NFDI4Objects
Community Cluster
Research Software
Engineering (RSE)

RESEARCH SQUIRREL ENGINEERS
Network

CAA Germany

Geodesist in the Humanities

... working with humanities people on Research Software and FAIRification Tools ...

Florian Thiery M.Sc.

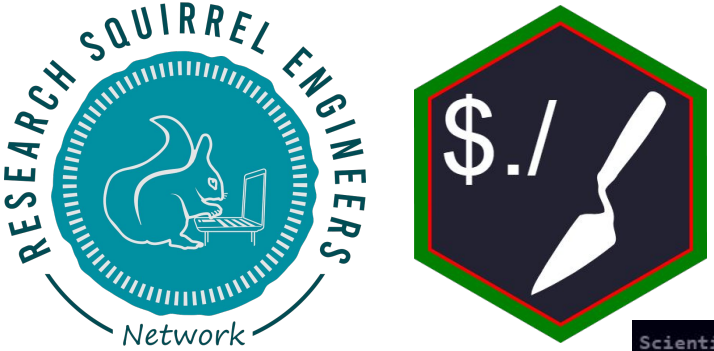
deRSE25 | 2025 | 25-27 Feb 2025 | Karlsruhe | Germany | 27/02/2025
Session: How to improve the visibility [...] of RSE(s) in NFDI

DOI 10.5281/zenodo.14920034

image: Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

This is me, **Florian Thiery M.Sc.**,

RSE in Task Area 2 “Collecting” within **NFDI4Objects** at LEIZA (Mainz).



Scientific Scripting Languages in Archaeology
A special interest group of CAA International dedicated to scientific scripting languages in archaeology.

Computer Applications &
Quantitative Methods in Archaeology

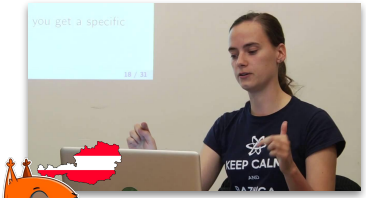
In N4O I am closely related to the CAA groups and the Squirrels

image: Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

My Goal: Open Data with Open Source: FAIR Data with FAIR4RS Tools

The Research Squirrel Engineers Network is a loose association of Linked Open Data/Wikidata enthusiasts, research software engineers and citizen scientists specialising in computational archaeology and geoinformatics.

Sophie
@SCSchmidt
0000-0003-4696-2101



Martina
@bellerophons-pegasus
0000-0003-0485-6861

Florian
@florianthiery
0000-0002-3246-3531

Timo
@situx
0000-0002-9499-5840

RESEARCH SQUIRREL ENGINEERS Network

WIKIDATA

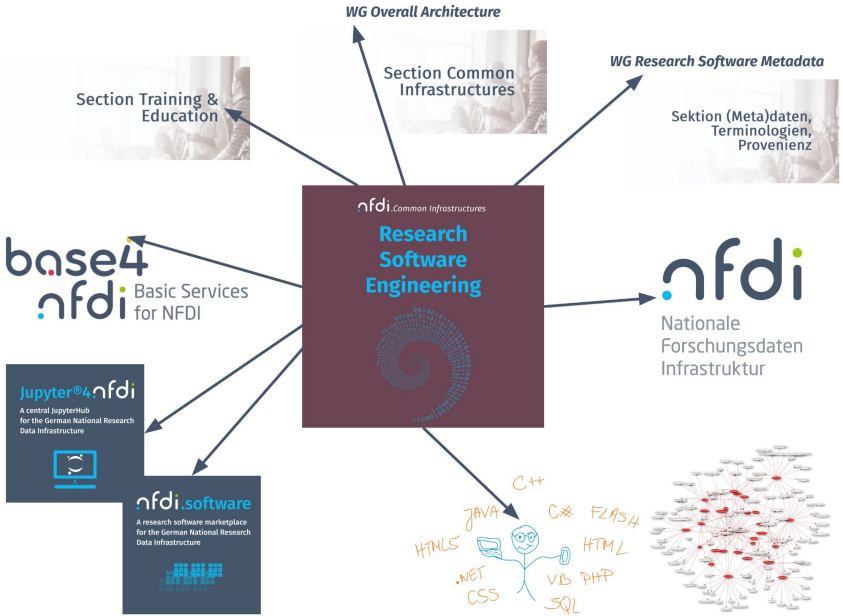
RDF

+ B. Danthine, A.-K. Distel, Peter Thiery, Fiona Schenk et al.

The members develop and maintain research and FAIRification and FAIRification tools and implement them in specific projects.

NFDI4**Objects**

Community Cluster
Research Software Engineering (RSE)



I am Chair of the N4O Community Cluster RSE
and the WG RSE within the INFRA section in the NFDI e.V.

NFDI4Objects

Community Cluster
Research Software Engineering (RSE)

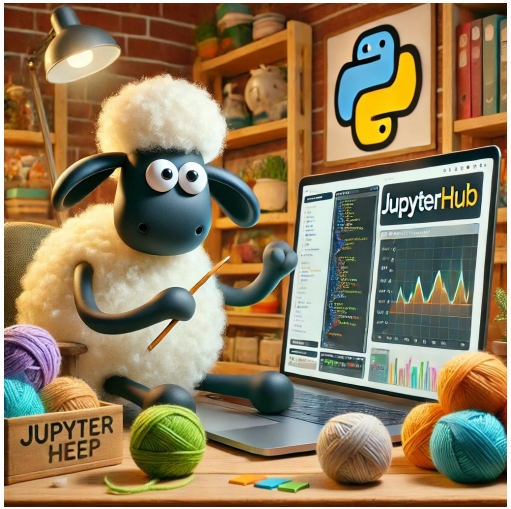
Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

rse.i | FAIR4RS

Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

rse.ii | n4o.software

Florian Thiery CC BY 4.0, created using OpenAI's DALL·E



rse.iii | jupyter4objects

We want to initiate RSE Working Groups

image: Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

RSE in Action ...

image: Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

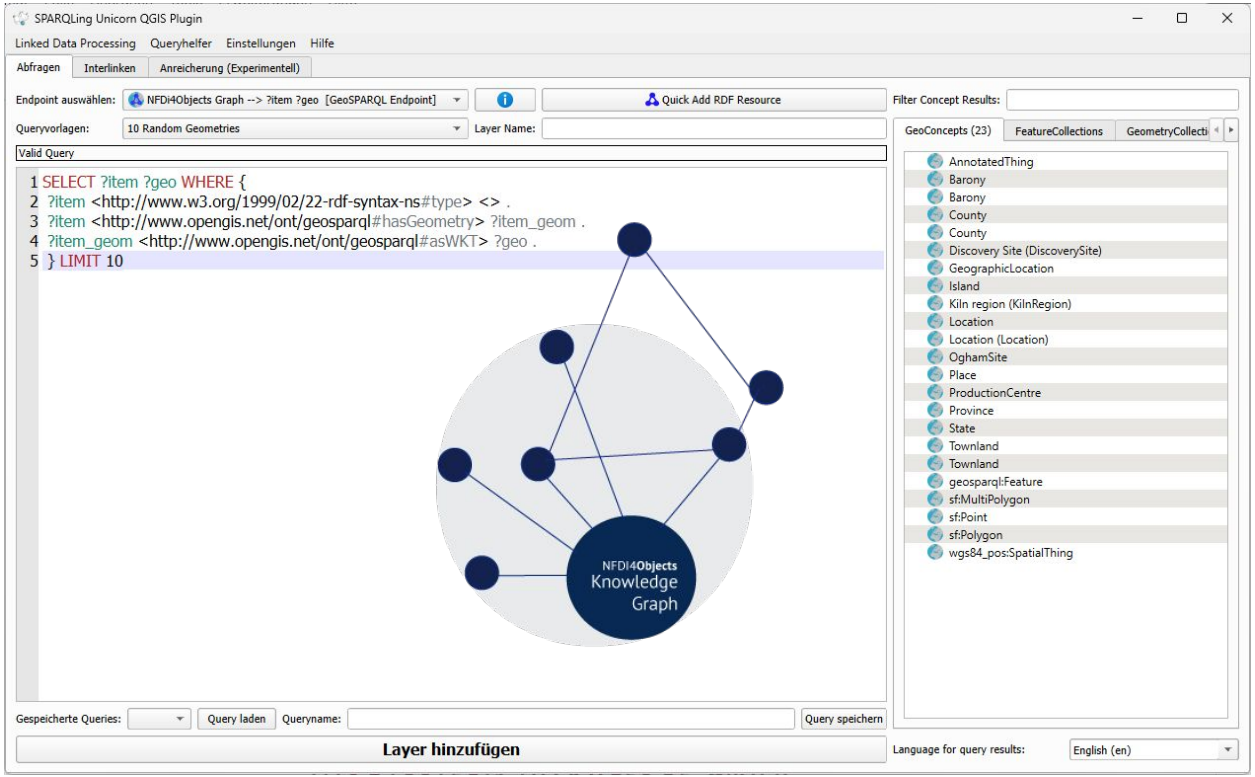
... using the **SPARQL Unicorn Research Toolkit**.

