So, in this webinar, we are going to use Adzuna datasets. I'll briefly show you all the information. So, Adzuna is a job aggregator website which has about 95 percent of job listing coverage in the UK.
So, it’s really easy to use.

You just search by job, company, or location and you get an aggregation from different sites listed there.

So, there are two datasets.

One is open, collected using an Adzuna API.

I added all the links in the presentation so you can follow along as well.

So, if you wish to collect open Adzuna datasets,
you can register for an API,

receive the key, and just collect the data yourself.

And this data, it's free and open data.

To reduce the size of it, it has been subselected to location_area_1

containing Scotland value.

It's only sample data,

mainly because of its size.

But I'd also like to introduce a licensed dataset available
for academic use at the Urban Big Data Centre website.

So, click. So, this dataset is cleaned and duplicated.

The only thing is you would need to apply for data to access it.

So, the difference between these two data is one is open,

it's messy, let's say,

due to its nature of scraping,

but there is work that could be done to derive insights.
And the reason why this data is available for academic use, it has been deduplicated, cleaned up, and nicely formatted.

The other thing is, in the data, because the description field containing information about the job ad, and in open data, it's truncated to 500 characters. So, quite often it's only a fraction of the job ad itself.

Right. In this lab or training,
we are going to use GeoPandas,

which is an open source project

which extends Pandas' frame as like a series of data frames

to GeoSeries and GeoDataFrame.

And the main difference between those two is that

GeoDataFrame has a special column, geometry.

Again, I've added the link for documentation

if you want to explore further.
So, let's start to open the data.

And I'm going to run this session alongside with you.

So, on this step, we are going to import Pandas to open our dataset,

add the data provided, really the first data provided,

using parquet format,

which is an open source file format which is designed to efficiently store the information.
I have included more information about how it works, its specifications, and other things.

So, in this lab, we are using it because of the size of the data.

So, it was just more convenient to pack it in a parquet and share it with you.

So, I'm specifying my data source and I'm going to open it.
Right. Let's do it like this.

My source and load it to Pandas GeoDataFrame.

Let's have a look at the data frame.

So, this is what our Adzuna data looks like.

It does have information about

the job title, its category and tag.

This is, by the way, created by Adzuna.
There is some categorisation of the data available in open data.

The dataset has information about contract type, well, in some cases.

Information about minimum salary.

About description.

As I mentioned before, it's truncated to 500 characters.

The date it was created and location information.

So, in this lab, we are going to use longitude and latitude
to make this data spatial.

So, before we do any kind of GIS,

analysis of GIS, we are just going to look at the data and try to import it.

So, in this lab,

on this step, I'm going to use Seaborn library for visualisation. You can see why shortly.

It has very nice jointplot, type of visualisation
to really showcase geographical data.

And I'm going to use contextily,

it's a library which provides background mapping.

So, on this line, I specify that I want to create this type of plot,

or the jointplot,

where X would be longitude and Y latitude from my data frame.

By default, I think, the colour is blue

so I changed it to red to make the data more visible.
On this line, I'm adding basemap from contextily

and I'm saying that I'm adding it on the same plot.

I'm specifying CRS.

So, I'm going to talk about CRS,

which stands for Coordinate Reference System

in more detail later.

And I'm specifying the source.
Source is the type of background mapping I would like to see.

Again, I'll talk more about it later and provide a few examples of what it could be.

There, alpha is set to 0.4 and alpha is transparency. Again, on these lines I've made plot slightly bigger than it is by default so you can see the data better.

So, if you're familiar with UK geography, you know that this area in red is Scotland.
And several job ads, well, they have these kind of outliers,

let's call them outliers,

they do have Scotland as value and location

but their coordinates are set to somewhere else

so we'll remove them later.

And if you see here,

our jointplot shows the maximum distribution of that.

And if you create intersection, you would see that it's Edinburgh
and Glasgow,

and most popular Third location is here in Aberdeenshire.

