Workshop 3: Exploring hydrological data Transcript

15 July 2025, 12:02pm

Samantha Rees started transcription

Samantha Rees 0:03
Lovely. Thank you.

Matt Dalle Piagge 0:04
There we go and we're ready to and we're ready to roll. All right. Welcome to lost track of which number workshop this is. But three, I think.

Samantha Rees 0:13
Yeah.

Matt Dalle Piagge 0:15
I hope so. And yeah, let's start with sharing a screen. That's what I was going to do. And let's start with this one. And I, I imagine by now we're familiar with the. Process of opening notebooks in Google Collabs. So as before I will put the link to the repository and the link to collabs itself, and when an annoying notification disappears out of the way of my chat box.
There we go. So as always, we're going into collabs, we're going into GitHub, and yeah, if this window hasn't opened, that's it's as a reminder, it's file and open notebook.

QF

Qidong Fang 1:00
Yeah.
Yeah.

Matt Dalle Piagge 1:06
Find the GitHub tab.
It's nice background noises there and paste in our lovely summer school link and then enter enter doesn't work. You have to actually click the search button for some reason, but.

Obviously wait for that to find lots of lots of notebooks and then they will after a particular a particular one. So there are three notebooks in this session which might be ambitious, but we'll see how we go. We're going to start with. It's coming up as a third one for me, it might be different for you.
But it's works. The name of it is workshop three-part one and it has some other words after that.
So let's load that.
And then the next thing as before is copy to drive or file and save a copy.
Let it do its thing.
There we go. Get rid of the previous tab. Let's close that. So don't get confused. And then we're we're ready to go. I will pause there in case anyone needs to catch up.
Well, whilst I.
Well, I'll introduce to notebooks, I suppose. So, yeah, this workshop is going to be split into three different parts, exploring 3 different types of data you're likely to come across in, in the hydrology and climatology and hydroclimatology and and hydroclimatic research.
In part one, we're going to look mostly at human readable text data. This is the data that you can read in a. You can open in Microsoft Word if you wanted to, and you could see all the characters and the data and the numbers. They're right there in front of you.
We'll also cover vector data, so that's shape files Geo. Jason, you may have heard of, but this is where you have. Well, yeah, it's data that contains shapes, basically, and that can be very useful for cutting things out.

Shagun Garg 3:00
Yeah.
Yeah.
Yeah.
Yeah.
Yeah.
Yeah.
Yeah.

Matt Dalle Piagge 3:11
And.

And that was an even more interesting white noise. And then part three, we'll be looking a bit more at what Malia introduced to introduce us to X-ray. So looking at net CDF data and and czar as well, which are these sort of N dimensional data.
But then yeah, we multi dimensional, we usually tend to, you can generalise as many dimensions as you want, but most of the time we're working in three or four dimensions. So you know latitude, longitude, time and perhaps depth or ensemble members as the as the 4th.
Is everybody OK? We all got the notebook up. It all looks. Looks like this. Looks like this one. So it's the right one. Shout. If not and we'll we'll continue and yeah, see lots of thumbs up. That's good and as.
As with the previous notebook, feel free to ask questions and interrupt and question things that don't make sense or don't or you don't understand it's you know, this is an interactive thing. It's not a, it's not a lecture, it's it's meant to be collaborative and interactive as much as is possible in an online forum.
But yeah, if we're all set to go, let's let's crack on with with looking at some some textual data, there would tend tend to be two main types that you'll come across in in hydrology to to sort of common formats. We'll start with looking at CSV.
Data. There's lots of acronyms going to be banded around, so I will try and always spell them out, but I may I may forget, but CSV is that one at least makes sense so it's comma separated values. It's basically I wish I should have had an example. Actually I didn't bring one in.
But it is essentially if you opened it up in a notebook, so not a notebook, it's a wrong, wrong word in Microsoft Word, or a notepad or an editor text editor like that, or even excel. But if you open up in a text editor to start with, you would see lots of numbers basically separated by commas.
Confusingly, sometimes they can be separated by tabs or other characters, and it's still called a CSV file or comma separated value file, but generally speaking it's it's commas and that makes it fairly simple to work with and simple to read in. And so yeah.
We're going to have a go at we're going to have a go at that. First of all, there's the standard commands to run and. But in these notebook sessions of importing the packages and installing the ones that we didn't have by default. So let's do that.
You may get. Oh, hello. I have too many active sessions. That was not. That was expecting to see. So bear with me whilst I disconnect. The ones I don't need right now. That's that one. I'm going to assume that one and that one. I don't need those.

So if that came up, let us know and we'll sort it out otherwise.
It's actually run that command and it might take a few. It might take a few seconds to run whilst that's running.
What we will do in this session, there'll be a few. There'll be a lot of me talking through some code that's already in the notebook, but there'll also be some places where you can code along with me. Where I'll be typing in code. Just small little lines. Nothing, nothing too much for and for you to type in as well. And just to get the hang of working with some of these commands.
OK, so that's run that's installed. You can import our packages.
First of all, because all the data that we're going to work with is on object storage, which similarly introduced us to in the last session. So it's all stored in the cloud. Then there's no no files will be stored on your.
In your collab session, so the file tab over here will will remain empty. Apart from this you know prebuilt folder that exists in there, but let's have I've got all sorts of data types. In this example data folder, so I'm just going to open that up and have a look at what's in it.
OK, that's a lot of stuff.
S3FS yes, I will basically.
So yeah, we're using S3FS to do this, which is a Python package. But basically what what it does and there's some text written somewhere else in this notebook. I can't actually remember where and that explains what that's doing, but essentially what it is is it acts as a sort of layer between you and the files.
And it basically pretends it pretends to your notebook to to you that it is a file system, so it reads in all the data that's on your in the cloud or in the object store, which is an S3 system. And then it pretends that it's a file system. That's the FS. So you've got S3 file system.
Yeah, all it is. All it is doing is essentially pretending that or making the files on your cloud, whatever cloud service or using act as if it's a local disc, which makes it means basically you can run it once and then it gets out the way and you can run your code as normal.
And that's what S3FS is. And we're using it here to just have a look at what data are thrown up into an example data bucket, which is equivalent to a folder in in sort of common parlance. We will explore some of these files, probably not going to get around to all of them.
We've got lots of funky extensions. A lot of these are part of shape files, which we'll

get to in in workshop too, and they often come with a lot of different extensions. What else have we got? We've got a dot, grd. That's another text file we'll get to in a little bit net. CDF ends with NC.
CSV which will open in a minute and I think then there's everything in there, there's our data restored somewhere else and I'll show you that later in the third part. But with that out the way, let's crack on with actually loading in and reading in some data. Yeah, and thanks guys for for explaining some more about SD.
Press in the chat. There'll be plenty of links for you to find out more information throughout the the notebook in the chat, and then in the recap afterwards and at the end. So yeah, if there's anything that isn't clear and that isn't, and there will be links to hopefully explain some of the stuff. If I don't go into it in too much detail here.
OK. So we're going to read in our CSV file. Well, that's a lot of information. We don't need to look at that right now. And what we're doing is using pandas, which I think we've got briefly mentioned.
In the previous session, that is another Python library that handles particular types of data. It's best suited for tabular data or two-dimensional data. I think perhaps is another way of putting it, of which most text files generally are most text related generally is CSV.
In particular, and pandas like, even if you hover over it, it even gives you a bit of information about it.
And pandas, like many libraries, can read directly and from object storage. No need to from S3 object storage cloud. They're all the same thing here that I won't use those terms interchangeably because I don't think it really has one particular name.
I can read read directly from that provided you give it particular information and that information is this endpoint URL which is the tenancy that Emilia mentioned in the previous workshop. It's basically the computer or the server on which the data is stored.
And we're also giving it this this Anon equals true or colon true here, which is what we use if the data doesn't need to have any credentials, it's not behind a login system that we need to access it so we can access it as anonymous. That's what. That's what all that means.
So if we run that.
Wait a few seconds for it to come in. Occasionally it can be a bit finicky with because it's connecting it over there because it's obviously reading this file from somewhere else on the Internet, so it can take a few seconds, but this time it seems to if it hasn't.

