melodies
exploiting open data

University of Reading

Building rich and interactive web applications with CoverageJSON

Jon Blower, Maik Riechert, Guy Griffiths,
Mridul Kumar, Riley Williams

Institute for Environmental Analytics,
University of Reading, UK

European Commission

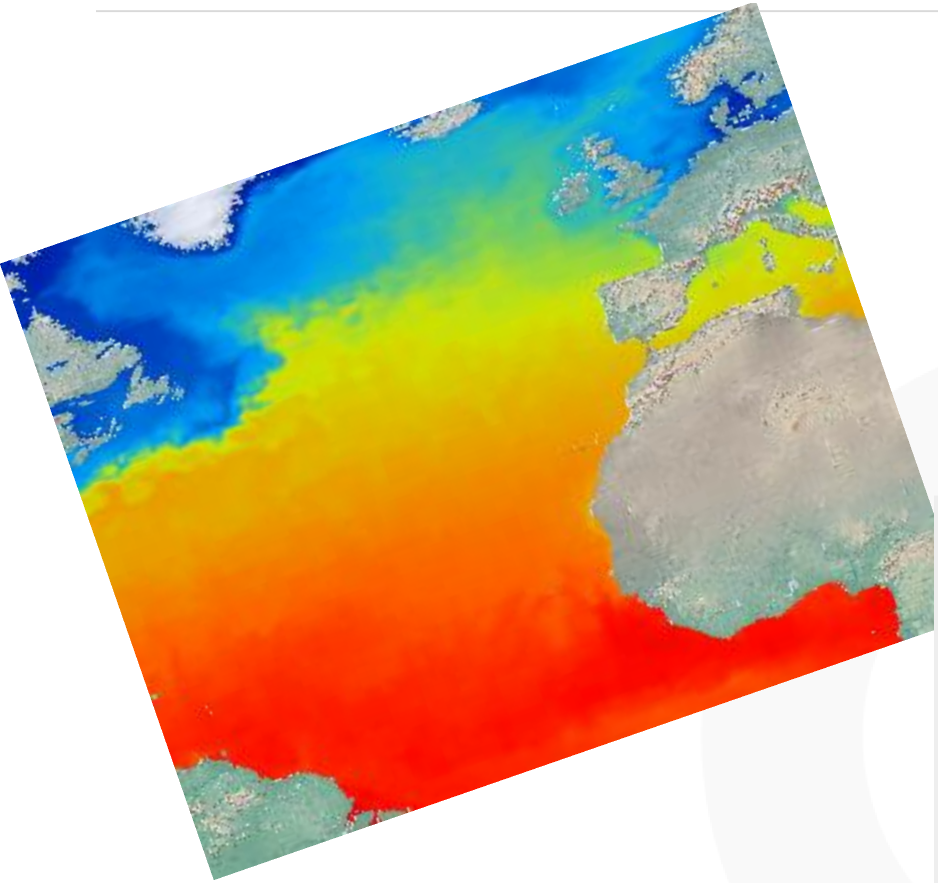
Introduction

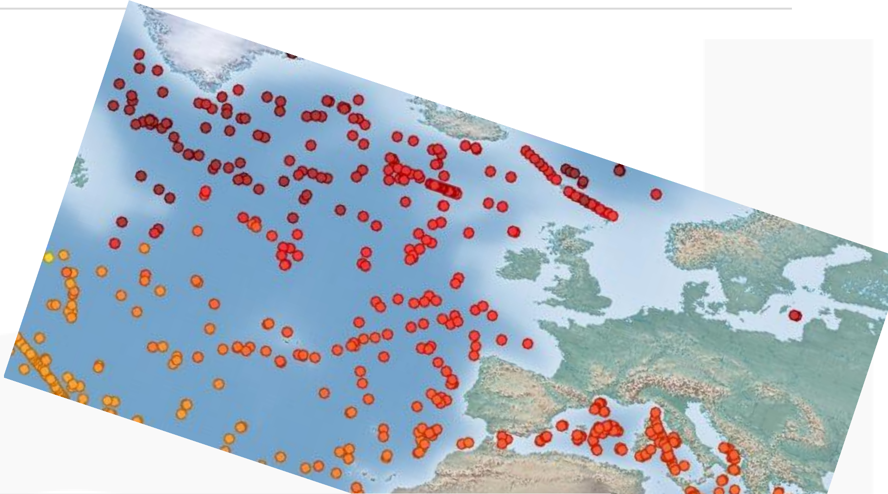
- Web browsers are becoming increasingly capable as visualisation and analysis platforms
- Lots of tools and libraries are built around images and “simple features”
  - GeoJSON, KML, OpenLayers, Leaflet ...
- Formats and tools for scientific / meteorological **data** are not always web-friendly
  - Complex, binary, desktop-oriented
  - Large variety, usually community-specific