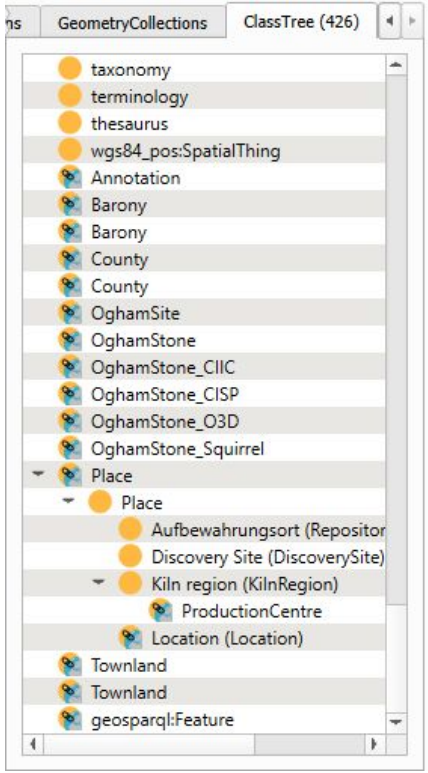
A FAIRification tool for digital data management is the SPARQL Unicorn and its implementation for QGIS

It contains of

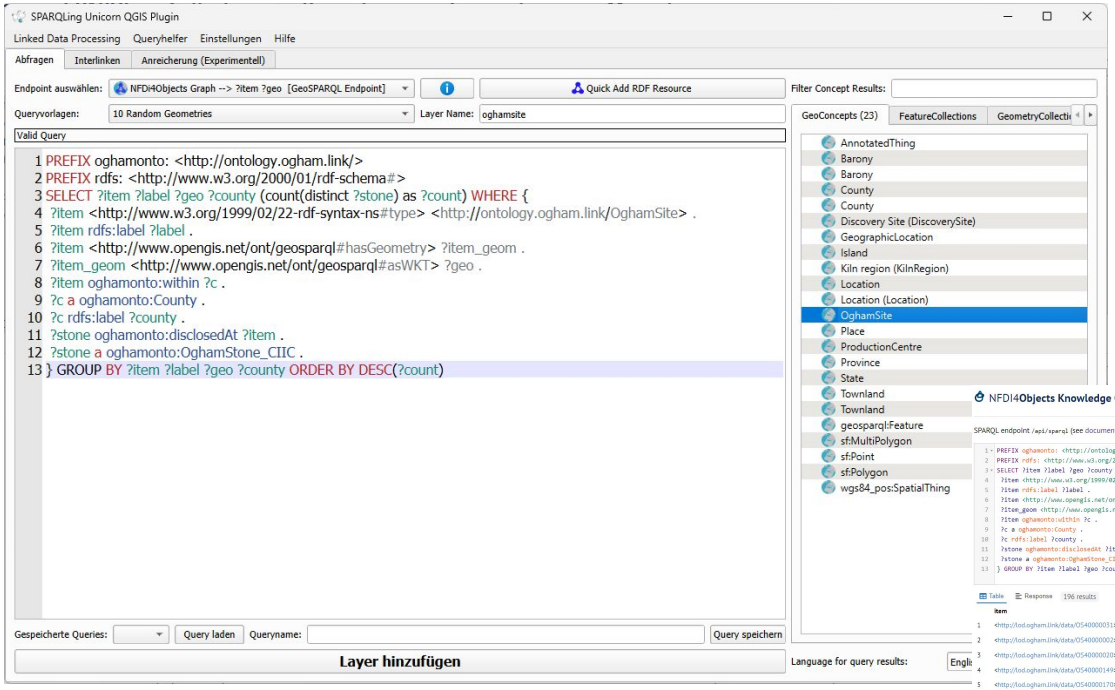
(i) the **SPARQLing Unicorn QGIS Plugin**

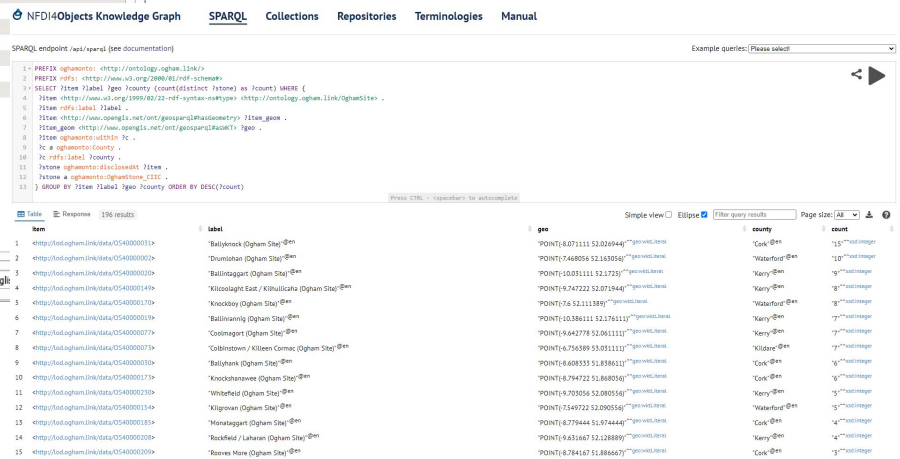
(ii) the **SPARQL Unicorn Ontology Documentation** tool





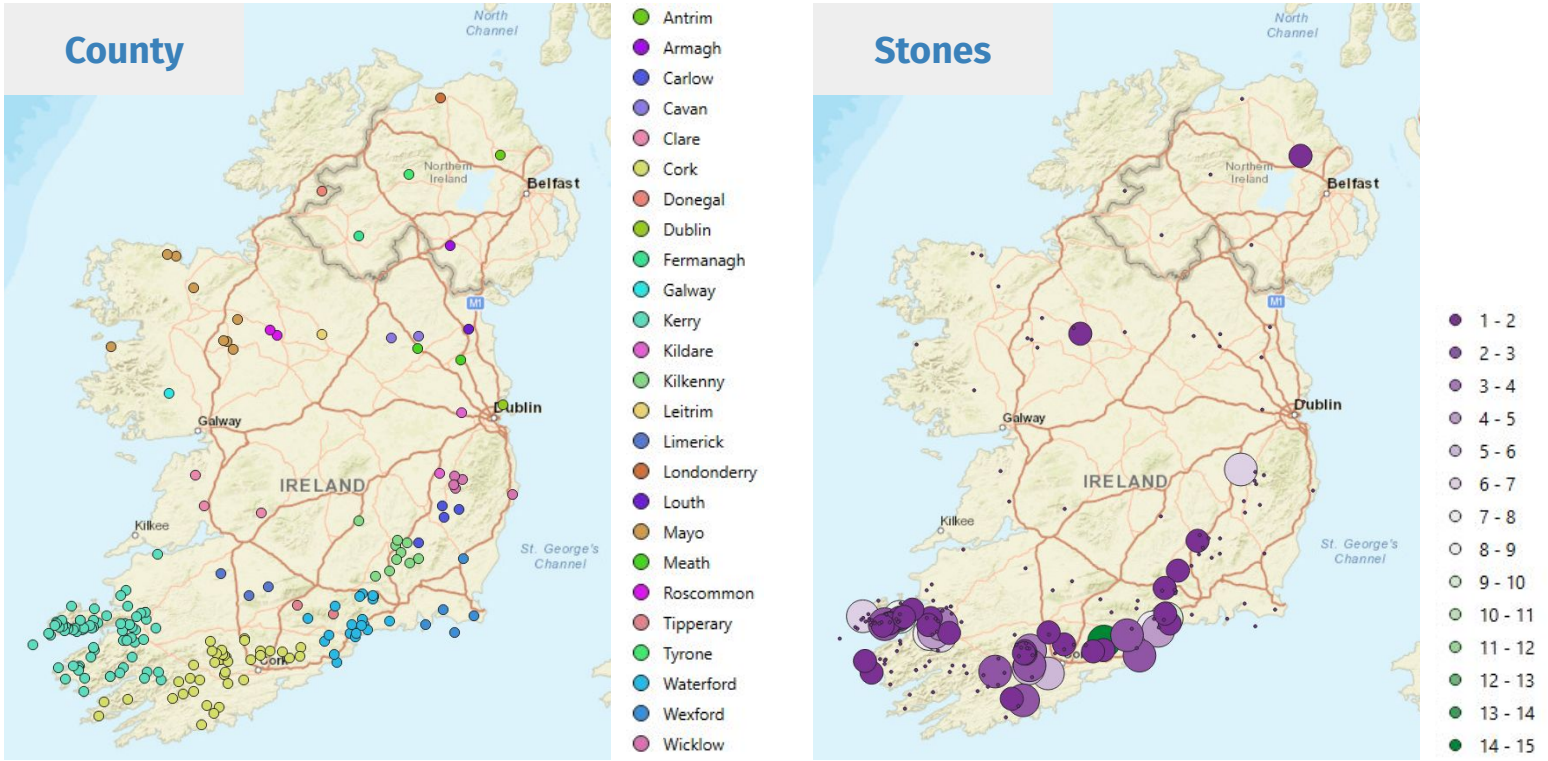
NFDI4Objects Knowledge Graph





https://t1p.de/v0lg7

N4O KG - Ogham Stones / Sites / Counties



N4O KG - Ogham Stones / Sites / Counties

SPARQL Unicorn Ontology Documentation

DOI 10.5281/zenodo.8190763

This repository hosts a standalone version of the HTML documentation feature included in the SPARQLing Unicorn QGIS Plugin.