If it takes more than a few seconds.
If you know usually if it gets to about a minute and it hasn't done another thing, then that's where I usually recommend cancelling it and starting again. But if that happens, let us know and I'll show you how to do that. It is one of the perils of working with data that is remote and not and not local.
Let's see what it looks like.
So it looks like a table. This is what it would look like pretty much if you opened it up in Excel. It would be a table columns, rows. In this case I happen to know what it is. It's actually not got much information associated with it or metadata which is.
Not how it should be, and I think in Workshop 5 you'll learn a bit about what then some of the metadata that should be be here, but you've at least got daemons here, so some time descriptions and these along the top are catchment headings. So they're basically a numeric identifier that says.
This is the data for a particular catchment or a particular river basin.
And lots of minus ones as well. We'll get to that. Get to what that means in a bit. That looks a bit strange.
OK. And so we sort of come down to the first.
First, back proper interactive bit. I'm going to zoom in a bit, sort of make my screen a bit easier to read. I hope that's good. That's probably big enough I suppose. So we've got we've got 2 lines that are currently empty. I'll have a have these.

Jay Lindle 12:56
Yes.
Yeah.


Matt Dalle Piagge 13:13
Big like capital Block letters in them. I'm going to type some stuff in here and I'll pause so that you can make sure that everyone's got it to show what we're doing and ultimately what we're doing here.

Jay Lindle 13:19
Yeah.


Matt Dalle Piagge 13:27
Is doing something with this time information, so you can see it's split into day,

month, year which is OK we can see what that's doing. That's helpful for us. But to make the most out of pandas.
What we want to do is combine this information into what's known as an index. So at the moment these rows are indexed by what's in bold. That's just numbers. It's not very helpful. The true the powerful functionality of planners is unlocked when.

Jay Lindle 13:43
Yeah.

Matt Dalle Piagge 13:57
You index your data with actually helpful information here. That will be time information. So at the moment pandas hasn't recognised that these are. This is time information. This is something it can work with and make working your data a lot easier down the line. So what we're going to do is tell pandas that.
These three columns are time information and then to treat it as such, and then afterwards I'll show you what we can do with that. Once we've done it.

Jay Lindle 14:20
Yeah.
Yes.

Matt Dalle Piagge 14:26
So right this is what I need to make sure I type in the right thing, so I need to bring up a different window over here.
Which is this one.
You won't be able to see this. That's fine. This is to make sure I don't make any mistakes whilst typing this in. That's all on my other screen.
I will have what I need to type so.
Pandas read just read CSV function. It has lots of reading in options that you can has. You know it can read from Excel. It can read many many different formats of textual data and it has a very has a lot of functions, one of which is a function.
One of which one of the which of the options of which is look it's filling it in for me because Google's very trying to be clever is an argument called past dates and what that does is tell is where you tell pandas what.
Where the date date information is in your file. Usually if it's former is it, it's clever

enough to pick this up automatically, but for some reason on this file it hasn't, probably because it's split into three different columns. It needs to be told to look across those 3 columns.
So I'm going to type this in type along with me.
If you're following, So what we're doing here is saying at the moment.
The type the information you want to combine the information in column 01 and two into a new column called times.
And be careful of all the twiddly brackets and stuff, and this tells us that when it's reading those columns, it'll encounter the day first, then the month, then the year. Otherwise it will try and read it the other way around.
Can everyone see that? All right, can everyone follow that? All right.
Looks good. Cool. We've got one more line to fill in.
And this is where the mag. Well, I just want to say this is where the magic happens. Not yet. Magic doesn't happen quite yet. That's later. Next. What we want to do is tell Python, Python, tell pandas to use this new column that we've created, which is called times, to be the new to be the index.
And that's a nice simple command which I will make sure I type in correctly.
So it's name of the data set followed by set index and that particular column and we want to make sure that that's in place equals true.
Let's check that I've done that correctly.
Yes, all right. If everyone's happy with that, we can run the cell.
Wait for it to to figure it all out, because it's reading it in again, so it might take a little a few seconds.
There we go. And now if we look at the data again.
You can see it's that it's combined these three columns that were separate before. It's a day, month, year into one. And so it's, they're all and it's formatted it nicely for us as well. So it's formatted it as a year, month to day.
One more thing we should do, I think.
Oh no, I'm getting ahead of myself.
Yeah, we know we're gonna have a look at the data first I think.
Before we get into the into the well, I'm going to say yeah, into the magic of working with pandas and time functionality. We're just going to have a quick look at the data, so I'm going to hey.

Jay Lindle 18:21
Yeah.

Matt Dalle Piagge 18:26
Was it a question or did I miss something?
I'll assume that a questions.
OK, I'm gonna continue.
But yeah, do jump in interrupt if needed. That's fine. I'm just. I'm going to assume everyone's following along. I will pause a case. Need to check, but yeah, if you've got stuck or or. Yeah. One of the commands isn't running or you missed what I typed in. Then. Yeah. Then do let us know and we'll we'll catch you up.
Yeah.
But yeah, right. Where was I?
Yeah. First of all, before we sort of get into the true magic, we're going to do a little bit of subsetting. So we're going to select out the time a particular time range and a particular catchment. So this is where we can use the column and row headings or which are called the indexes to select out a particular part of the data set that we've read in.
Because we've now told pandas to use time as the index, we can basically write in the dates that we wanted to select out and it will know what that means. So we don't have to say, oh, I want row figure out what row we want. We don't say I want row five, column ten, we can.
Say I want these dates and I want this particular column, in this case the columns are catchment IDs. So let's see, I'm going to put in what? What dates did I put in? Let's find out.
I'm going to use 2000 to 2015 so again feel free to type along with me so that we're on the same page. So I'm going to go from 2002, the first of first of January 2001.
To the end of 2015, we're going to select out that data.
And let's pick a particular catchment. I've picked a fairly random ID that I know happens to exist.
Yeah.
And when you've done that?
You can press yeah, press the run button, run the, run the, run the cell and we can see what it looks like.

Yeah.
Another another yeah. The next cell should be blank for you to type into. In this case, we're going to just plot it. And again the the nice stuff about using pandas and accelerator workgroup data is that they both integrate very nicely with pythons plotting functionality. So we can literally just do this dot plot.
After the name of our data set and it will go away and plot it for us without us having to do anything. So what we've got here actually I might pause there and check that that is all making sense to people and that everyone's got a nice.
Vague plot that looks vague like this doesn't have to look the same if you put in different dates or different catchment ideas, it might look different, but yeah.
How we all getting on all good.
Awesome.
OK.
So that's a little that's.
A bit of a basic bit of plotting.
Ah, Ali, you've run into that, that, that.
Not reloading problem that not not loading problem that I have that has come up before. I'm assuming it's going to be this cell if we said that was it this cell.

Ali Rudd 22:10
Yeah.

Matt Dalle Piagge 22:11
Yeah, what I do here is go into runtime this runtime menu and interrupt execution. I think there is an easy way of doing it, that's just the way I've learned to do it. That should stop it running and then.

Ali Rudd 22:26
Yeah.

Matt Dalle Piagge 22:28
Basically, try it again and hopefully it should run in a few seconds.
Everyone, everyone hold your breath.

Ali Rudd 22:36

Hmm.

Matt Dalle Piagge 22:44

I generally say if it goes beyond half a minute, it probably.
It's having a problem. There you go. Yeah, they said the vague reason of working with data over the Internet rather than than locally. Sometimes it doesn't. It gets a bit stuck. That's all right. So yeah, that's also a demonstration of what, what to do if anyone else gets stuck at those particular locations.

Ali Rudd 22:48

Oh yeah, it's done it now. Yeah, fab.
Thank you.