Right. But it's important to say that,

at this stage, this data is not spatial data yet.

To make it spatial,

we are going to convert the data frame into GeoDataFrame

so we are able to perform geospatial analysis.

We are going to import GeoPandas
and the convention is to shorten the library names

that we are going to import Pandas as GPD.

Again, GeoPandas is a project which allows us to work with geospatial data

and Python environment.

On the background, it uses Shapely library.

I've added documentation here.

So, if you click the link, you can read about it later.
And it uses matplotlib for plotting.

So, let's import matplotlib as we are going to use it for plotting GeoPandas.

And in this session, we are going to use Point and Polygon so I'm importing them as well.

So, now we need to create our GeoDataFrame based on available X and Y.

And the convention, really,
the command is GeoPandas creates points from XY,

which kind of says everything for itself.

So, let's do this.

So, on this step, I'm creating a GeoDataFrame and I'm calling it GDF.

I'm saying GeoPandas data frame for command using my DF, data frame,

the job ads data loaded on a previous step.
And for geometry column,

we are going to use a longitude column and a latitude column

and create points based on them.

So, let's see our data.

So, we have exactly the same data as before.

If you noticed on the previous steps,

when we looked at the data we had only 23 columns

and now we have one extra column called geometry
which is a Point

and it has information similar to what we have here.

So, now we have a spatial dataset.

Let's check our Coordinate Reference system.

We could have specified it on a previous step.

If it's not specified, by default it's set to none.

And in this section, I'm going to talk very, very briefly about
Map projections and Coordinate Reference System

For those who are not familiar with this kind of concept, which is very, very crucial in the GIS world.

Map projection, in very simple terms, tries to transform the Earth from its spherical shape to a flat, planar shape or flat map.

And Coordinate Reference System, CRS for short, defines how this two-dimensional space
projected on a map in your geographical system

relates to real places on Earth.

So, the decision of which map projection and Coordinate Reference System to use depends on the region and country.

I've included a source link where you can read more.

But please bear in mind it's very, very important for GIS.

So, I have included a few visual examples just to showcase it.
Depending on the choice, you could have different distortions.

In this case, it's distortion by area.

And this is another example just to illustrate how the shape changes.

But it could also change distances and angles depending on the projection.

But what you see on the screen would change based on what you specified.

Again, there is a source to check
and maybe learn a bit more about it on your own.

So, now we are going to set coordinates to WGS84 using the set_crs function, which takes EPSG codes, which stands for European Petroleum Survey Group, and value of the system.

So, for WGS84, the value of EPSG is 4326.
And I'm printing it as well just to check that it worked.

I've also included for you several quite useful links, I think.

So, one is Spatial Reference where you can find many, many, many codes and references.

coordinate references. So, we'll just open one.

So, it does contain Bounds or a link and a Google.

And the other one is also quite useful.
Let's just open it.

It does also have visual representations of what you would expect your data to be like after you project it.

And also, it does contain transformational values.

Let's move on.

So, now we've set up our coordinate reference system and we have our geospatial data,
let's do some visualisation.

So, we are going to use matplotlib

and the function dot plot.

And by default,

the map in GeoPandas is created based on the geometry column.

Let's run it.

So, it might take a while.

What I'm saying is I want to plot my GeoDataFrame,
I want my dots to be red,

and I want it to show.

So, if you see here, it's a small plot

but you can see the outline of Scotland here.

Similar to what we've seen

when we were visualising data using just X and Y and Seaborn.

On this step, I would like to show how to open different file formats of
spatial data in Pandas.

So, in this step, we are going to load the boundary of Scotland using our open and free Ordinance Survey data BoundaryLine,

which is available for download for everybody.

And it is a collection of data available.

Trying to find the description.
boundaries of civil parishes, wards, communities,

electoral divisions.

So, it does contain a lot of information

and it's available for download.

So, to read the file,

you just need to specify GeoPandas, read file,

and what it does, the library reads the data

and it returns it as a GeoDataFrame object.
Run it here.

