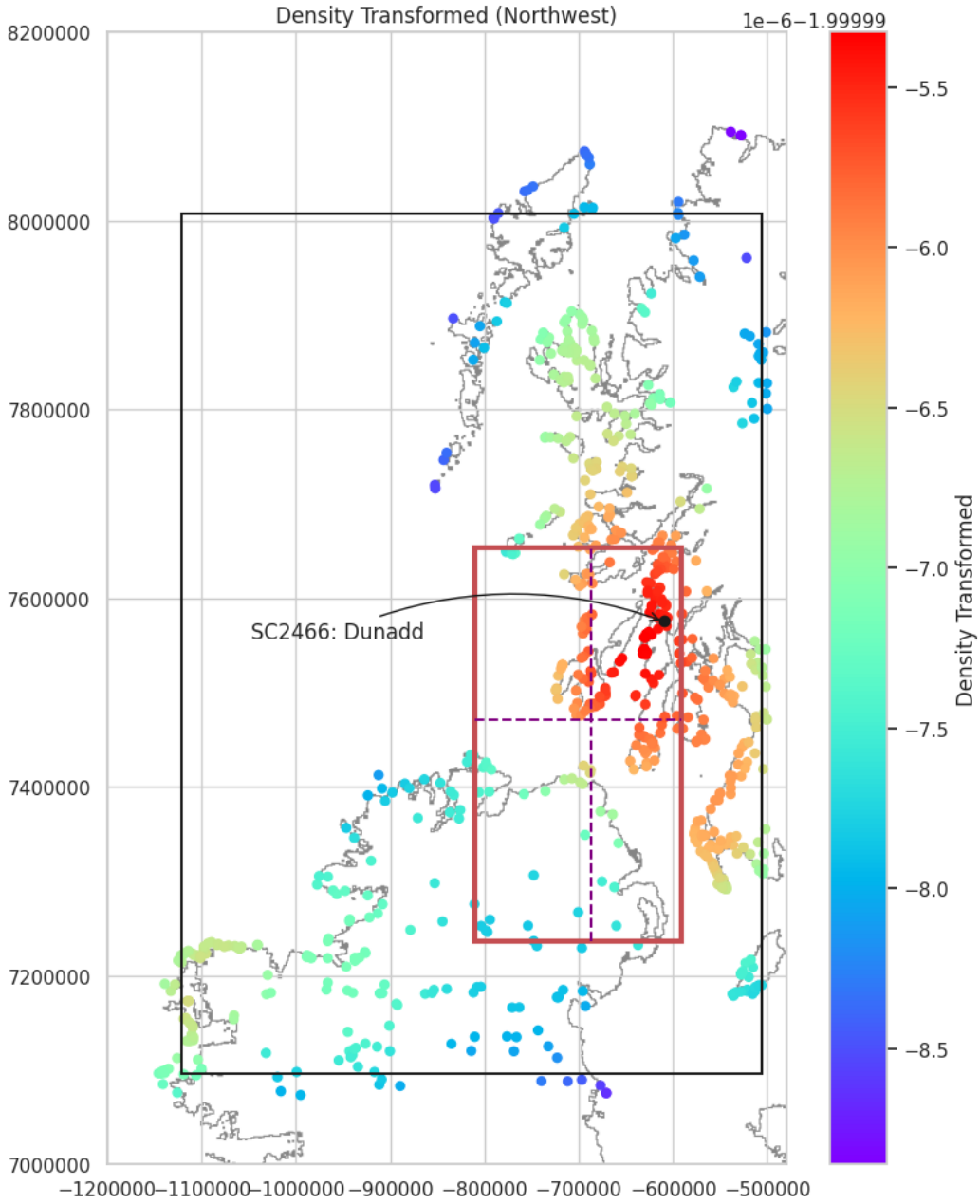
Northwest Data Mapped

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

The elongated IQR along the 'Location_X' and ‘Location_Y’ axes suggests the Irish data is pulling the median value into the Irish Sea and this may indicate that the Northwest cluster

should be assessed independently of the Irish data. See: Northwestern Data Plotted Minus Ireland.

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



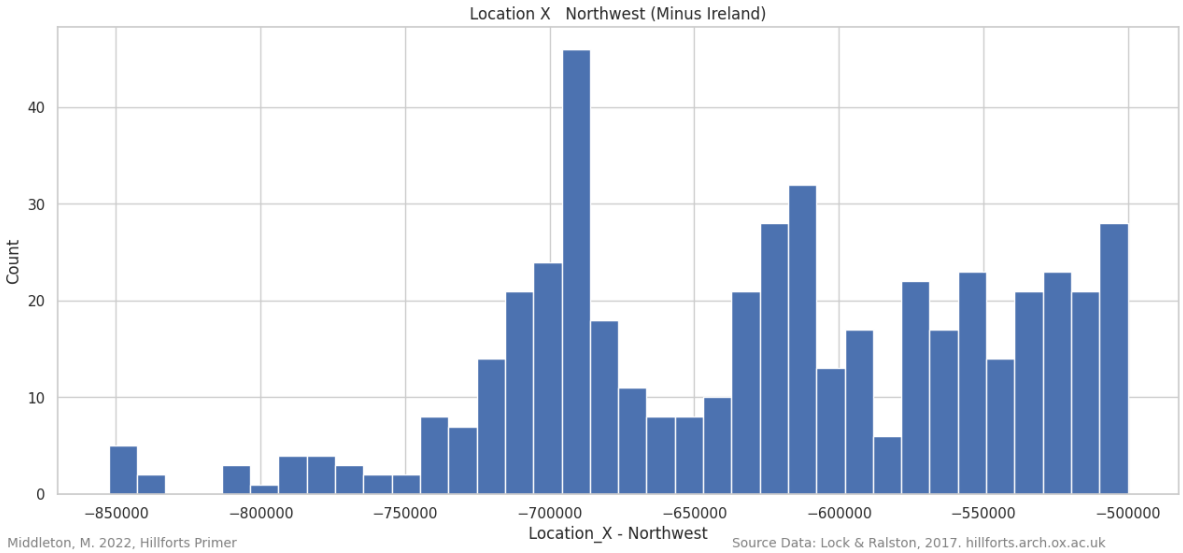
See: Density Map showing Extent of Northwest Boxplot
See: Northwestern Data Minus Ireland Mapped

Northwest Data Minus Ireland Plotted

The histogram now contains only the Northwestern data in Scotland. See: Northwest Data Plotted.

In [ ]:
```
plot_histogram(cluster_north_west['Location_X'], 'Location_X - Northwest', 'Locati
```

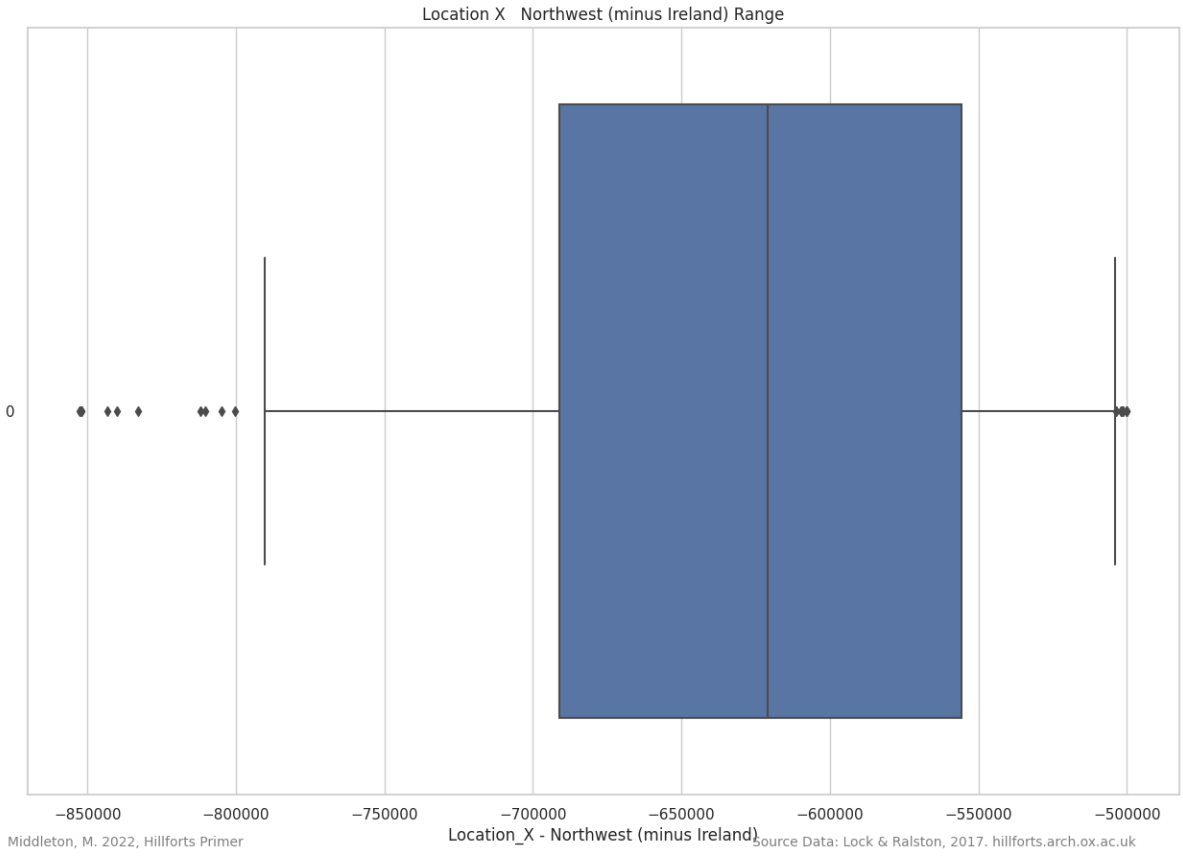
```
10000
```



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

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

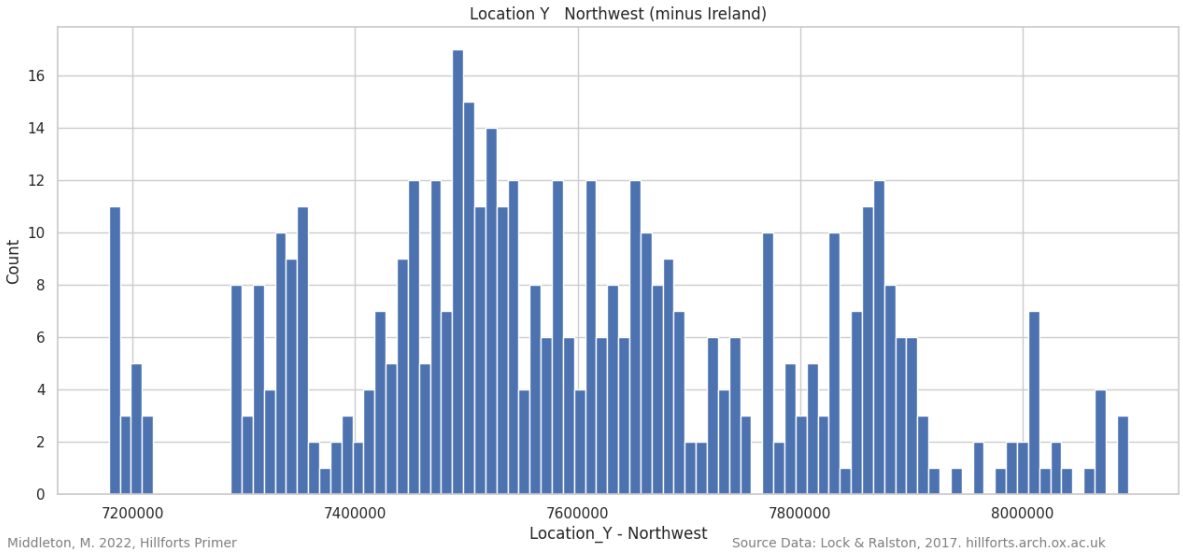
In [ ]:
```
location_X_cluster_north_west = plot_data_range(cluster_north_west['Location_X'],
```



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

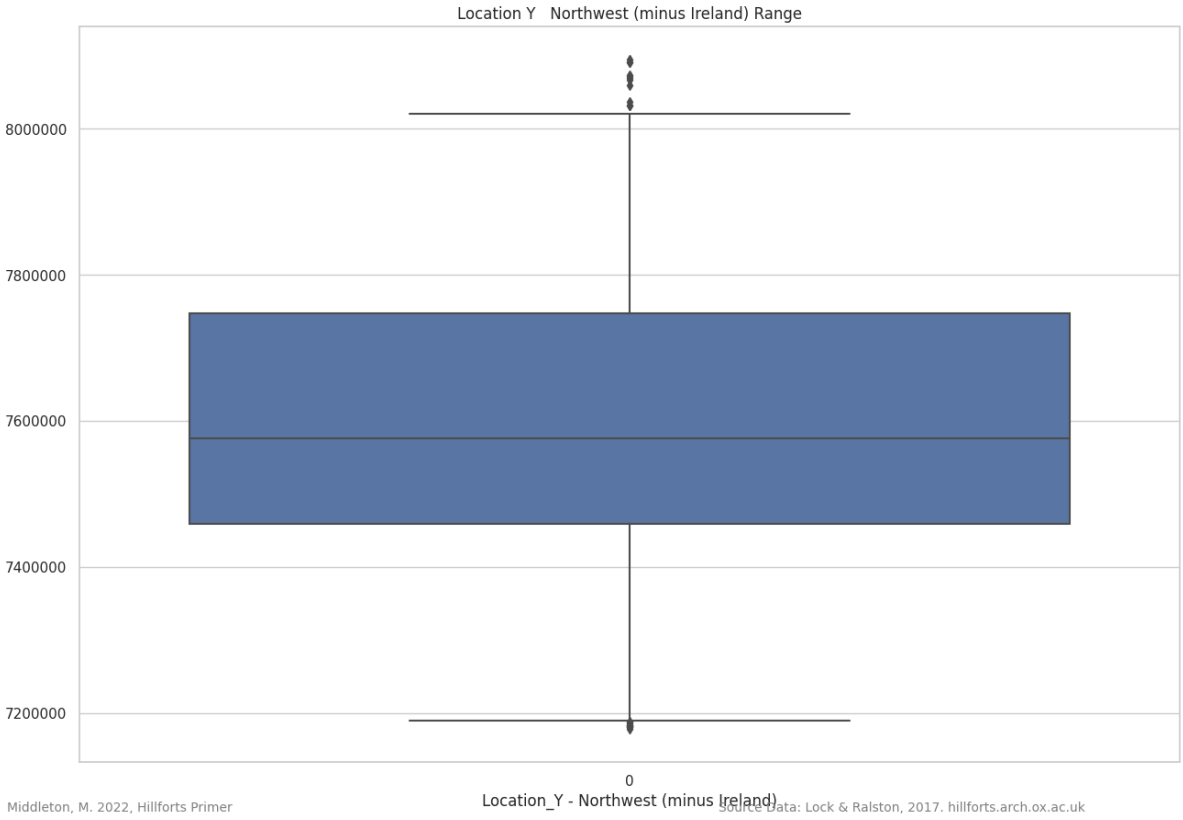
In [ ]:
```
plot_histogram(cluster_north_west['Location_Y'], 'Location_Y - Northwest', 'Locati
```

```
10000
```



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

In [ ]:
```
location_Y_cluster_north_west = plot_data_range(cluster_north_west['Location_Y'],
```

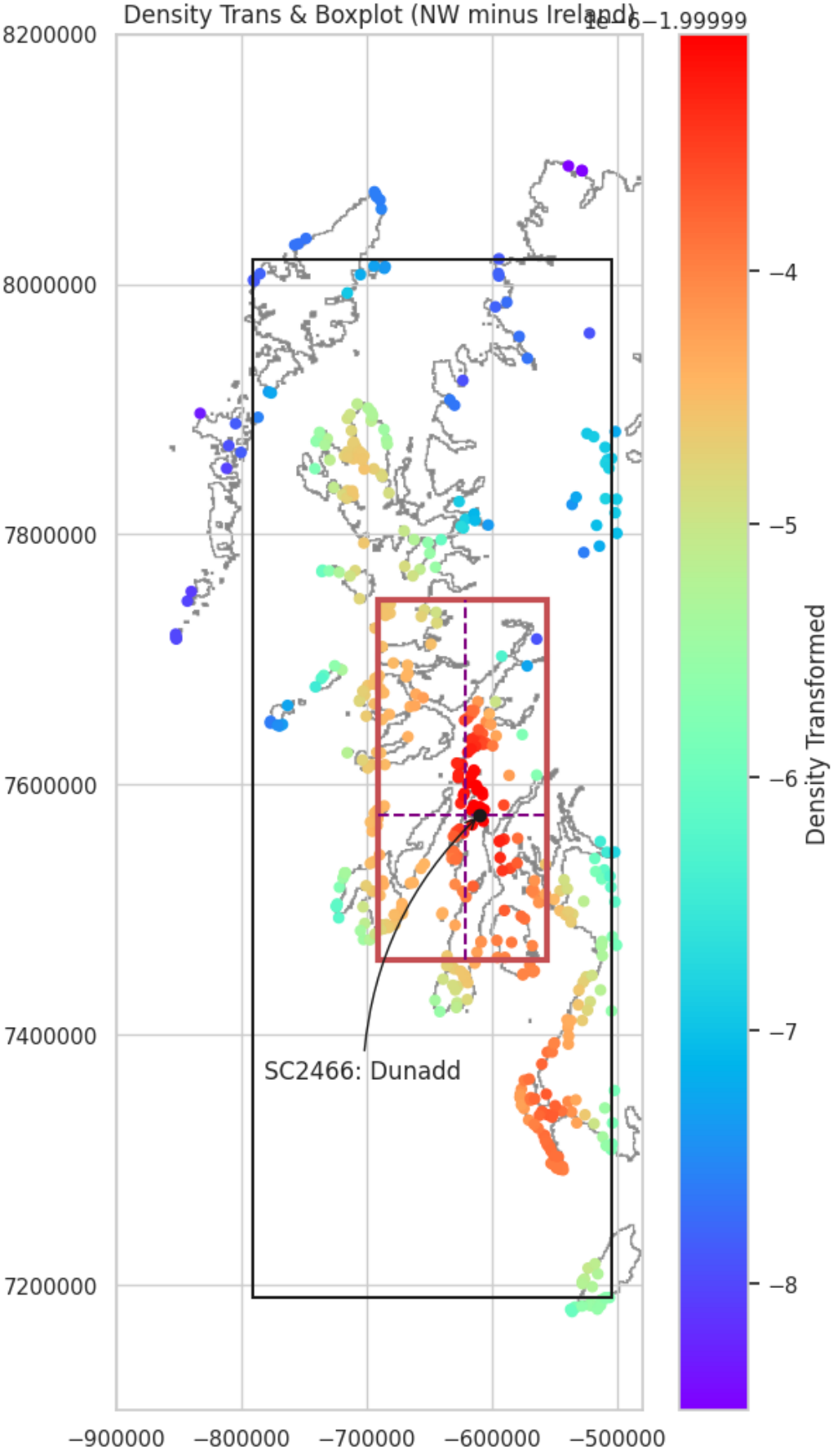


Northwestern Data Minus Ireland Mapped

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

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

In [ ]:
```
plot_north_west_dnesity_minus(show_eildon, cluster_north_west, location_X_cluster_
```



Middleton, M. 2022, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Northeast Density

The density values are refreshed in the clipped Northeastern data.