Matt Dalle Piagge 23:05

Right, let's do a bit more fancy plotting, so Python pandas X-ray can do some nice basic plotting, but if we want to make it look a bit nicer, we have to basically do, modify and characterise things ourselves.
So I'm going to do a little bit of there's quite a few, quite a few bits of of lines here. It looks like, but there's actually not much to type in. This is just showing you. If you wanted to customise this plot a bit, for example, if you were making it, making it public, or need to do a publication, or.
We're doing more than just a cursory look at the data. Here's here are some commands that you can use to basically make it look nice, and I again, you know I would, I would type these in and you can follow along with me and I'll try not to go. I'll try not to go too fast.
I tend to do this quite rapidly, but let's see make sure I've got the right file up.
I type in the right things.
Right. So the first line is pretty simple. It's basically what we did before, which is saying that's that's our data set. We want to plot.
This time we want to plot it on a particular set of axis that we created in the two lines above or the one particular line above. In this case, we want to give it a label. We can name it anything we want. I'm just going to call it observe flows that will do. That makes it easier to refer to in in in further commands.

If we look at this plot, we can notice that the Y axis is a bit strange. For some reason it's going below 0, which is not something we should ever be having in in observed river flow data.
Umm.
So we can make sure that the the axis stops at 0, so that's a nice as simple as doing set Y limb 0.
Let's put it at what do we put it at the top? It goes up to about 25.
So that's that one.
Title giving it a title very simple.
Plot Dot title fun title because I'm not particularly imaginative today, it would seem this PLT thing here. Notice it's different to to axe, I think you can actually use axe here as well, but just use PLT here. That's the name of the plotting library that we're using that so is running underneath.
Underneath pandas plotting plotting code. Oh, we can even put a label on the Y axis because there's nothing at the moment. All good garage have a label on the Y axis. I'm very guilty of not doing that. Sometimes. Again, plot that Y label.
Nice Y axis label for example, and then it would automatically plot a legend as well, which shows you what the different lines on the plot mean. Given there is just one here, it doesn't really matter.
Umm.
Because yeah, it's clear what what the plotting is just a one line, but then you know, if there were many lines, you could give all of the plots labels like this and then add a legend and it would automatically show which line is what colour and so on.
Pause a moment.
Before running it and now we.
And that shouldn't try and get both on one page so that you can still see the commands zoom out a bit.
And that looks a bit nicer.
Nice and simple. So far nothing complicated, but not necessarily obvious. If you didn't know how to. If you didn't know the exact commands or the exact.
The the syntax, which is something I always forget and something to remember with these things, is that I can do this by heart now because I've been doing it a lot, but for a lot of anything more, slightly more complicated than this, I always have to look it up before I do it, and that's that's that's fine.
You know, no one expects anyone really to remember exactly how to do fancy

coding and and and plotting it's, you know, all wait. The documentation is basically there to be read when you need to do something, and I absolutely recommend.
If you're not sure, searching the documentation and and and looking for looking for it online and that's fine. That's how a lot of coding is done in the sciences. I must admit. All right, it's a simple bit of plotting is one more little thing to do before we move on from this kind of data.
And that is, we know I noted I pointed out earlier, if I scroll all the way up, but there was an awful lot of minus ones here, which seems a bit strange now, you know, somewhere in this file there should be some information telling you that.
What this means, but as this is not particularly good data, it doesn't actually have that I happen to know for the purposes of this workshop that's that is signifying that there is missing data there.
Usually you'll find missing data is a number that is unrealistic for the actual data that's being contained in the file. So in this case it's roofer flow data which is always positive. So we've got someone is using a negative number to represent.
Missing data.
That, however, is quite unhelpful if we didn't. If we don't tell pandas about it again, because if we then wanted to calculate means and or do any kind of calculation numeric calculation with this data, it's going to use these minus ones and that's going to give us the wrong answer because it's going to assume that there are genuine -1 flows some of the time, which is not.
What we want so you can tell it to basically ignore these missing values by adding an extra line to our reading in code. And again there's a blank line here for us to fill in, and I have to remember what it is I'm supposed to put in here.
Sometimes again, it's clever enough to read these in automatically, sometimes it's not. So in this case it's not, and it can be a bit hit and miss whatever it is or isn't. But that's why it has all this extra functionality that you can add in that tells you that tells it what it needs to do.
So in this case it's the command is NA values which is saying you know which values are not applicable, which values are not actual, not actually they show and that's -1.
So now if we run that and again give it a few seconds.
And if it takes more than a few seconds, then we might have to rerun it, but we'll see. I get impatient. I think I'll wait till, like maybe 20, maybe 30 seconds, and then I'm going to start it again.
OK, I'm getting impatient. I'm going to run that again.

There we go. Now did it in in nine seconds? Yeah.
And now if you have a look at the data again, it's now replaced those minus ones with NANDS, which stands for not a number in this case. Basically all that tells it to do is ignore it when it's doing calculations.
So if we run the next cell which is more plotting stuff, I won't make you type it all in this time because it's fairly simple. We can see that it's removed those nasty spikes that we had before where it was treating those one less ones as actual data. Those have now disappeared.
Do do, do, do, do, do, do.
And I if I run the next cell.
Where I've just selected out a much smaller amount, much smaller period of data. You can see that there's a small gap has appeared in in a little bit. Actually I think there might be two in this plot. No, that's just one and basically it's basically treated that that area as a blank.
I say it won't sharpen your plots.
OK. How are we doing?
I'll wait for a chorus of thumbs up. I see a few. That's good. Excellent. Nice and simple so far. I hope that's the idea.
Now I'm going to quickly, uh, talk about um.
All that stuff I mentioned earlier about the pandas time, date, time and time handling functionality, which can be very, very useful when we're doing analysis of hydrological data.
So let's have a look at the data again and just remind remind you that it has these Times Now as the columns as the rows excuse me and and the catchment IDs as the as the column headings. If we as I mentioned earlier.
The word index and the row is A and the rows. The row headings are what pandas likes to refer to as an index.
And what we can do is pull it out by doing OBS data data. Obs data is the name of my data set dot index. I'm just going to show you this. You don't have to follow along here.
Basically, and it and it shows that it's a special kind of data. It's more than just a list of the date, the dates and times. It's this special object called a Date Time index, which means we can do something. We can do good stuff with it. What it what it basically has.
Is is aware of what we mean when we say days, months, years, seasons, quarters,

weekdays, seconds, any kind of time unit you care to mention. It knows what that means in reference to the row headings. Yeah, in the index and the row.
Headings. So for example you can type this initial with me if you want it, it knows what the year is of all the all the rows. I forgot I shouldn't put brackets there, so yeah, it's telling you what the what the year is of every single row. It can tell you what.
The quarter is that's a bit of an interesting one, but anyway which which quarter of the year it's in, this can be very useful for calculating seasonal means if you know if you assume each quarter of the year each three months of the year is if we're different different season.
I think it can do the day of the week as well. Dot index dot weekday no day.
I have week. Let's see. Yes.
So now it's telling you it now it knows which day of the week each row is, even though that information arguably we didn't give it that it's told you that automatically calculated that automatically, which means we can do.
It makes it a lot easier to calculate stuff like a yearly mean or a seasonal mean or or, you know, just have to be a mean. It can be a sum or it's only a mathematical calculation that needs to know what you know, whether it's a which year you're in or which month you're in, or which which season you're in, or or whatever it is you want to do it.
So in this case, we're just going to calculate an annual mean.
Of every catchment. And so we're using the pandas group by functionality and that is basically fancy talk for it's going to group everything into group into buckets of years and then it will take the mean over each of those buckets. So calculate a mean for each year.
The era there is of data. I don't do that for every single catchment, so it's a common thing. You may end up doing when you're working with with hydrological data surprisingly common. So I will make sure I type in the right code again.
So I am slightly juggling lots of different screens here. Tense. I look looking back between one screen and the other right. So what we want to what we want to group by is years in this case and we saw earlier that to access years.
We do of data dot index dot year so that says group by this particular information and where every different year makers own group and then the dot mean on the end basically tells it to calculate the mean over each of those.
Each of those groups.
And if we look at that now, what we'll find is the Times column has reduced to just

years and the values that are now in the inner data set are these annual means for each of the catchments we've got. So it's telling you the annual mean flow.
For each year for it, for every catchment and how it handles the these the missing data. The Nands is is fairly important to know if we'd left these as minus ones, they'd be they would be included in our calculation. It would give us wrong, wrong, wrong answers.
Whereas now it sees that there are any nands in and if there are, it ignores those in the calculation, and if the entire year is a group of, net is all missing. That's when it will produce. It will continue that it will say that it can't calculate the mean of nothing, so it's still not a number.
Keep the work keep referring to the data as missing.
And that I think I'll do one more plot just for fun. I think we've got time. So I'm just going to pull out two catchments.
And see what they look like together, just to to bring this sort of back down to a hydrological path, as it were. So far, we've been lots of just data wrangling. So what I plot just to sort of, yeah, simple plot go on. Yeah.