So, in mine, I used GeoDataFrame OS data and I'm saying read file, which is saved in the data folder.

So, it's district_borough_unitary shape.

So, the full dataset provided is GeoPackage with many layers so that it's packaged to reduce the size and make it easier for you to work with the data.
I preloaded it into a shapefile.

If you decide to load the full dataset,

please command on this line and use this.

So, very similar, we are reading the file

and for that, set it to GeoPackage

and its name and specify the way we need it to be loaded

so it would provide the same result.
Let's check the Coordinate Reference System.

00:15:23,145 --> 00:15:27,154
So, this data is projected dataset to the British National Grid,

00:15:27,254 --> 00:15:30,302
EPSG code 27770.

00:15:32,314 --> 00:15:35,712
Right. Let's visualise our datasets together.

00:15:35,812 --> 00:15:40,220
So, I'm going to load my OS data

00:15:40,320 --> 00:15:43,387
and my Adzuna and job listings together.

00:15:44,529 --> 00:15:47,133
It might take some time.

00:15:48,814 --> 00:15:51,789
Again, I specified my Adzuna data to be red.
And if you see it, it doesn't look right.

So, we have our UK and our Adzuna data here,

over here, but we would expect it there.

This is one of the typical and common problems which could occur when you use GeoPandas.

It is mismatch of projection.

Do you remember our GeoDataFrame was WGS84?

And OS data is British National Grid.
So, if you visualise your data and it's something like this,

the first thing would be to check the projection.

So, on this step, let's make the plot bigger

and fix the projection issue using to_crs Scotland.

So, I'm going to run it.

So, on this line, I'm setting my plot to a bigger size

and I'm going to reproject my both datasets to Web Mercator

to align it with the background map.
my OS data using command to_crs,

specifying a new Coordinate Reference System.

Similarly, on this line, I'm doing the same with my Adzuna data from the GeoDataFrame and I'm setting the colour as red

so we can clearly see it.

And on this line, I'm adding basemap.

So, now our visualisation is better.
So, data aligned.

286
00:17:26,511 --> 00:17:31,730
And what I want to talk about next is

287
00:17:31,830 --> 00:17:33,488
contextily basemaps.

288
00:17:34,305 --> 00:17:40,767
So, there are different ways you can use reprojection

289
00:17:40,867 --> 00:17:42,455
and align with this basemap.

290
00:17:42,555 --> 00:17:44,283
So, just a reminder that,

291
00:17:44,590 --> 00:17:47,894
on this step, we projected both layers

292
00:17:48,382 --> 00:17:51,341
and didn't do any reprojection of the basemap file.
It could have been done in different ways.

So, we could have reprojected only one layer to match the Coordinate Reference System of another one and then specify the same EPSG code in our basemap.

And it's the output we would expect.

So, the data is, it's the same location but it is visualised in a slightly different way.
What I mean by this is, try to compare this picture and this.

So, the data is the same, just a different Coordinate Reference System and a slightly different visualisation.

So, now I'd like to show you available backgrounds.

So, if you specify contextily providing this,

you would get a list of all available background map providers.

And then, if you add contextily providers
and instead of keyname add something like "HERE"

and said dot keys,

you would get a list of these background maps and visualisations of different styles available from these providers.

So, in this example for provider HERE,

we have this many different styles
available for you to use.

317
00:19:14,687 --> 00:19:16,707
Right. Spatial Join.

318
00:19:16,997 --> 00:19:22,076
This is quite a popular operation in GIS.

319
00:19:22,366 --> 00:19:26,724
I've added a link for how it works in GeoPandas.

320
00:19:26,954 --> 00:19:28,723
So, here it is, spatial join.

321
00:19:29,084 --> 00:19:30,904
It's different arguments.

322
00:19:31,744 --> 00:19:35,523
Again, it's something for you to have.

