Met Office INFORMATICS LAB

You do you.

How next-gen data platforms can stop weather and climate scientists from being software engineers and other perversions

Theo McCaie - Data Engineering Research Lead. UK Met Office and Joint Center for Excellence in Environmental Intelligence

hello@informaticslab.co.uk @informatics_lab informaticslab.co.uk

80% of data science is...

...data munging.

...formating

...cleaning

...wrangling

...etc

...validation

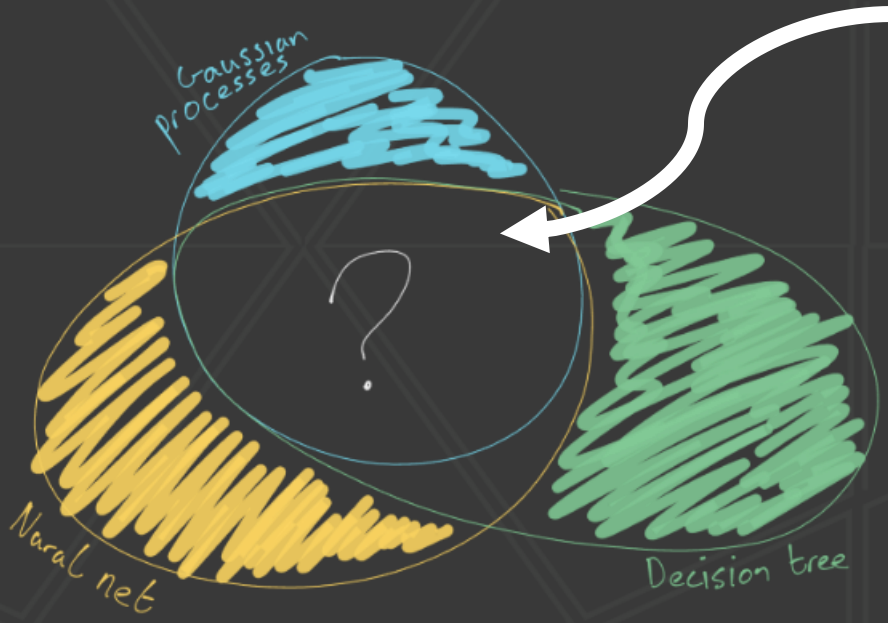
...discovery

80% of data science is data wrangling

Data wrangling:

- Acknowledged as data science
- But oft distanfally
- But if you could make one part of this diagram 50% more efficient which would have most impact…

More the same than different



Most machine learning approaches share many of the same foundations

- Access to clean, organised, data
- Pipelines to format data
- Compute infrastructure to run on
- Running many experiments e.g. hyperparameter tuning

