2021 Edition

CHESS Course: An Introduction to FAIR Data Management for Geoscientists, Syllabus

Day 2 - Session 2

Exploiting data for analysis

Torill Hamre, Nansen Environmental and Remote Sensing Center

NorDataNet
Norwegian Scientific Data Network

Session Topics

- Interfaces to data
  - OGC interfaces (WMS, WFS, WCS)
  - OPeNDAP
  - RESTful (Restful in general)
  - OpenAPI Specification (OAS)
- Benefits of using interoperable data.
- Integration in tools using Python, R
- Jupyter Notebook
- Example - aggregating daily data to monthly means

NorDataNet
Norwegian Scientific Data Network

Learning Objectives

- Familiarise with some standard protocols for data access
- Looking into some widely used protocols
- Understanding why standardising access protocols is important
- Investigate examples of using access protocols in Python, R, matlab
- Aggregate daily to monthly data

Assignment 2:
Exploiting data available through standard protocols, possibly in combination with your/course datasets

NorDataNet
Norwegian Scientific Data Network

Interfaces to data

- NorDataNet (and other infrastructures) offer standard interfaces to data, through e.g. OPeNDAP ("Open-source Project for a Network Data Access Protocol")
    - Metadata
        - Dataset Attribute Structure (DAS)
        - Dataset Descriptor Structure (DDS)
    - Data
        - Values (DataDDS)
- Other standards often supported by scientific data providers include
    - HTTP(S), FTP
    - OGC WMS, WFS, WCS

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

- “The Open Geospatial Consortium (OGC) is an international consortium of more than 500 businesses, government agencies, research organizations, and universities driven to make geospatial (location) information and services FAIR - Findable, Accessible, Interoperable, and Reusable.” (OGC About)
- Experts collaborate on developing standards for geospatial data and services; all specifications are open and free at https://www.ogc.org/docs/is
- Implemented by numerous tools/systems, Open Source and commercial
- Sample standards for data access:
  - **Web Mapping Service (WMS)** - returns data on a map (raster image)
  - **Web Feature Service (WFS)** - returns vector data as a file
  - **Web Coverage Service (WCS)** - returns gridded data as a file

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Mapping Service (WMS)

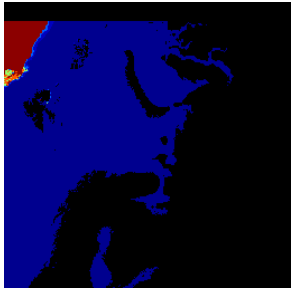
- Interface:
  - **GetCapabilities (Mandatory)**
    - Arguments: VERSION=version, SERVICE=WMS, REQUEST=GetCapabilities, FORMAT=MIME_type, UPDATESEQUENCE=string
  - **GetMap (Mandatory)**
    - Arguments: VERSION=1.3.0, SERVICE=WMS, REQUEST=GetMap, LAYERS=layer_list, STYLES=style_list, CRS=namespace:identifier, BBOX=minx,miny,maxx,maxy, WIDTH=output_width, HEIGHT=output_height, FORMAT=output_format, TRANSPARENT=TRUE|FALSE, BGCOLOR=color_value, EXCEPTIONS=exception_format, TIME=time, ELEVATION=elevation, ...
  - **GetFeatureInfo (Optional)**
    - Arguments: VERSION=1.3.0, SERVICE=WMS, REQUEST=GetFeatureInfo,QUERY_LAYERS=layer_list, INFO_FORMAT=output_format, FEATURE_COUNT=number, I=pixel_column,J=pixel_row, J=pixel_row, EXCEPTIONS=exception_format
- Implemented by many systems and tools, e.g. ncWMS (bundled with Thredds), GeoServer, MapServer, QGIS, ArcGIS.