323
00:19:38,839 --> 00:19:42,378
As always, let's look at the data.
And on this step, we are going to remove outliers but outside of Scotland and limit it to make it smaller and faster.

We're going to remove everything outside of Glasgow.

So, just a quick recap.

So, our GeoDataFrame with Adzuna listings contains 24 columns now.

23 of which are the Adzuna data itself
and one is the geometry that we created.

I'm going to subselect my Ordinance Survey data only to have Glasgow.

Sorry, Glasgow.

That's set and I'm going to go with Glasgow.

Will it just quickly work?

What it looks like. Again, if you're familiar,
340
00:20:36,355 --> 00:20:37,822
Just a visual check.

341
00:20:38,802 --> 00:20:41,982
Let's check what Coordinate Reference System it is in.

342
00:20:42,201 --> 00:20:45,459
So, it is in Web Mercator.

343
00:20:45,827 --> 00:20:47,636
If you remember, on the previous step,

344
00:20:47,823 --> 00:20:50,218
we reprojected OS data

345
00:20:51,189 --> 00:20:56,725
and Glasgow GeoDataFrame inherited it.

346
00:20:57,885 --> 00:21:00,154
Right. Let's make spatial join.

347
00:21:01,004 --> 00:21:06,001
So, to make a join, we need to specify our GeoDataFrame

we'd like to join.

And in this example, we are going to select

everything which falls within.

And, if you can see, we've got an error.

What do you think caused an error?

Sorry.

So, if you read here the user warning,
it says it's a mismatch between left and right geometry.

And it says that our left GeoDataFrame is in WGS84, 4326,

and our Glasgow data frame has the EPSG code 3857, which is Web Mercator.

So, because there are different projections,

GeoPandas cannot figure it out and throw an exception.

So, what we need to do is to reproject one of the layers to match the projection of another.
Let's run it.

It might take some time if you run it on your computer.

So, again, to remind you,

we are going to select all the job ads within Glasgow.

Let's have a look at what we've got.

We're going to spatially join

our Adzuna dataset and Glasgow boundary.

Right.
I'm going to check the length of my join.

So, now it's quite a small and manageable dataset.

So, I'm going to remember this number because I'm going to use it for something else as well.

Right. Let's check what we have inside.

And the first thing you would notice is now we have 41 columns.
our Adzuna dataset has been spatially joined with Glasgow

00:22:54,926 -- 00:22:58,132
and all the records are saved information

00:22:58,506 -- 00:23:01,455
from the data layer related to this location.

00:23:01,743 -- 00:23:05,133
In this case, it's only information about

00:23:05,792 -- 00:23:06,871
our Glasgow...

00:23:08,319 -- 00:23:10,688
Sorry. The same information for all the points

00:23:10,788 -- 00:23:13,427
but it could be used in a more powerful way,

00:23:13,527 -- 00:23:14,914
which I'll show you later.
But an important thing to note and to take from it is that by spatially joining two of our GeoDataFrames,

we receive information from both of them.

In this step, I really wanted to show that you can use your GeoDataFrame in a similar way as your data frame and you can use typical functions. So, here, for instance, I used a group by column to find
the most paid categories

by finding the average minimum salary and then sorting values by the salary.

So, in this step, we just did a summary of our GeoSpatial data frame of geographical data and received some statistics.

Right. Let's quickly visualise data.
And I would like to introduce you to, in this step, to geoplot,

which is a fantastic library for geospatial data visualisation.

I will quickly show you the gallery, just to introduce

the different visualisation styles and techniques available from here.

Again, please have a look when you have some time.

And, as always, we need to import this library.

And here I’m going to use pointplot,
where my colour would depend on the value in the salary min column.

So, the data parameter is just legend true to have the legend here.

Edge colour is set to light grey but you can actually see it

so it's edge of the point.

And its line width.

So, we do have visualisation

but clearly most of the values fell in this region

and we can't actually see a difference between
different areas of Glasgow in terms of salary min.