Rather than initiating the documentation generation within the SPARQLing Unicorn QGIS Plugin, this python script allows the generation of the documentation standalone or as a Github Action.

The standalone script does not rely on QGIS classes and does not provide the full functionality available in the SPARQLUnicorn QGIS Plugin.

Deviations from the SPARQLing Unicorn Plugin are listed as follows:

- Support for less geometry literals: Only WKT and GeoJSON literals are supported for rendering

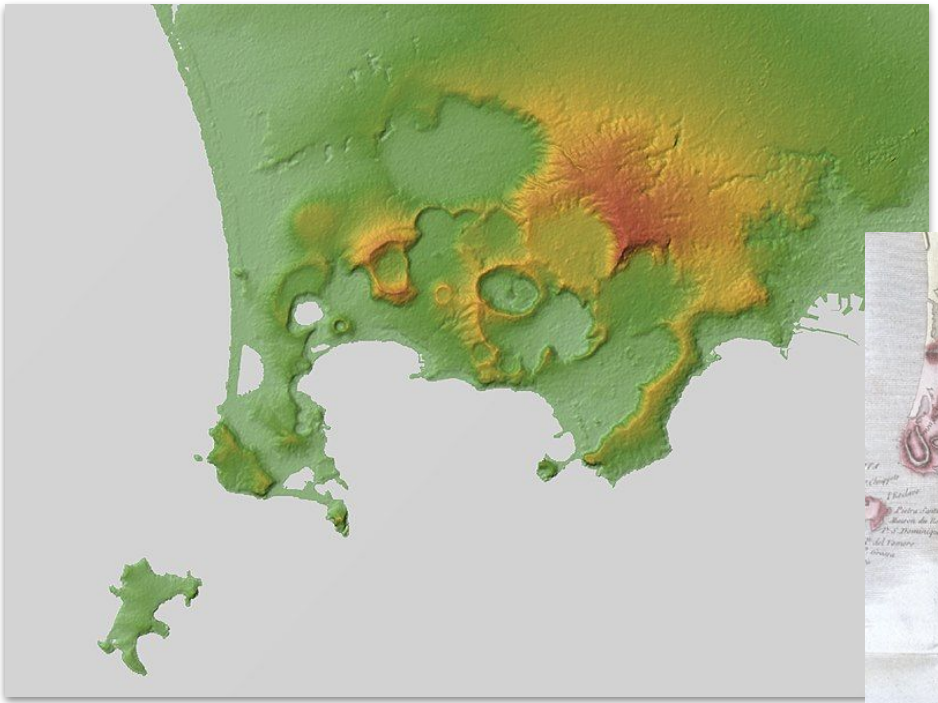
Usage Example as Github Action

For a usage example please refer to this repository: https://github.com/sparqlunicorn/sparqlunicornGoesGIS_testdata

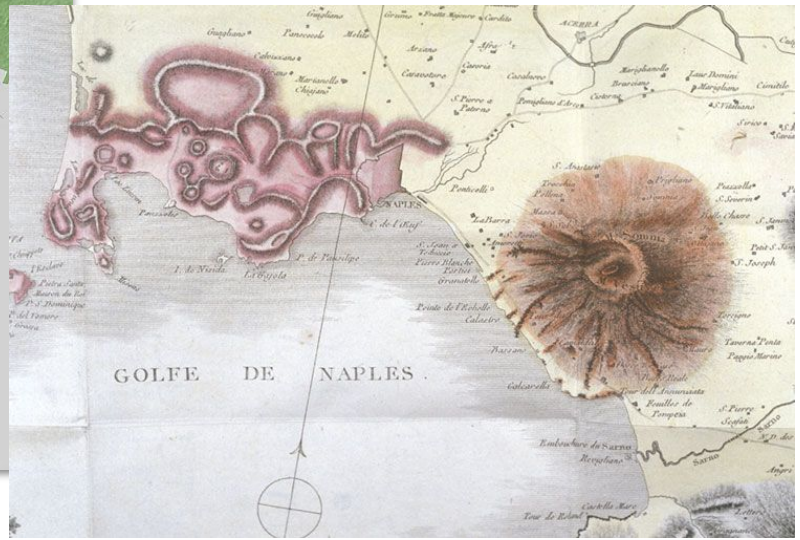
https://github.com/sparqlunicorn/sparqlunicornGoesGIS-ontdoc
via 10.5281/zenodo.8190763

HTML creation with the help of Unicorn and GitHub Actions

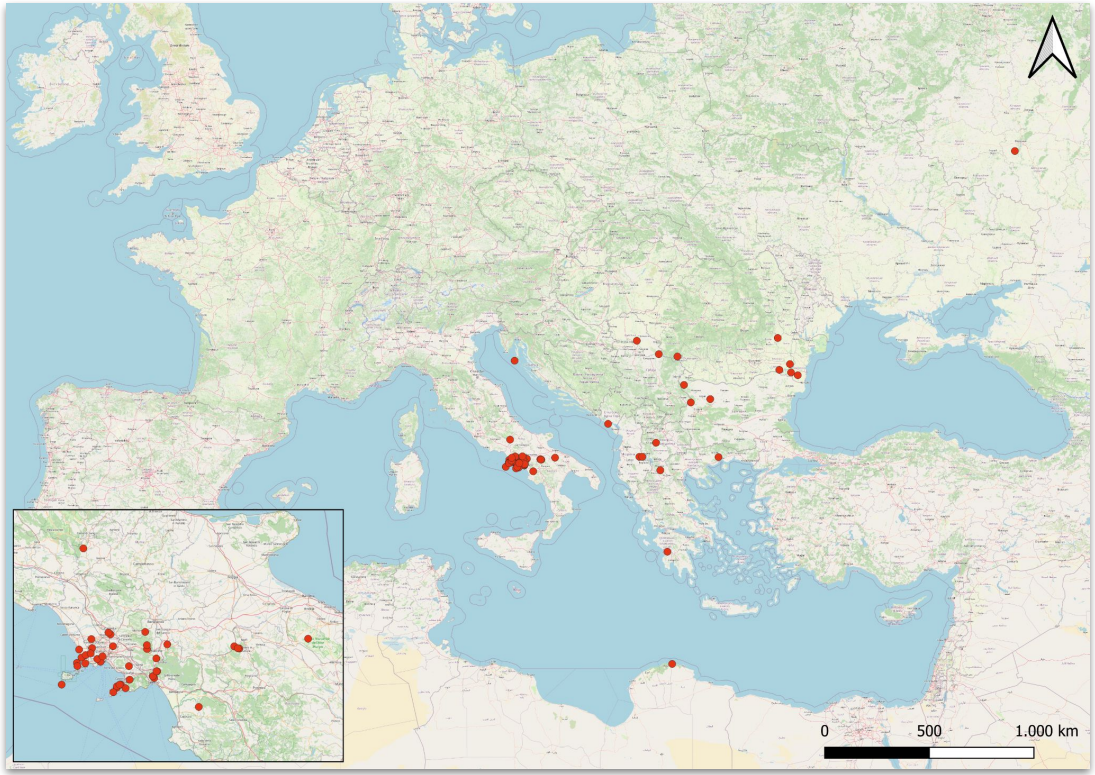
Batholith, Public domain, via Wikimedia Commons



Breislak, Scipione (1748-1826). Voyages physiques et lythologiques dans la Campanie. Paris: Dentu, Imprimeur-libraire, 1801, via https://vulcan.lindahall.org/12_large.shtml



About 40,000 yr b2k ago, the largest eruption of the Campanian Ignimbrite (CI) took place in the Phlegraean Fields.



Evidence of the ash fall from this late Pleistocene volcanic event can be found throughout Central Europe.

via https://doi.org/10.1029/139GM20 © Fedele et al.