NorDataNet
Norwegian Scientific Data Network

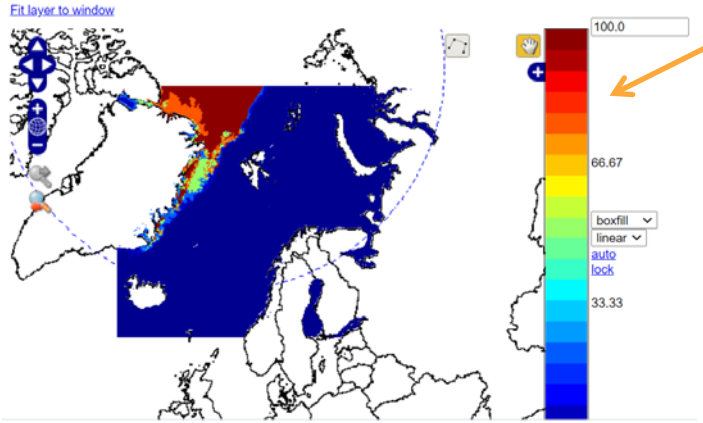
OGC interfaces

Web Mapping Service (WMS) - Example

- https://thredds.met.no/thredds/wms/sea_ice/SIW-METNO-ARC-SEAICE_HR-OBS/ice_conc_svalbard_aggregated?service=WMS&version=1.3.0&request=GetCapabilities returns an XML file with the WMS capabiltiiles (what operations are supported, what image formats are supported, layer list, etc.)
- https://thredds.met.no/thredds/wms/sea_ice/SIW-METNO-ARC-SEAICE_HR-OBS/ice_conc_svalbard_aggregated?SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&CRS=EPSG:32661&BBOX=1012565.33,358281.13,2575886.32,1834909.74&WIDTH=600&HEIGHT=500&LAYERS=ice_concentration&ELEVATION=0&TIME=2020-09-07T14%3A00%3A00.000Z&TRANSPARENT=true&STYLES=boxfill%2Frainbow&COLORSCALERANGE=0%2C100&NUMCOLORBANDS=20&FORMAT=image/png (returns an image of the ice chart; addtional call needed to genrate the legend)



(N call)



Extension: GetLegendGraphics

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Feature Service (WFS)

- Interface:
  - GetCapabilities (discovery operation)
  - DescribeFeatureType (discovery operation)
  - GetPropertyValue (query operation)
  - GetFeature (query operation)
  - GetFeatureWithLock (query & locking operation)
  - LockFeature (locking operation)
  - Transaction (transaction operation)
  - CreateStoredQuery (stored query operation)
  - DropStoredQuery (stored query operation)
  - ListStoredQueries (stored query operation)
  - DescribeStoredQueries (stored query operation)
- Implemented by many systems and tools, e.g. GeoServer,UMN MapServer, QGIS, ArcGIS.

Table 1 — Conformance classes

Conformance class name	Operation or behaviour	WFS Conformance Test	FES Conformance Test(s)	GML Conformance Test(s)
Simple WFS	The server shall implement the following operations: GetCapabilities, DescribeFeatureType, ListStoredQueries, DescribeStoredQueries, GetFeature operation with only the StoredQuery action. One stored query, that fetches a feature using its id, shall be available but the server may also offer additional stored queries. Additionally the server shall conform to at least one of the HTTP GET, HTTP POST or SOAP conformance classes.	A.1.1	ISO 19143:2010, A.1	ISO 19136:2007, A.1.1, A.1.4, A.1.5, A.1.7, B.3, B.5, B.2.3
Basic WFS	The server shall implement the Simple WFS conformance class and shall additionally implement the GetFeature operation with the Query action and the GetPropertyValue operation.	A.1.2	ISO 19143:2010, A.2, A.4. A.5, A.6, A.7, A.12, A.14	ISO 19136:2007, B.4
Transactional WFS	The server shall implement the Basic WFS conformance class and shall also implement the Transaction operation.	A.1.3		
Locking WFS	The server shall implement the Transactional WFS conformance class and shall implement at least one of the GetFeatureWithLock or LockFeature operations.	A.1.4		
HTTP GET	The server shall implement the Key-value pair encoding for the operations that the server offers.	A.1.5		
HTTP POST	The server shall implement the XML encoding for the operations that the server implements.	A.1.6		
SOAP	The server shall implement XML encoded requests and results	A.1.7		