Amulya Chevuturi 37:38
Matt Ali asked to show the mean line of code again mean line of code. Mean line code, yeah.

Matt Dalle Piagge 37:41
Ah.
Absolutely. Where was that? It's this bit here, I think.
All right. Yeah, give, shout if we need to go back to that again. I'm not. Yeah. I can't always see the chat. So you're welcome. Do do. If I seem to have missed something in the chat, absolutely fine to, to to say so.

Ali Rudd 38:11
Thank you.

Matt Dalle Piagge 38:21
So here we're just comparing two different rivers. The annual mean flow of both of them. In this case I've I've pulled out two particular catchments, which is that of the the Thames and the and the Tyne, which is a river up in northeast England.

Two very large 1-2 large river basins which perhaps you know to to me who isn't a hydrologist by training, believe it or not that even though it's you know they're in the same arguably rather small country, the annual mean flows are quite different. Between them and not even just the the amounts because they are different size Ripper basins, but also the variability. But that is where we're going to leave it. I think for this type of data there is a lot more in this notebook.
In particular, I will just briefly Scroll down. There is a whole section on another type of text to those you might come across, which is known as sort of just. Technically, I think it's a proprietary format, but it's very well used in in hydrology, particularly in hydrological models which are known as ASCII grid files.
I'll have a I'll quickly show you what they look like.
Just so you know, if you come across them, they'll have this particular heading information at the top, which basically tells you information about about the grids that you're reading in, and then the data itself is separated by tabs. But again, it's human reasonable and it's textual and.
In this case you can just see a lot of missing data at the top of the in the corner of the grid, so there's a lot more in this notebook that will talk you through how to work with this data. Essentially, it's very similar to what we've what we've done with pandas. It's a case of reading in the data, which takes a little bit more work, but most of the code is dedicated.
To helping you do that and then once it's in pandas or I think actually X-ray or one or two of those, those libraries it it functions as if it was as if it was the same kind of data we've just been working with and you can use very similar commands and very similar.
Analysis, techniques and plotting commands to work with it, but I think I will. We will pause there on text to data so that we have enough time to cover the other two types to anyone. This is a good opportunity to ask questions I think.
Probably won't have too long, but if there are any shortest questions, do shout now, otherwise save them to the end and afterwards and we can go through in more detail. Or if you're stuck, the facilitators can help you out some more. How we're doing.
I think everyone's good. Fantastic.
In which case it's time to go back-to-back to back to basics. That's not what I meant. Time to start, open a new notebook. So what we're going to do is do what we did before. So file open notebook, go to GitHub. I'm not sure this is going to be the right

link, but it is. I will post that link again.
So that people have it and we're going to open a different notebook this time.
So where we go, we're workshop three and it's this time it's workshop three-part 2.
They should see Part 2 in the file name.
Once that's come up, it's the same, same, same deal as always. It is save a copy and drive or use this copy to drive button at the top.
Close to previous tabs, I've only got 1 copy open.
And was and I'll run this first couple of cells because again it might take a little while to install and we couldn't wait a while to get those up.
I'll just have a quick drink. It's thirsty work. All this talking.
All right. I'll assume everyone's cracking on with that or getting somewhere with that.
In this section, we're going to look at vector data, which is a fancy name for basically shapes, shapes and lines, usually Geo reference. So they usually come with a coordinate system. So you know where on the earth they're referring to.
These are very, very useful if you want in hydrology for focusing on particular catchments or particular river basins or particular rivers. So you'll often find you'll have files that are literally, genuinely called shape files.
Which are basically containing a list of different shapes which refer to different geological features, usually in hydrology there'll be river basins and rivers, but you can get shape files of all sorts of things. It can just be borders of countries.
Where lakes are I already mentioned rivers. I'm running out of examples.
But yeah, any kind of spatial data that you know that is, that is a shape. I mean, you'll probably find it in a shape file. You may also find it in a new Earth. It's I'd not try, actually if it's a newer, but I've I've only seen it around more recently.
In Geo Jason format and yeah, good point, Matt. Sometimes it's not a shape, it's just literally a point on a map and that says something interesting is in this point that's a good point. That's a good point.
So we've loaded, we've loaded the loaded, our other packages that we need to to work with this kind of data. Again, we're going to read it in from object storage from using S3FS.
There's a lot more information here about shape files, some of which I, some of which I covered in in my previous wafflings.
But the bit that we really need to know now is that we're going to, we're going to going to use pandas, but we're going to use a particular brand of pandas, I suppose, which is called Geo pandas very imaginatively. And basically, it exacts, it works in

exactly the same way as to all the stuff you've just seen, but it.
Adds in another column which associates the shape and the geometry with each row of the table. Better if we just crack on and see it. So let's set up the the file and the object storage system as we did before.
And then notice we're using GPD now to read it in rather than PD, so it's Geo pandas nicely only took a second for me. That's the quickest it's ever been, which is good.
And then we'll have a look at what it looks like and you can see it's it's basically representing this.
The expected data as as a table looks very similar to what we saw before. You've got row headings and you've got column headings, but you've got this extra thing which is the geometry column which makes Geo pandas a bit special. And this is where it contains to shape information.
And then here it's saying we've got lots of polygons which are basically lots of vector shapes. The other columns that you've got is information about each particular shape. Again, I'm sort of.
What? What am I doing? I'm.
Leading you into teasing for Workshop 5, which is more about how data should look if it's done properly. This measure data isn't wonderfully descriptive. There's a lot missing if you have to kind of know what data is in order to know what these column headings mean.
I'll we'll go through it together. And so we'll you know, we'll know what it means by the end of this with this workshop. But yeah, strictly speaking, there should be a lot more columns here that tell you more information about the about which what shape actually is. Yeah. But yeah, shape files.
And Geojson files and other kind of vector formats should have as well as just the raw shapes themselves. They'll have some metadata alongside them which describes information about the shapes and what's actually in the file. OK.
Yeah.
So we're going to do a quick bit of live coding. Again, nothing, nothing too complicated here. We're just going to.
Finally, make it hang on. Sorry, multitasking. Bringing up notebook on my computer. We're just going to select out one of the rows. In this case, one of the shapes, and we're going to see all that. Use Geo pandas, pandas plotting capabilities to see what it looks like. Let's say you can do this with any.
Any of the shapes or multiple of the shapes in the file. It's a name of our data set is

helpfully called shapefile so.
That's I will start typing in the right code cell. There we go so shapefile.
In this case, notice we're using something called I lock. I explain a little bit more about what that is in the text I think, but basically it's just another way of selecting out rows and columns. In this case we're selecting out the 1st row in our data set.
And we're gonna see what it see what it looks like.
And yeah, well, that's that's the shape, I guess. Again, if it had more metadata, we might be able to tell a bit more about it. But as a as a first go at seeing what's in the shape file, I guess that might be useful if you know what you're looking at.
What we might want to do is find out some more information about what's in this shape file, so I should have paused that. Did everybody get that something?
Command. OK and then was able to type that out again. I'm sort of rattling ahead without.
Without pausing, but I think we're good. OK, so I can. I will continue in that case.
If we just look at the information itself in the next cell rather than the the plot of it, you can see we've got we have got this metadata which you know can can be helpful.
In this case it probably isn't. Sometimes you may find some more information in this thing called attributes.
Contributes, which you can access in Geo pandas as well, but here it's very helpfully blank. And if this data was a.
Well, yeah, if this data was fair, which is an acronym, you may well come across later on in the in the summer school. That was not what I meant to do. This could have something more information about it in it. But to be, to be honest, a lot of data you will come across.
It's gonna be a bit like this. You have to struggle struggle a bit to find some of the information that you that you need to work with it.
Next, we're going to plot this entire file. That might be something that's helpful to do. Again, nice and easy.
Name of the shape none. The name of the file. If I did dot plot off we go.
And varies and it just knocks up something nice and quick and easy for you, so we can at least see that this is really we are working in the UK. This looks a bit familiar as a map of the UK with bits bit chunks taken out of it.
Umm.
But again, we might want to do is make this a little bit easier to read and we want to add some parameters to our plotting command to to see a bit better what this data