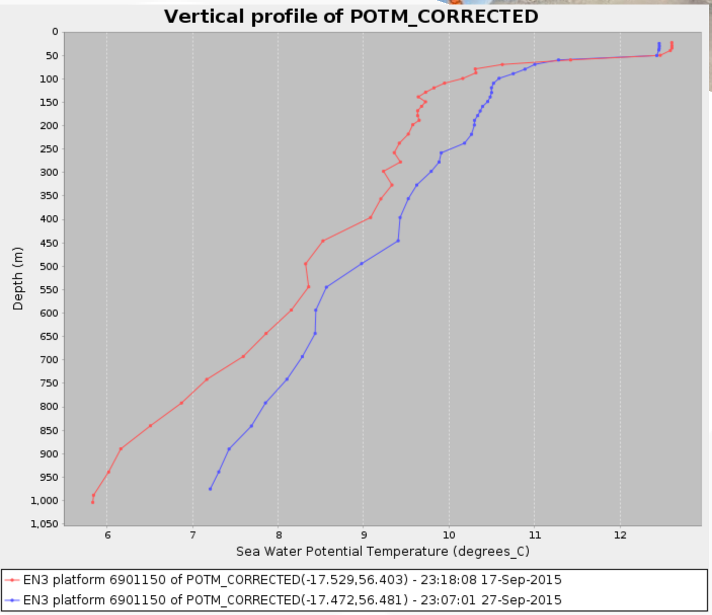
- => Lots of people building ad-hoc solutions for web applications

- We want to bring scientific data within the reach of more Web and mobile app developers
  - Web-friendly formats (i.e. JSON)
  - More powerful and reusable visualisation/analysis tools

“Coverages”: a unifying concept






Vertical profile of POTM_CORRECTED
Depth (m)
0
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1,000
1,050
6
7
8
9
10
11
12
Sea Water Potential Temperature (degrees_C)
EN3 platform 6901150 of POTM_CORRECTED(-17.529,56.403) - 23:18:08 17-Sep-2015
EN3 platform 6901150 of POTM_CORRECTED(-17.472,56.481) - 23:07:01 27-Sep-2015

data = f(position, time, ...)

The CoverageJSON data format

- **Rich and efficient JSON** encoding of coverage data
- “As simple as possible but no simpler“

- Gridded and non-gridded data
- n-Dimensional data
- Continuous and categorical data
- Internationalisation
- Embedded semantics
  - (some interoperability with RDF through JSON-LD)

Skeleton CoverageJSON document

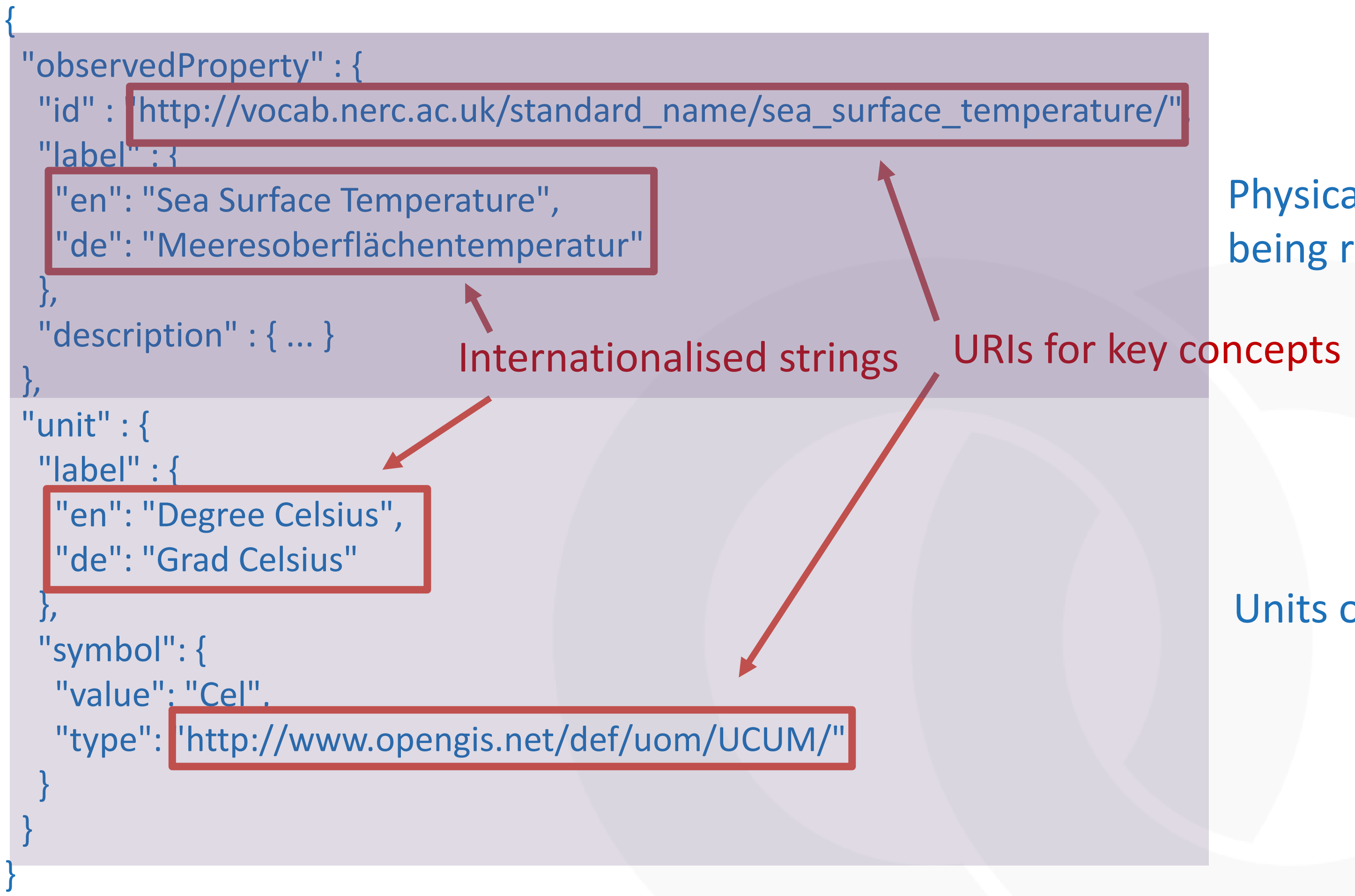
```
{
  "domain" : {
    ...
    "referencing" : [ ... ]
  },
  "parameters" : {
    ”SST" : { ... },
    ”sea_ice" : { ... }
  },
  "ranges" : {
    ”SST" : { ... },
    ”sea_ice" : { ... }
  }
}
```