This is the realm of Data Engineering

Met Office INFORMATICS LAB

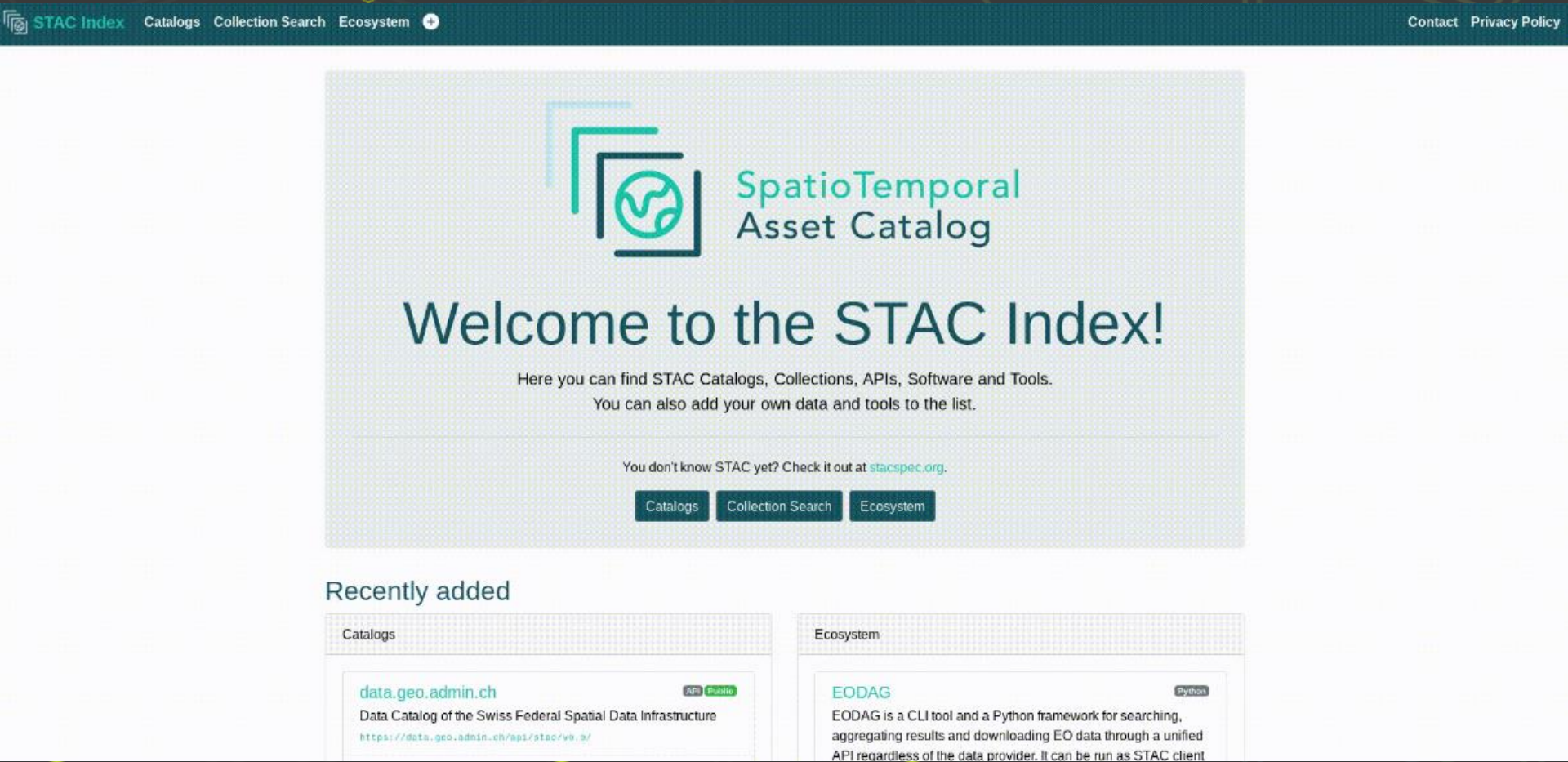
Better data engineering will be the most effective accelerant of data science.

hello@informaticslab.co.uk @informatics_lab informaticslab.co.uk

STAC

SpatioTemporal Asset Catalogs:

- “a common language… more easily be indexed and discovered”
- “new code doesn't need to be written whenever a new data set or API is released.”
- Seeks to remove burdand of building unique pipelines, spur common tooling.


STAC Index
Catalogs
Collection Search
Ecosystem
Contact
Privacy Policy
SpatioTemporal
Asset Catalog
Welcome to the STAC Index!
Here you can find STAC Catalogs, Collections, APIs, Software and Tools.
You can also add your own data and tools to the list.
You don't know STAC yet? Check it out at stacspec.org.
Catalogs
Collection Search
Ecosystem
Recently added
Catalogs
data.geo.admin.ch
API Public
Data Catalog of the Swiss Federal Spatial Data Infrastructure
https://data.geo.admin.ch/api/stac/v0.9/
Ecosystem
EODAG
Python
EODAG is a CLI tool and a Python framework for searching, aggregating results and downloading EO data through a unified API regardless of the data provider. It can be run as STAC client

```
In [1]: import nc_ds_utils
        import numpy as np
        import iris
        import datetime
        from time import mktime
        import os
```