To fix this or to enhance it,

I'm going to use mapclassify.

Mapclassify is a library which provides classification schemes

there are different parameters.

You will see.

So, I'm going to import this mapclassify.

The library is mc.
And I'm going to use scheme Quantiles

and my salary min column from join.

Now, as I've said before,

this part is the same.

The only change I made in here is
to specify the scheme I'd like to use.

So, the legend changed.

But now you can see that the data has been reclassified
and now we can see the distribution based on the salary min column.

Again, there are many schemes and classification types available.

It's a very, very powerful to create nice looking visualisations.

Right.

The other geospatial function I'd like to introduce is clip.

GeoPandas does clip to what you would expect to get
So, one layer is clipped by another

00:27:01,143 --> 00:27:03,383

based on the geometries.

00:27:03,562 --> 00:27:05,931

Again, very important, both layers must be

00:27:06,031 --> 00:27:08,099

the same Coordinate Reference System.

00:27:11,890 --> 00:27:15,590

So, for that reason, let's check what we have now.

00:27:15,690 --> 00:27:19,888

So, one of our layers is in Web Mercator

00:27:19,988 --> 00:27:23,056

and the other one is in WGS84.

00:27:25,317 --> 00:27:30,025

So, let's reproject both layers to Web Mercator.
00:27:32,031 --> 00:27:34,801
Actually, now I'm thinking we do not need to reproject one of them.

00:27:35,270 --> 00:27:37,089
We need to reproject only one of them.

00:27:39,299 --> 00:27:41,225
And let's do the clip.

00:27:41,402 --> 00:27:44,641
So, we are going to clip

00:27:44,741 --> 00:27:47,680
our Adzuna job listings stored in GDF

00:27:47,780 --> 00:27:49,389
by the Glasgow boundary.

00:27:49,786 --> 00:27:52,406
And, again, remove everything else outside.

00:27:52,694 --> 00:27:56,726
So, let's check the length of our dataset.
So, if you remember, it is exactly the same number as we had when we were joining the data.

The length looked quite similar.

Let's look at the clipped GeoDataFrame.

And, again, there are only 24 columns.

So, the way clip works is it subselects all the features.
So, what we have got here, we've got our Adzuna job ads subselected or clipped only to be within Glasgow.

So, no extra additional information is added when using clip.

Now, let's plot the data.
I'm plotting my clipped GeoDataFrame,

00:28:53,025 -- 00:28:54,163
my Glasgow GeoDataFrame,

00:28:54,263 -- 00:28:58,940
set in some transparency so we can see both datasets.

00:28:59,040 -- 00:29:01,588
And I'm adding my basemap.

00:29:01,993 -- 00:29:04,511
Now, I'm using OpenStreetMap Mapnik.

00:29:04,864 -- 00:29:06,306
This is what we've got.

00:29:07,879 -- 00:29:12,281
If you can see, no point is outside of the Glasgow boundary.

00:29:15,518 -- 00:29:19,184
So, on this step, I just really wanted to show you that
using join and clip in this way

provided us with the same result

and show the way it works if you run assert statement

and it doesn't return anything or returns true.

So, if it doesn't return any output,

it means the statement is true.

And if it throws an error, it means it's false.

So, in our case,
both geospatial functions provided the same result.

I don't have to explain this slide.

So, when you work with geoplot,

what you could often come across is

some clearly wrong result but you do not get any errors.

And it's a little helpful hint.

So, in geoplot,

coordinates need to be within this range.
So, they should be between minus 180 and plus 180.

I'll showcase what I mean here.

So, I'm going to create, clip, and reproject it.

I'm going to create a new GeoDataFrame by reprojecting an available clipped data frame.

And I want to show you bounding boxes of our newly reprojected data frame.
and our existing clipped data frame.

503
00:30:46,883 --> 00:30:48,057
So, if you can see,

504
00:30:49,125 --> 00:30:53,212
this GeoDataFrame and bounding box are

505
00:30:53,312 --> 00:30:56,779
within minus 180 and plus 180.