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Feature Service (WFS) - Example

- https://opendata.fmi.fi/wfs?service=WFS&version=2.0.0&request=getCapabilities (get a list of what is offered)


```
  </ows:AccessConstraints>
  </ows:ServiceIdentification>
▶ <ows:ServiceProvider>
  ...
  </ows:ServiceProvider>
▼ <ows:OperationsMetadata>
  ▼ <ows:Operation name="GetCapabilities">
    ▼ <ows:DCP>
      ▼ <ows:HTTP>
          <ows:Get xlink:href="https://opendata.fmi.fi/wfs/eng"/>
          <ows:Post xlink:href="https://opendata.fmi.fi/wfs/eng"/>
        </ows:HTTP>
      </ows:DCP>
    ▼ <ows:Parameter name="AcceptVersions">
      ▼ <ows:AllowedValues>
          <ows:Value>2.0.0</ows:Value>
        </ows:AllowedValues>
      </ows:Parameter>
    </ows:Operation>
  ▼ <ows:Operation name="DescribeFeatureType">
    ▼ <ows:DCP>
      ▼ <ows:HTTP>
          <ows:Get xlink:href="https://opendata.fmi.fi/wfs/eng"/>
          <ows:Post xlink:href="https://opendata.fmi.fi/wfs/eng"/>
        </ows:HTTP>
      </ows:DCP>
    ▼ <ows:Parameter name="outputFormat">
      ▼ <ows:AllowedValues>
          <ows:Value>application/gml+xml; subtype=gml/3.2</ows:Value>
          <ows:Value>application/gml+xml; version=3.2</ows:Value>
          <ows:Value>text/xml; subtype=gml/3.2</ows:Value>
          <ows:Value>text/xml; version=3.2</ows:Value>
        </ows:AllowedValues>
      </ows:Parameter>
    </ows:Operation>
  ▼ <ows:Operation name="ListStoredQueries">
    ▼ <ows:DCP>
```


Returns data in XML
(a client (Software) can
determine what is offered)

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Feature Service (WFS) - Example