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

There are 1598 forts in the Northeast data package.

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

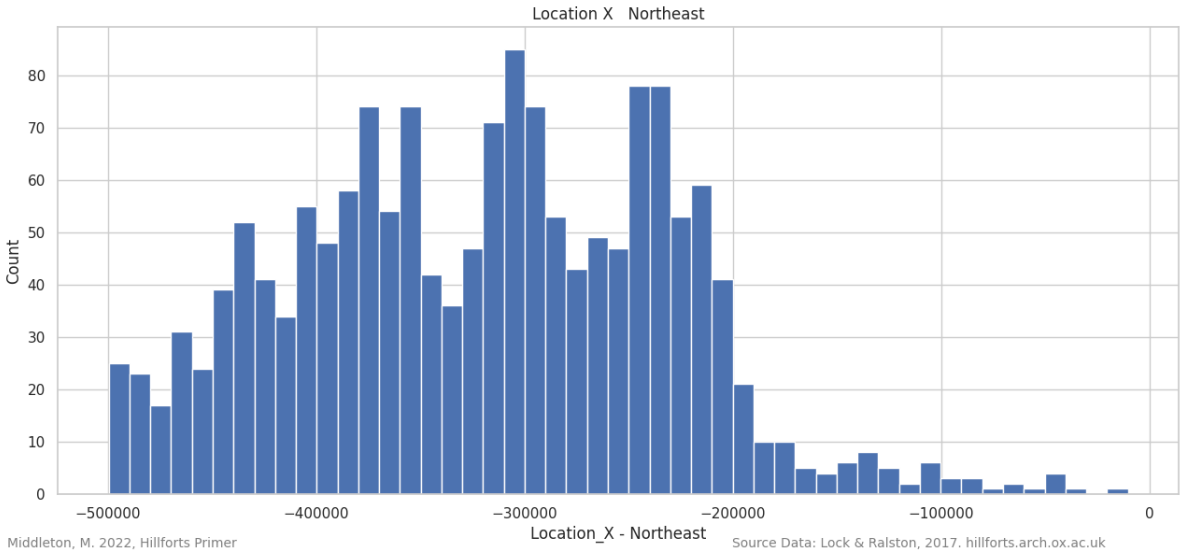
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1598 entries, 0 to 1597
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Location_X     1598 non-null   int64  
 1   Location_Y     1598 non-null   int64  
 2   Density  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

Northeast Data Plotted

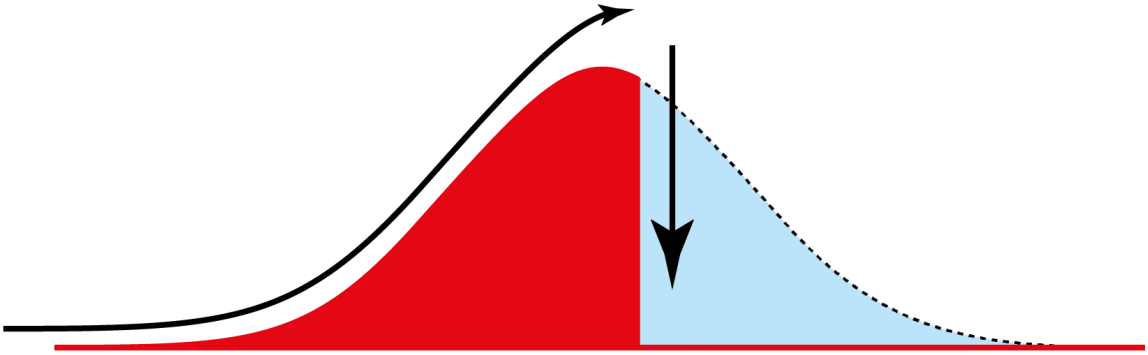
There is an increasing concentration of hillforts from west to east up to the North Sea coast. At this point the distribution quite literally, 'falls off a cliff'. There is very little land to the east of -200,000.

In [ ]:
```
plot_histogram(north_east['Location_X'], 'Location_X - Northeast', 'Location_X - N
```

```
10000
```



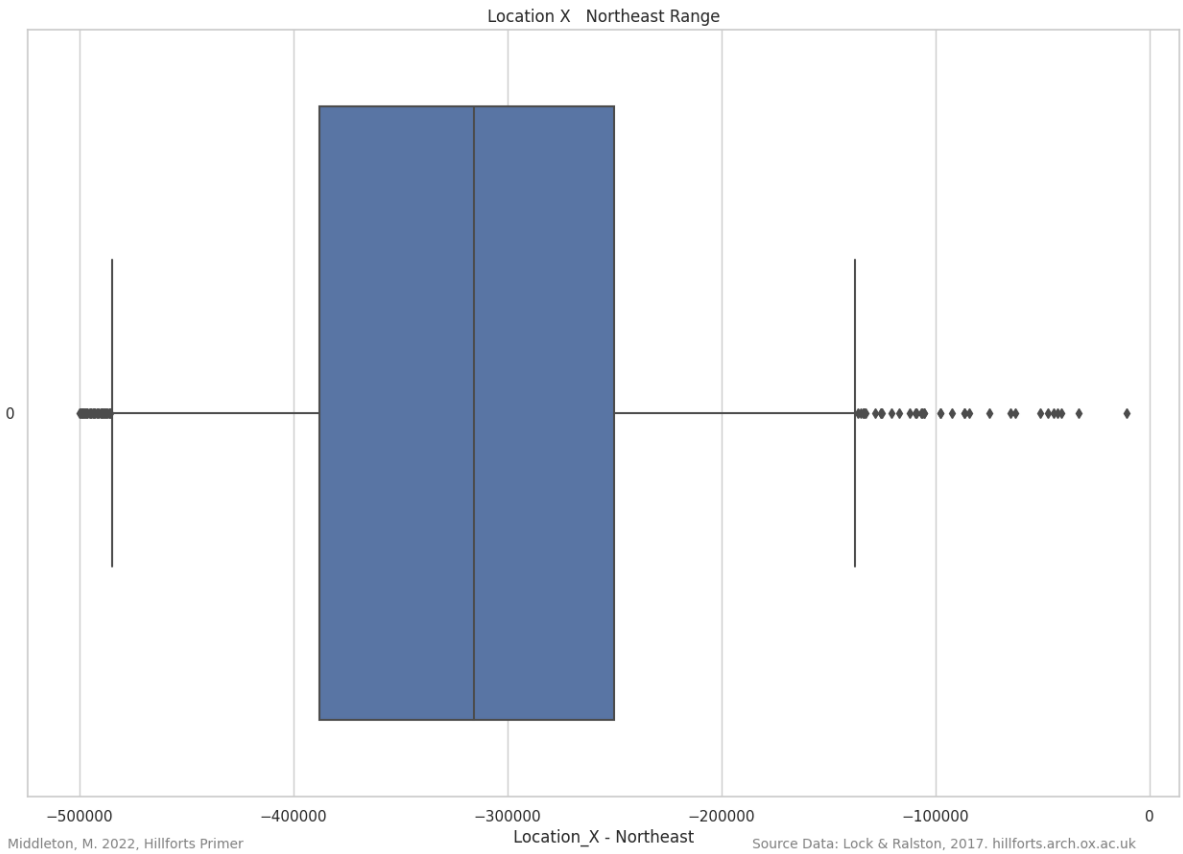
The density calculations assume the data will be spread across the entire area mapped. Where there is a peak in the data the density algorithm will expect the density to fall away on either side. The coast interrupts the expected spread, and the data is truncated immediately after the peak. This results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



A data cliff. This figure shows how a peak in the data would be expected to plot in a normal distribution (blue) compared to an abstraction of how the 'Location_X' data actually plots in the Northeast, next to the coast (red) (see histogram above). As a boxplot is best used when the data has a normal ditribution, in this case, the cliff causes the median value and the interquartile range to be offset to the left (west).
Mike Middleton *CC BY-SA 3.0**

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

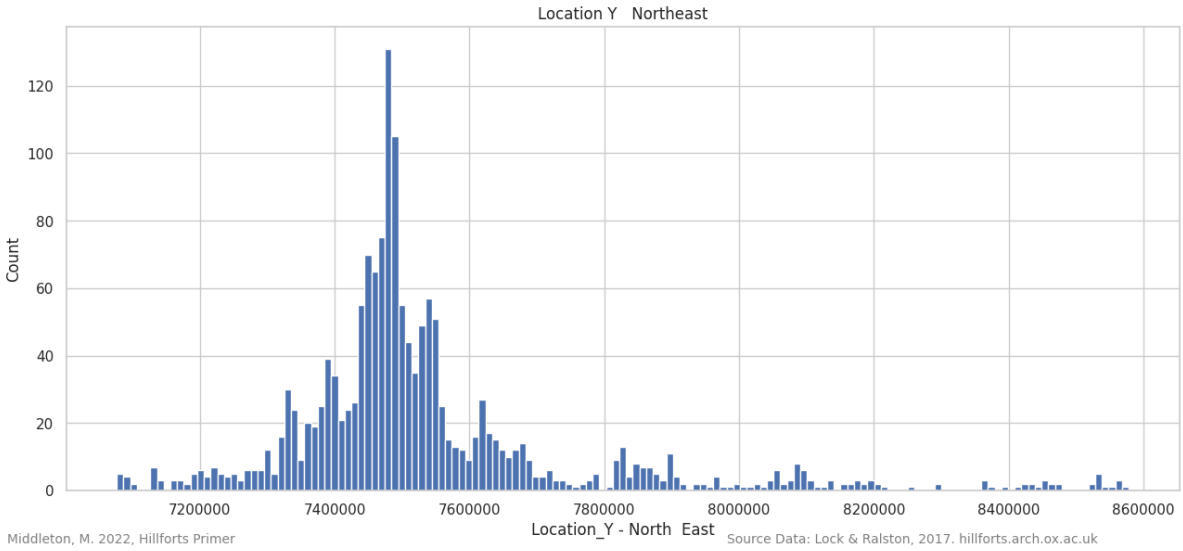
In [ ]:
```
location_X_north_e_data = plot_data_range(north_east['Location_X'], 'Location_X - N
```



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

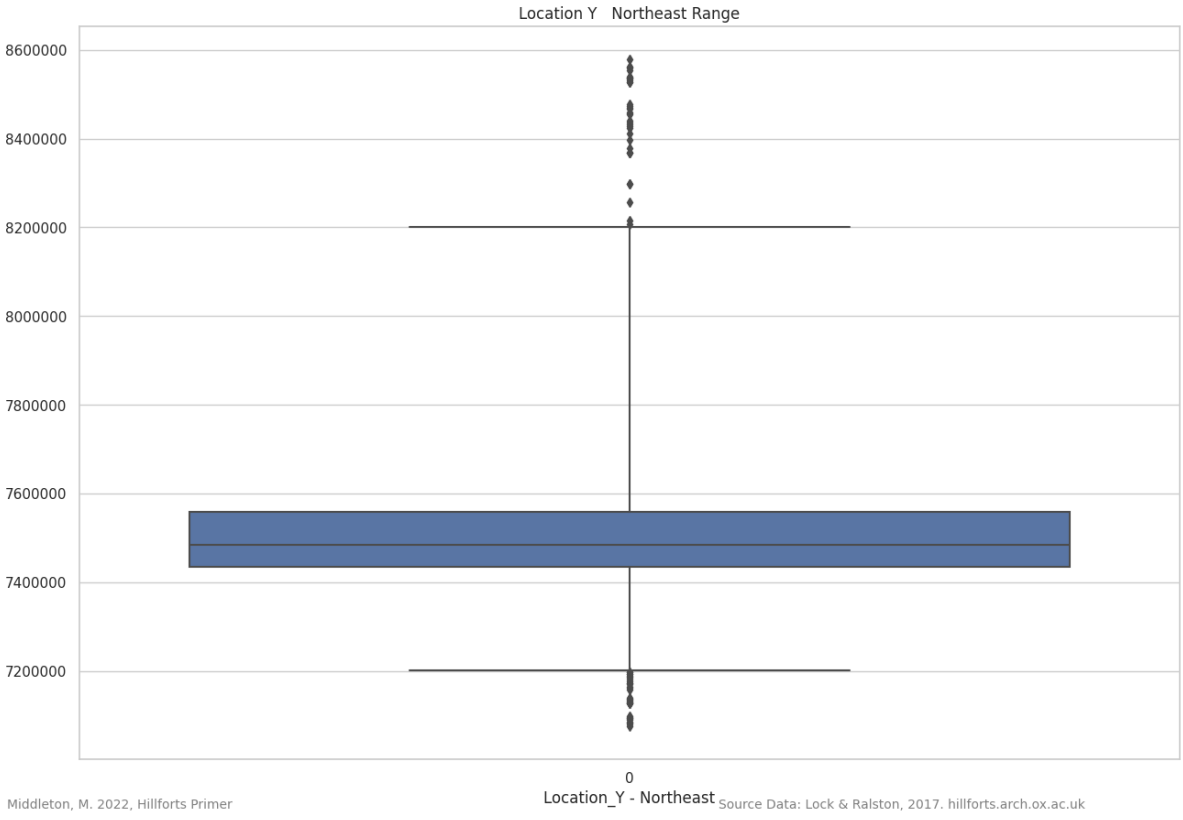
In [ ]:
```
plot_histogram(north_east['Location_Y'], 'Location_Y - North  East', 'Location_Y -
```

```
10000
```



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

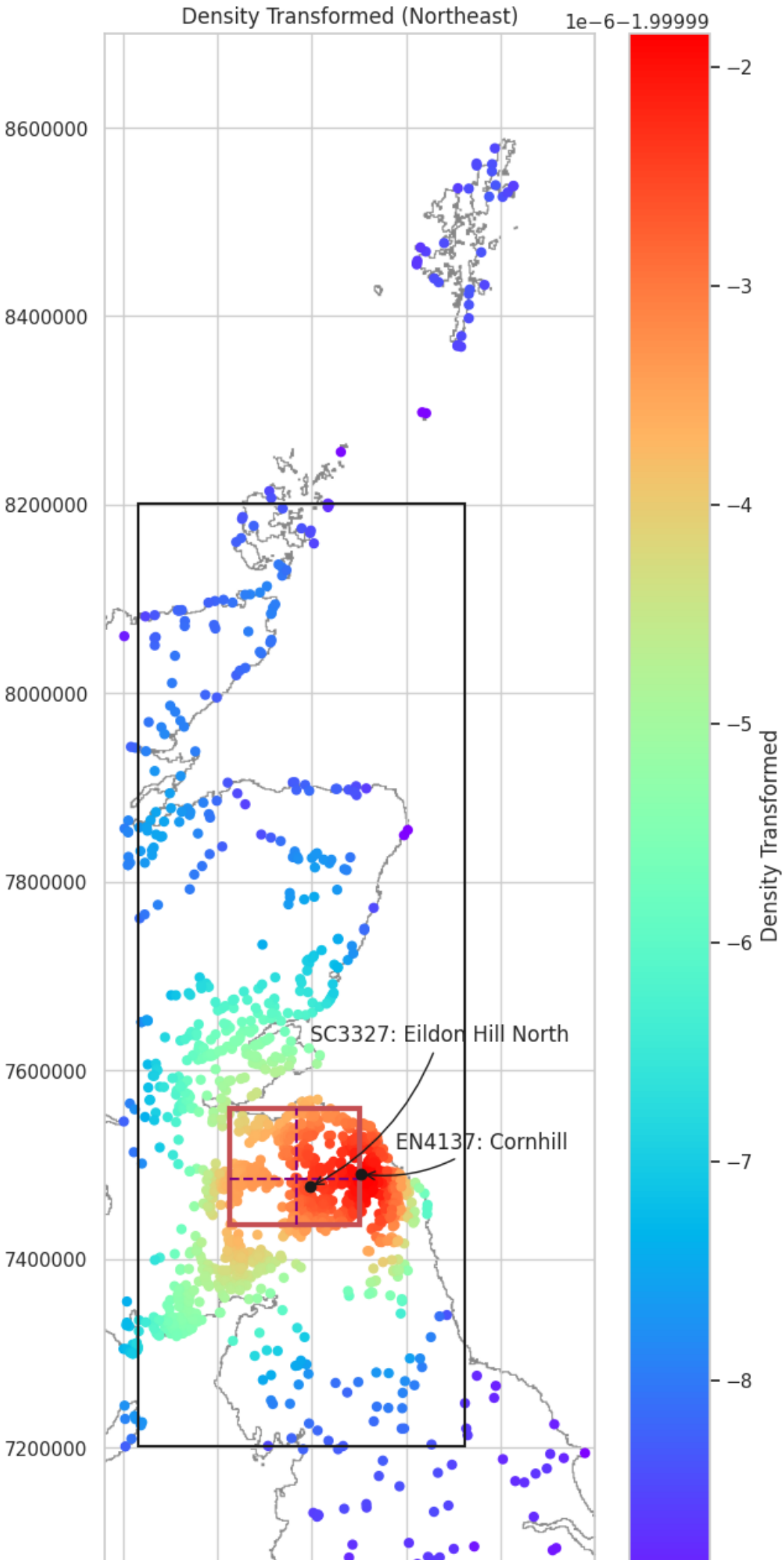
In [ ]:
```
location_Y_north_e_data = plot_data_range(north_east['Location_Y'], 'Location_Y - N
```

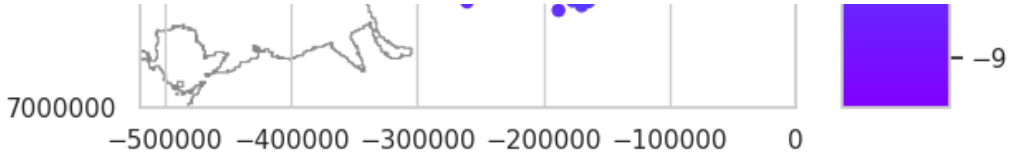


Northeast Data Mapped

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

In [ ]:
```
plot_north_east_density(show_eildon,north_east, location_Y_north_e_data)
```