actually is. So what I'm going to do here in this bit to add.
Is we're gonna make.
You're going to remove the face column at the American spelling, which we all love. Face colour equals none. And remember the quotation marks are going to remove the colour from it, so it will just plot the borders.
Of the shapes and I guess you could argue that's not particularly helpful either. So what we're going to do now is zoom in a bit and just look at one particular catchment. So I happen to know that.
The Thames catchment, which we briefly saw earlier, has a particular catchment ID which is 39001 I keep meaning keep doing that and we're going to select that out from the data set and have a look at it specifically rather than data set as a whole so that we can, you know, it's easier to work with.
And this is where I also need to refer to my notes to make sure I get the right command in.
So this is another example of how we can locate and subset our data using the column headings rather than the the in rather than the sort of the numeric indexes so we can do Thames.
Shapefile dot lock which is pandas subsetting function and what we want.
Is to select out every row.
Where the ID string which is one of the column headers.
Is 390039001. There's quite a lot going on there, so I'll pause so that people could have got that down.
What that should do is select out the catchment that refers to.
But first, the Thames River basin.
And if that's all going through, OK, we can plot this particular Ripper Basin.
I'm just going to zoom out a bit so that I can we can keep that plotting combined at the top.
Keep the subset command at the top.
Now I happen to know that that is the Thames. But you know the the Thames Basin. But you you you might not. So we might want to basically do other things with the the plotting to make it look a little bit more, a little bit nicer and more obvious what it is to people who perhaps might not know what the the Thames is.
The time space isn't where it is. All the rest of it.
So we've got some more plotting commands coming up and some more substituting commands coming up. I'm not going to make you type these these out mainly for

the sake of time, but we can do some fancy things here.
Is where we're using pandas subsetting command to select out all the catchments that are labelled between 39,000 and 40,000 or 39000 and forty 000 because again, I happen to know that.
All the catchments and sub catchments within the Thames will start with a #39, so we can tell pandas to just pull out all the rows that start with an ID of 39.
Which we've got here and then we can just plot those, which looks the same.
Sorry, which is not helpful. We've selected out a whole load of catchments that are within the terms, but all we just see is 1 blue BLOB. That's because we forgot to turn the colouring off again. So if we want to see the outlines of the shapes, that's where this face colour equals none is a very useful command. So we'll run that.
That one, and now you can see a lot more detail. So you can see all the different sub catchments that were within the Thames. So all the little rivers that make up the main River Thames and Main catchment basin.
Which I think actually looks quite pretty. It's almost a work of art in some way anyway.
What's next? Because I've actually forgotten what I've got next. OK, let's do one more plot. I think I'm just going to keep scrolling down for a bit. I don't think we're going to do this. This is a bit too. Don't need to do that. There's just some more.
Information on doing calculating you can and just can calculate the area of the the the shapes the vectors for you. So you can select out you know what the biggest catchment and the smallest catchment and order them by size.
And if there are other catchment properties within the shape file, you could use those to select out specific catchments that are perhaps the most steep or the least steep, or I've forgotten what other catchment parameters are there or length of the of the river of the river in it. That kind of stuff.
I remember I found myself doing a lot of this in my first year at CEH and I was throwing some hydrological data and had to work with it. But for the sake of time I won't go for this here. The notebook we hear that with the code will be filled in for you.
There's a complete version in the repository as well, which we'll link to at the end and send around, so everything will be there for you to go through in your own time. If this is something that's of interest, or use it or or useful.
Right. I'll pause there.
And check for questions. Thanks, Amelia. That's that is the website that I got this data

from, I think. Yeah, yeah. Yeah. How's everyone doing?
Lots of thumbs up, lots of nothing. That is good. That is good. That means we're doing all right for time.
Uh.
Right. One more thing to do, I just need to think about best where to start this. I'm just do a bit of scrolling, don't feel need to follow me just yet. There's one more bit of plotting that I think would be really, really nice to show.
I think let me just see if this Thames plot rivers. Yes, if we can Scroll down, scroll your notebook down until you get this cell.
I was skipping a few from all your work previously.
This is where this is where we start to do some more fancy plotting, so pandas and Geo pandas plotting capabilities are fine, usually more if you just want to have a quick look at the data. But if you want to do more fancy stuff with coastlines and more Geo referenced and.
And grid lines and where it is on the planet. Kind of plots or more professional looking plots then we bring in another package called Carter Pie which is essentially a map, a map plotting package for Python which knows how to handle different coordinate systems and knows what to do with shape files and how to plot them.
And and you basically it's a nice way of making your plots look pretty, which which I probably spend too much time doing. But if we run this cell, what should happen?
Should be this this warning here, which is fine to ignore, it's just sourcing some data, it's going to plot.
The Thames catchment on the map of the UK and also show the major rivers of the UK as well. I go into this a bit more in a bit more detail in the notebook. What each line of this commander is doing.
Because essentially it's like it's, you know, it's slightly different plotting commands. Now we're using car to pine knock pandas.
But ultimately, there's nothing too complicated and and if you scroll below I I write more information about what what is going on with each one.
And if you want to go really fancy, what we can do is Scroll down even further.
I'm sorry, This is why I I should have done that beyond all the text that I've written.
Until you get to these cells, so a cell where we're reading in rivers shape file, which is a different shape file, and this one which contains the more rivers of the UK. So this is now not not coherent shapes but.
Lines that denote where the rivers are in the UK. Again, we'll read it in using

geopounder's capability to read in from the Object Store from S3. We'll use a similar plot to what we did earlier.
To see what it looks like.
And also got the tennis catchment on there as well. So that's that's sort of a basic Carter pie plot. But if we want to make this interactive in the recent years, Geo Pandas added a really cool capability which which basically produced interactive maps that you can zoom in and out of.
Umm.
I would just demonstrate, I hope.
It gets there.
And other with other cool features. Let's have a look at it.
So what it can do with just one one command basically is it times 2 files zoomed in. It basically creates an interactive map that you can zoom in and out of click and drag. See where the shapes files are. If you hover over it, it shows you the metadata that's also there.
And yeah, and also you can customise so much about these maps. This is just a default at the moment. It is that you know it's showing you. I think it's the open street map background, but you can change the background to be satellite view or all sorts of different background tiles you can plot.
Multiple shape files on one map, which is what I do in the next command.
So here I put the rivers on as well.
You can change the colour what what the tool tips are the the things that come up when you hover over it doesn't work for the rivers because I don't think that shape file has any information in it, but for the Thames catchment it does.
Yeah.
And one final thing you can do and notice there was a question chat I will get to that in a second is if you want to export this map, share it with people, show it to people, they don't have to be able to run these commands.
It has once you've created the plot, you can save it as an HTML file and that actually I. So I lied earlier about you not having to use anything in your files. It will be in your files and then if we.
I'm not going to do it here, but what you could do is download the file, send it to someone and they could open it up in their web browser and they'd get an exact copy of this map. Also interactive also able to zoom in and out and do all that sort of fancy stuff. It's sort of starting to turn Python into a bit.