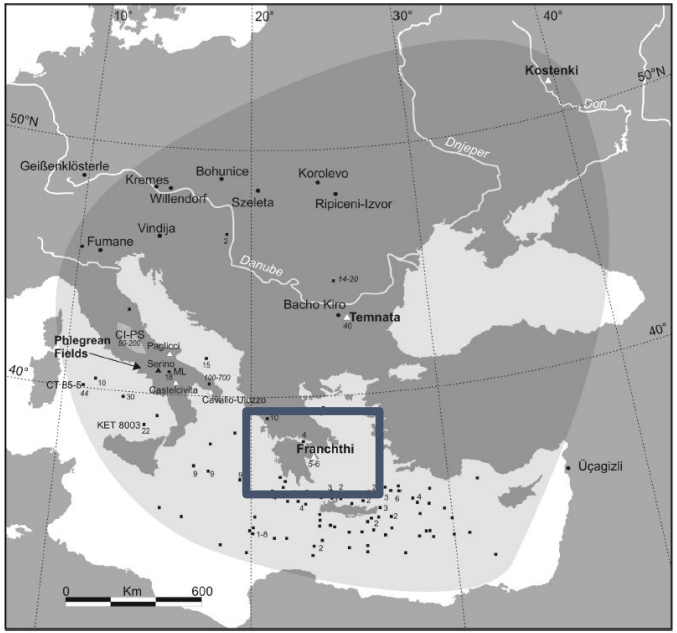


Figure 1. Geographic distribution of the Campanian Ignimbrite deposits, including archaeological and sampling sites mentioned in the paper. Solid squares: CI tephra occurrences and related thickness in cm (in italics if reworked) [modified from *Cornell et al.*, 1983; *Amirkhanov et al.*, 1993; *Cini Castagnoli et al.*, 1995; *Narcisi and Vezzoli*, 1999; *Fedele et al.*, 2002; *Upton et al.*, 2002]. Dotted area in central Italy: distribution of the CI-derived paleosol (CI-PS [*Frezzotti* and *Narcisi*, 1996]). Solid triangles: archaeological sites with CI ash layer. Blank triangles: archaeological sites with ash layer attributed to the CI on the basis of cultural-stratigraphic position and ^{14}C dating (*Fedele et al.*, 2002; *Giaccio* and *Isaia*, on file). Solid circles: selected European Palaeolithic sites within the area potentially affected by the CI air-fall.

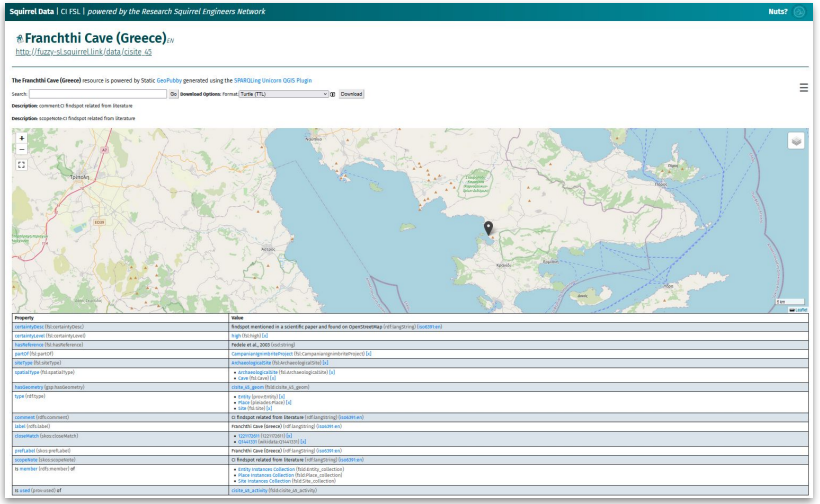
Franchthi Cave (Greece) in Fedele et al. (2003)

Zde, CC BY-SA 4.0, via Wikimedia Commons

Zde, CC BY-SA 4.0, via Wikimedia Commons

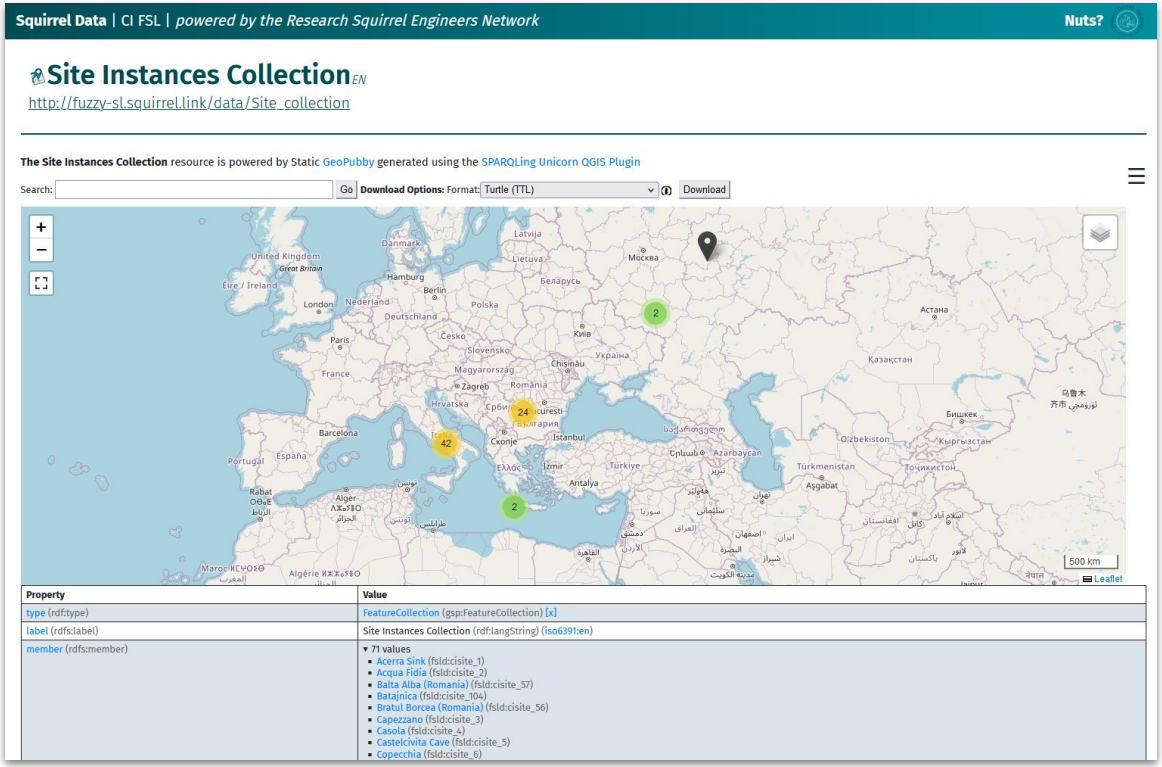
Xalaros, Attribution, via Wikimedia Commons

https://research-squirrel-engineers.github.io/campanian-ignimbrite-geo/cisite_45/index.html



Franchthi Cave as Archaeological Site in the LOD Cloud

https://research-squirrel-engineers.github.io/campanian-ignimbrite-geo/Site_collection/index.html