Density Transformed (Northeast)
1e−6−1.99999
8600000
8400000
8200000
8000000
7800000
7600000
7400000
7200000
SC3327: Eildon Hill North
EN4137: Cornhill
−2
−3
−4
−5
−6
−7
−8
Density Transformed

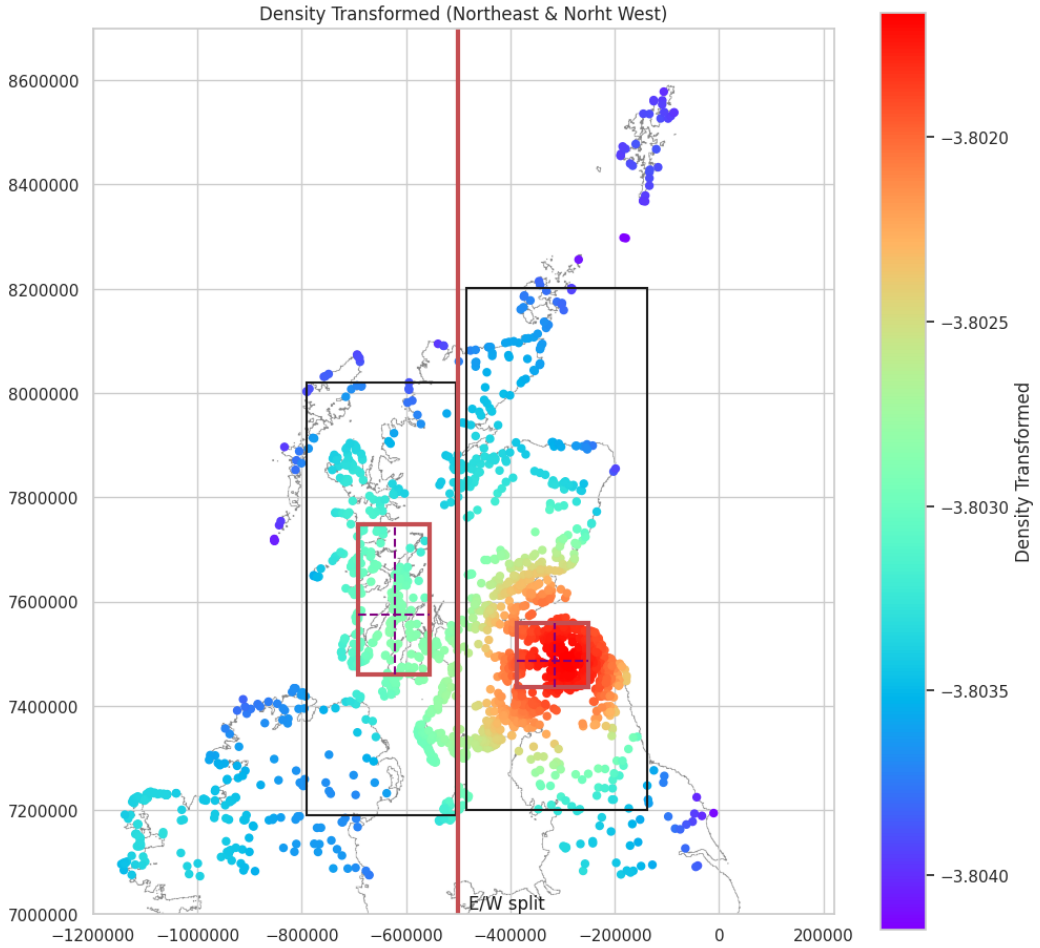


Middleton, M. 2022, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Density Map showing Extent of the Northern Boxplots

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

In [ ]:
```
plot_norhtern_density_boxplots(show_eildon, north, location_X_cluster_north_west,
```



Middleton, M. 2022, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: Northwestern Data Minus Ireland Mapped

See: Northeast Data Mapped

Southern Density

Southern Location Data Plotted

There are 1833 records in the southern data package.

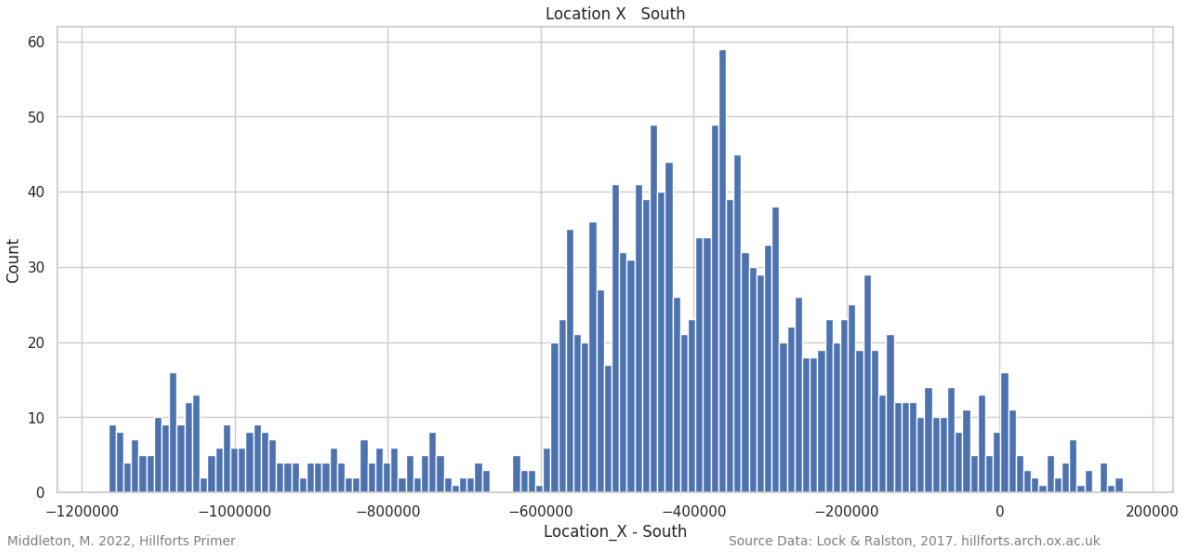
In [ ]:
```
south.info()
```

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

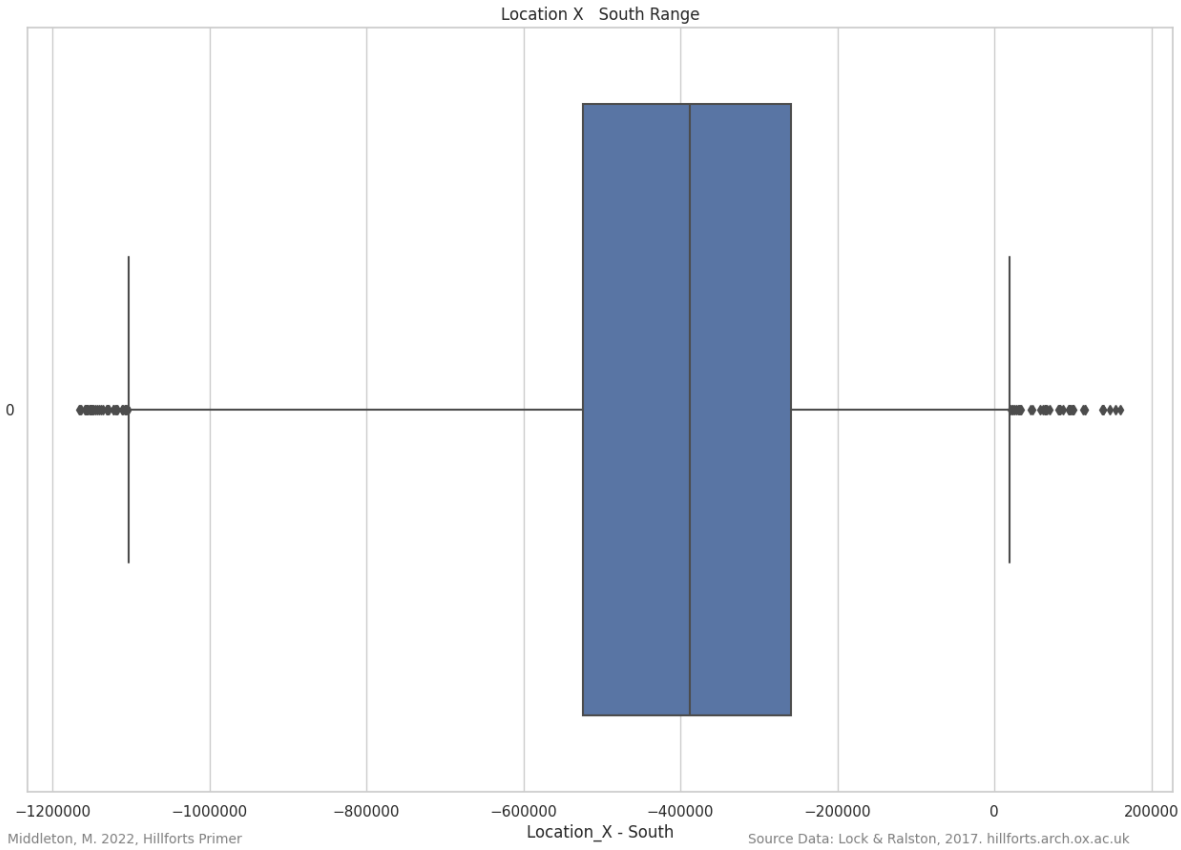
There are two distributions of data in the southern package. To the west, a low density of hillforts over Ireland and to the east, a more intense concentration of hillforts over Wales and into England. In the Welsh and English data there is a broad peak that increases in density from east to west. The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

In [ ]:
```
plot_histogram(south['Location_X'], 'Location_X - South', 'Location_X - South', 10(
```

```
10000
```



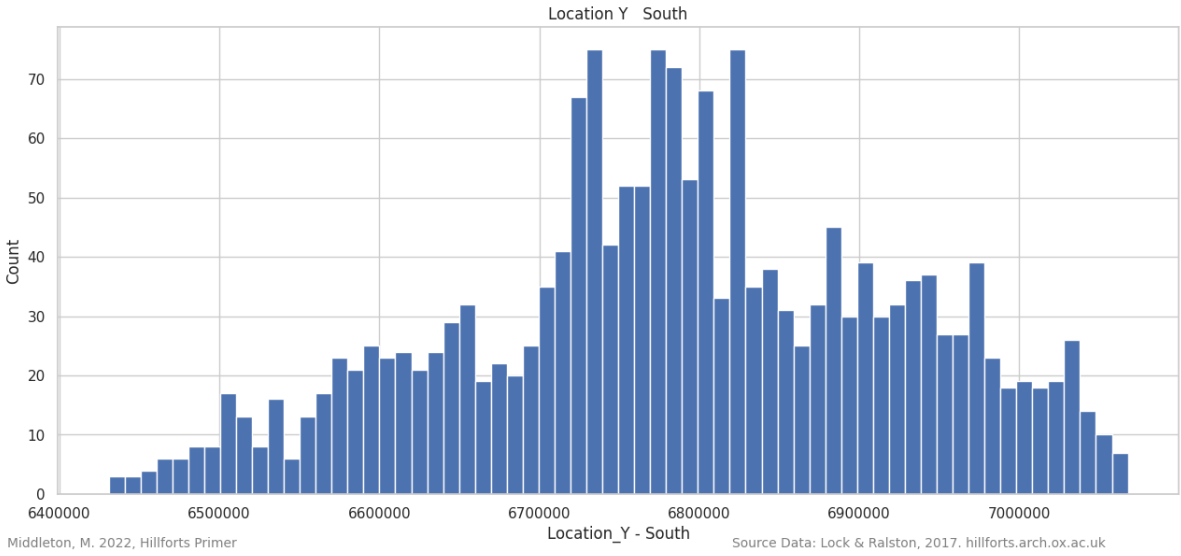
In [ ]:
```
location_X_south_data = plot_data_range(south['Location_X'], 'Location_X - South',
```



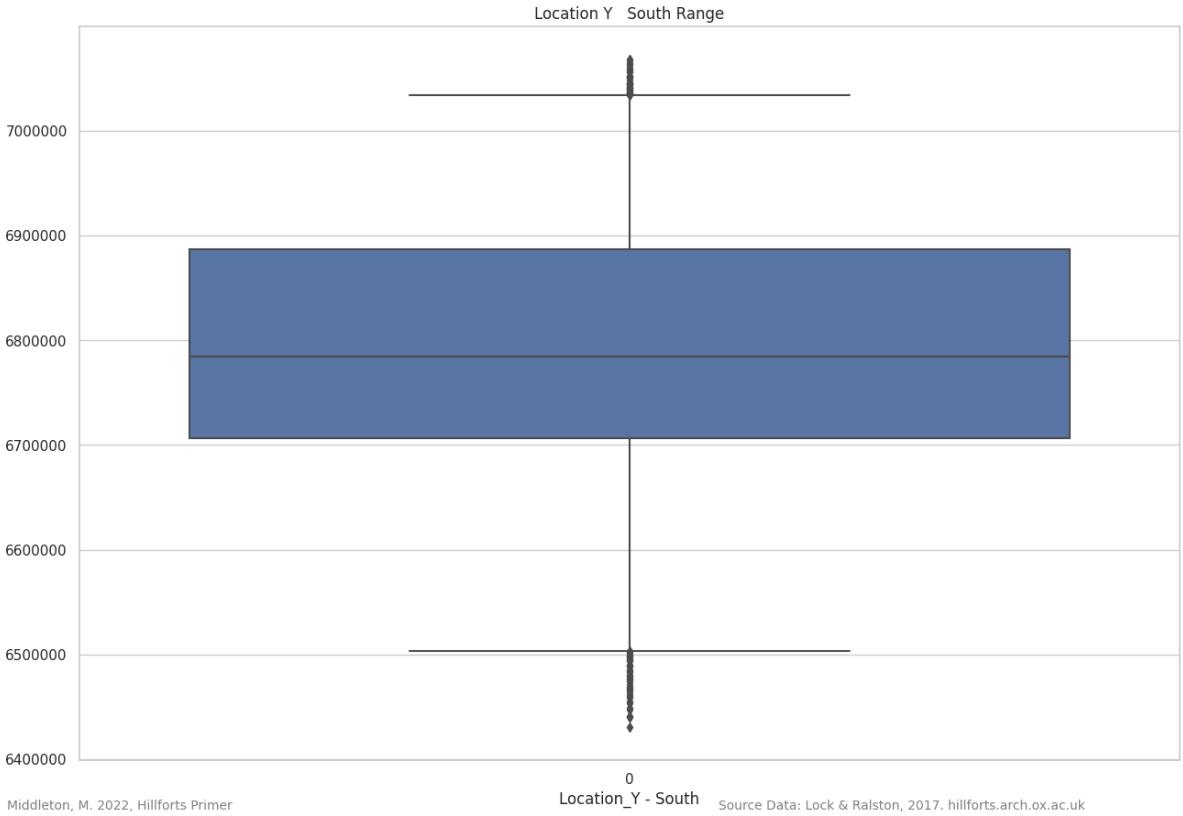
There is a broad peak in the 'Location_Y' data. Toward the centre of the broad peak, a narrow band of high density peaks can be seen between 6,720,000 to 6,820,000.

In [ ]:
```
plot_histogram(south['Location_Y'], 'Location_Y - South', 'Location_Y - South', 10(
```

```
10000
```



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

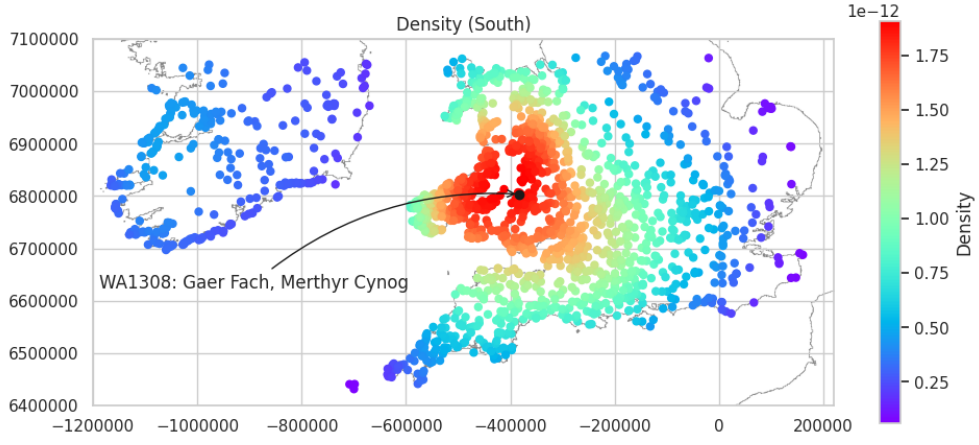


Southern Data Density Mapped

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

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

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



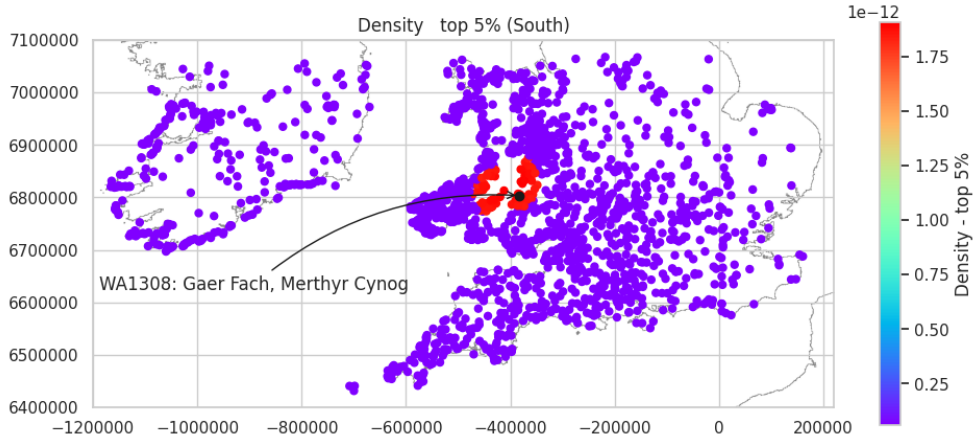
Southern Data Density Mapped (top 5%)

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

In [ ]:
```
south_top5 = south.copy()
south_top5['Density'].where(south_top5['Density'] > south_top5['Density'].quantile
south_top5['Density'].describe()
```