STASH

source

Batch by time

```
In [8]: start, stop = next(iter(time_window_batch((1851, 1, 1, 0, 0), (1861, 1, 1, 0, 0), datetime.ti
        (hours=24*50))))
        filter_files_for_dates(FILES, start, stop)

Out[8]: ['/data/cssp-china/mini-dataset-24-01-19/nc/hourly/apepda.pj51150.nc',
         '/data/cssp-china/mini-dataset-24-01-19/nc/hourly/apepda.pj511f0.nc',
```

Mapping the work so there i

Iris can not open the same HDF5 file (NetCD
the tasks without overlapping the files being a

```
In [11]: TaskSet = namedtuple('TaskSet',['inc

         def create_task_set(i, start,stop):
             return TaskSet(i, start, stop, f

         windows = time_window_batch((1851, 1
                                      (1861, 1
                                     step=dat
                                    overlap=0

         tasks = [create_task_set(i, start, s
```

```
In [12]: def do_overlap(task, other_tasks):
             for other_task in other_tasks:
                 for file in task.files:
                     if file in other_task.fi
                         return True
             else:
                 return False

         task_sets = []
         for task in tasks:
             for task_set in task_sets:
                 if not do_overlap(task, task
                     task_set.append(task)
                     break
             else:
                 task_sets.append([task])
         len(task_sets)

Out[12]: 2
```

Process the tasksets on distributed cluster

```
In [13]: Result = namedtuple('Result', ['success', 'value'])

         def error_catch_rmean(start, stop):
             try:
                 result = rolling_mean_at_point_for_date_range(start, stop)
                 return Result(True, result)
             except Exception as e:
                 return Result(False, e)
```

```
In [14]: DataSet = namedtuple("DataSet",

         def to_datasets(i, result):
             if result.success:
                 with dask.config.set(sch
                     cube = result.value
                     data = cube.data
                     time_unit = cube.coo
                     times_raw = cube.coo
                     times = [datetime.da

                     return DataSet(data,
             else:
                 return result
```

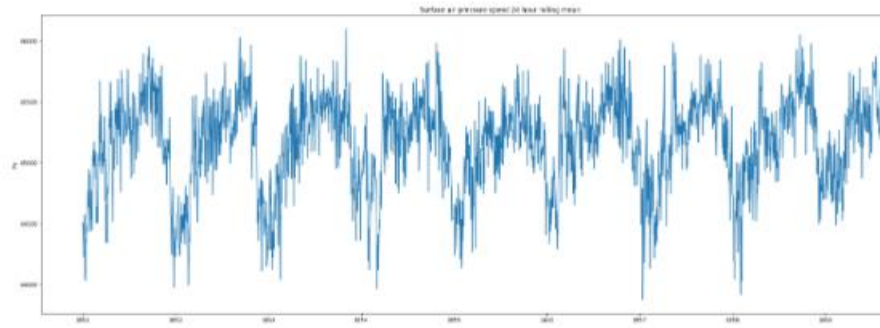
```
In [15]: import distributed
         from dask_kubernetes import Kube
         import dask.bag as db
```

```
In [16]: cluster = KubeCluster()
         cluster.scale_up(20)
         display(cluster)

         VBox(children=(HTML(value='<h2>K
         >\n    ...
```

```
In [19]: %%time
         collections = []
         for task_set in task_sets:
```