- http://opendata.fmi.fi/wfs?service=WFS&version=2.0.0&request=getFeature&storedquery_id=fmi::forecast::hirlam::surface::point::multipointcoverage&place=helsinki&


```
▼<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs/2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:om="http://www.opengis.net/om/2.0" xmlns:omso="http://inspire.ec.europa.eu/schemas/omso/3.0" xmlns:ompr="http://inspire.ec.europa.eu/schemas/ompr/3.0"
  xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gco="http://www.isotc211.org/2005/gco"
  xmlns:swe="http://www.opengis.net/swe/2.0" xmlns:gmlcov="http://www.opengis.net/gmlcov/1.0" xmlns:sam="http://www.opengis.net/sampling/2.0"
  xmlns:sams="http://www.opengis.net/samplingSpatial/2.0" xmlns:target="http://xml.fmi.fi/namespace/om/atmosphericfeatures/1.0" timeStamp="2020-09-07T16:19:28Z"
  numberMatched="1" numberReturned="1" xsi:schemaLocation="http://www.opengis.net/wfs/2.0 http://schemas.opengis.net/wfs/2.0/wfs.xsd http://www.opengis.net/gmlcov/1.0
  http://schemas.opengis.net/gmlcov/1.0/gmlcovAll.xsd http://www.opengis.net/sampling/2.0 http://schemas.opengis.net/sampling/2.0/samplingFeature.xsd
  http://www.opengis.net/samplingSpatial/2.0 http://schemas.opengis.net/samplingSpatial/2.0/spatialSamplingFeature.xsd http://www.opengis.net/swe/2.0
  http://schemas.opengis.net/sweCommon/2.0/swe.xsd http://inspire.ec.europa.eu/schemas/omso/3.0 http://inspire.ec.europa.eu/schemas/omso/3.0/SpecialisedObservations.xsd
  http://inspire.ec.europa.eu/schemas/ompr/3.0 http://inspire.ec.europa.eu/schemas/ompr/3.0/Processes.xsd http://xml.fmi.fi/namespace/om/atmosphericfeatures/1.0
  http://xml.fmi.fi/schema/om/atmosphericfeatures/1.0/atmosphericfeatures.xsd">
 ▼<wfs:member>
  ▼<omso:GridSeriesObservation gml:id="WFS-
   vGcAOUiygPZ.M1zR9XVgT5YdebyJTowuYWbbpdOs2_l1x4efR060aeWzDtdOufXlmw48rp1w36d3R0629dnTTw36d3THv7ZeWHPlhaWLLn07qmnbltSfwqU7hZzCY2P1zrUi0Kcd06aMmrhnZd2Spp25bUn8Kl54TXnBm07s
   -">
   ▼<om:phenomenonTime>
    ▼<gml:TimePeriod gml:id="time-interval-1-1">
      <gml:beginPosition>2020-09-07T17:00:00Z</gml:beginPosition>
      <gml:endPosition>2020-09-09T04:00:00Z</gml:endPosition>
     </gml:TimePeriod>
    </om:phenomenonTime>
   ▼<om:resultTime>
    ▼<gml:TimeInstant gml:id="time-1-1">
      <gml:timePosition>2020-09-07T15:20:06Z</gml:timePosition>
     </gml:TimeInstant>
    </om:resultTime>
    <om:procedure xlink:href="http://xml.fmi.fi/inspire/process/hirlam"/>
   ▼<om:parameter>
    ▼<om:NamedValue>
      <om:name xlink:href="http://xml.fmi.fi/inspire/process/hirlam"/>
     ▼<om:value>
      ▼<gml:TimeInstant gml:id="analysis-time-1-1">
        <gml:timePosition>2020-09-07T12:00:00Z</gml:timePosition>
       </gml:TimeInstant>
      </om:value>
     </om:NamedValue>
    </om:parameter>
    <om:observedProperty xlink:href="http://opendata.fmi.fi/meta?
    observableProperty=forecast&param=GeopHeight,Temperature,Pressure,Humidity,WindDirection,WindSpeedMS,WindUMS,WindVMS,MaximumWind,WindGust,DewPoint,TotalCloudCover,Wea
   ▼<om:featureOfInterest>
    ▼<sams:SF_SpatialSamplingFeature gml:id="enn-s-1-1-">
     ▼<sam:sampledFeature>
```