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

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



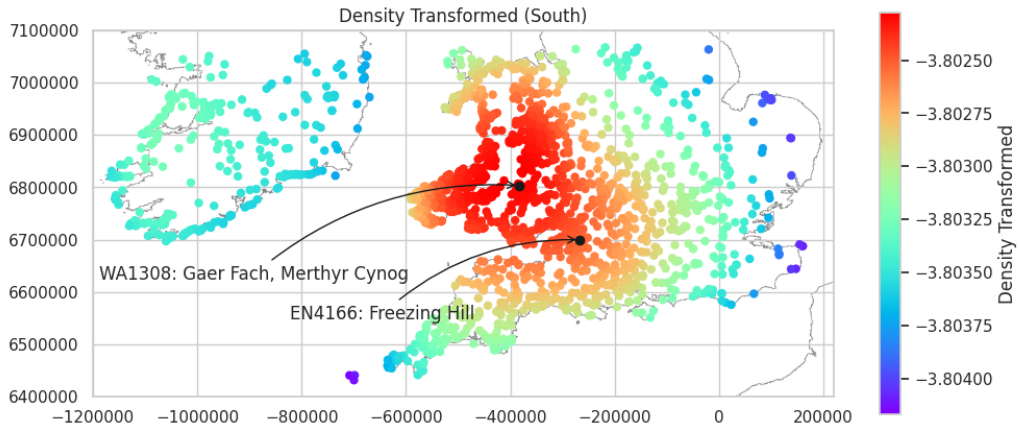
Middleton, M. 2022, Hillforts Primer

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

Southern Data Density Mapped (Transformed)

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

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



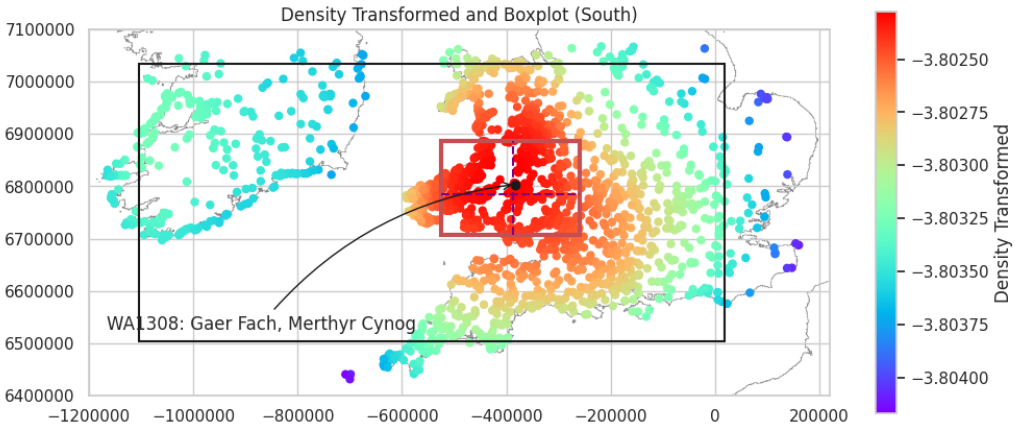
Middleton, M. 2022, Hillforts Primer

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

Southern Density Boxplot

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

In [ ]:
```
plot_southern_density_boxplot(show_gaer_fach, south,location_X_south_data, locatio
```



See: Southern Location Data Plotted

See: Southern Density Boxplot Minus Ireland

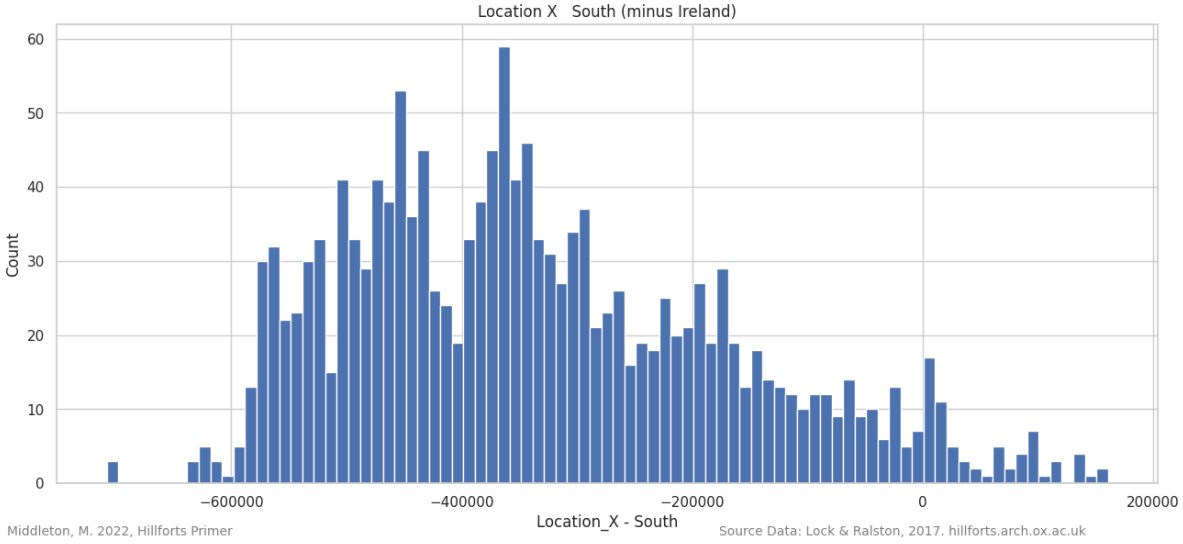
Southern Location Data Minus Ireland Plotted

The histogram now contains only the southern data, extending over Wales and England.

See: Southern Location Data Plotted

In [ ]:
```
plot_histogram(cluster_south['Location_X'], 'Location_X - South', 'Location_X - So
```

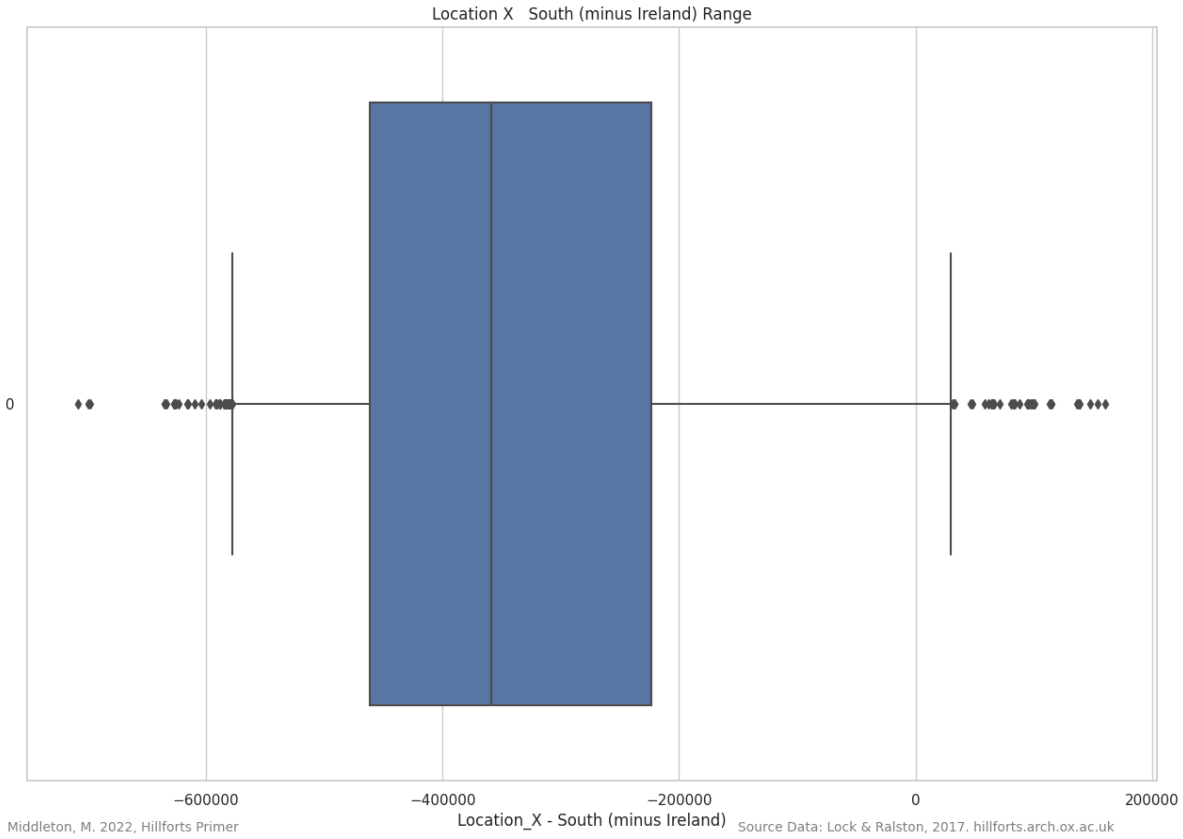
```
10000
```



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

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

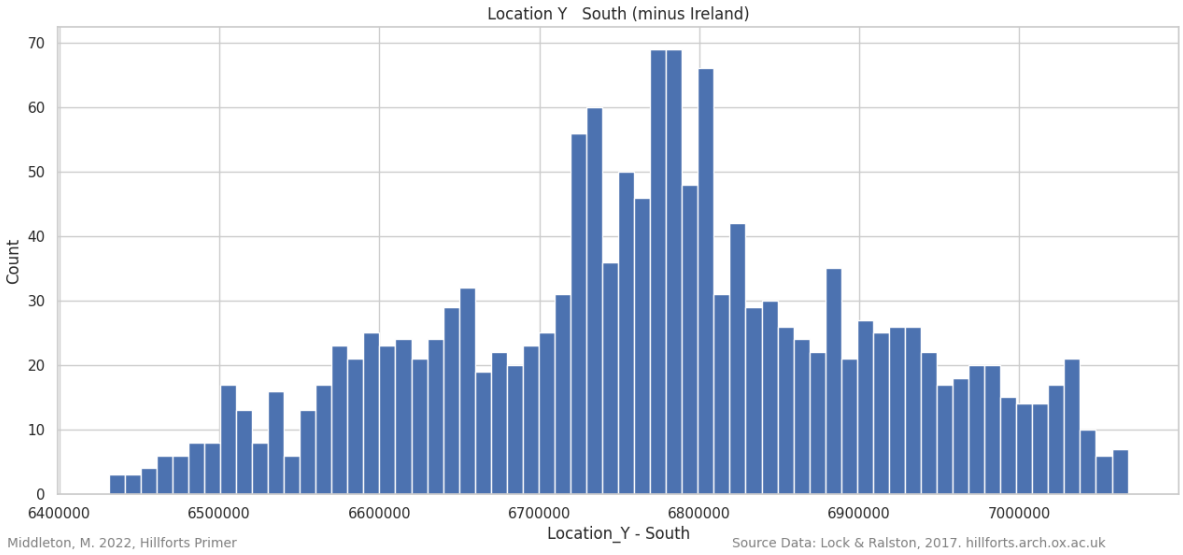
In [ ]:
```
location_X_cluster_south = plot_data_range(cluster_south['Location_X'], 'Location_
```



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

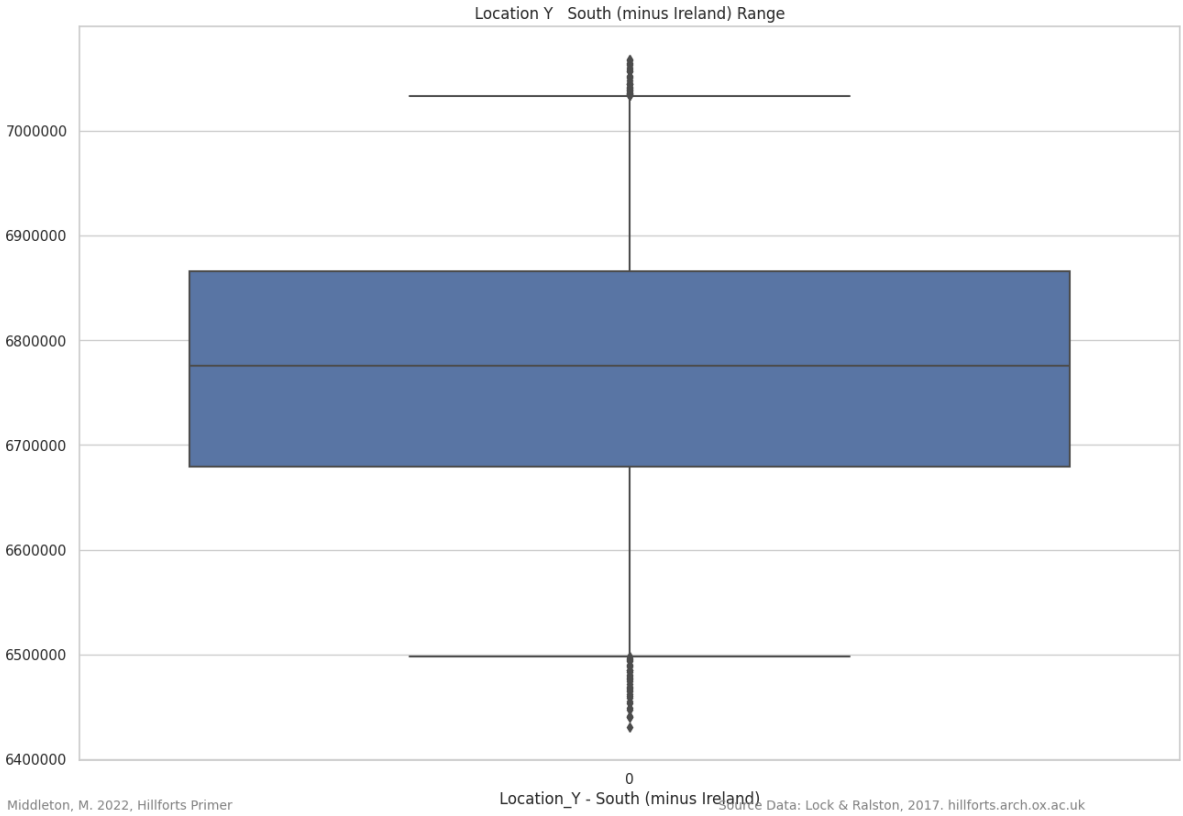
In [ ]:
```
plot_histogram(cluster_south['Location_Y'], 'Location_Y - South', 'Location_Y - So
```

```
10000
```



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

In [ ]:
```
location_Y_cluster_south = plot_data_range(cluster_south['Location_Y'], 'Location_
```



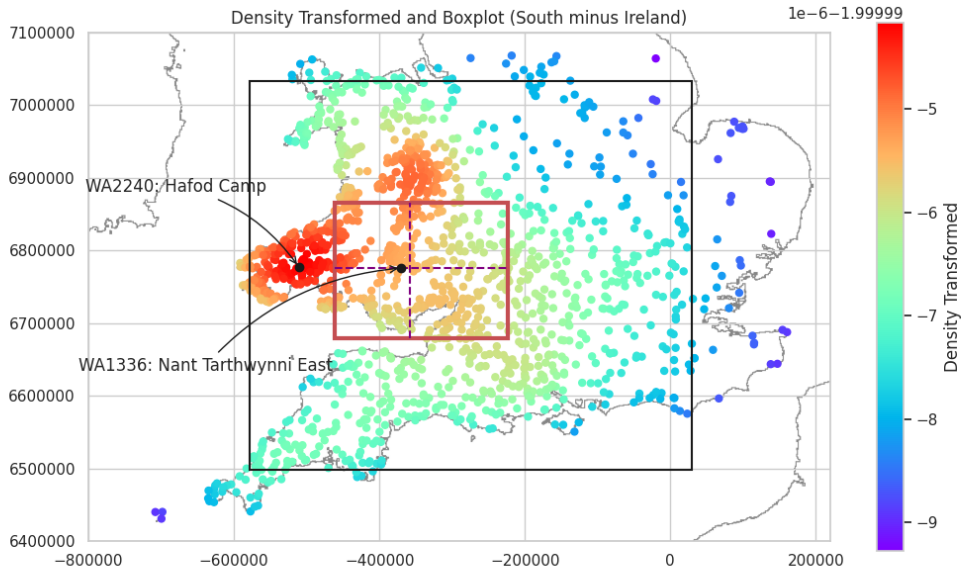
Southern Density Boxplot Minus Ireland

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

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

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

In [ ]:
```
plot_south_density_transformed_boxplot(show_gaer_fach, cluster_south, location_X_c
```



Middleton, M. 2022, Hillforts Primer

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

Ireland Density

Ireland Location Data Plotted

There are 507 hillforts in the Irish data.