506
00:30:57,259 --> 00:30:59,156
And because of the different projections of

507
00:30:59,256 --> 00:31:00,311
the clipped data frame,

508
00:31:00,411 --> 00:31:05,310
they are out of these limits.

509
00:31:05,804 --> 00:31:07,977
Again, a bit of an explanation.
So, if you plot our clipped GeoDataFrame,

because it does fall within those bounds,

you would get some visualisation.

So, in this case,

this is our clipped GeoDataFrame

where the size of the points depends on

the information stored in the salary min column.

Limits just specify the minimum and maximum size of the points.
So, figure size is just the size of the plot.

Alpha is the transparency.

Legend is set to true again.

Probably, it's better to reduce the size from 50 to something else so it does fit better.

And, on this line, I'm also adding my Glasgow boundary on this plot.

Now, let's take a look if we use the clipped GeoDataFrame which bounding boxes are outside of those values.
And see, you only have Glasgow,

the Glasgow outline but nothing from the clipped data frame.

And it doesn't throw you any errors.

So, if anything like this happens when you're using geoplot,

just bear in mind that you might need to use different projections

so your bounding box is within those limits.

Just some hints I wanted to share.
In these many examples,

we would look into transformation between GeoDataFrame and data frame.

As I mentioned before, you can use most Pandas functions in your GeoDataFrame.

And I'm going to add in another dataset.

It is Data Zone boundaries.

I've added the link here.

So, it's an open dataset containing Data Zones,
which, if you’re not familiar,

it’s a kind of statistical area in Scotland.

So, the data provided on the website comes in a shapefile but I have converted it to geojson just to show how easy it is to open this file format.

And it’s quite a typical file format available and used.

So, again, the Data Zone boundaries are clipped for Glasgow,
just to give you the size, so let's open it.

So, similarly, we just say in GeoPandas to read file and specify the file location.

Let's check the Coordinate Reference System.

So, it's British National Grid,

code 27700.

Because I'm going to use it with my GeoDataFrame containing Adzuna ads,
I want to check what it is now, after the several reprojections we made. So, it is in Web Mercator code. So, let's convert our GeoDataFrame to match Data Zones, specifying CRS Scotland and the British National Grid code.
Let's look at our Data Zones.

00:34:36,935 --> 00:34:40,923
So, you can see this is a Glasgow boundary and small, tiny polygons inside which we are going to use.

00:34:44,838 --> 00:34:47,615
to make a spatial join again.

00:34:47,715 --> 00:34:49,189
But this time I'm going to use my Adzuna data and my Data Zone,

00:34:54,444 --> 00:34:57,863
assigning values of a Data Zone to
every point of Adzuna job ads based on their location.

Saying that the operation we're going to use for this is

the point should be within the border, completely within.

So, depending on the spec of your computer,

it might be faster or take longer

so be patient.

Let's have a look at the data.

So, we do see that we have received extra columns.
Let's have a look.

So, we have our Adzuna data here and Data Zone information attached to it.

But, just an important difference,

when we did the join with Glasgow,

we only received one set of values for every Adzuna point so it also going to be different for every point,

or for most of the points.
Let's find average salary advertised per Data Zone.

And to do this, I'm going to use group by.

So, I'm going to group my dataset by Data Zone name

and I'm going to find the average value of salary min column.

Right. Let's have a look.

So, our data would look like this.

They would have Data Zone name and information on average salary min.
On this step, I'm going to rename my columns

because it's not really salary minimum anymore.

It's average salary - mean.

So, I'm going to rename the column,

just to make it clear.

And now, let's join our GeoDataFrame Data Zone

with this statistical information of salaries.

So, I'm going to, as I just did, based on the name.
Let's have a look at the data.

So, now we do have the name of the Data Zone, average salary, and the information of the Data Zone.

Now here it does contain a geometry column.