Coordinates of data points and referencing information

Metadata describing data values

Data values as nD arrays

Metadata sample



```
{
 "observedProperty" : {
  "id" : "http://vocab.nerc.ac.uk/standard_name/sea_surface_temperature/",
  "label" : {
   "en": "Sea Surface Temperature",
   "de": "Meeresoberflächentemperatur"
  },
  "description" : { ... }
 },
 "unit" : {
  "label" : {
   "en": "Degree Celsius",
   "de": "Grad Celsius"
  },
  "symbol": {
   "value": "Cel",
   "type": "http://www.opengis.net/def/uom/UCUM/"
  }
 }
}
```

Internationalised strings

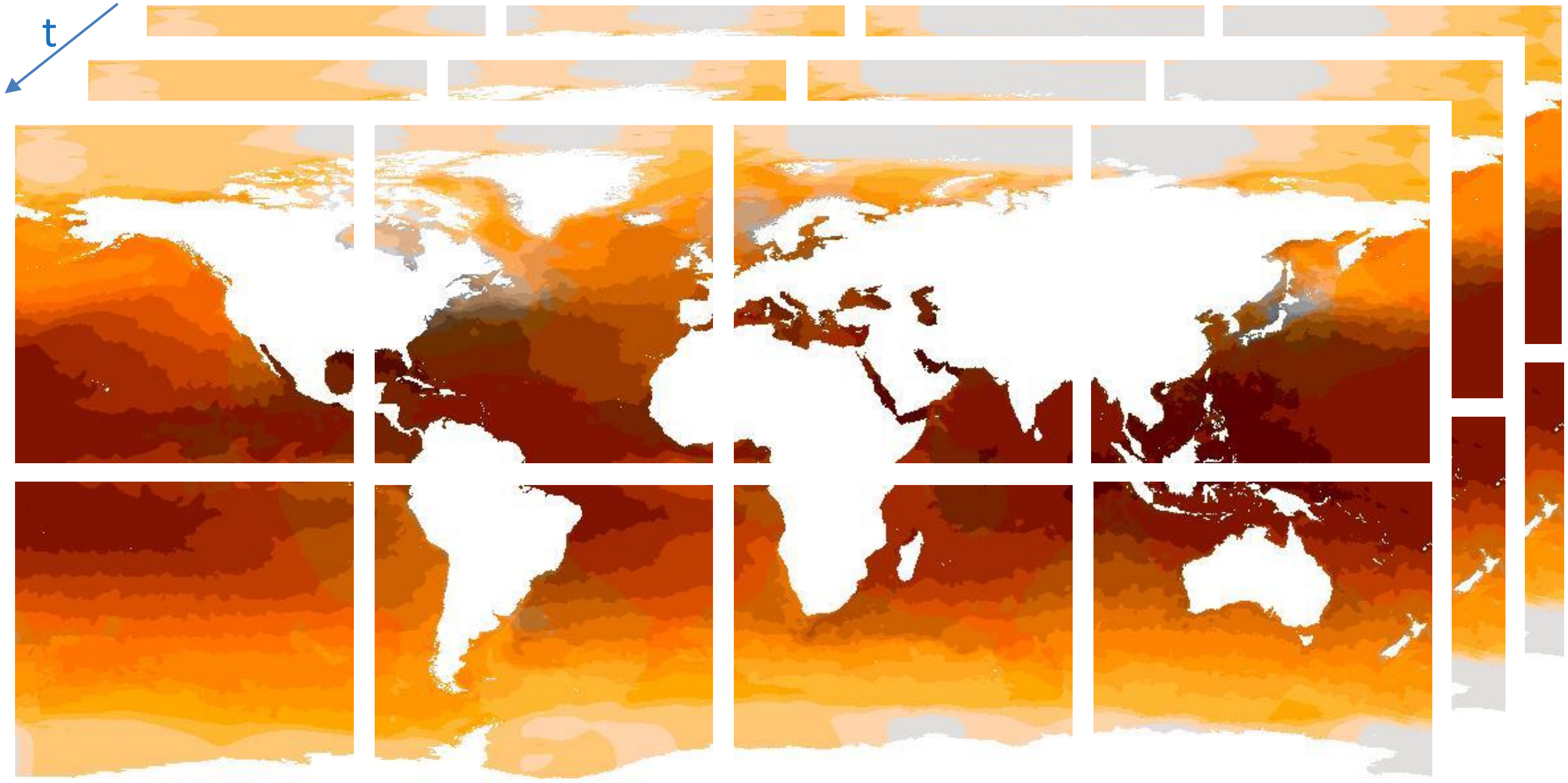
URIs for key concepts

Physical quantity being recorded

Units of measure

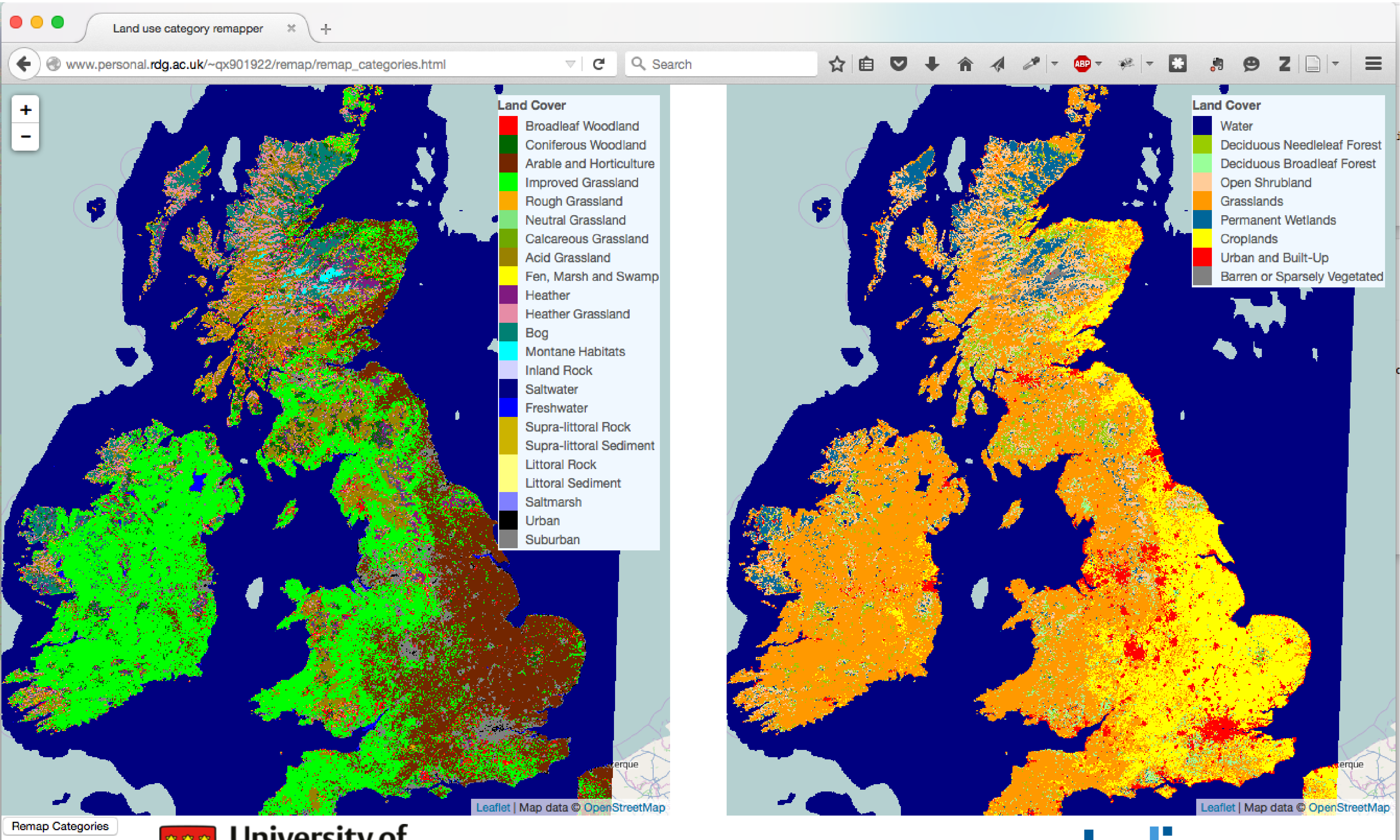
Scalability through tiling

- Large data files can be split into several JSON documents
- Each document holds part of the nD array
- Reduces need to run complex servers (cf. Web Map Tiling)