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

ireland_location_data = roi_and_ni_data[location_features_short].copy().reset_inde
```

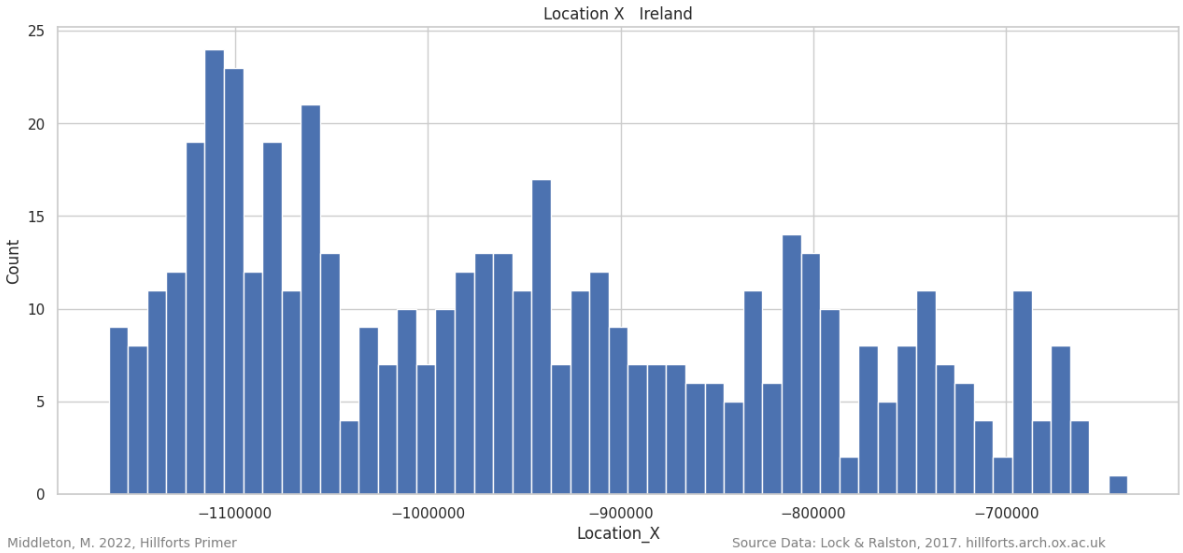
In [ ]:
```
ireland_location_data.info()
```

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

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

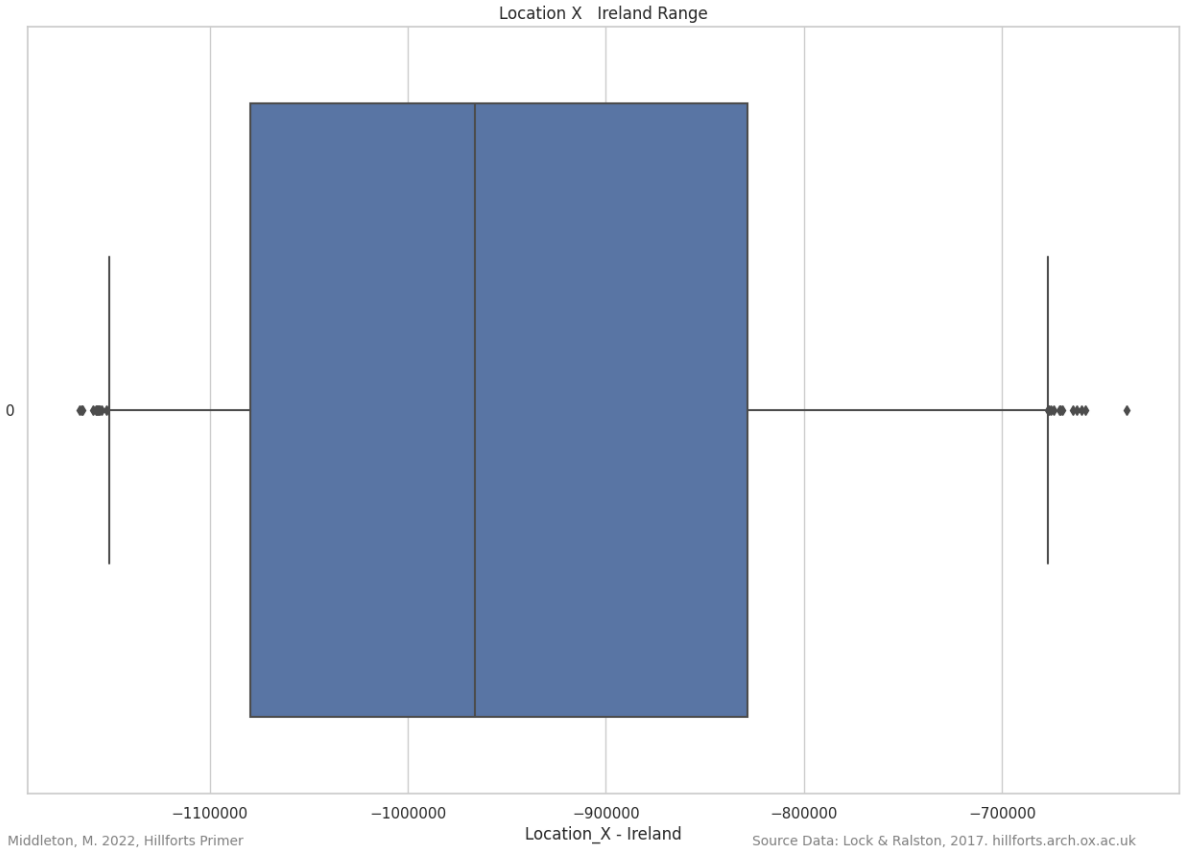
In [ ]:
```
plot_histogram(ireland_location_data['Location_X'], 'Location_X', 'Location_X - Ir
```

```
10000
```



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

In [ ]:
```
location_X_ireland_data = plot_data_range(ireland_location_data['Location_X'], 'Lo
```

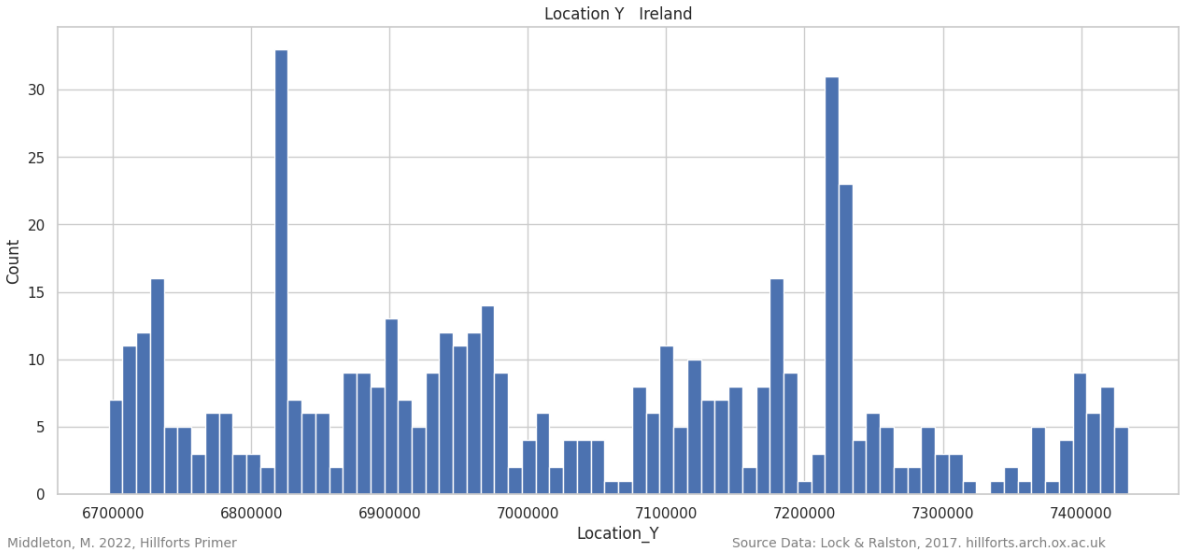


The 'Location_Y' histogram shows a number of shallow peaks and two, taller, narrow peaks. The peak at 6,820,000 corresponds to an unconnected grouping of forts along the south east coast of Ireland from Rosslare to Waterford and another group of forts along the Dingle Peninsula. This peak is a result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

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

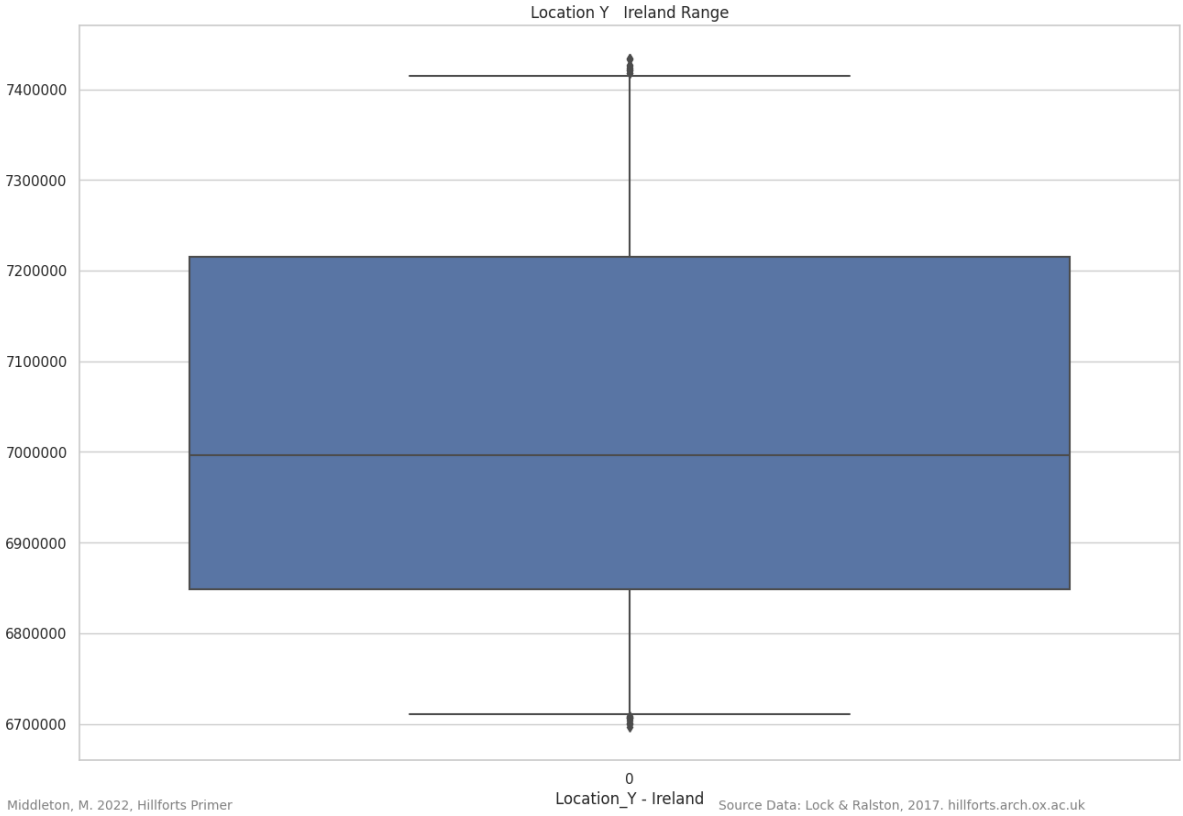
In [ ]:
```
plot_histogram(ireland_location_data['Location_Y'], 'Location_Y', 'Location_Y - Ir
```

```
10000
```



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

In [ ]:
```
location_Y_ireland_data = plot_data_range(ireland_location_data['Location_Y'], 'Lo
```



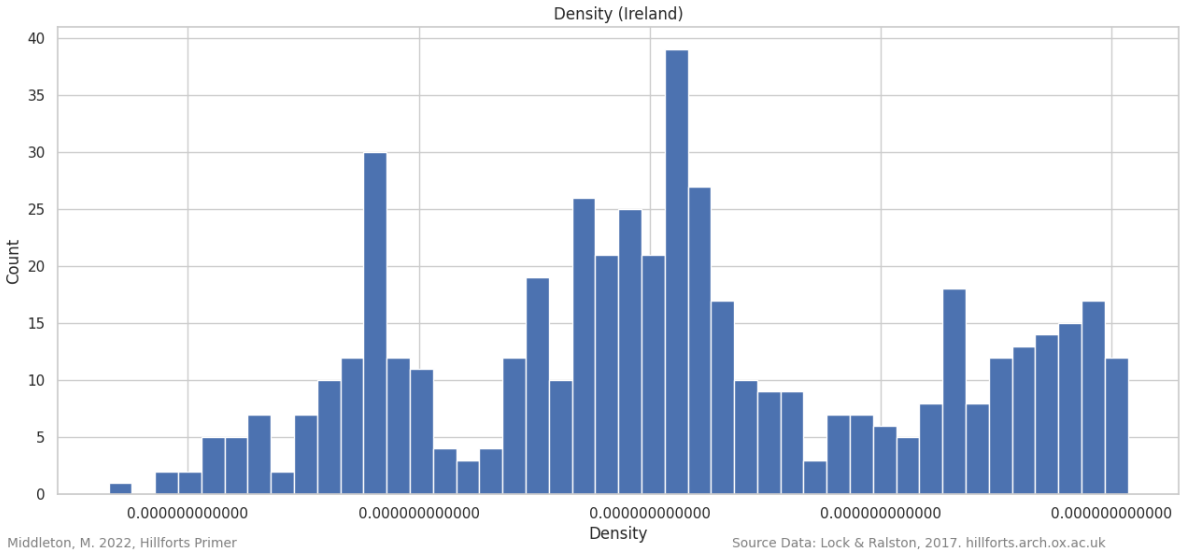
Ireland Density Data Plotted

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

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

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

None

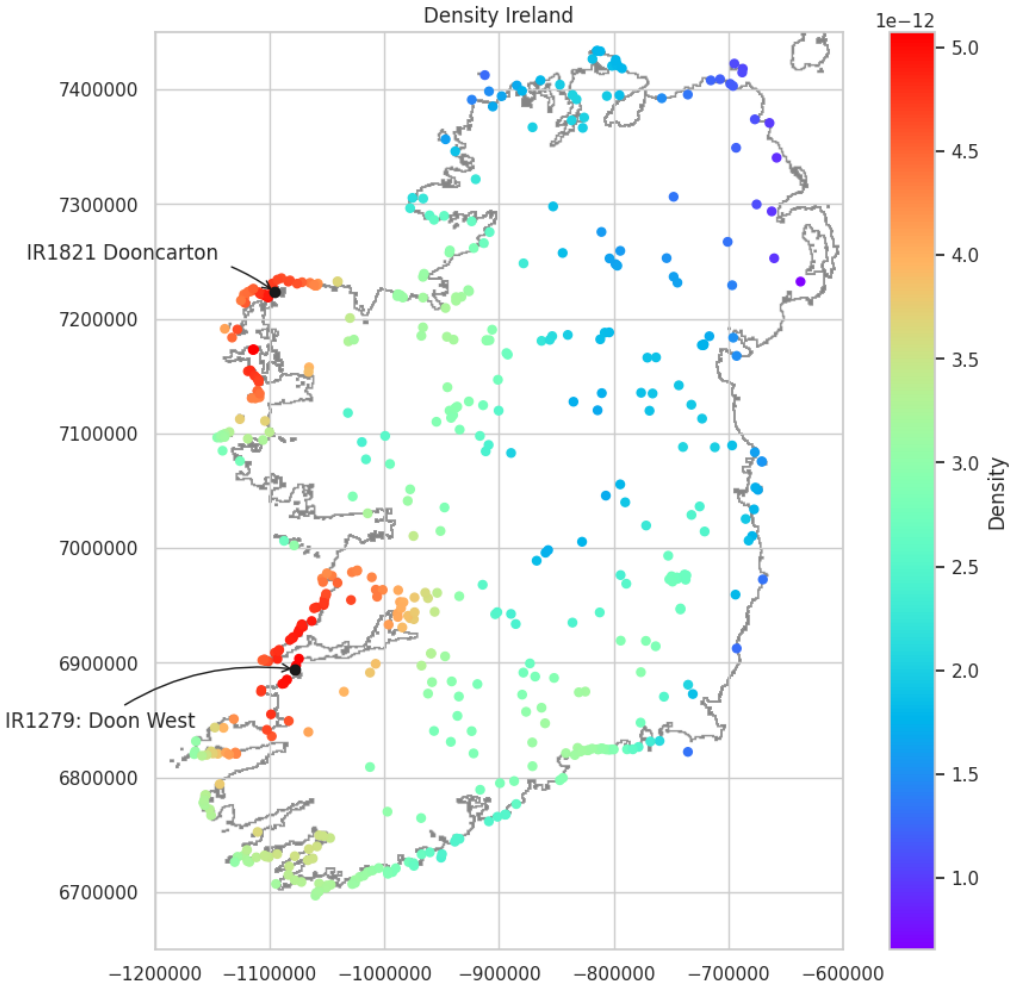


Ireland Density Data Mapped

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

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

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



Middleton, M. 2022, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

See: Location Data: Density Data Mapped.

See: Location Data: Density Data Transformed Mapped.

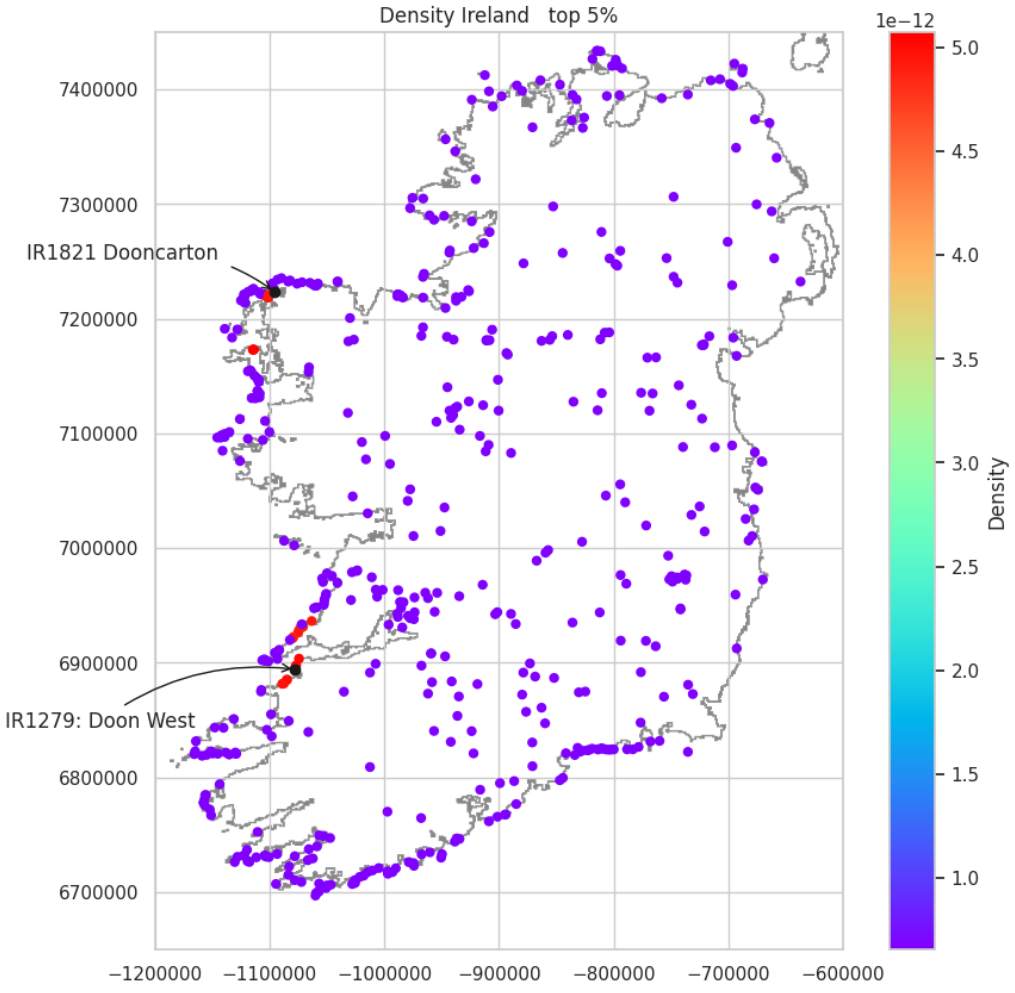
Ireland Data Density Mapped (top 5%)

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

In [ ]:
```
ireland_density_top5 = ireland_density_data.copy()
ireland_density_top5['Density'].where(ireland_density_top5['Density'] > ireland_de
ireland_density_top5['Density'].describe()
```

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

In [ ]:
```
plot_density_ireland_top_5(show_dooncarton, ireland_density_top5)
```



Middleton, M. 2022, Hillforts Primer

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

Ireland Data Split

Split Clusters in Irish Data

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

In [ ]:
```
split_location = 7060000
south_cluster = ireland_density_data[ireland_density_data['Location_Y'] < split_lo
north_cluster = ireland_density_data[ireland_density_data['Location_Y'] >= split_l
```

There are 278 records in the South Ireland data package.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278 entries, 0 to 277
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Location_X     278 non-null    int64
 1   Location_Y     278 non-null    int64
 2   Density        278 non-null    float64
 3   Density_trans  278 non-null    float64
dtypes: float64(2), int64(2)
memory usage: 8.8 KB
```