Let's check the type of salaries by Data Zone.

And Pandas says, actually, it's a data frame, despite having a geometry column.

And it is recognised as a data frame.
So, let's plot it to verify that.

And, you see, it's not geographical representation or a map we see with some plots.

Pandas clearly recognised our salaries dz as a data frame.

So, to do that, we need to say that or create a new GeoDataFrame salaries dz saying that our salaries dz data frame we created in the previous step has got geometry, and this geometry is called geometry.
But pretty much, please use this column geometry as our geometry.

Now, let's see the columns.

Yeah. So, we still have geometry. Just to check.

And let's plot our new GeoDataFrame to check that we do have a map now.
So, we do have a map now

with some empty values.

So, these empty values or empty spaces are

the Data Zones which did not have any Adzuna job listings

so there is no average salary minimum,

sorry, average salary available for those Data Zones.

So, they just disappear from the dataset because

there are no values for them.
Right. Let's do some choropleth mapping.

So, just wanted to say it's not normalised data for analysis,

just an example. So, for our plot,

we are going to use the salary mean column

Colourmap is yellow, green, blue,

but you can use anything else.

Figure size is set up on this line.

And the scheme I'm using is Quantiles.
So, this is our distribution of average salaries between Data Zones in Glasgow. Again, blank spaces are those Data Zones where no information was available.

And here, I would like to show you a link for Colourmaps available.

Matplotlib.

Hold on. Give me a second.

So, this is how they will look,
what's available, and the names of them

which you could use in your code are just provided on the left-hand side.

Right. Let's do some cartogram

using geoplot.

Just to showcase that it does contain lots of visualisation.

So, I'm going to

convert my salaries Data Zone GeoDataFrame
to WGS84.

00:40:27,033 --> 00:40:30,353
The scale of my data would depend on

00:40:30,453 --> 00:40:34,460
the information on the values stored in the salary mean column.

00:40:36,142 --> 00:40:41,171
I guess the edge colour is the colour of those polygons inside.

00:40:41,948 --> 00:40:44,737
That colour also depends on the value.

00:40:44,837 --> 00:40:47,504
And the colourmap used here is red.

00:40:47,696 --> 00:40:49,755
Let's try to do something.

00:40:50,664 --> 00:40:51,782
This should work.
No. Purples.

No. Not much better.

Well, yeah, you can play with it to make it better.

I’m not sure what you see on your screen.

Mine, it’s quite pale.

Let’s try something else.

I think this one is better. So, you can see that

data polygon size and its colour depends on the value in the salary mean column.
Another quite popular type of visualisation you probably could have seen on the internet.

It just shows the dependency of the true size and of some value, and the size represented by some value.

Right. And as a last step, I want to show how to export your data on your GeoDataFrame outside. And really, it is just using the command to_file
and setting the destination.

Or if you're exporting it to a database,

you would need to specify the connection details.

Location, connection details, schema,

and, really, what you want to export.

Again, more information is available if you follow this link.

So, I'm going to export my clipped GeoDataFrame.
to_file. By default, it's a shapefile.

And there is an important warning that the shapefile does only support 10 characters in the column names, so all the names would be truncated.

It is known limitation of a shapefile.

So, please bear in mind that if your GeoDataFrame has a very descriptive column, it probably would be better to export it into some other format.
I'll probably show you how our data looks.

Okay. Give me a second.

So, what I'll do is, I'm going to share some other screen.

So, it's going to be QGIS.

A kind of open source traditional GIS software.

And I'm going to open my exported clipped GeoDataFrame.
So, by just dragging and dropping.

It’s similar to what we’ve seen in our notebook.

I’m going to share my notebook again.

And this is it.

Thank you very much for your participation.

Please visit our website to find out more about our upcoming webinars.
And I'm ready for Questions and Answers.

If you have any.

Thank you.

So, I'm going to open chat and check if there are any questions.

I see that quite a few people are asking where to get the data.