Interactive, in-browser reclassification of land cover maps

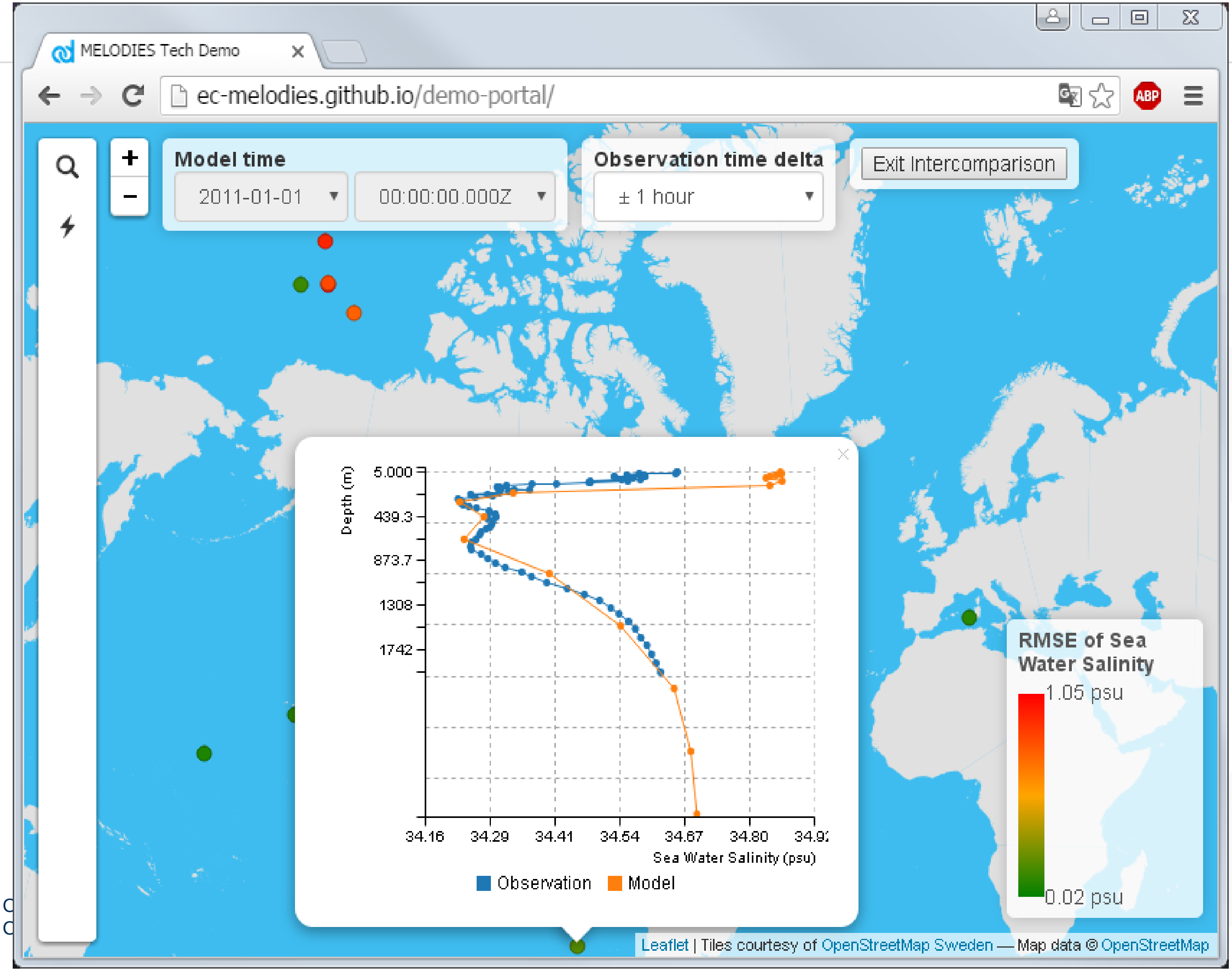
youtu.be/dxfmTkBdn90



University of Reading

melodies
exploiting open data

In-browser intercomparison of models and observations



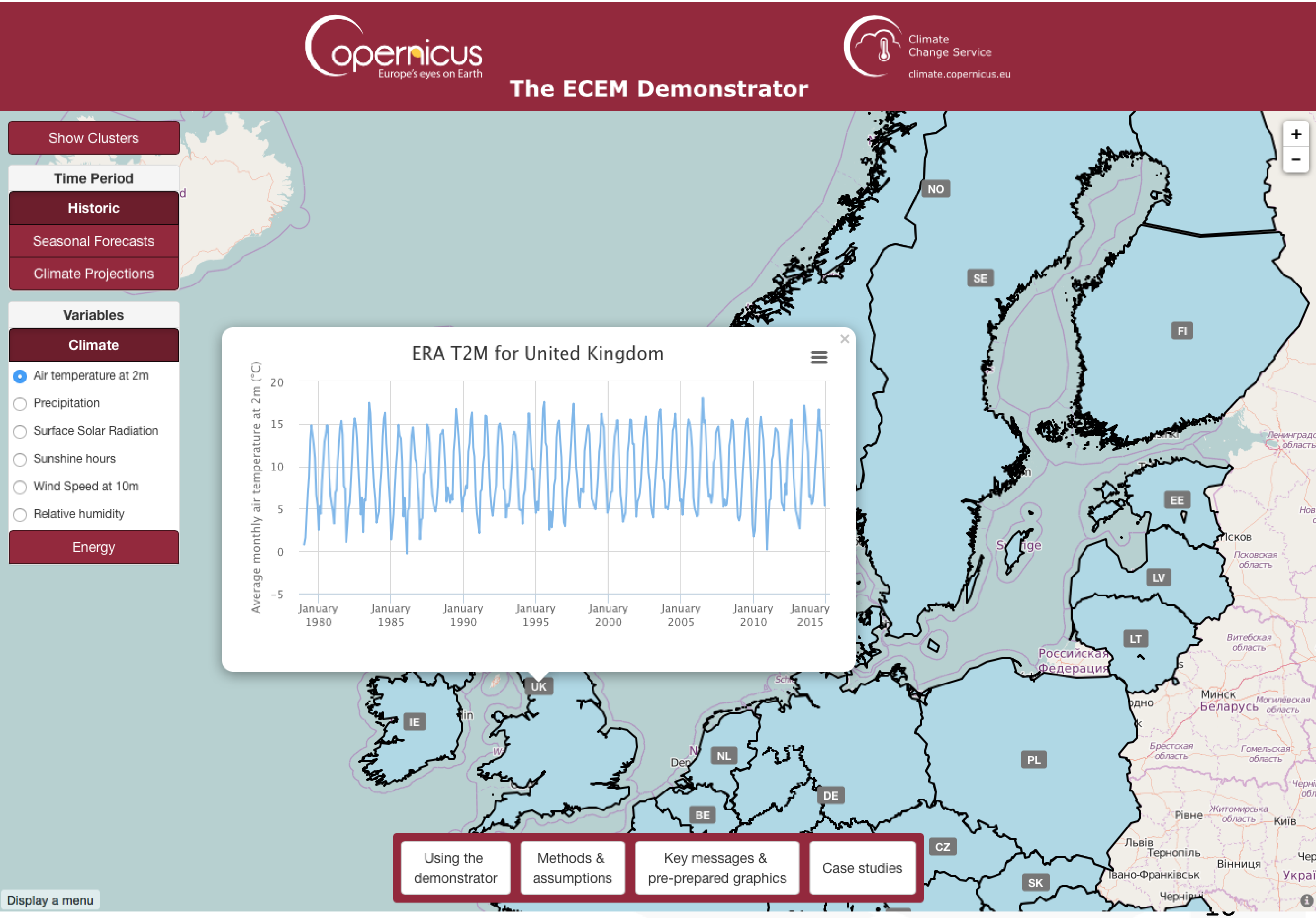
MELODIES Tech Demo

ec-melodies.github.io/demo-portal/

Model time: 2011-01-01 00:00:00.000Z

Observation time delta: ± 1 hour

Exit Intercomparison



The ECEM Demonstrator

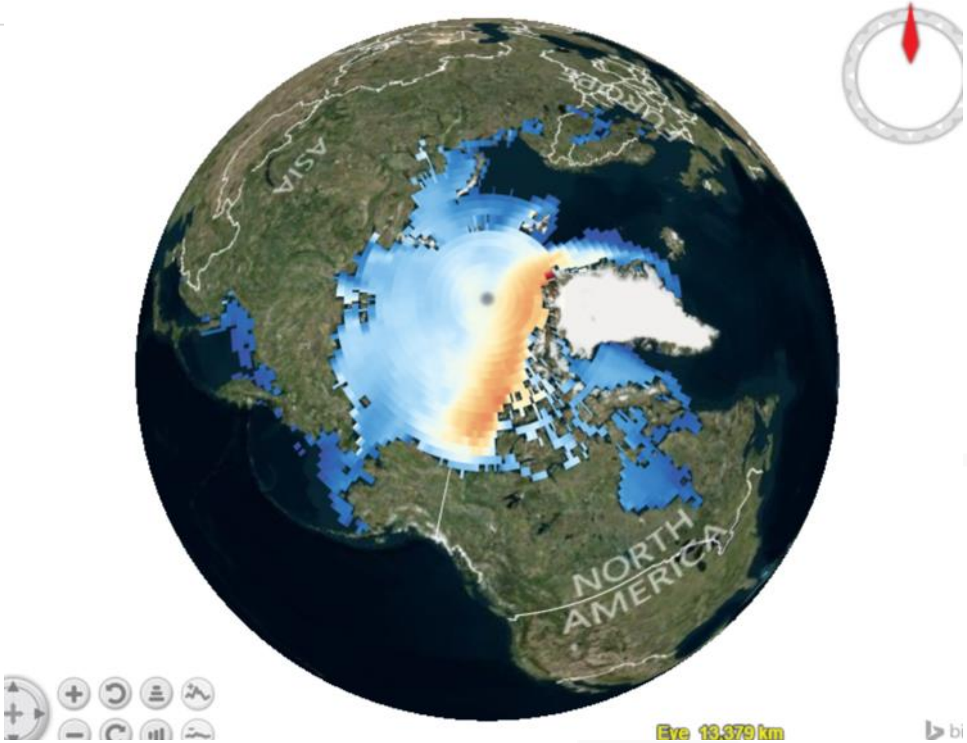
Time Period: Historic

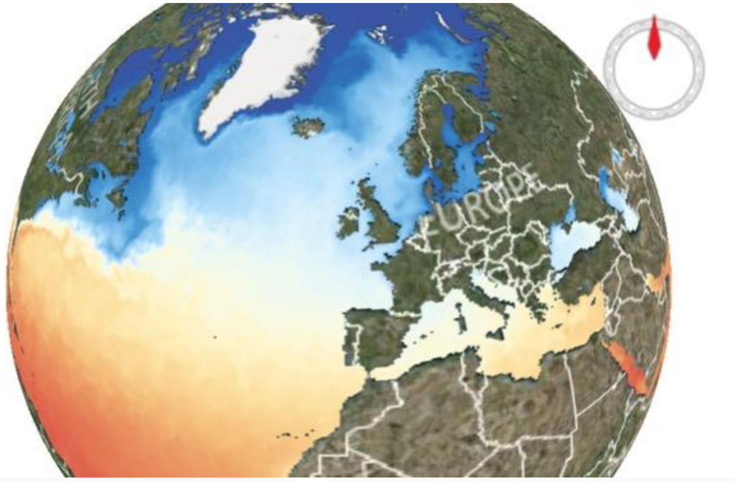
Variables: Climate