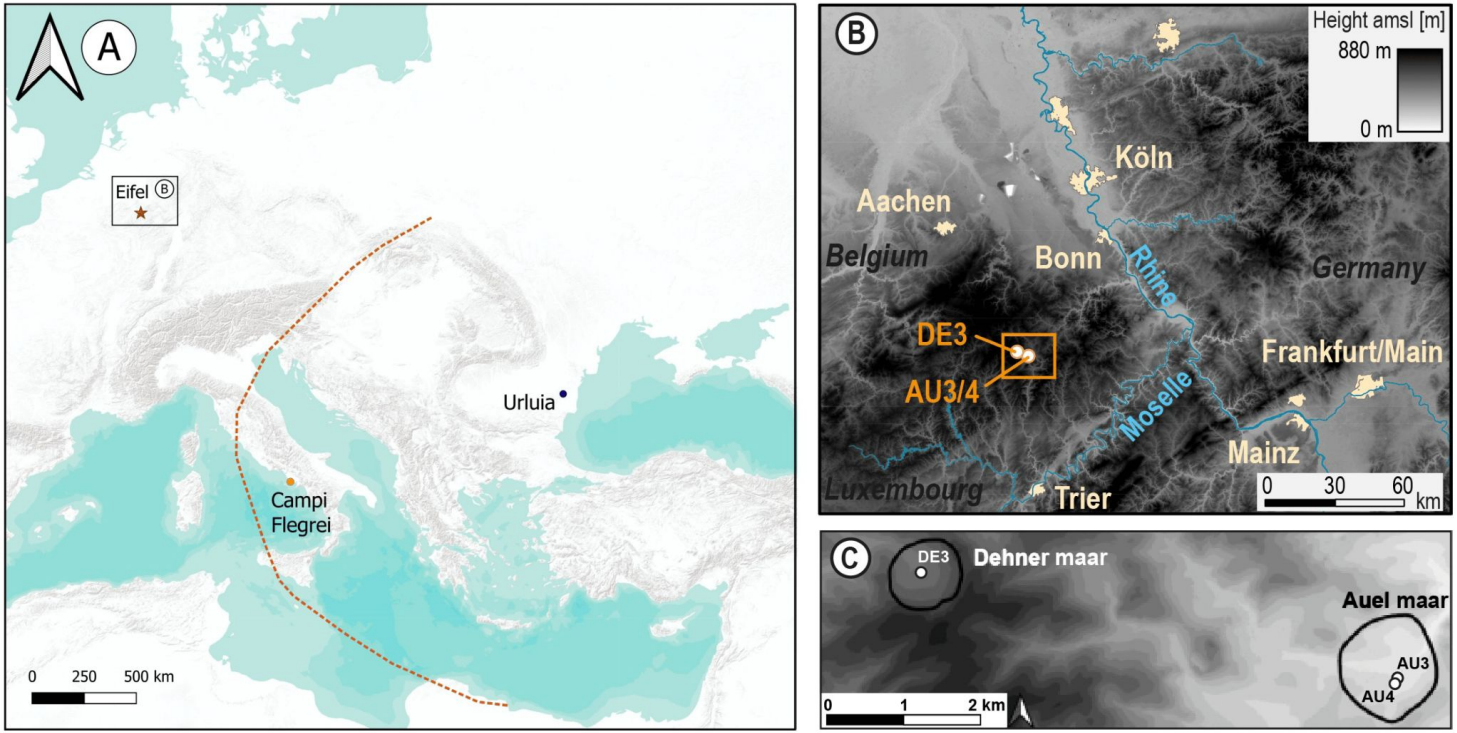


All these sites and information can be visualised via the Unicorn

image: Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

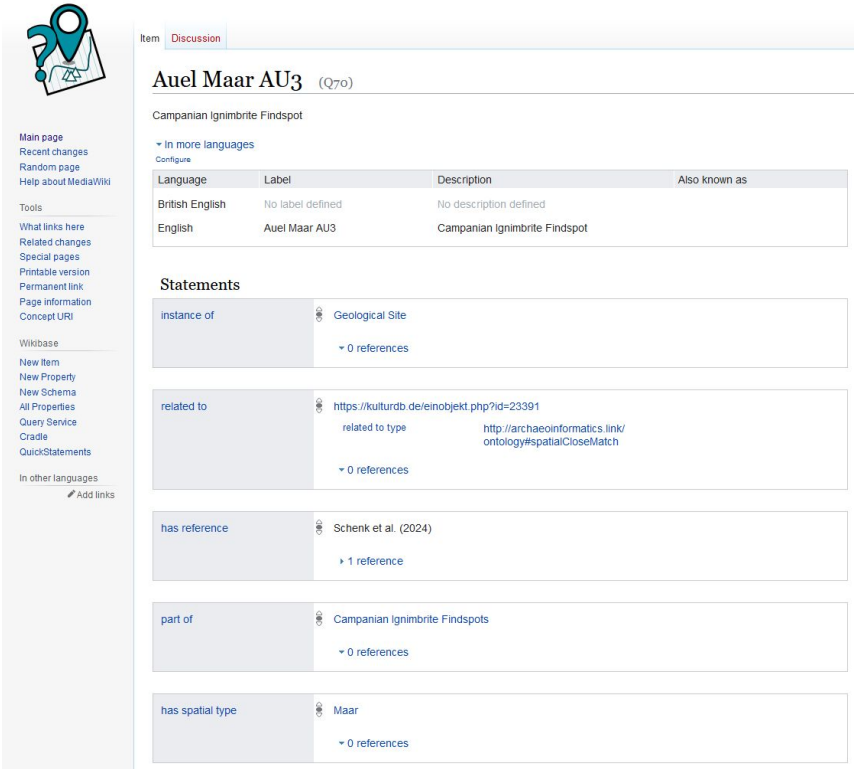
Using **Python Minions** these data could be also transformed into **QuickStatements** within the **fuzzy-sl Wikibase**

via https://doi.org/10.3390/quat7020017, Fig. 1

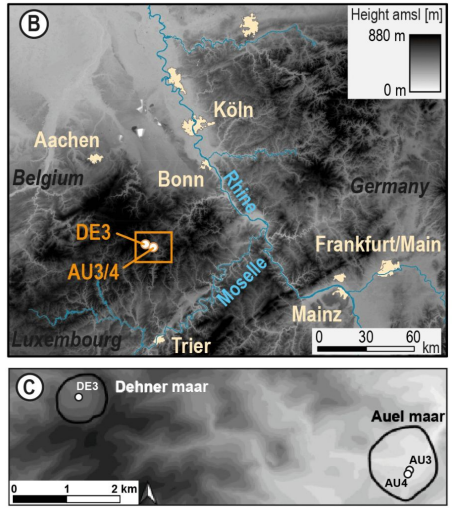


Auel Maar (AU3/4) & Dehner Maar (DE3) sites in Schenk et al. (2024)

via https://fuzzy-sl.wikibase.cloud/wiki/Item:Q70



via https://doi.org/10.3390/quat7020017, Fig. 1



Auel AU3 [Q70] in the fuzzy-sl Wikibase

image: Florian Thiery CC BY 4.0, created using OpenAI's DALL·E

Jupyter Python Minions as **Jupyter Notebooks**

Define SPARQL query service

```
import os
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
import contextily as ctx  # For adding OpenStreetMap basemaps
from matplotlib.patches import Patch
from scipy.stats import gaussian_kde
import numpy as np

def querySparql(query):
    sparql = SPARQLWrapper("https://query.wikidata.org/sparql")
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
    results = sparql.queryAndConvert()
    return results['results']['bindings']

# Define the GeoJSON file path
geojson_file = os.path.join(os.getcwd(), "gs_ireland_island.geojson")  # Adjusted for Jupyter Notebook
```

Define the SPARQL Query

```
# SPARQL Query
oghamQuery = """
SELECT ?item ?itemLabel ?geo ?site ?siteLabel ?county ?countyLabel WHERE {
  ?item wdt:P31 wd:Q2016147.
  ?item wdt:P189 ?site.
  ?site wdt:P31 wd:Q72617071.
  ?item wdt:P189 ?county.
  ?county wdt:P31 wd:Q179872.
  ?item wdt:P625 ?geo.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
"""
```

Fetch Data and Convert to DataFrame

```
# Fetch data using the SPARQL query
sparql_results = querySparql(oghamQuery)

# Convert SPARQL JSON results into a DataFrame
data = []
for result in sparql_results:
    geo = result['geo']['value'] if 'geo' in result else None
    lat, lon = (None, None)
    if geo:
        lon, lat = map(float, geo.replace("Point(", "").replace(")", "").split())
    data.append({
        "item": result['item']['value'],
        "itemLabel": result['itemLabel']['value'],
        "site": result['site']['value'],
        "siteLabel": result['siteLabel']['value'],
        "county": result['county']['value'],
        "countyLabel": result['countyLabel']['value'],
        "latitude": lat,
        "longitude": lon,
    })

df = pd.DataFrame(data)
df
```

WIKIDATA

https://research-squirrel-engineers.github.io/jupyter-nb-lod/wikidata-ogham-sites-map

	item	itemLabel	site	siteLabel	county	countyLabel	latitude	longitude
0	http://www.wikidata.org/entity/Q70892459	CIIC 71 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85394002	Ahalisky (Ogham Site)	http://www.wikidata.org/entity/Q162475	County Cork	51.678889	-8.846944
1	http://www.wikidata.org/entity/Q70892460	CIIC 72 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85394004	Aultagh (Ogham Site)	http://www.wikidata.org/entity/Q162475	County Cork	51.770833	-9.088056
2	http://www.wikidata.org/entity/Q70892463	CIIC 73 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393926	Carhoovauler (Ogham Site)	http://www.wikidata.org/entity/Q162475	County Cork	51.688333	-8.956111
3	http://www.wikidata.org/entity/Q70892466	CIIC 74 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393926	Carhoovauler (Ogham Site)	http://www.wikidata.org/entity/Q162475	County Cork	51.688333	-8.956111
4	http://www.wikidata.org/entity/Q70892468	CIIC 75 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393928	Keenrath (Ogham Site)	http://www.wikidata.org/entity/Q162475	County Cork	51.759444	-9.185833
...	...	...	...	...	...	...	...	...
328	http://www.wikidata.org/entity/Q70892620	CIIC 156 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393964	Ballintaggart (Ogham Site)	http://www.wikidata.org/entity/Q184469	County Kerry	52.172500	-10.031111
329	http://www.wikidata.org/entity/Q70892623	CIIC 157 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393964	Ballintaggart (Ogham Site)	http://www.wikidata.org/entity/Q184469	County Kerry	52.172500	-10.031111
330	http://www.wikidata.org/entity/Q70892626	CIIC 158 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393964	Ballintaggart (Ogham Site)	http://www.wikidata.org/entity/Q184469	County Kerry	52.172500	-10.031111
331	http://www.wikidata.org/entity/Q70892629	CIIC 159 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393964	Ballintaggart (Ogham Site)	http://www.wikidata.org/entity/Q184469	County Kerry	52.172500	-10.031111
332	http://www.wikidata.org/entity/Q70892630	CIIC 160 (Ogham Stone Concept by RAS Macalister)	http://www.wikidata.org/entity/Q85393964	Ballintaggart (Ogham Site)	http://www.wikidata.org/entity/Q184469	County Kerry	52.172500	-10.031111