There are 229 records in the North Ireland data package.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 229 entries, 0 to 228
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Location_X     229 non-null    int64  
 1   Location_Y     229 non-null    int64  
 2   Density        229 non-null    float64
 3   Density_trans  229 non-null    float64
dtypes: float64(2), int64(2)
memory usage: 7.3 KB
```

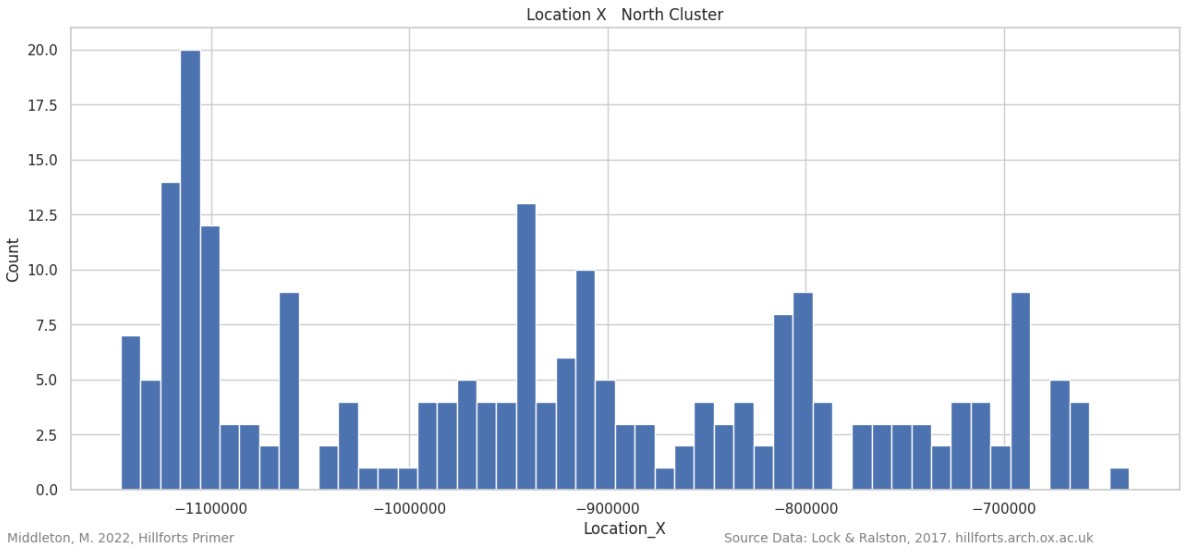
North Ireland Density

Nothern Location Data Plotted (Ireland)

The distribution of forts in North Ireland is very similar to that seen in Ireland Location Data Plotted.

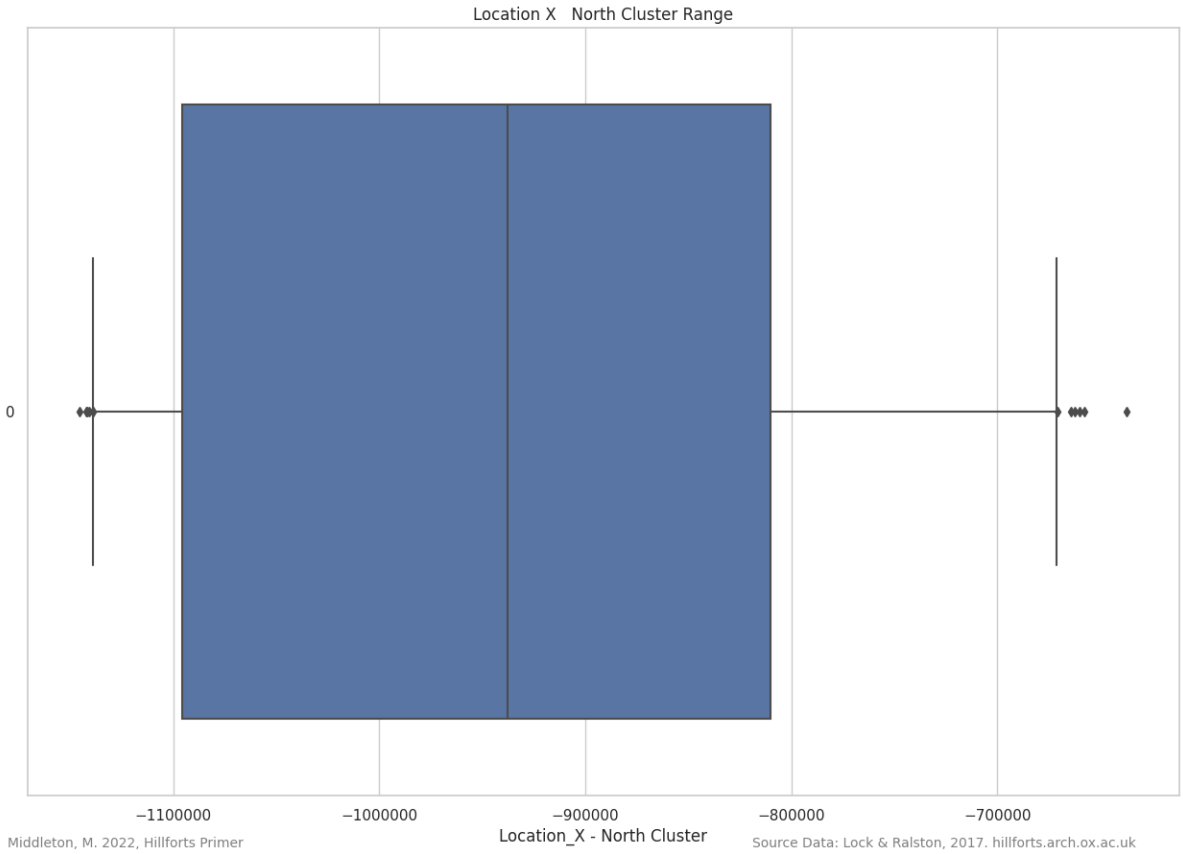
In [ ]:
```
plot_histogram(north_cluster['Location_X'], 'Location_X', 'Location_X - North Clus
```

```
10000
```



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

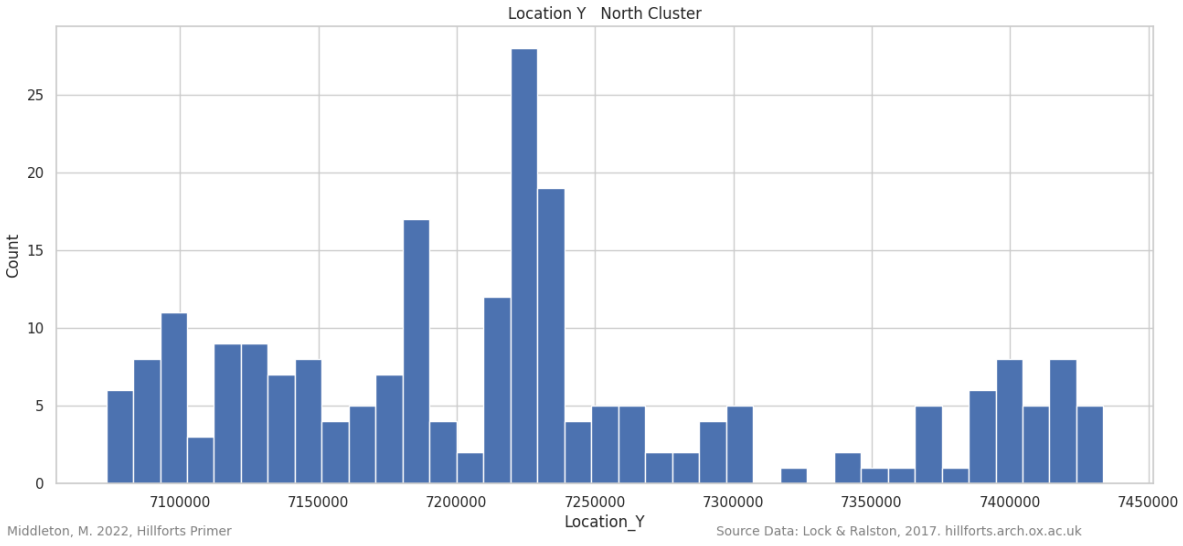
In [ ]:
```
location_X_north_data_ireland = plot_data_range(north_cluster['Location_X'], 'Loca
```



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

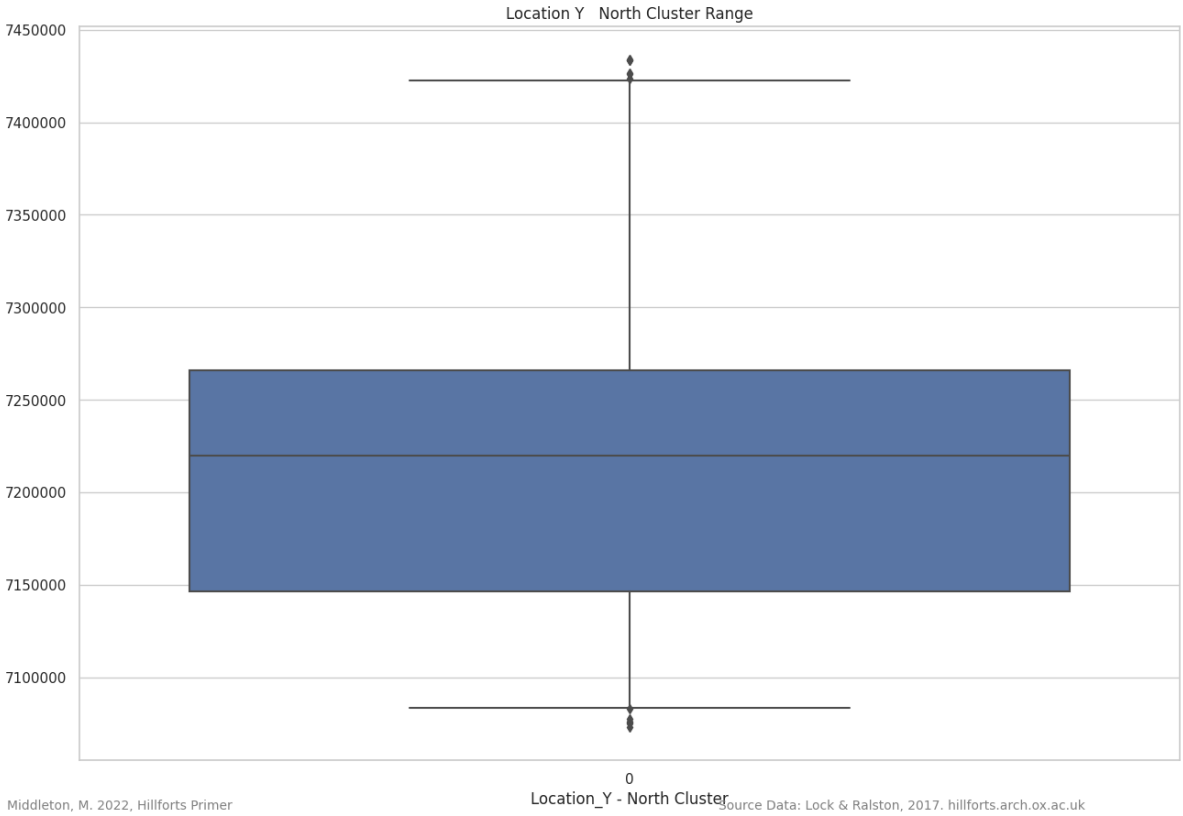
In [ ]:
```
plot_histogram(north_cluster['Location_Y'], 'Location_Y', 'Location_Y - North Clus
```

```
10000
```



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

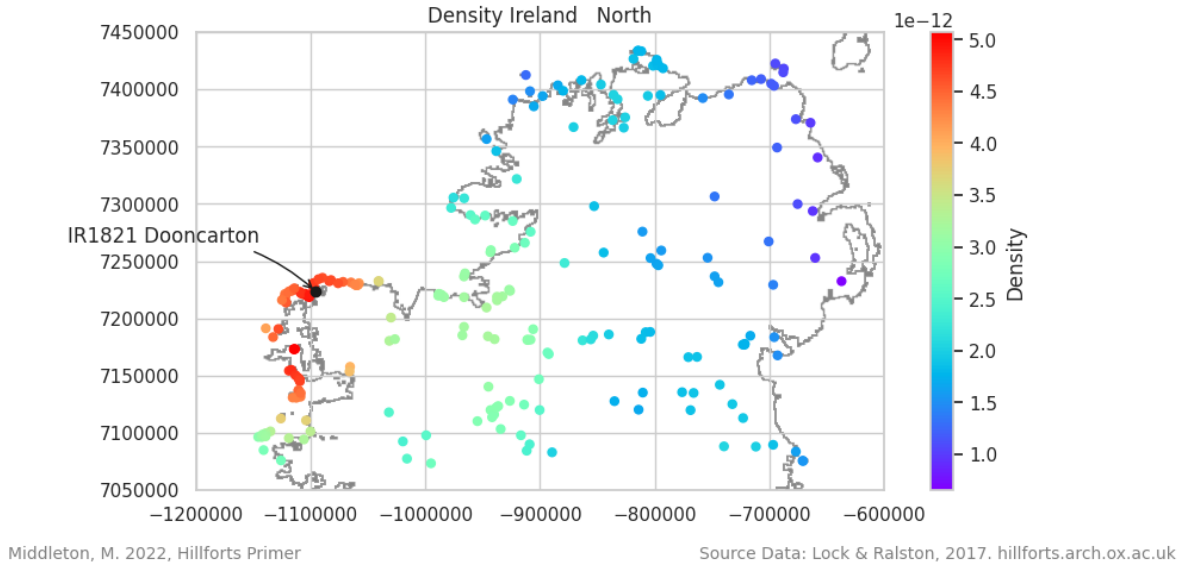
In [ ]:
```
location_Y_north_data_ireland = plot_data_range(north_cluster['Location_Y'], 'Loca
```



Nothern Cluster Location Data Mapped (Ireland)

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

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



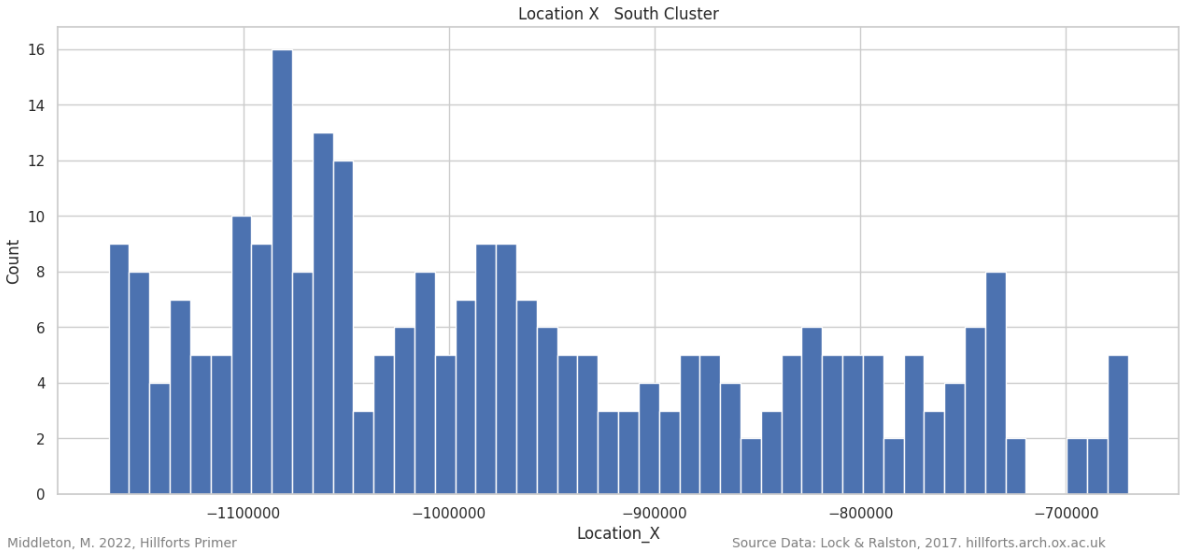
South Ireland Density

Southern Location Data Plotted (Ireland)

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

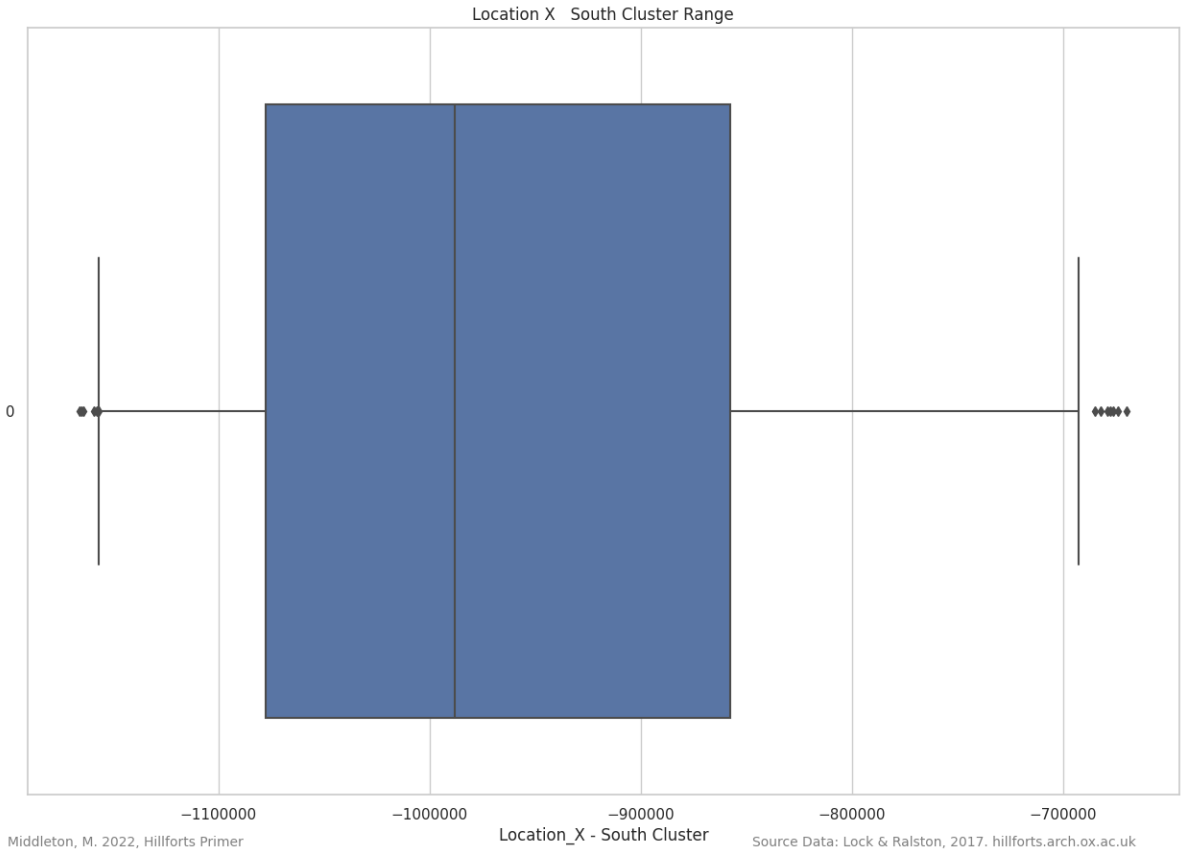
In [ ]:
```
plot_histogram(south_cluster['Location_X'], 'Location_X', 'Location_X - South Clus
```