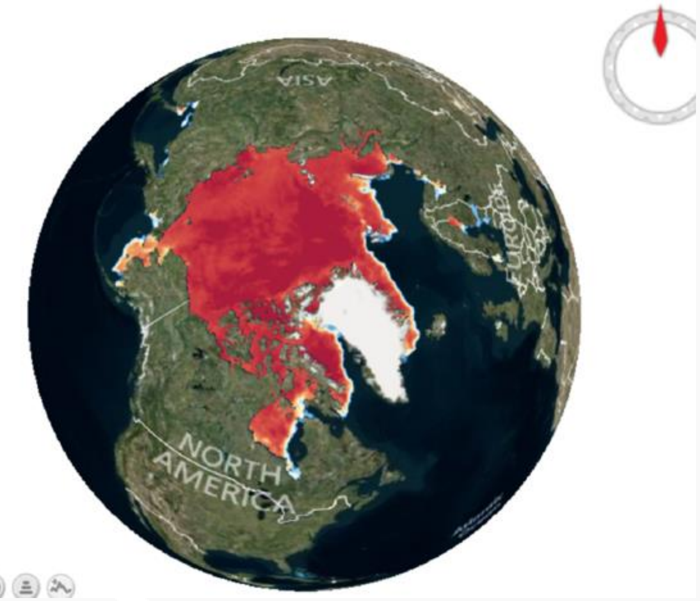
● Air temperature at 2m

ERA T2M for United Kingdom

NASA Web World Wind and CovJSON







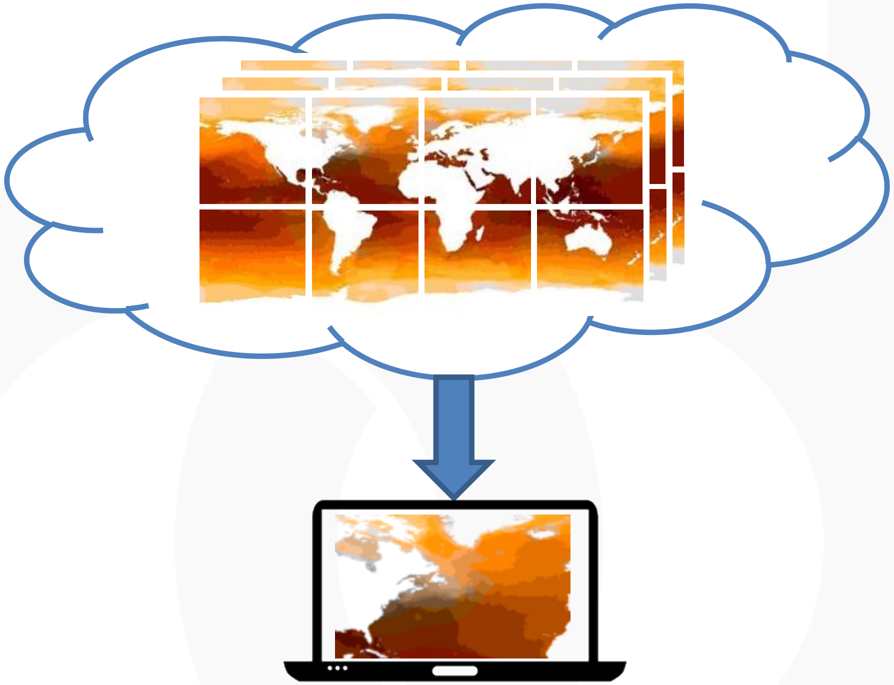
https://webworldwind.org

Beyond visualisation: Big Data analytics over the Web

- Put CovJSON tiles on a web server
  - or content delivery network
- Write simple analysis script in Python
  - Use Dask to treat tiles as single virtual dataset
  - Dask automatically downloads *only the required tiles*

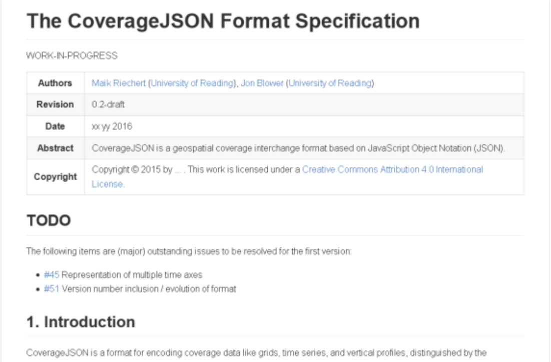
=> work on big datasets more easily, **without the need for a complex server**

Calculate mean sea surface temperature over certain region:



```
dataset = getDataset("http://my.covjson.doc")
sst = dataset["sst"]
result = da.mean(sst[0,:450,:]).compute()
```

CoverageJSON Resources



The CoverageJSON Format Specification

WORK-IN-PROGRESS

Authors	Maik Riechert (University of Reading), Jon Blower (University of Reading)
Revision	0.2-draft
Date	xx yy 2016
Abstract	CoverageJSON is a geospatial coverage interchange format based on JavaScript Object Notation (JSON).
Copyright	Copyright © 2015 by ... . This work is licensed under a Creative Commons Attribution 4.0 International License.