333 rows × 8 columns

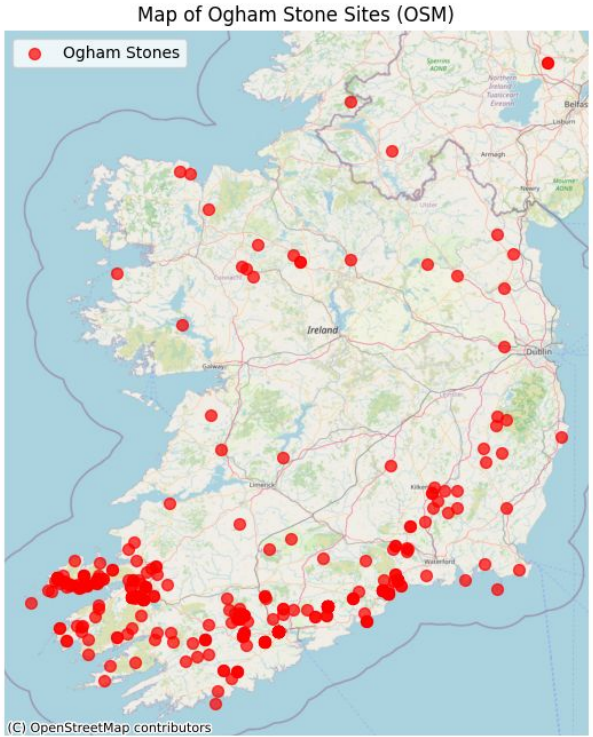
SPARQL query to create DataFrame

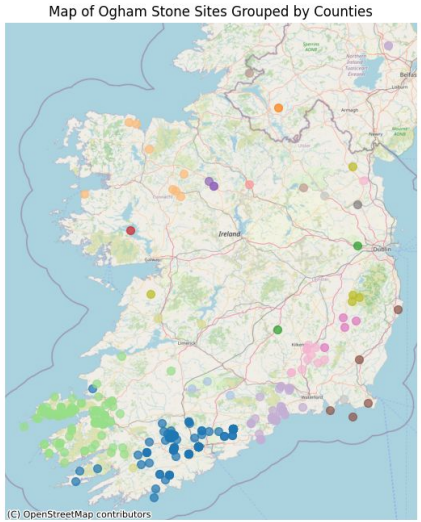
Visualise the Data as Maps

```python
# Check if DataFrame is populated
if df.empty:
    print("No data retrieved from the query.")
else:
    # Filter rows with valid coordinates
    df_with_coords = df.dropna(subset=['latitude', 'longitude'])

    # Create a GeoDataFrame
    gdf = gpd.GeoDataFrame(
        df_with_coords,
        geometry=[Point(xy) for xy in zip(df_with_coords['longitude'], df_with_coords['latitude'])],
        crs="EPSG:4326"
    )

    # Convert to Web Mercator for OSM basemap
    gdf_mercator = gdf.to_crs(epsg=3857)

    # Load Ireland boundary from GeoJSON
    ireland_boundary = gpd.read_file(geojson_file)
    ireland_boundary = ireland_boundary.to_crs(epsg=3857)

    # Map 1: Plot points without text decorations
    fig, ax = plt.subplots(figsize=(12, 8))
    gdf_mercator.plot(ax=ax, color='red', markersize=50, alpha=0.7, label="Ogham Stones")
    ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik, zoom=8)
    ax.set_axis_off()
    plt.title("Map of Ogham Stone Sites (OSM)")
    plt.legend()
    plt.show()
```

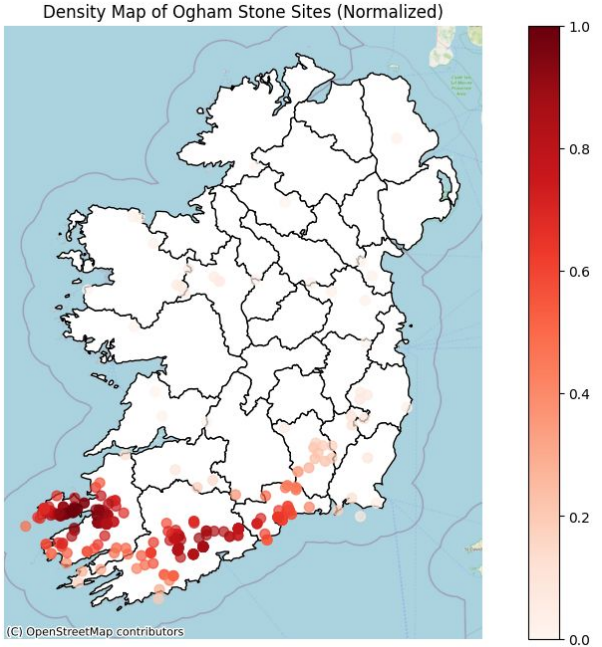
```python
# Map 2: Plot with points colored by county and fix legend
fig, ax = plt.subplots(figsize=(12, 8))
unique_counties = gdf['countyLabel'].unique()
colors = plt.cm.tab20.colors[:len(unique_counties)]  # Generate unique colors
county_colors = {county: colors[idx] for idx, county in enumerate(unique_counties)}

patches = []
for county, color in county_colors.items():
    county_data = gdf_mercator[gdf_mercator['countyLabel'] == county]
    county_data.plot(ax=ax, color=color, markersize=50, alpha=0.7)
    patches.append(Patch(color=color, label=county))  # Add patch for legend

ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik, zoom=8)
ax.set_axis_off()
plt.title("Map of Ogham Stone Sites Grouped by Counties")
plt.legend(handles=patches, title="Counties", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()

# Map 3: Density map with normalized values
x = gdf_mercator.geometry.x
y = gdf_mercator.geometry.y

# Calculate kernel density
xy = np.vstack([x, y])
kde = gaussian_kde(xy)
density = kde(xy)

# Normalize density to range [0, 1]
density_normalized = (density - density.min()) / (density.max() - density.min())
gdf_mercator['density'] = density_normalized

fig, ax = plt.subplots(figsize=(12, 8))
ireland_boundary.plot(ax=ax, color="white", edgecolor="black")
gdf_mercator.plot(ax=ax, column='density', cmap="Reds", markersize=50, alpha=0.7, legend=True)
ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik, zoom=8)
ax.set_axis_off()
plt.title("Density Map of Ogham Stone Sites (Normalized)")
plt.show()
```

Create Maps


Map of Ogham Stone Sites (OSM)
Ogham Stones
Ireland
Dublin
(C) OpenStreetMap contributors


Map of Ogham Stone Sites Grouped by Counties
Ireland
(C) OpenStreetMap contributors


Counties
County Cork
County Limerick
County Donegal
County Mayo
County Kildare
County Kerry
County Galway
County Leitrim
County Roscommon
County Waterford
County Wexford
County Cavan
County Carlow
County Kilkenny
County Louth
County Meath
County Wicklow
County Clare


Density Map of Ogham Stone Sites (Normalized)
(C) OpenStreetMap contributors
1.0
0.8
0.6
0.4
0.2
0.0

Output

https://research-squirrel-engineers.github.io/jupyter-nb-lod/n4okg-ogham-sites-county

Define SPARQL query service

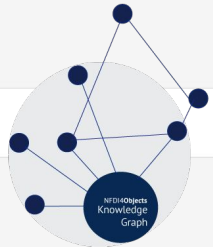
```
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import matplotlib.pyplot as plt

def querySparql(query):
    sparql = SPARQLWrapper("https://graph.nfdi4objects.net/api/sparql")
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
    results = sparql.queryAndConvert()
    return results['results']['bindings']
```

Define the SPARQL Query

```
# SPARQL Query for Samian Ware Kiln Sites
oghamQuery = """
PREFIX oghamonto: <http://ontology.ogham.link/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?county (count(distinct ?stone) as ?count) WHERE {
 ?item <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://ontology.ogham.link/OghamSite> .
 ?item oghamonto:within ?c .
 ?c a oghamonto:County .
 ?c rdfs:label ?county .
 ?stone oghamonto:disclosedAt ?item .
 ?stone a oghamonto:OghamStone_CIIC .
} GROUP BY ?county ORDER BY DESC(?count)
"""
```

Fetch Data and Convert to DataFrame

```
# Fetch data using the SPARQL query
sparql_results = querySparql(oghamQuery)

# Convert SPARQL JSON results into a DataFrame
data = []
for result in sparql_results:
    data.append({
        "county": result.get("county", {}).get("value", None),
        "count": int(result.get("count", {}).get("value", 0))
    })

df = pd.DataFrame(data)
df
```



	county	count
0	Kerry	127
1	Cork	81
2	Waterford	47
3	Kilkenny	12
4	Kildare	8
5	Mayo	8
6	Wexford	5
7	Wicklow	5
8	Carlow	4
9	Clare	3
10	Limerick	3
11	Roscommon	3
12	Antrim	2
13	Cavan	2
14	Meath	2
15	Tipperary	2