Bit like a a GIS system. Not quite, but it's getting there.
Right. That is the cool thing. I wanted to save, wanted it, wanted to talk, wanted to get to again. There's a little bit more information in this notebook than I've gone through today. There are is a section below on other vector data formats you might encounter, but again.
Once you've read them in with pandas or Geo pandas in this case it's pretty simple to work with them. You use the same commands that we've just been through. The only difference is it's a slightly different command for reading them in which I've got here. So Geo Jason is one I see around quite a lot these days.
So yes, there are still some lines that are currently with the insert code here lines in them. What we'll do rather than go through those now, I decided to skip over them for the sake of time. What we'll do is I will share.
A full version of the notebook at the end of the session, it will come around in the recap material. It's in the repository with the with the ones we're working with now, and that will be filled in. The code will be filled in there the the slightly edited one was only for this this session to for us to.
To to get you to type some things in.
All right.
I keep asking it, but how is everyone doing? So far it's been a chorus of positive answers, which is great. Still with us half an hour to go and we're not quite half an hour to go, 25 minutes to go. Promise you it's worth it.
Or something like that.
We're going to do next. We're going to do next. I see. I even forgot myself. Next. We're going to look at some gridded data. So this is now the end dimensional stuff. This is X-ray. Net CDF and Czar, which we talked about earlier. So same routine as always.
File open notebook.
Have I got that paste in that? Oh, I didn't. I forgot to go to GitHub, paste in that link, which I'll also put in the chat as a reminder again.
Wait for it to find notebooks in the repository. This time we're going down to workshop three-part 3, which is we're Bridget.
That one here.
Once that's come up, the usual stuff, let's make a copy of it. Save a copy in our drive.
Wait for that to load up.
Close the old one and run the 1st 2 commands as they take as. This one takes a bit

longer. I think it takes about a minute or two to install these packages as it's slightly more slightly more chunky than before.
All right, so done two types of data, one more to go and this arguably the one that is most useful and you'll come across most often, I think this is net CDF czar gridded data sets.
Or to be, you know, used proper parlance of multi dimensional data types or N dimensional data types. I've heard them referred to as both. So basically.
Yeah, the data sets that can have any number of dimensions, but most of the time I'll be working with three or four dimensional data. So you'll have this is classic stuff that is that tends to come out of hydrological models or any kind of.
Of system model to be honest, often it's also the input data or observed data that you might feed into models. Or you might want to work with directly. So for example we saw earlier and I think we're working with in this notebook a gridded observes.
Rainfall data sets.
Which?
Yeah, it's, it's, it's another example of a gridded data set and overcomplicating things, but you can get observed data sets that are also made onto a grid rather than at individual points. Now boat of three, that's another one of these packages for handling.
Umm.
Working with data that's stored on the cloud or on S3 or on object storage, you might hear it referred to with any of any, any and all of those. We don't actually we don't. I think I don't think we actually work with it directly. It just needs to be there under the hood to handle the.
All the web requests that we're going to be doing for requesting lots and lots of data created data, lots of chunks of data from from where it's stored in the cloud. So in the case of the text files and the shape files, it was basically a case of those files will be read straight in to our memory. They're small enough that's not a problem.
Into the memory of of. In this case our Google collab tension, but we've gridded data and and they mentioned data. Data sets get a lot lot bigger and that can be a problem. So we need to have extra packages that handle that and make sure we don't overload.
Overload the server with all our requests. Essentially, there's more things that that size has implications for, and I'll cover that in a SEC.
Right. So those packages have now installed. I'm going to ask for another another

chorus of thumbs up for when those have installed for you. And I said it takes a couple of minutes from when you first run it.

Ashling Laffey 1:08:00
Hi, Matt. Sorry, just wondering what time should you give up and run it again in terms of how long it's been loading for just mine seems to be taking a while this time.

Matt Dalle Piagge 1:08:08
Mine took two minutes.
There we go. Mine took two minutes. What about this one does take a while.

Ashling Laffey 1:08:18
OK, I'm up on the three now, so I might just give it another go.

Hywel Lloyd 1:08:20
You know, well, I I just took 3 minutes.

Matt Dalle Piagge 1:08:23
OK.

Ashling Laffey 1:08:23
OK, cool. I'll hang on another second. Thanks guys.

Matt Dalle Piagge 1:08:25
Yeah, I think this one, give it a few minutes. There is quite a lot to install. I'll, I'll hang about here for another few of a couple of minutes before we dive into the rest of the notebook. So no one gets lost or left behind.
OK.
I'm just gonna Scroll down a bit so I know what's coming or we can let people read through this.
Yeah. So some more information whilst we're waiting for the packages to install.
We are again using the package that you may have seen earlier, which is FS spec.
Actually, I think we saw SS3FS before and S3FS is a package that is sort of a a sub package of the broader Python package which is called FS spec which stands for file

system specification.
And again, what it does.
Is.
But yeah, it does what it what it was doing before, which is to essentially take any data storage system which and there are many different types out there beyond you know S3 and beyond your standards as a file system that you might be familiar with. And translate it into a file into into a normal. What we would think it was a normal files and folders directory system as if it behaves to our code as if it is a local file system that we can work with which is.
Again, you sort of run it once it gets out the way, and then you can get on with the rest of your code without having to worry about it too much. There are a couple of things you do have to think about, which I'll talk through in a second, but I think now that most people are have had the packages installed and loaded.
We can we can crack on.
So like before, we're going to use S3 file system S3FS which is the sub spec of FS spec to read in data. In this case we're reading in a czar data set.
Which is.
As Emilia mentioned before, it's a it's a strange name, but essentially it's just a grid of data format that is adapted for working on the cloud and working on object storage. Net CDF is its twin, essentially very similar.
And you will see you will see more net CDF to begin with than Czar. But it's sort of slowly becoming more and more popular as to get bigger.
But Nets CDF doesn't work so well if you if you put it into a into a cloud storage system like we're doing here, or like we're going to work with here. And I think earlier and will you share and I think I've also at the bottom of this notebook, got a link with where me and then will you talk through?
How data is adapted and converted for use on object storage? So if you'd like more information about that, there is a lot more out there. We spent maybe too much of our lives figuring that out at one point.
Right, that's enough of that. Now let's load it in and have a look at it. We can. We're using X-ray and nice and simple. You either use open data set or in this case openzar. It knows how to handle it.
And it will look very familiar to what we came up with in an earlier session. Previously. Let's assume in a bit. So you've got. It's a nice view of your data, the actual data variables are down here and then you have the coordinates above.

There's a lot of coordinates, but essentially you just look at the top ones to know what we're really working with, which is in this case it's X&Y in time. As you can see, there's a lot of time dimension. There's a large number of time coordinates here. That's because the data set we're working with is an hourly gridded rainfall data set that runs for.
Several years I can't remember quite what it is on the top of my head, and so that that adds up.
And so we need to start being a bit more clever and a bit more careful about how we work with this data.
As some more go for it.

Amulya Chevuturi 1:12:38
Just sorry, Matt, to the time dimension will tell you the 1990 to 2016, you can see it.

Matt Dalle Piagge 1:12:42
Oh, yeah, yeah. You can see it in there. I wasn't even reading what was in front of me. Classic. Yeah. Yeah, yeah, yeah. Yeah. X-rays can be helpful. And it tells you these things without you having to work too hard to find them out. Yeah, 1990 to 2016. So that's.

Amulya Chevuturi 1:12:48
Fair enough.
Yeah.