It has been uploaded to GitHub.
It's available for you.

00:44:35,023 -- 00:44:39,417

And I'm going to add this presentation, this slideshow presentation,

00:44:39,976 -- 00:44:43,915

to what you have on GitHub.

00:44:44,431 -- 00:44:47,326

So, I'm going to at it on a later date.

00:44:47,664 -- 00:44:52,413

So, it can be accessible for you quite soon.

00:44:56,003 -- 00:44:57,032

Thanks, Nadiia.

00:44:57,132 -- 00:45:03,371

Yeah, and I am also posting the link that they usually put

00:45:04,300 -- 00:45:09,788

for past presentations and resources afterwards.
You wait a few days, I think.

Yeah, it's here.

Okay, yeah. Maxwell has a question on choropleth maps.

Cool. Choropleth maps, yes.

Can these choropleth maps be made interactive?

Yeah, I would also want to know that.

Yes, but it does require different libraries.
and it's, kind of, another complexity.

I did not show anything in this lab just to keep it simple and introductive.

But yeah, it is possible. Just a different set of libraries to use.

So, something like Bokeh maybe or Dash.

So, both of them can display geographical information easily
but do provide some interactivity.

Yeah, and I was wondering,

earlier you mentioned something about

the boundary files.

Pretty early on in the presentation.

Could you also share those links?

Right. I'm going to share my screen again.
So, in the chat, Chau Man has sent you a link to GitHub

where you can find the data, the exercise itself.

So, it’s an exercise folder.

So, if you open this GeoPython lab,

you would have...

So, because I have it here,

you would have this notebook with all the links attached.

So, please have a look there.
I'm trying to find my...

This presentation and notebook, I'll update on GitHub later.

And it is pretty much the same as what you have received, with very tiny modifications and additions.

So, please see GitHub and the notebook, if you scroll down.

you will have all the links,

all the datasets.

So, it's BoundaryLine.
I think this is the one mentioned.

As well, I can send those in the chat.

I need to stop sharing.

This is it.

And there are many open datasets available on the Ordnance Survey website, something like Open Roads, Code-Point, and Zoomstack, which are fantastic for visualisation and to work with for geospatial analysis.
They’re not very detailed in some respects but may be useful as datasets to use when you’re learning new libraries or a tool.

Are there any more questions?

Does the projection used tend to vary by country?

Well... Right.

The question is, does the projection used
tend to vary by country?

Well, yes, it does.

Typically, every country has its own projection which is selected based on their best fit of the area.

And a different projection which preserve either directions, angle or areas.

And again, those three would be used depending on what you want to have less distortion of, let's say.
And the second part of...

So, yes, different countries tend to have different projections.

Or does that tend to be a standard reference?

So, data collected, via GPS devices, so mobile phones, are provided in WGS84.

Absolutely forgotten what this stands for.

So, it's World Geodetic System.

1984 is the year it was created.
So, this is quite a popular CRS to use for this type of data. The type of data which you see, something like Google Maps, it's called Web Mercator. It's a slightly different transformation than the plot. But what I wanted to say.

If your data is projected to some CRS,
If you know where your data is and what projection it is,
you can easily reproject it to something else.

So, for Britain or, well, government datasets are provided in the British National Grid.

But again, if you're new to it,

if you show information from the internet,
you can easily reproject them,
those data to WGS84 or the other way round to British National Grid.

I hope I explained it well.

It's quite a complex topic so I would really suggest you scroll through the links I've provided and do additional reading to understand that better.

Thank you.

So, as I mentioned before,
there's a video recording of this webinar

which will be available in an accessible format at a later date.

So, you can view it. And my presentation/notebook will be uploaded to GitHub as well.

So, I think this is it.

Thank you everybody for joining. Thank you for all these questions.

If you're interested in the data, you can contact UBDC or register with Adzuna for the API.
And I think in a few weeks' time

there's going to be a call for interest

in the licensed Adzuna dataset.

Thank you everybody.