SPARQL query to create DataFrame

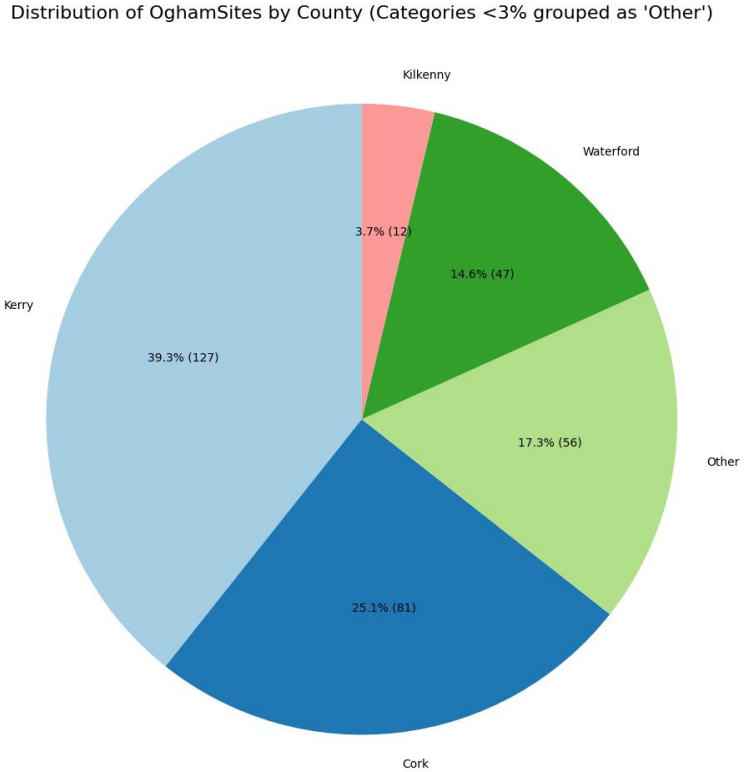
Pie Chart

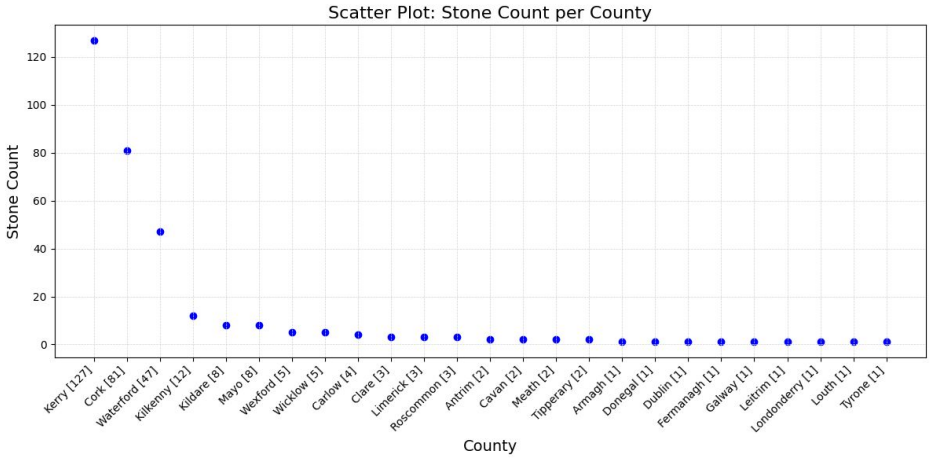
```
# Calculate the total number of stones.
total_count = df["count"].sum()

# Compute the percentage for each county.
df["percentage"] = df["count"] / total_count * 100

# Separate counties with a percentage >= 3% from those with less than 3%.
major_df = df[df["percentage"] >= 3].copy()
minor_df = df[df["percentage"] < 3].copy()

# If there are counties with less than 3%, combine them into an 'Other' category.
if not minor_df.empty:
    other_count = minor_df["count"].sum()
    major_df = pd.concat([major_df, pd.DataFrame([{"county": "Other", "count": other_count}])], ignore_index=True)

# Optional: Sort the categories by count in descending order.
major_df = major_df.sort_values("count", ascending=False)

# Define a function to format the autopct text to include both the percentage and the stone count.
def make_autopct(values):
    def my_autopct(pct):
        total = sum(values)
        count = int(round(pct * total / 100.0))
        return '{:.1f}% ({:d})'.format(pct, count)
    return my_autopct

# Create a pie chart displaying the distribution of OghamSites by county.
plt.figure(figsize=(10, 10))
major_df.set_index("county")["count"].plot(
    kind="pie",
    autopct=make_autopct(major_df["count"]),
    startangle=90,
    colors=plt.cm.Paired.colors
)
plt.title("Distribution of OghamSites by County (Categories <3% grouped as 'Other')", fontsize=16)
plt.ylabel("")  # Remove the default y-label.
plt.tight_layout()
plt.show()
```

Scatter Plot

```
# Create a scatter plot with county on the x-axis and stone count on the y-axis.
plt.figure(figsize=(12, 6))
x_positions = range(len(df))
plt.scatter(x_positions, df["count"], color="blue")
plt.title("Scatter Plot: Stone Count per County", fontsize=16)
plt.xlabel("County", fontsize=14)
plt.ylabel("Stone Count", fontsize=14)

# Create customised x-axis labels with county and count in square brackets.
xtick_labels = [f"{county} [{count}]" for county, count in zip(df["county"], df["count"])]
plt.xticks(x_positions, xtick_labels, rotation=45, ha="right")

# Add a light grey grid.
plt.grid(color="lightgrey", linestyle="--", linewidth=0.5)

plt.tight_layout()
plt.show()
```

Create Charts


Distribution of OghamSites by County (Categories <3% grouped as 'Other')
Kilkenny
Waterford
Kerry
Other
Cork
3.7% (12)
14.6% (47)
39.3% (127)
17.3% (56)
25.1% (81)

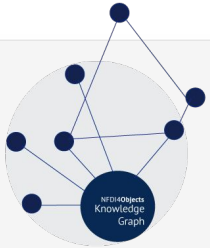

Scatter Plot: Stone Count per County
Stone Count
County
0
20
40
60
80
100
120
Kerry [127]
Cork [81]
Waterford [47]
Kilkenny [12]
Kildare [8]
Mayo [8]
Wexford [5]
Wicklow [5]
Carlow [4]
Clare [3]
Limerick [3]
Roscommon [3]
Antrim [2]
Cavan [2]
Meath [2]
Tipperary [2]
Armagh [1]
Donegal [1]
Dublin [1]
Fermanagh [1]
Galway [1]
Leitrim [1]
Londonderry [1]
Louth [1]
Tyrone [1]

Output

https://research-squirrel-engineers.github.io/jupyter-nb-lod/n4okg-ogham-sites-maps

Define SPARQL query service

```
import os
from SPARQLWrapper import SPARQLWrapper, JSON
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
import contextily as ctx  # For adding OpenStreetMap basemaps
from matplotlib.patches import Patch
from scipy.stats import gaussian_kde
import numpy as np

def querySparql(query):
    sparql = SPARQLWrapper("https://graph.nfdi4objects.net/api/sparql")
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
    results = sparql.queryAndConvert()
    return results['results']['bindings']

# Define the GeoJSON file path
geojson_file = os.path.join(os.getcwd(), "gs_ireland_island.geojson")  # Adjusted for Jupyter Notebook
```



Define the SPARQL Query

```
# SPARQL Query
oghamQuery = """
PREFIX oghamonto: <http://ontology.ogham.link/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?item ?label ?geo ?county (count(distinct ?stone) as ?count) WHERE {
 ?item <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://ontology.ogham.link/OghamSite> .
 ?item rdfs:label ?label .
 ?item <http://www.opengis.net/ont/geosparql#hasGeometry> ?item_geom .
 ?item_geom <http://www.opengis.net/ont/geosparql#asWKT> ?geo .
 ?item oghamonto:within ?c .
 ?c a oghamonto:County .
 ?c rdfs:label ?county .
 ?stone oghamonto:disclosedAt ?item .
 ?stone a oghamonto:OghamStone_CIIC .
} GROUP BY ?item ?label ?geo ?county ORDER BY DESC(?count)
"""
```

Fetch Data and Convert to DataFrame

```
# Fetch data using the SPARQL query
sparql_results = querySparql(oghamQuery)

# Convert SPARQL JSON results into a DataFrame
data = []
for result in sparql_results:
    geo = result['geo']['value'] if 'geo' in result else None
    lat, lon = (None, None)
    if geo:
        lon, lat = map(float, geo.replace("POINT(", "").replace(")", "").split())
    data.append({
        "item": result['item']['value'],
        "label": result['label']['value'],
        "county": result['county']['value'],
        "count": int(result.get("count", {}).get("value", 0)),
        "latitude": lat,
        "longitude": lon,
    })

df = pd.DataFrame(data)
df
```

	item	label	county	count	latitude	longitude
0	http://lod.ogham.link/data/OS40000031	Ballyknock (Ogham Site)	Cork	15	52.026944	-8.071111
1	http://lod.ogham.link/data/OS40000002	Drumlohan (Ogham Site)	Waterford	10	52.163056	-7.468056
2	http://lod.ogham.link/data/OS40000020	Ballintaggart (Ogham Site)	Kerry	9	52.172500	-10.031111
3	http://lod.ogham.link/data/OS40000149	Kilcoolaght East / Kilhullicaha (Ogham Site)	Kerry	8	52.071944	-9.747222
4	http://lod.ogham.link/data/OS40000170	Knockboy (Ogham Site)	Waterford	8	52.111389	-7.600000
...	...	...	...	...	...	...
191	http://lod.ogham.link/data/OS40000062	Castletimon (Ogham Site)	Wicklow	1	52.909167	-6.063611
192	http://lod.ogham.link/data/OS40000168	Knickeen (Ogham Site)	Wicklow	1	52.998056	-6.535000
193	http://lod.ogham.link/data/OS40000045	Boleycarrigeen (Ogham Site)	Wicklow	1	52.947012	-6.608898
194	http://lod.ogham.link/data/OS40000098	Donard (Ogham Site)	Wicklow	1	53.016944	-6.617778
195	http://lod.ogham.link/data/OS40000040	Baltinglass (Ogham Site)	Wicklow	1	52.966389	-6.624444