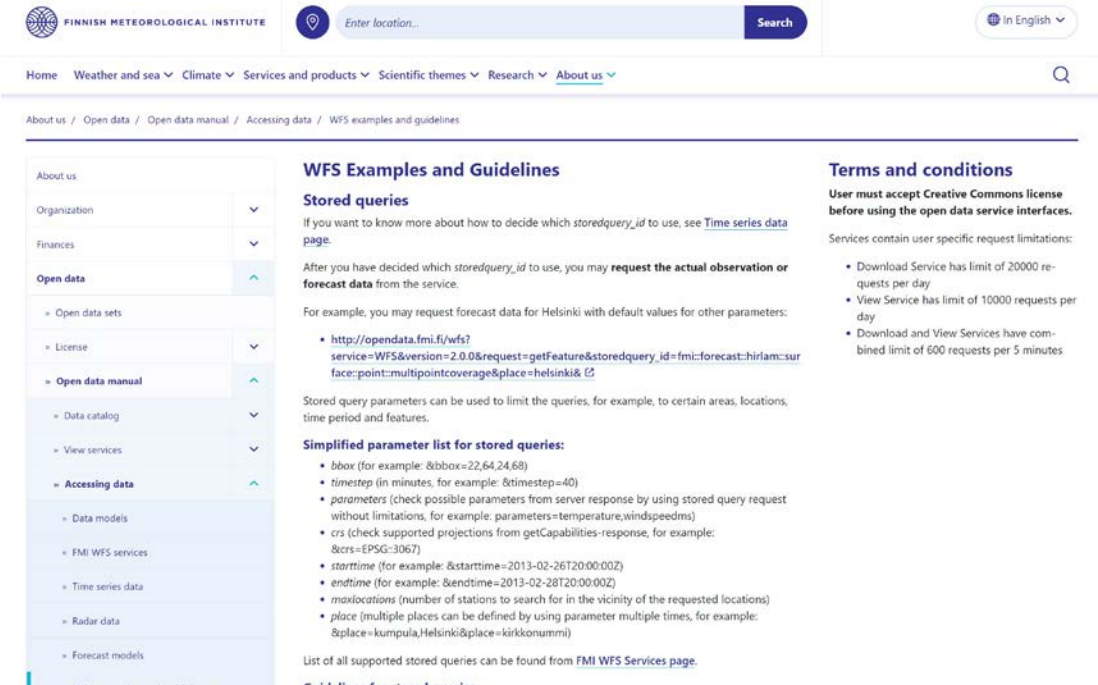

Returns data in XML (GML – Geography Markup Language)

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Feature Service (WFS) – Tutorial

- https://en.ilmatieteenlaitos.fi/open-data-manual-wfs-examples-and-guidelines



NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Coverage Service (WCS)

- Interface:
  - **GetCapabilities** - return all service-level metadata and a brief description of the data in GML format
  - **DescribeCoverage** - return a full description of one or more coverages within the service in GML format.
  - **GetCoverage** - return a coverage in one of the supported formats: e.g. GeoTIFF, NITF, HDF, JPEG, JPEG2000, PNG

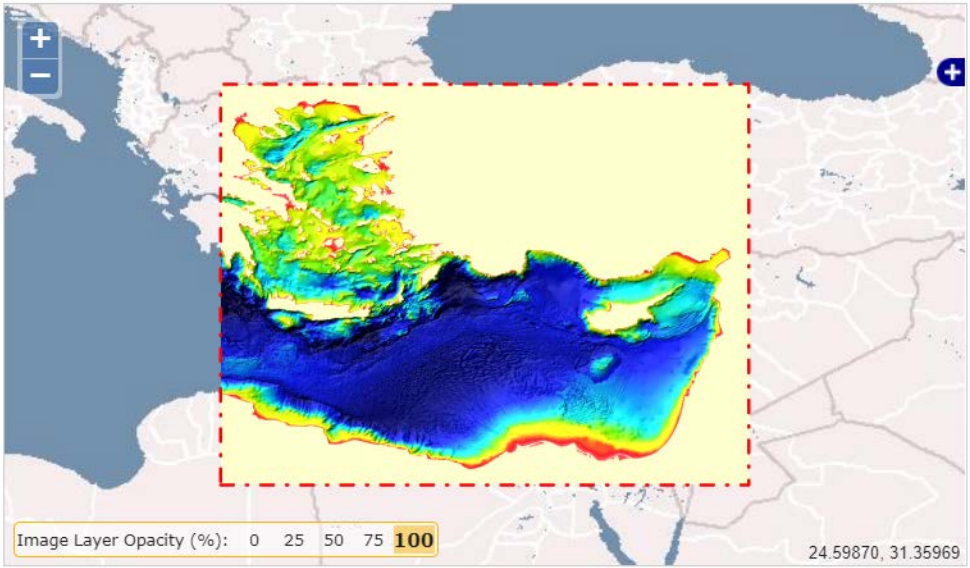
- Implemented by many systems and tools, e.g. GeoServer,UMN MapServer, QGIS, ArcGIS, Rasdaman.