```
10000
```



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

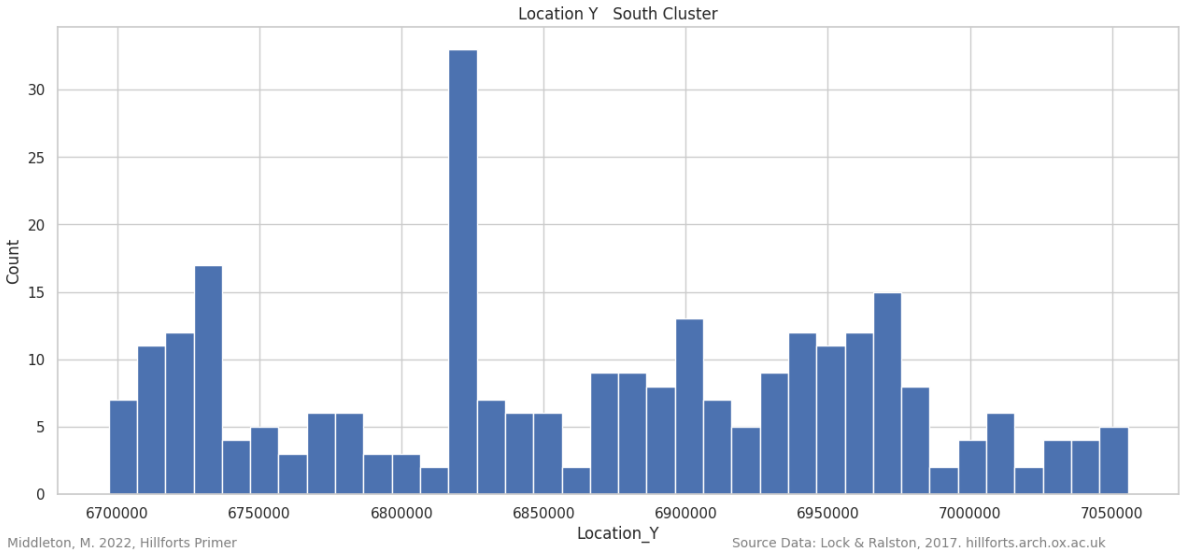
In [ ]:
```
location_X_south_data_ireland = plot_data_range(south_cluster['Location_X'], 'Loca
```



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

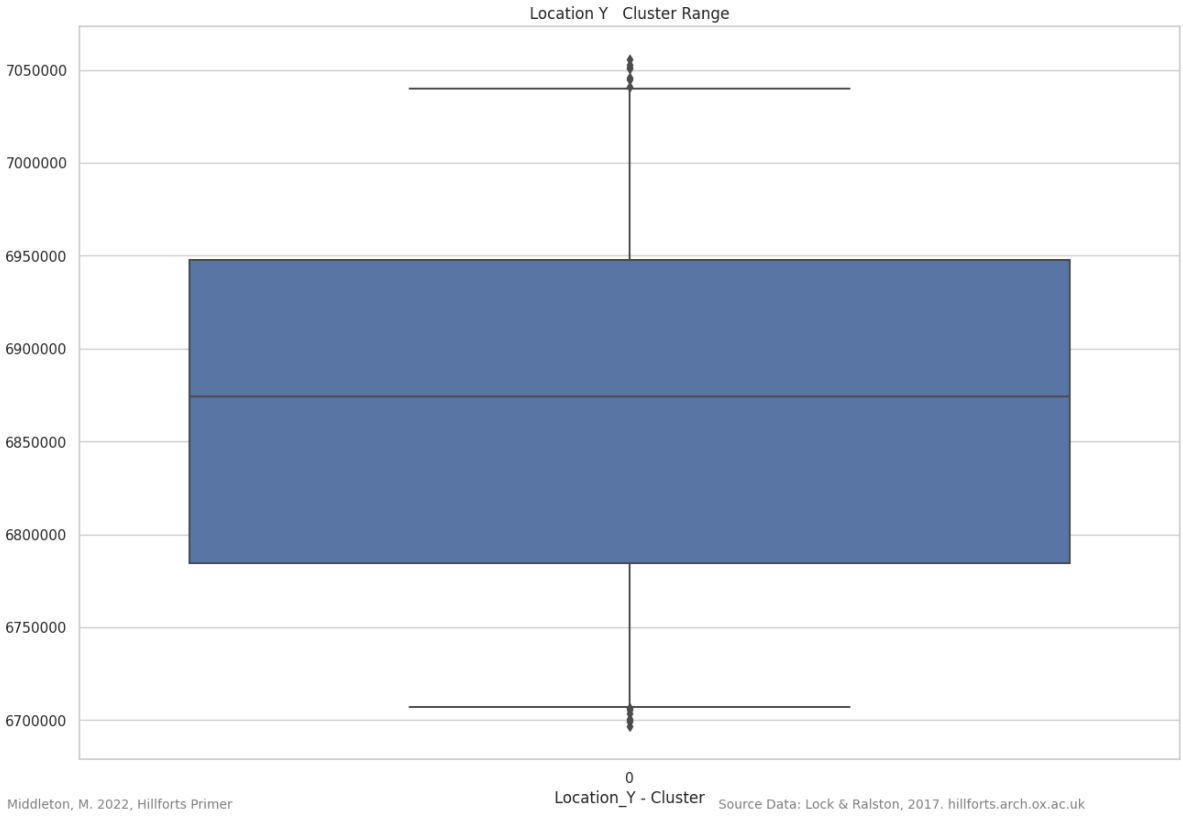
In [ ]:
```
plot_histogram(south_cluster['Location_Y'], 'Location_Y', 'Location_Y - South Clus
```