Matt Dalle Piagge 1:13:00
25 years or so of hourly data. That's quite a lot.
I'm going to skip over the next bit. It's not so important for now. It's a lot more information of what I've just talked through, talking you about about the data sets and and how it's structured and how it's formatted. But what we really want to see is coming down to this cell here.
Which is sort of ends in dot data if you use that command, it pulls out the underlying arrays that are sitting beneath sitting with X arrays, a wrapper around X arrays are essentially a wrapper around gridded data, and this is showing you what as a visual representation of the actual data. That's.
Underneath that you're working with and you get more information about it,

including the size of it. So this is a 1 1/2 terabyte data set, so it's not one that we could just read in and read into our memory and work with. We have to be a bit clever.
And that cleverness comes from the second column, which talks about chunks. Now the reason X-ray. I'm sorry. The reason Czar works better on object storage is that it inherently supports chunking of data. So whereas in net CDF you might have several net CDF files to make up a data set.
In ZAR, that that functionality is inbuilt, so one ZAR data set is made-up of lots and lots of different chunks of data and these chunks are the objects that you would find when you put it on an object store.
And you only ever work with the chunks of the data that set that you really really need to. So if you want to extract out a bit of the data, particularly of the data, you don't have to load the entire thing. It just finds the nearest chunk, or chunks that contain those you need, and it will load those and only those.
So you can basically read the specific bytes, give or take that you need from your data set and no more to make sure that you never you don't read in too much data. That's the main reason is this sort of a shift towards this format as data sets get bigger and bigger and harder and harder to handle.
If you're working on a local laptop or you know you know, or even on even on some cloud computer service.
The next thing that's quite important to get our head around is related to what I was just saying and related to what Emily mentioned earlier in that in X-rays lazy loading capabilities.
This is, yeah, this is an important part of the library and it goes a step further when we work with data that's on cloud storage systems.
In particular, with a library known as Desk one more, one more library for you to get your head around, but not really get your head around or introduce. I think this is another thing that we use when we're working with large data sets on the object storage and this.
And the Dask library is something that works underneath the hood, sort of a layer down from what you see when you're typing in your X-ray commands, and it automatically handles. Works out how to best use the compute resources that you've got.
To parallelize your computation and and evaluate it as quickly as possible and as efficiently as possible without overwhelming the memory, the memory in your

computer, the RAM, the memory that you have in your session that you're working with.
We're not going to go into the details of desk here. It's again, it's a module in itself. It's a workshop in itself. There's a lot, a lot that goes into using it well. But by and large, the default behaviour that comes with X-ray whenever you read in a gridded data set from object storage is but.
Is generally fine. You don't need to worry about it too much except for a couple of small points that I've I'm going to go into detail I've detailed here and that is it works in conjunction with X-rays lazy loading. So you mentioned we mentioned earlier that X-ray only ever loads the data.
From disc or in this case from new object storage when you need to see it. So if you type in a command that is say, calculating the mean, it won't actually it will actually do that until you say show me the data, plot it for me, save it to disc. I want to see the numbers.
Until you do that, it doesn't. Actually, it doesn't actually calculate it. It sort of stores it in a buffer, and when you want it to be calculated when you want to see it, then it that's how it calculates it, and that's what desk does as well. So it ties in with this and it will only execute the computation.
That you need to see the data that you've asked for when you've requested to see it. So again, if you do the commands that is, you know take a mean select out this data, process this data in some way, it won't actually compute it until.
Until you you until you see it, until you request to see it and we're going to, you know, I'll show you how this works in a moment. So it's a bit clearer. And then that brings in two function. The two functions of that that you kind of do need to know about.
I would actually say arguably you're only used to know one of them and that's this compute method that I've highlighted here. If you do, don't do it now. But if you do click on it, it will take you some more information about it. But what one?
Not problem, but I was a feature or bug. I'm not sure. I think it's a feature of DAS is that when you if you run a command that says show me the data, show me the numbers, plot it for me it will go away and do that and it will plot it. But then if you use further on down the line you say OK, I want the results of that command. I want that plot. I want those numbers.
I want to do something with that. It will actually go back and calculate it all the way again from the start. From scratch it won't store those numbers in memory because

this is all part of the memory saving function of it. The problem with that is that can be really slow. So if you said I I've calculated the mean.
And now I want to do something with this mean I don't necessarily want it to go back and have to calculate the mean again, especially if it was a large data step where it takes a few minutes. I want that to be stored in memory because it's small enough and that's where using the dot compute.
Method is essential because what that does is it will save those the actual numbers. It will save the actual data in memory rather than just what desk does, which is essentially the instructions for calculating that. If that doesn't make sense, I'm going to go through it now a little bit more detail in an example.
Hopefully it'll become clearer. If not, we can talk it through, talk it through a bit more at the end and there's a lot more information out there that I've written here and in in links you can click through to to read more in, reading more in more detail.
Oh, look, another in Zoke. Insert code here. Bit. OK, let's do that. Let's see what I wanted us to do that I think that is nice and simple. And that is just plotting the data just to just to prove to you that X-ray in pandas work very similar.
You don't really need to worry too much about if you don't want to fancy a plot, you don't need to worry too much about it.
It can actually just plot it.
Let's plot out a particular time step.
So what I want to do is select out a particular time step, so I'm going to this is how you select how variables with X-ray. So we want to select out the rainfall amount variable we want to we can do simple indexing it can do really a lot more fancy. But for now we're just going to keep it simple we know.
So it's if we looked at this earlier, we know is ordered with time YX. So we want the 36th time step. I think actually it's the 7th because Python starts at 0. So we've selected out a time step of data and let's see what that looks like.
But because we're going to be reading things in using DAS because it will any data set you read in which is a Xar file will use DAS under the hood automatically, we can use a progress bar function.
To tell us how long it's taking.
It'll take a few seconds, I think. Hopefully I tried to tell earlier it was taking you for a few seconds or a few minutes, but this is quite useful for getting an idea of how long your computations will take, which as as they get more complex and larger, will will take. It will take a while, therefore having a progress bar is very helpful.

So what this is doing in the background is whilst this is running, is pulling out the chunks of data of the data that you need to do to make this plot, which I think should only be no, it will be a. It'll be five or six chunks worth of data. It will combine them together and it will plot them.
How's that getting on? Does that work for everybody?
Cool. Do shout. If not, I know we're getting a bit close on time, but I think this is quite important. We go through it so that'll be fine. If not, if it is, that's excellent.
Now this next bit demonstrates what I meant earlier, so if I.
What I'm gonna do?
Don't have to feel that you have to copy along here. I'm just going to demonstrate something for you. So I'm going to remove this.
He says doing live coding, which is always dangerous. So if this doesn't work, I apologise. So let's see what looks like.
OK.
So what we've done here is I have selected out the same point as I did before.
Now, in theory, you'd think if I now want to look at that point and see what the data actually is in there, I would just you know, I would just look at the variable when it would be there, but that's not quite how dusk works. This is just that slight difference in behaviour. What it's actually done is it's showing you.
Well, it's showing you a schematic of what the data would look like once it is calculated and it's showing you how many calculations it would need to calculate it, but it hasn't actually calculated it yet. To get it to calculate it. Sorry to get it to calculate.
That's when you had to add the doc, compute to it, and then the actual results of that calculation, which is in this case is just selecting out a particular time point in our data set. Until we do that.
All you'll ever get from that variable.
Talk, talk, talk.
Is not the actual data, but sort of dasks instructions for calculating it, which is not wonderfully helpful if you actually want to work with it. So instead we're going to use the doc compute to actually pull that out all that data into memory, and then we can work with it fine, no problem. Now it's no longer associated with desk, it's its own thing.
We just got the actual data in memory, so whilst I wait for that to happen.
Let's have a look further down and see what we've got. I'm going to Passover this