NorDataNet
Norwegian Scientific Data Network

OGC interfaces

Web Coverage Service (WCS) - Example

- http://ogcdev.bgs.ac.uk/ogcclient/WCS/GetCoverage_v2_0_1.html



Returns data from a WCS server as an image.
Integrated in OpenLayers for overlay with other data

NorDataNet
Norwegian Scientific Data Network

OPeNDAP

OPeNDAP (Open-source Project for a Network Data Access Protocol)

Initiative and protocol to simplify access to scientific data

«The goal is to allow end users, whoever they may be, to access immediately whatever data they require in a form they can use, all while using applications they already possess and are familiar with.» (https://www.opendap.org/about)

Widely adopted in many scientific communities

Open source software, e.g. THREDDS Data Server (TDS), Hyrax (reference), ERDDAP

Used by numerous data centres and research infrastructures around the world

Benefits

- Allows reading data as a stream
- Subsetting (extract a part of the dataset)
- Many libraries and tools available

OPeNDAP

Easy to integrate into your code e.g.

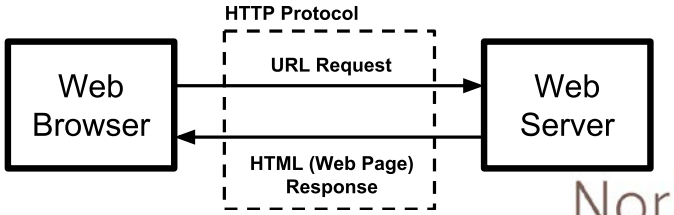
- R, Python, Matlab, …
- Excel (!) - NETCDF4Excel on GitHub (http://netcdf4excel.github.io/)

Many GUI tools support it, e.g.

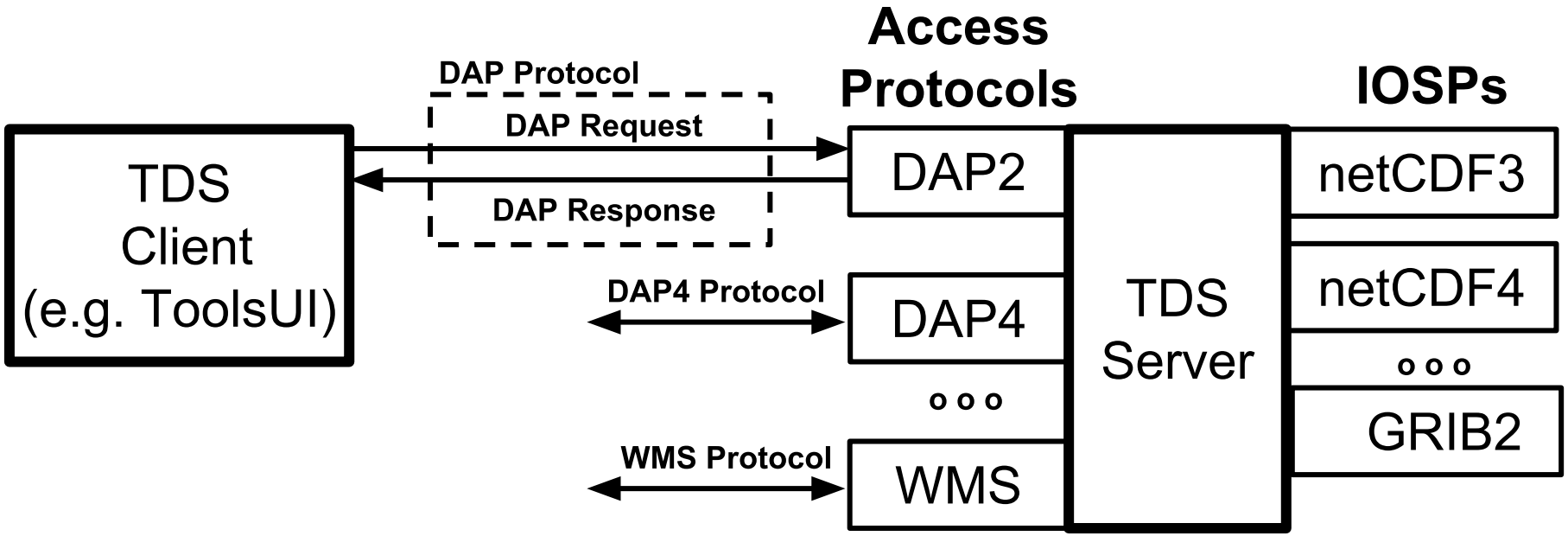
- Panoply (https://www.giss.nasa.gov/tools/panoply/)
- Ncview (http://meteora.ucsd.edu/~pierce/ncview_home_page.html)
- Ferret (https://ferretop.pmel.noaa.gov/Ferret/)