```
10000
```



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

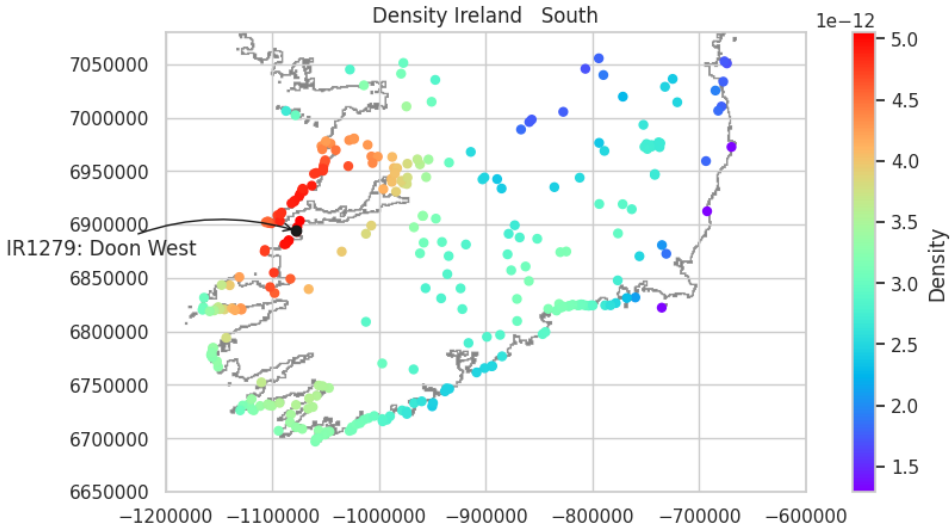
In [ ]:
```
location_Y_south_data_ireland = plot_data_range(south_cluster['Location_Y'], 'Loca
```



Southern Cluster Location Data Mapped (Ireland)

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

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

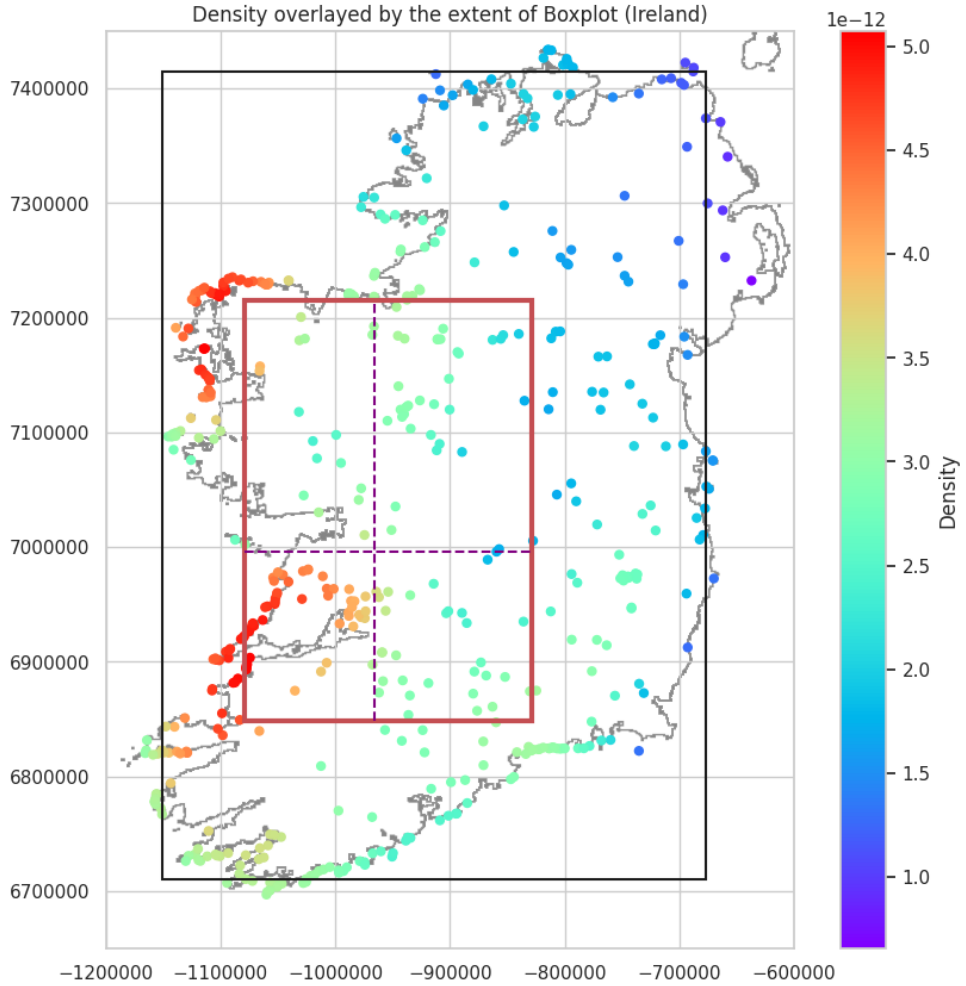


Middleton, M. 2022, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Ireland Boxplots

Density Map showing Extent of Boxplots (Ireland)

In [ ]:
```
plot_ireland_one_boxplot(ireland_density_data, location_X_ireland_data, location_Y_
```



Middleton, M. 2022, Hillforts Primer Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

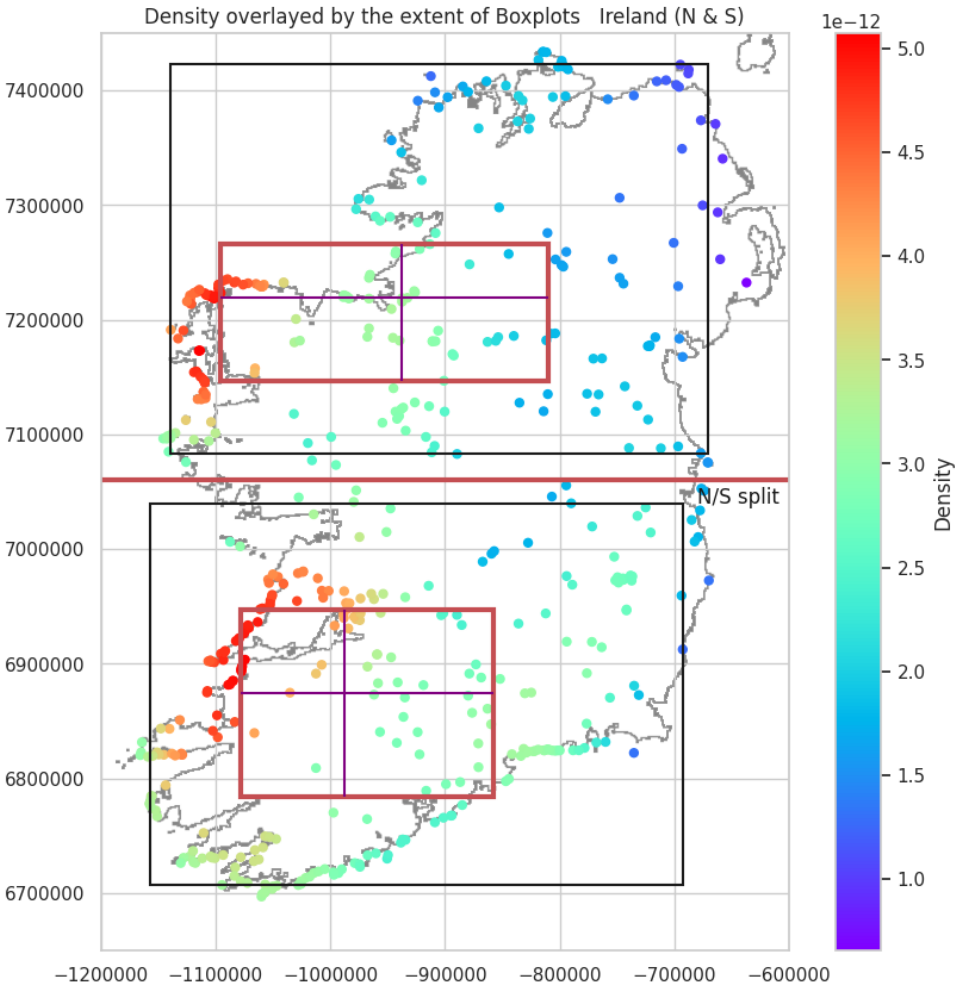
A boxplot over Ireland is stretched between two clusters in the 'Location_Y' axis. In the 'Location_X' axis the boxplot is broad because of the relatively even distribution of sites over

most of the island.

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

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

In [ ]:
```
plot_ireland_boxplots(ireland_density_data, location_X_south_data_ireland, locatio
```



Middleton, M. 2022, Hillforts Primer

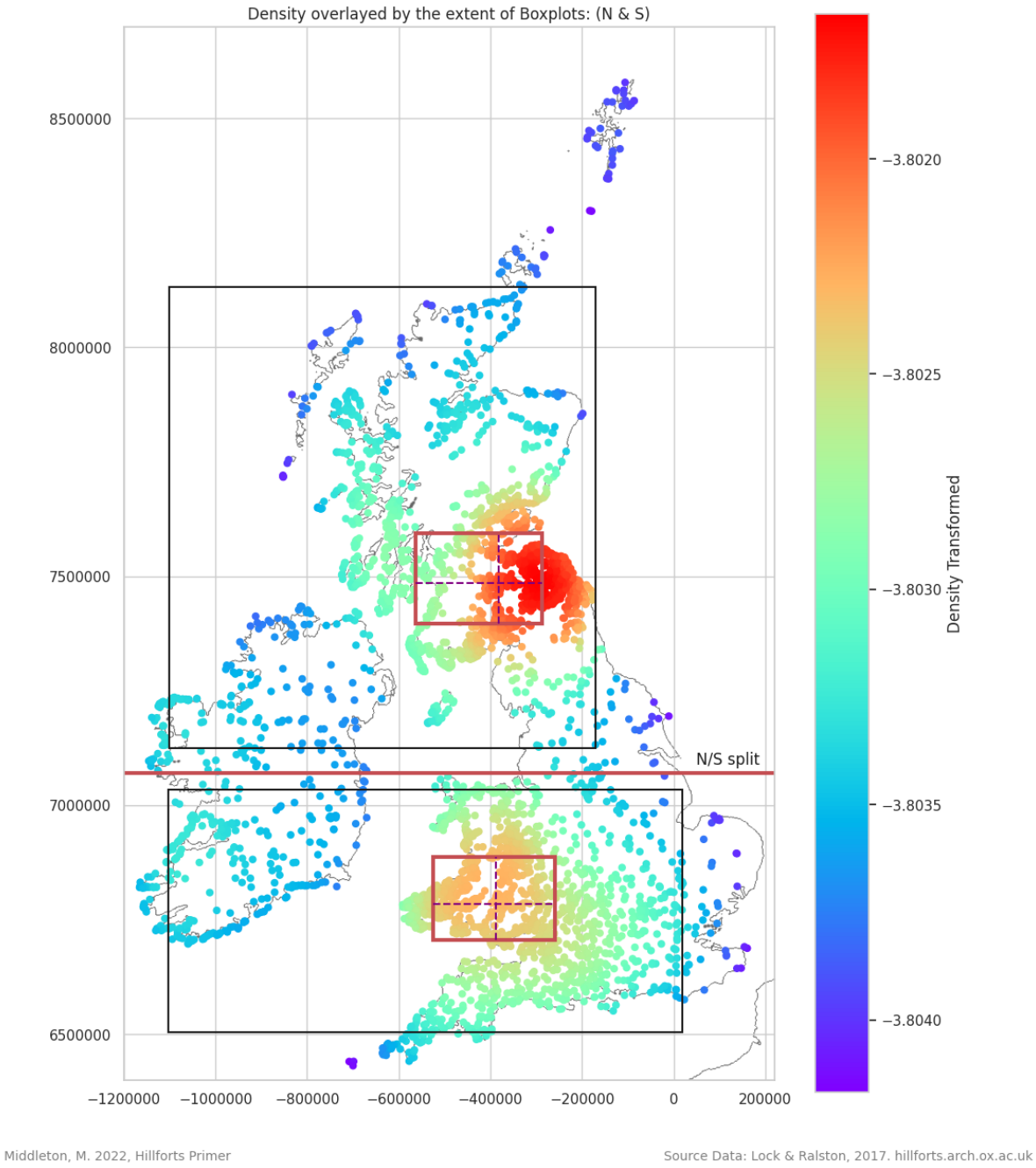
Source Data: Lock & Ralston, 2017. hillforts.arch.ox.ac.uk

Density Boxplots (All Data)

Density Map showing Extent of Boxplots (N & S)

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

In [ ]:
```
plot_ns_boxplots(transformed_location_numeric_data_short, location_X_south_data, l
```



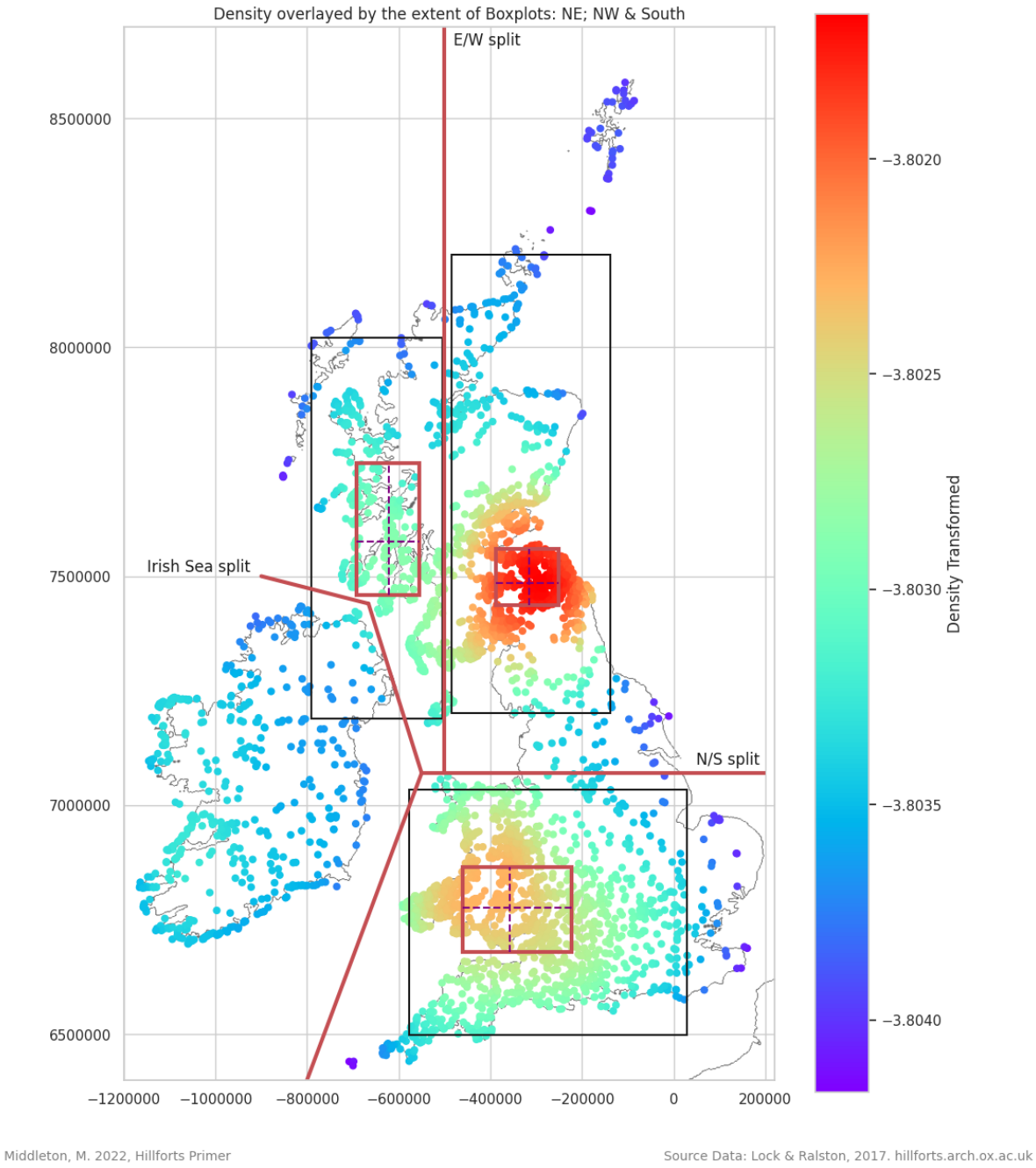
See: Density Map showing Extent of Boxplots (North)

see: Density Map showing Extent of Boxplots (South)

Density Map showing Extent of Boxplots Northeast, Northwest and South

The northern boxplot in Density Map showing Extent of Boxplots (N & S) was stretched to the west indicating secondary clusters. The northern data was then spit to reveal a cluster focussed over the west coast.

In [ ]:
```
plot_ne_nw_s_boxplots(transformed_location_numeric_data_short, location_X_cluster_
```



See: Density Map showing Extent of Boxplots (South)

See: Northwestern Data Minus Ireland Mapped

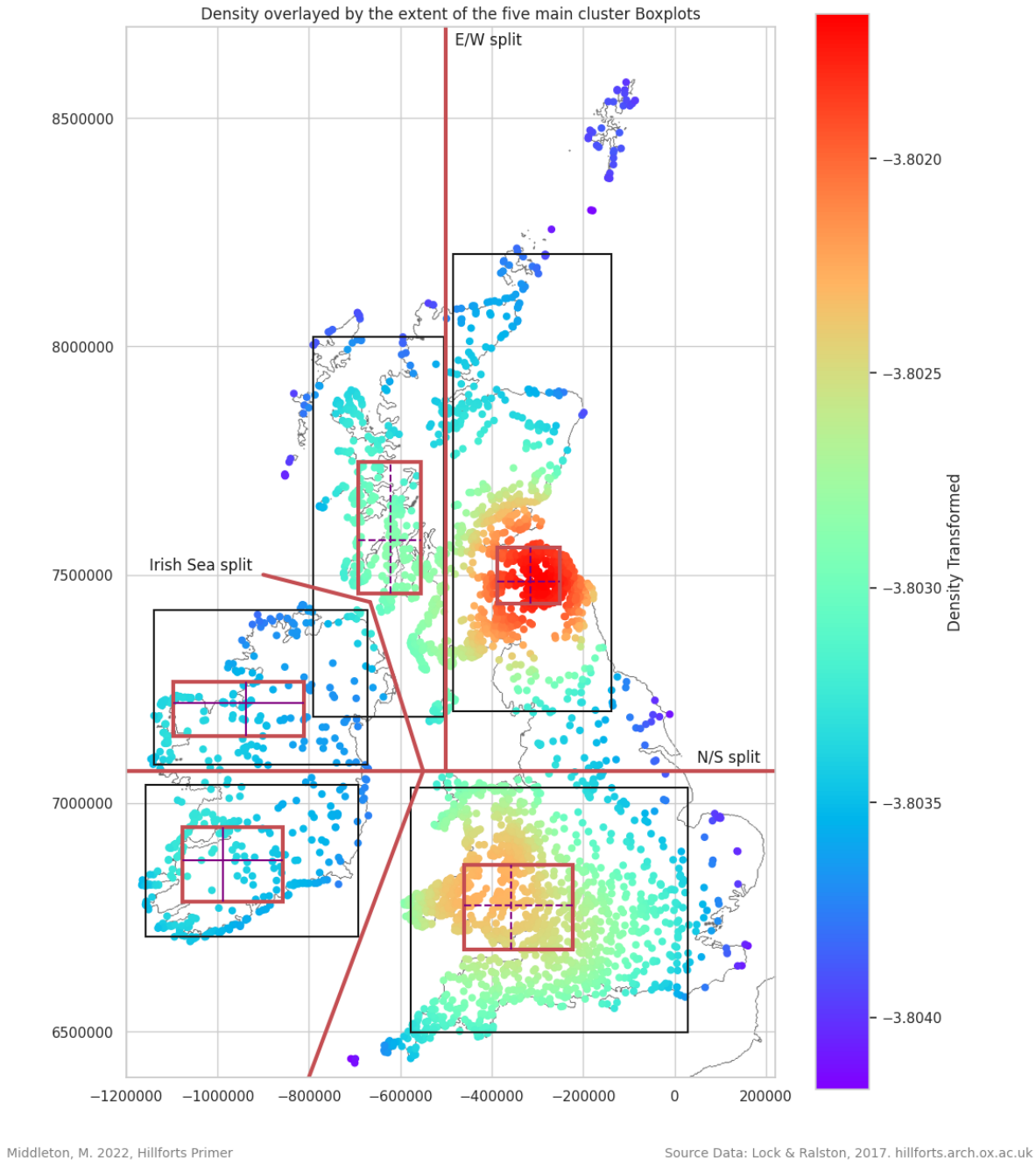
See: Northeast Data Mapped

See: Density Map showing Extent of Boxplots identified in the Atlas Data

Density Map showing Extent of Boxplots identified in the Atlas Data

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

In [ ]:
```
plot_five_clusters(transformed_location_numeric_data_short,location_X_cluster_nortl
```

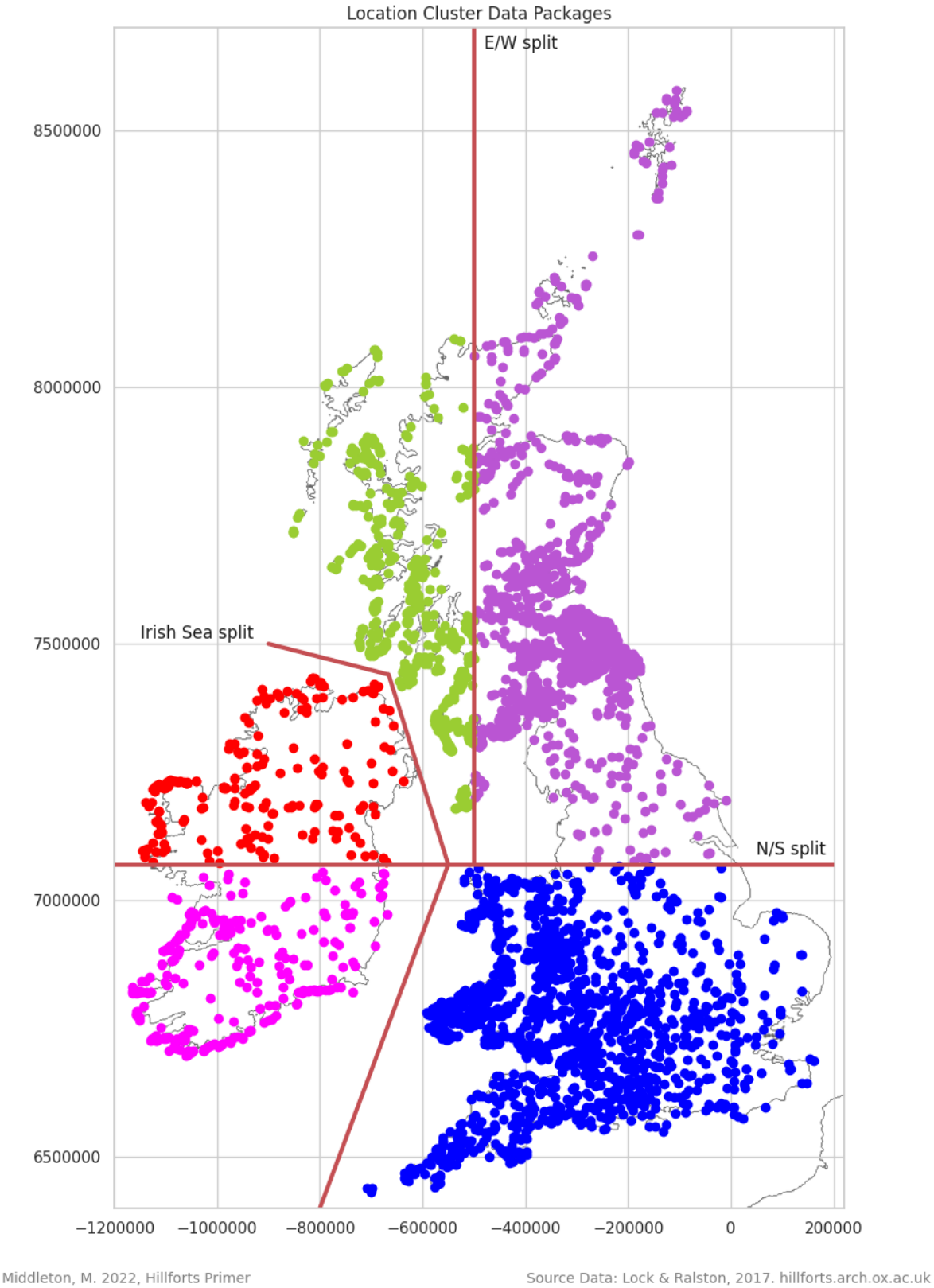


See: Southern Cluster Location Data Mapped (Ireland)
See: Nothern Cluster Location Data Mapped (Ireland)
See: Density Map showing Extent of Boxplots (South)
See: Northwestern Data Minus Ireland Mapped
See: Northeast Data Mapped
See: Density Map Showing Clusters Adjusted by Region

Cluster Data Packages Mapped

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

In [ ]:
```
plot_clusters(cluster_north_ireland, cluster_south_ireland, cluster_south, cluster_
```

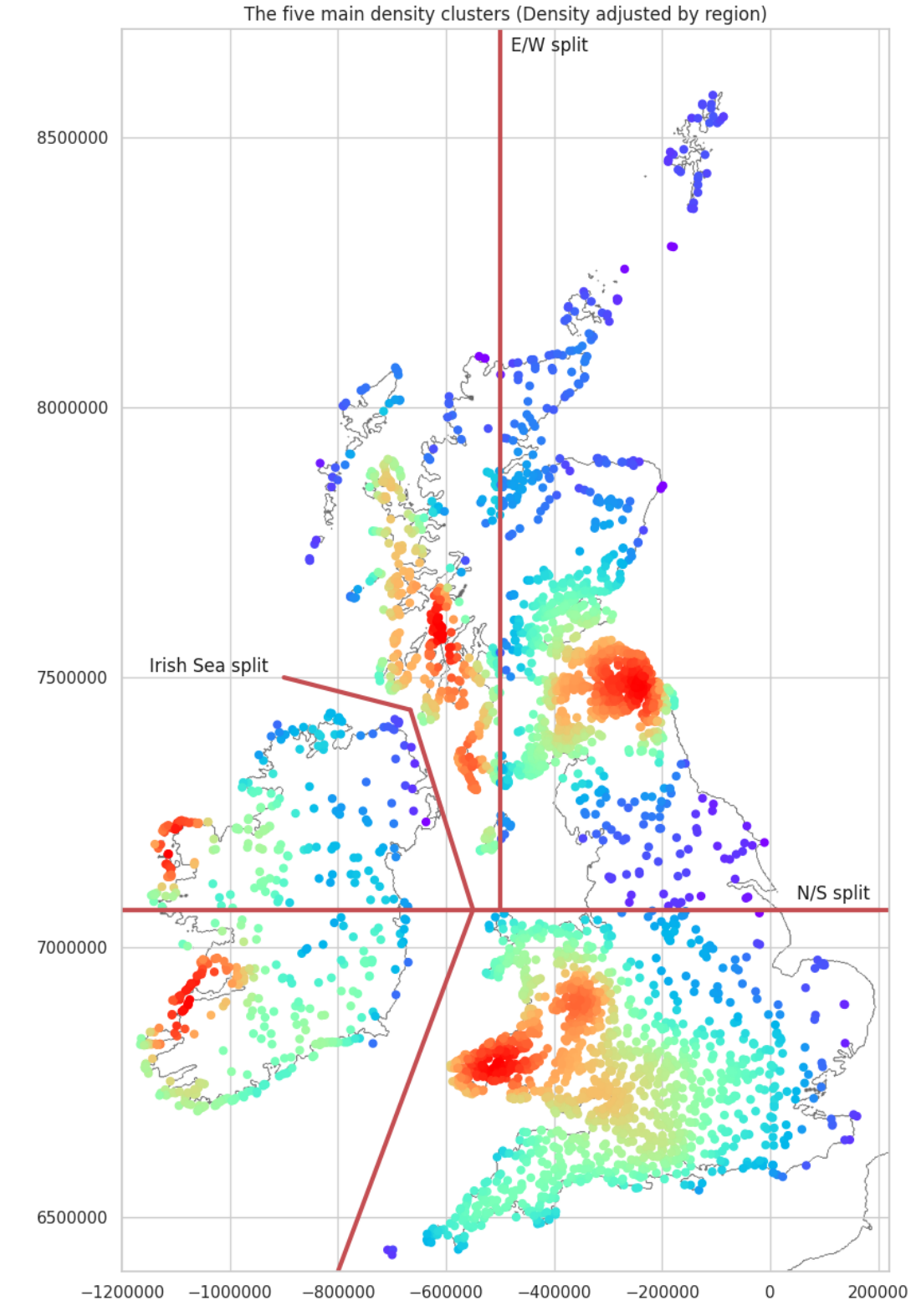


Density Map Showing Clusters Adjusted by Region

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

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

In [ ]:
```
plot_adjusted_density(ireland_density_data,cluster_south,cluster_north_west,north_
                      location_X_cluster_south, location_Y_cluster_south,
                      location_X_cluster_north_west, location_Y_cluster_north_west,
                      location_X_north_e_data, location_Y_north_e_data,
                      location_X_south_data_ireland, location_Y_south_data_ireland,
                      location_X_north_data_ireland, location_Y_north_data_ireland)
```



See: Souythern Cluster Location Data Mapped (Ireland)

See: Nothern Cluster Location Data Mapped (Ireland)

See: Density Map showing Extent of Boxplots (South)

See: Northwestern Data Minus Ireland Mapped

See: Northeast Data Mapped

See: Density Map showing Extent of Boxplots identified in the Atlas Data

Location Data Packages

Pre-processed location data.

In [ ]:
```
location_data_list = [transformed_location_numeric_data_short, location_text_data
download_location_data_list = [location_numeric_data, transformed_location_numeric_
```

Download Location Data

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

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

Save Figure List

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

Part 2: Management & Landscape

Colab Notebook: Live code

HTML: Read only

HTML: Read only topographic