196 rows × 6 columns

SPARQL query to create DataFrame

Visualise the Data as Maps

```
# Filter rows with valid coordinates
df_with_coords = df.dropna(subset=['latitude', 'longitude'])

# Create a GeoDataFrame
gdf = gpd.GeoDataFrame(
    df_with_coords,
    geometry=[Point(xy) for xy in zip(df_with_coords['longitude'], df_with_coords['latitude'])],
    crs="EPSG:4326"
)

# Convert to Web Mercator for OSM basemap
gdf_mercator = gdf.to_crs(epsg=3857)

# Load Ireland boundary from GeoJSON
ireland_boundary = gpd.read_file(geojson_file)
ireland_boundary = ireland_boundary.to_crs(epsg=3857)
```

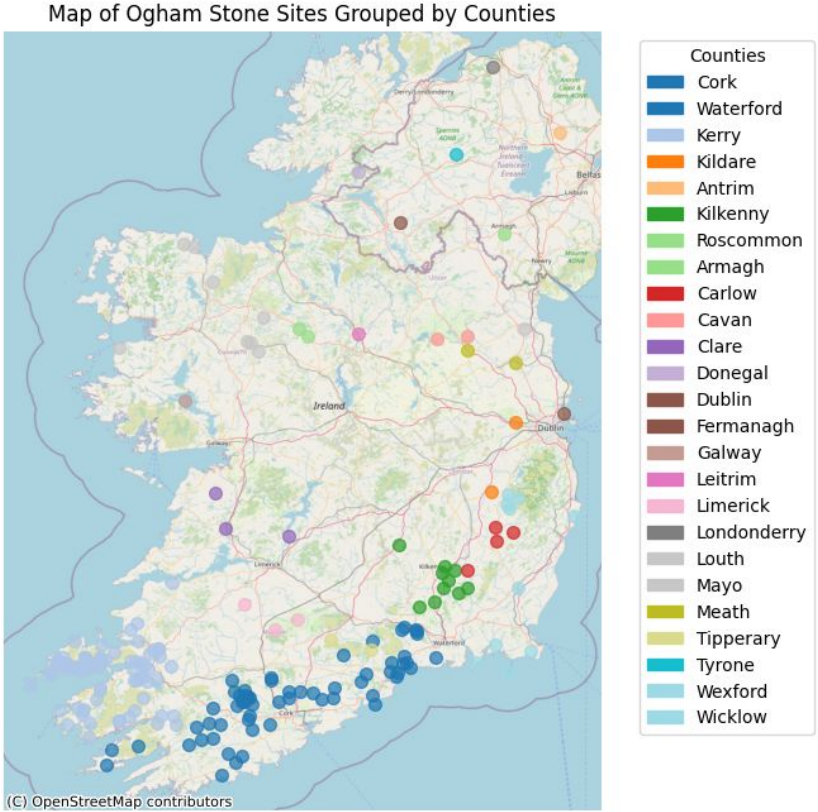
Map 1: Plot with points coloured by county

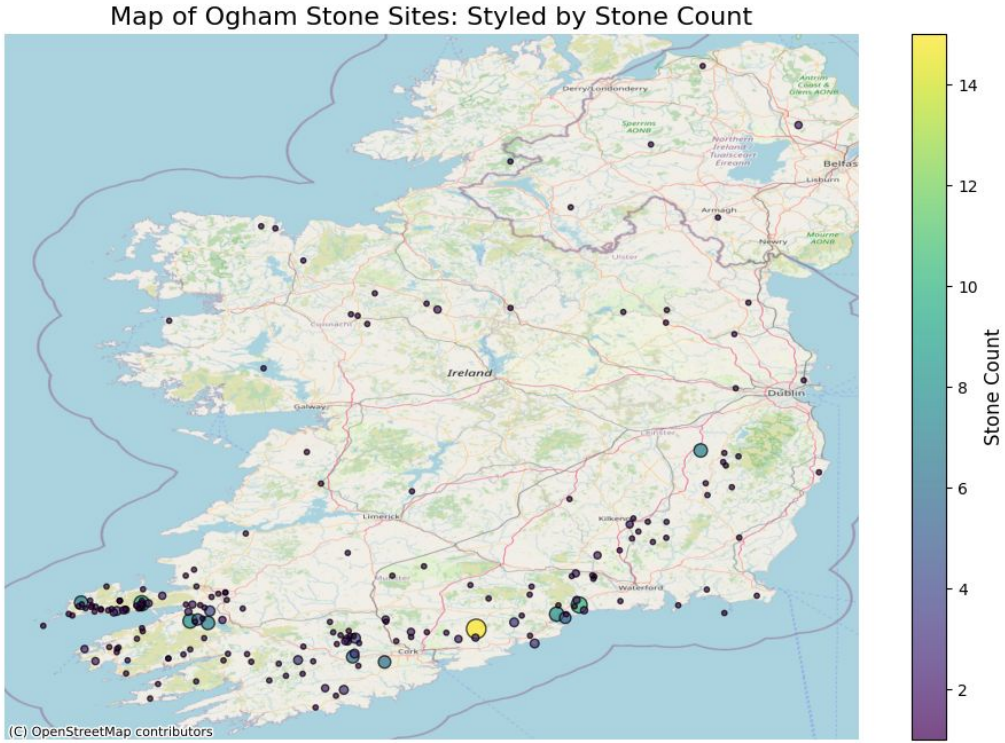
```
# Map 1: Plot with points coloured by county
fig, ax = plt.subplots(figsize=(12, 8))
unique_counties = gdf['county'].unique()
# Create a colormap with as many colours as unique counties
cmap = plt.get_cmap('tab20', len(unique_counties))
county_colors = {county: cmap(idx) for idx, county in enumerate(unique_counties)}

patches = []
for county, color in county_colors.items():
    county_data = gdf_mercator[gdf_mercator['county'] == county]
    county_data.plot(ax=ax, color=color, markersize=50, alpha=0.7)
    patches.append(Patch(color=color, label=county))  # Add patch for legend

ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik, zoom=8)
ax.set_axis_off()
plt.title("Map of Ogham Stone Sites Grouped by Counties")
plt.legend(handles=patches, title="Counties", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

Map 2: Plot with point colours and sizes grouped/styled by stone count

```
# Map 2: Plot with point colours and sizes grouped/styled by stone count.
fig, ax = plt.subplots(figsize=(12, 8))
# Extract x and y coordinates from the GeoDataFrame.
x = gdf_mercator.geometry.x
y = gdf_mercator.geometry.y

# Define a scaling factor for point sizes.
size_factor = 10
sizes = gdf_mercator["count"] * size_factor

# Plot the points using a continuous colormap (e.g. 'viridis') based on the stone count.
sc = ax.scatter(x, y, s=sizes, c=gdf_mercator["count"], cmap="viridis", alpha=0.7, edgecolor="k")
ctx.add_basemap(ax, source=ctx.providers.OpenStreetMap.Mapnik, zoom=8)
ax.set_axis_off()
plt.title("Map of Ogham Stone Sites: Styled by Stone Count", fontsize=16)

# Add a colour bar with label.
cbar = plt.colorbar(sc, ax=ax)
cbar.set_label("Stone Count", fontsize=12)
plt.show()
```

Create Maps


Map of Ogham Stone Sites Grouped by Counties
Counties
Cork
Waterford
Kerry
Kildare
Antrim
Kilkenny
Roscommon
Armagh
Carlow
Cavan
Clare
Donegal
Dublin
Fermanagh
Galway
Leitrim
Limerick
Londonderry
Louth
Mayo
Meath
Tipperary
Tyrone
Wexford
Wicklow
Ireland
(C) OpenStreetMap contributors


Map of Ogham Stone Sites: Styled by Stone Count
Stone Count
14
12
10
8
6
4
2
Ireland
(C) OpenStreetMap contributors

Output

Finis!
Thx!

Questions?

Florian Thiery M.Sc.
Research Squirrel Engineers Network
mail@fthiery.de
ORCID: 0000-0002-3246-3531

DOI 10.5281/zenodo.14920034