OPeNDAP defines a series of services (methods) to access metadata and data

- Dataset Attribute Structure (DAS)
- Dataset Descriptor Structure (DDS)
- Values (DataDDS)



OPeNDAP – Client-Server communication (TDS)



NorDataNet
Norwegian Scientific Data Network

OPeNDAP - Dataset Attribute Structure (DAS)

DAS Call met.no

```
Attributes {
    time {
        String long_name "reference time of sea ice file";
        String units "seconds since 1981-01-01 00:00:00";
        String calendar "standard";
    }
    yc {
        String axis "Y";
        String long_name "y-coordinate in Cartesian system";
        String units "m";
    }
    xc {
        String axis "X";
        String long_name "x-coordinate in Cartesian system";
        String units "m";
    }
    lat {
        String long_name "latitude";
        String units "degrees_north";
    }
    lon {
        String long_name "longitude";
        String units "degrees_east";
    }
    crs {
        String grid_mapping_name "polar_stereographic";
        Float32 straight_vertical_longitude_from_pole 0.0;
        Float32 latitude_of_projection_origin 90.0;
        Float32 standard_parallel 90.0;
        Float32 false_easting 0.0;
        Float32 false_northing 0.0;
        String proj4_string "+proj=stere lon_0=0.0 lat_ts=90.0 lat_0=90.0 a=6371000.0 b=6371000.0";
        DODS {
            Int32 strlen 0;
        }
    }
    ice_concentration {
        String long_name "sea ice concentration";
        String standard_name "sea_ice_area_fraction";
        String units "%";
        String coordinates "lon lat";
        String grid_mapping "crs";
        String source "met.no";
        Int16 _FillValue -99;
        Float32 scale_factor 1.0;
        Float32 add_offset 0.0;
    }
...
```

information about data, i.e. metadata

OPeNDAP - Dataset Descriptor Structure (DDS)

DDS Call met.no

```
Dataset {
    Int32 time[time = 1];
    Float32 yc[yc = 2980];
    Float32 xc[xc = 3812];
    Grid {
     ARRAY:
        Float32 lat[yc = 2980][xc = 3812];
     MAPS:
        Float32 yc[yc = 2980];
        Float32 xc[xc = 3812];
    } lat;
    Grid {
     ARRAY:
        Float32 lon[yc = 2980][xc = 3812];
     MAPS:
        Float32 yc[yc = 2980];
        Float32 xc[xc = 3812];
    } lon;
    String crs;
    Grid {
     ARRAY:
        Int16 ice_concentration[time = 1][yc = 2980][xc = 3812];
     MAPS:
        Int32 time[time = 1];
        Float32 yc[yc = 2980];
        Float32 xc[xc = 3812];
    } ice_concentration;
    Grid {
     ARRAY:
        Int16 concentration_range[time = 1][yc = 2980][xc = 3812];
     MAPS:
        Int32 time[time = 1];
        Float32 yc[yc = 2980];
        Float32 xc[xc = 3812];
    } concentration_range;
} myocean/siw-tac/siw-metno-svalbard/2020/09/ice_conc_svalbard_202009071500.nc;
```

”This request returns information about data types, so that a receiving program can allocate space appropriately.”