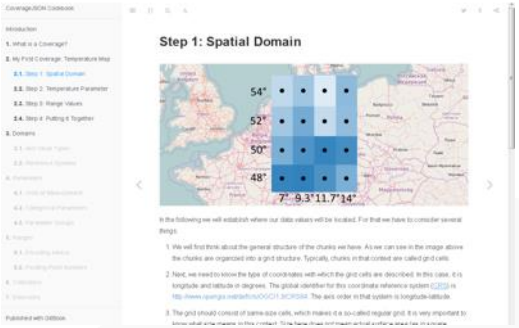
TODO

The following items are (major) outstanding issues to be resolved for the first version:

- #45 Representation of multiple time axes
- #51 Version number inclusion / evolution of format

1. Introduction

CoverageJSON is a format for encoding coverage data like grids, time series, and vertical profiles, distinguished by the



CoverageJSON Cookbook

Introduction

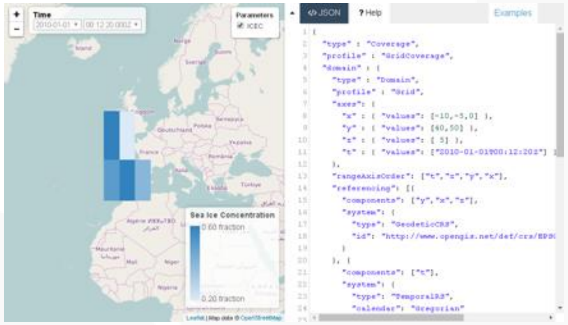
1. What is a Coverage?
2. My First Coverage: Temperature Map
   2.1. Step 1: Spatial Domain
   2.2. Step 2: Temperature Parameter
   2.3. Step 3: Range Values
   2.4. Step 4: Putting it Together
3. Domains

Published with GitBook

Step 1: Spatial Domain

In the following we will establish where our data values will be located. For that we have to consider several things:

1. We will first think about the general structure of the chunks we have. As we can see in the image above the chunks are organized into a grid structure. Typically, chunks in that context are called grid cells.
2. Next, we need to know the type of coordinates with which the grid cells are described. In this case, it is longitude and latitude in degrees. The global identifier for this coordinate reference system (CRS) is http://www.opengis.net/def/crs/OGC/1.3/CRS84. The axis order in that system is longitude-latitude.
3. The grid should consist of same-size cells, which makes it a so-called regular grid. It is very important to know what size means in this context. Size here does not mean actual surface area (as in square



```
{
  "type" : "Coverage",
  "profile" : "GridCoverage",
  "domain" : {
    "type" : "Domain",
    "profile" : "Grid",
    "axes": {
      "x" : { "values": [-10,-5,0] },
      "y" : { "values": [40,50] },
      "z" : { "values": [ 5] },
      "t" : { "values": ["2010-01-01T00:12:20Z"] }
    },
    "rangeAxisOrder": ["t","z","y","x"],
    "referencing": [{
      "components": ["y","x","z"],
      "system": {
        "type": "GeodeticCRS",
        "id": "http://www.opengis.net/def/crs/EPSG
      }
    }, {
      "components": ["t"],
      "system": {
        "type": "TemporalRS",
        "calendar": "Gregorian"
```

Specification

Cookbook

(start here!)

Playground

https://covjson.org

Tools (https://covjson.org/tools)

- JavaScript data-reading library
- Visualisation
  - Leaflet plugin
  - Web World Wind demos
- Conversion
  - Python library to convert from NetCDF to CovJSON
  - Java libraries
- Servers
  - Export CoverageJSON from ncWMS/THREDDS

CovJSON vs OPeNDAP etc

- OPeNDAP can also deliver data to web clients
  - Binary format
  - Requires special server (e.g. Hyrax, THREDDS)

- CovJSON has pros and cons vs OPeNDAP
  - More friendly format for web developers
  - Can be served as static documents for scalability (with tiling)
  - Better support for semantic content
  - Less efficient (but compression helps a lot)

- Remember: CovJSON is just a format, which can be created in many different ways
  - On-the-fly or statically
  - Hence can be used as output format from THREDDS, WCS etc.

Conclusions

- CoverageJSON is a *simple but not simplified* format
  - Handles many kinds of data, include satellite images, derived products, in situ observations, numerical model data ...
  - A bit like a JSON version of NetCDF, with enhancements
  - Friendly for web developers
  - Supported by documentation, tools and examples
- Will be published soon as joint OGC/W3C document
- Potential future output format for Web Coverage Service
- We want to enable the community to generate new and exciting data-driven websites and apps!

Thank you!

@Jon_Blower

https://covjson.org

http://www.melodiesproject.eu