'cause. I think we did some of it earlier.
This is just again showing you how to plot X-ray based plot with X-ray rather than plotting with with Geo pandas, but it's very very similar, so I'll leave that for the for the interested interested participants to go through themselves and this includes a section on what you might want to do.
To create animations of the data set. So if you want to run through say I want to see what the first month looks like every hour, animate it. For me, I've got some code here that will tell you how to do that. What I'll do is though is show you how that looks like at the end, just as a fun bonus.
We're going to skip over that for now. In the name of time, because there's quite a lot of code and we're all we're going to do is go all the way down to catch 1 extraction.
So if you if you can join me there, what we're going to do here is combine some of the combine the vector data with the gridded data and use.
And use the shape files and use a shape within the shape files to extract out subsets of of our gridded data set.
Which is, yeah, something. It took me a while to learn how to do so, hopefully.
This will make it easier for those going forward, which is what this is all about.
What we're doing here is I've written some code, I've gone away and written some code in the past quite a while ago now, but it still works, which is great, which is a convenience function which does this for you. Otherwise it's quite a long. You need to write in quite a lot of code to make it work nicely, so we'll be using this function that I wrote.
And to get that function we need to download the script file from the from the repository. That's this W get command which you which by now you probably recognise from from the last workshop running that.
Essentially pulls down a script file which contains the function that we're going to use. The next cell imports that function into our notebook, and now we're ready to use it. So first, what we're going to do is read in.
Region the gridded data set. This is a slightly different one. It's actually a climatology that I've calculated from the rainfall. I'd say the main data set we were working with just now, which I've skipped over in the name of time because it does take a while to calculate, but for the the demo, the important the.
But for the purpose of this section doesn't matter what the data is, we're going to have a graded data set that we're going to read in. Again, this might take a few

seconds. Let's see. In this case, it only took four for me. If it. Yeah. If it gets to sort of over a minute.
That's when it's time to cancel and and restart it. That might happen here.
And we're going to read in a shape file that we're going to use to subset that data.
We're going to use the same shape files that we were using from earlier, which was the one with all the catchments in it. And this is one bit where you have to code along with me.
And I'll make sure I've got the right command.
To.
OK.
Yeah. OK. So that's where our shape file is. We're going to read it in using.
S3FS as we have before.
And Google is trying to be helpful by having it all there for me.
I'm just going to type it in and anonymous equals true, which you may recognise Tau it where to look for the data which server the data is stored on, which I've typed in so many times. I almost know by heart now.
See how many typos I can make.
If anyone spotted the typo, do say I'm sure I'll have made one.
None.
That's because I didn't. I did that wrong. There you go.
I still did it wrong.
I tried to make this as simple as possible. Turns out I didn't.
What did I do?
Endpoint should we endpoint URL there we go.
Again, it'll likely take a few seconds, but you can use those seconds to make sure you're typed it in right as you. As you can see, it's very easy to type in the wrong names all the time, but what this is doing is what it did, like what it did before.
When we're in the Exa data workshop or the data part is reading in that shape file, so we can use it for for subsetting and just see what it looks like.
Yeah, you're taking too long, I think.
There we go. He has it in two seconds.
That should look familiar to everybody.
That shape value worked with earlier, and now we're going to look at do the fun stuff and then that will be it. I don't know where we're already slightly over time, so I will stop there.

So there is a function that I have written in my code which we downloaded just a second ago which is called catchment subset shapefile.
Which?
There's some information on it coming up. If you hover over it with your mouse, it seems because I actually was good about this and documented it, which is not something I always remember to do what we're going to do is just use that that function convenience function to pull out.
A part of the gridded data set that we've loaded, which is related to the the hourly rainfall data set. We're going to pull out a part of it and that part of it is going to be a catchment that we select. We'll probably use the Thames just for convenience, but one way of getting more information about a.
A function I just wanted to show here if you're not sure what a function does if the hovering over doesn't work, I think that's a Google collab specific thing. It has it. You can put a question mark after it.
And then run that and it will bring out some more information about how the function works and what you need to use to put into it to use it. So we're going to use that information to fill in this bit of code.
And then we'll be done.
So the code requires us to tell it.
The data set we want a subset which is our climatology data set. The name of the shape file that we're going to be using, which I believe I put came earlier, was actually S file, not SF name.
What is it? SF name? No, it is SF name. Maybe I shouldn't doubt myself. It's just the name of the fire.
Once again, we're going to pass in that pesky endpoint. This time it is endpoint and not endpoint URL. Just to be confusing.
We want to tell it which shape we want to we want to subset to. We're going to do that by saying we want to use the column that's called ID string, which if you remember is one of the many columns that comes with our shape file. It's one of the ways we can identify which shape file I want to use.
And we want to use the particular straight file that has the ID string of 39001 and we're going to add the drop command as well and you can read in the documentation what that does if you're interested.
Valid syntax. What do I do?
To do there shouldn't be brackets there, there should not be a brackets there. There

should be brackets there.
That was if you type in the name of the function first there you go.
Let's move that along to make it look a bit nicer.
I managed to type that in correctly in a bit quicker than I thought edged.
Then you should find that this thing that it's spat out is looks a lot like another X-ray data set. You'll notice that the X&Y coordinates have shrunk a lot. That's because we've we've we've cut out bits of it, and So what we're going to do now is just plot it to prove to you.
But I have indeed cut it out and they and there you go. And then there's some more stuff later on, which basically makes the plot more fancy, which I don't think we need to do. Did it before 7 minutes over, given the amount of stuff I had to cover, I think we did all right.
If you people are happy to stay on, I'm happy to answer questions. You feel free to make use of these notebooks a lot. I didn't cover and ask questions that can be in the chat for us to answer offline by e-mail for us to pass if Sammy to pass on to us later.
Or if you, I'd recommend having a break, but if you want to ask them now, by all means do. But yeah, I think I'll stop there for now. I hope that was useful.

Samantha Rees 1:35:47
We have got a hand up.

SG

Subhajit Ghosh 1:35:49
Yeah. How much? Yeah. It's not related to the exercise. Basically, I want to know why do you abandoned the previous tools like we use CDO or grads in CLM.

Matt Dalle Piagge 1:35:49
Yeah, go for it.
That's OK.

SG

Subhajit Ghosh 1:36:05
Choose Python in this is it really giving you any computational advantage or or the just particularly for your organisation you are working with different type of data sets. That's why you choose this part.

Matt Dalle Piagge 1:36:20

I think those, yeah, those software I'm I'm loosely aware of them. They do still exist and I think I think they are still quite well used. For me personally it's essentially this is what I'm used to working with. I I learned Python And actually I grew up with it. I didn't quite I I learned Python as part of my job.

Yeah, as as I you know, change jobs and whatnot, but ultimately for working with now no longer speaking personally, but now that data sets are getting bigger, the original software that we've we've had to, we could work with before is no longer really sufficient. It's going to struggle with someone.

And these data sets and that's where using Python gives you the flexibility and the and the capability of working with data sets that are much, much larger than this software could could use before. I think that's the main reason what why? To be honest, some of the yeah, some of the other things.

That I've shown here you could probably do with that those software I like the flexibility of Python ultimately.

Amulya Chevuturi 1:37:25

Just to add to what Matt has said and what Matt has posted on in the chat, definitely I would agree with all the points that they're making, but also I used grads and CDO during my PhD. And once you start using Python you realise it's like one stop shop for everything.

Matt Dalle Piagge 1:37:26

Go for it.

Amulya Chevuturi 1:37:44

CDO, ultimately, you know you end up having like multiple files output files, so you process something, you get an output file, then you process it, then you know you have a lot. You generate a lot of data and grads won't have this remote access functionality which in all of these.

Python have and most of it. Most of the important point is Python is free and Python has like a lot of community support. So anything you want to do you would find information online. So I would suggest if you're looking for making a switch, Python is a pretty good.

Package. I'm sorry. A language to turn to another good language in hydrology is R, particularly Matt, and I don't use it, but a lot of people within the hydrology sphere sphere use R so that's another language that you can explore if you want to.

Subhajit Ghosh 1:38:41
Thank you.

Matt Dalle Piagge 1:38:49
All right. If no further questions, go have a break.

Samantha Rees 1:38:54
Thank you. So our next session is on zoom. So I'll put a link in in the in the in this chat again so that you can access it and it's about communicating with science. Yeah, communicating science not with science, communicating science. So I'll put the chat in there now, not the chat. I'll put the link in the chat now and if you have any questions about this session.
Yeah, send it to me and I can pass it on to Matt and and the rest of the workshop leads. Thank you.

Ashling Laffey 1:39:23
Thanks very much.

Samantha Rees 1:39:24
Thank you everyone.

Amulya Chevuturi 1:39:42
Do we wait till? OK, everyone's gone. Yeah. Yeah. Bye.

Matt Dalle Piagge 1:39:44
Good. I think everyone gone have a tea break now.

Samantha Rees 1:39:45
No, you can. You can. Yeah. No, you can see you later. Thank you so much.
Yeah.

◉ **Samantha Rees** stopped transcription