Sort results and plot

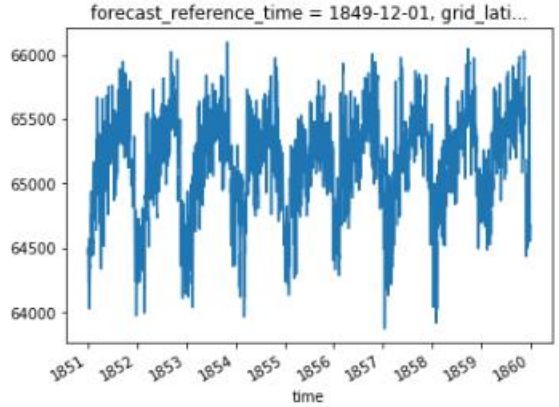
```
In [18]: %%time
         sorted_ds = sorted([ds for ds in chain(*collections) if isinstance(ds,DataSet)], key=lambda x
         datas, times = zip(*sorted_ds)
         plt.title("Surface air pressure speed 24 hour rolling mean")
         plt.ylabel(sample_cube.units)
         plt.plot(list(chain(*times)), list(chain(*datas)))

         CPU times: user 469 ms, sys: 4.55 ms, total: 474 ms
         Wall time: 467 ms

Out[18]: [<matplotlib.lines.Line2D at 0x7fe66aeb9210>]
```



In [2]:
```
import zarr
import xarray as xr
%matplotlib inline
```

In [12]:
```
zarr_store = zarr.storage.ABSStore('cssp-china', prefix='zarr_hourly_1851-1859', account_name=accoun
t, account_key=key)
ds = xr.open_zarr(zarr_store)
ds.surface_air_pressure[:,100,100].rolling(time=24, center=False).mean().plot()
```

Out[12]: [<matplotlib.lines.Line2D at 0x7f70f056e950>]



Analysis Ready Data

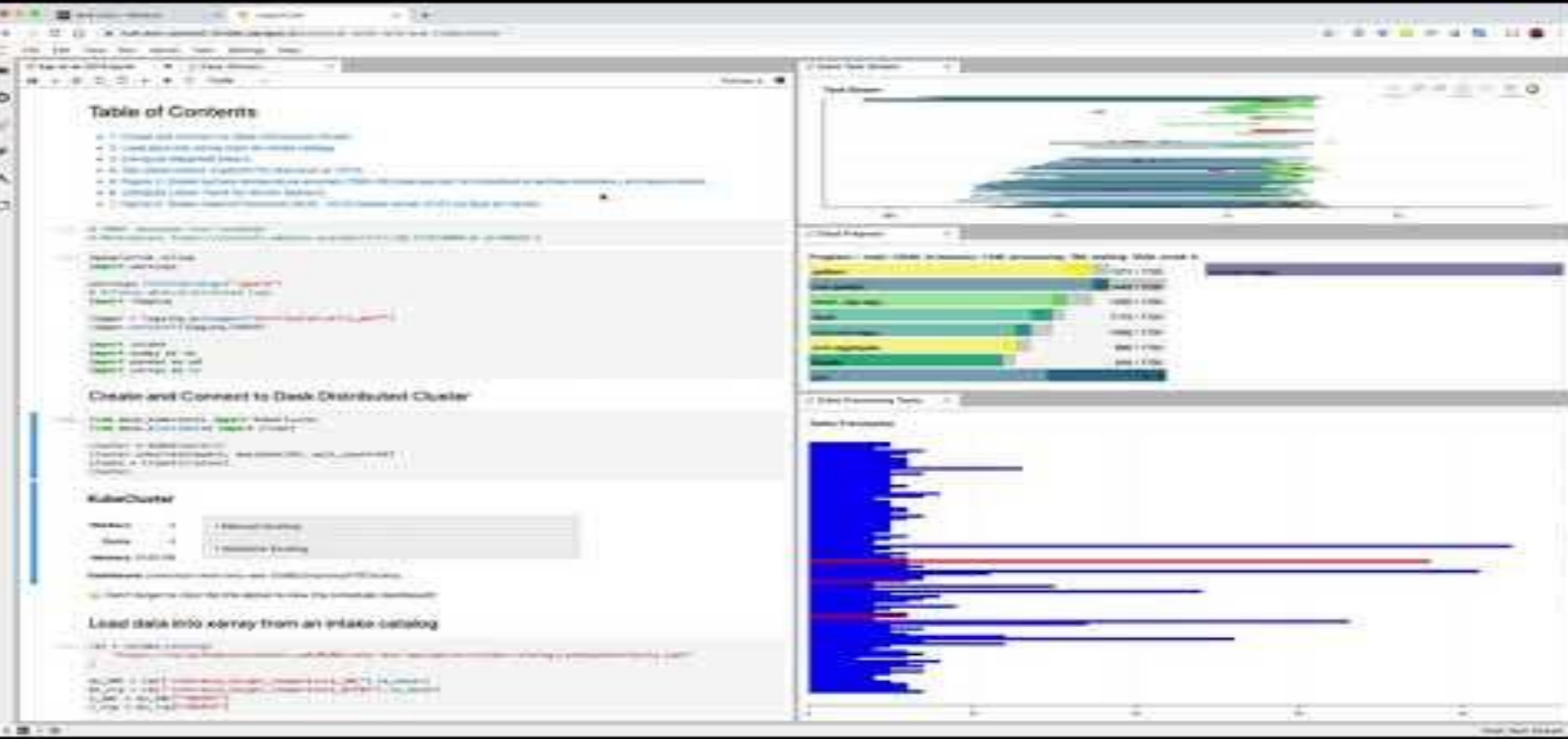
- Analysis Ready data is a dataset that is trivial to load and analyse
- ARCO (Analysis Ready Cloud Optimized) takes this to the next step ensuring that the data is efficiently to use on public cloud infrastructure.
- ARCO applies to a whole dataset (many thousands of millions of files/blobs/chunks)
- Data provider does most of the cleaning, amalgamating, indexing, etc
- Some example formats: Zarr, TileDB

Pangeo

- Open community
- Open source software
- Open source infrastructure
- Have working groups on ML amongst others

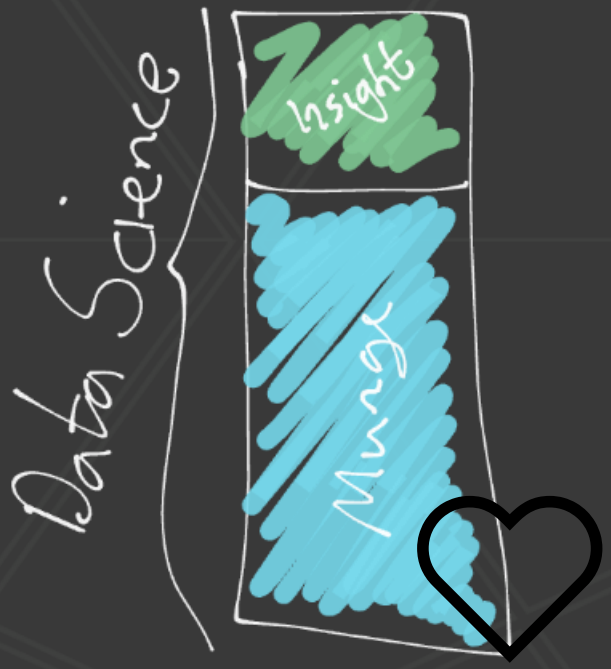
"Pangeo is first and foremost a community of people working collaboratively to develop software and infrastructure to enable Big Data geoscience research."

https://pangeo.io/ — https://discourse.pangeo.io/



Source: Joe Hamman - https://www.youtube.com/watch?v=Afri3dNQSoQ

Conclusions



- Give love to data wrangling
- Embrace and engage with data engineering efforts
- This will accelerate the data science revolution faster than anything else

Thank you!

Links:

- This presentation: https://bit.ly/3ePaJLE
- Me: Tho McCaie theo.mccaie@informaticslab.co.uk
- Pangeo: https://pangeo.io/
- STAC: https://stacindex.org/
- Blog about Analysis Ready Data
- More work from my team: https://medium.com/informatics-lab