OPeNDAP - DataDDS (DDX)

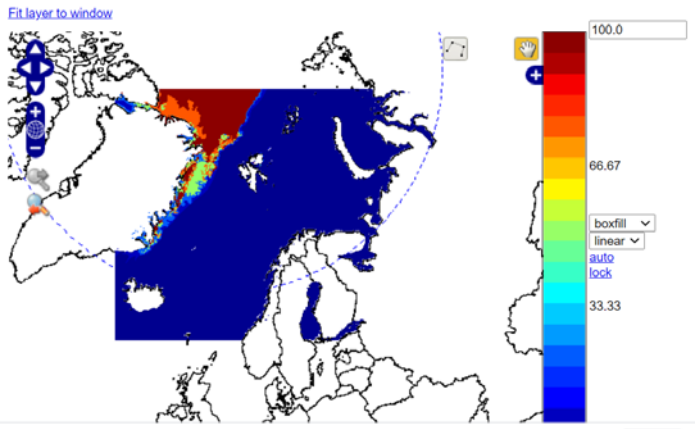
DDX Call met.no

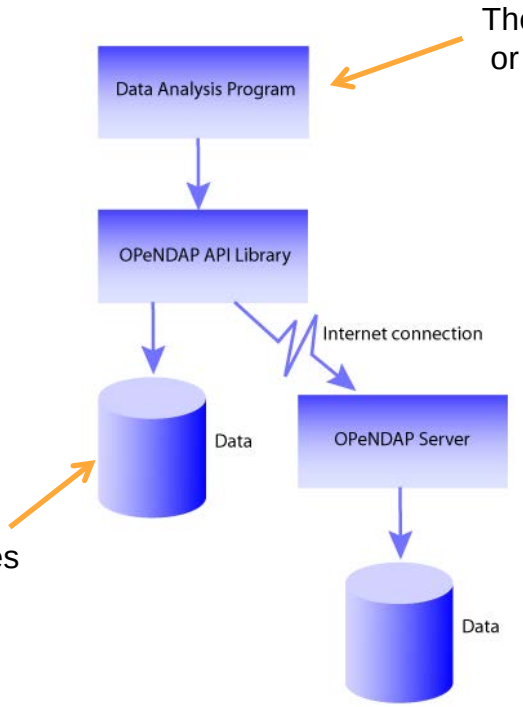
an XML version of the Data Attribute and Data Description replies


```
<?xml version="1.0" encoding="UTF-8"?>
<Dataset name="myocean/siw-tac/siw-metno-svalbard/2020/09/ice_conc_svalbard_202009071500.nc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xml.opendap.org/ns/DAP2"
xsi:schemaLocation="http://xml.opendap.org/ns/DAP2  http://xml.opendap.org/dap/dap2.xsd" >

            <Attribute name="NC_GLOBAL" type="Container">
                        <Attribute name="title" type="String">
                                    <value>Arctic Svalbard &amp; Barents Ice Concentration, L4, 1km daily  (METNO-ARC-
SEAICE_CONC-L4-NRT-OBS)</value>
                        </Attribute>
                        <Attribute name="Conventions" type="String">
                                    <value>CF-1.4</value>
                        </Attribute>
                        <Attribute name="netcdf_version_id" type="String">
                                    <value>3.6.3</value>
                        </Attribute>
                        <Attribute name="creation_date" type="String">
                                    <value>2020-09-07T13:46:18Z</value>
                        </Attribute>
                        <Attribute name="produced_date" type="String">
                                    <value>2020-09-07T14:00:00Z</value>
                        </Attribute>
```

OPeNDAP – Client-Server

Clients use these calls to gather data from distributed servers, inspect, plot and analyse data





The client, e.g. Godiva2 (ncwms) or Panoply

Remote data servers (>>1)

Your local NetCDF files

NorDataNet
Norwegian Scientific Data Network

OPeNDAP

Resources:

- https://www.opendap.org/
- https://opendap.github.io/documentation/QuickStart.html
- http://conference.eresearch.edu.au/wp-content/uploads/2018/10/0910-1010-Mingfang-Wu-Fri-Lake.pdf
- OPeNDAP clients: https://www.opendap.org/support/OPeNDAP-clients
- OPeNDAP servers: https://www.opendap.org/index.php/support/OPeNDAP-servers
- Tutorial on DAP2 and DAP4 Protocol Services: https://www.unidata.ucar.edu/software/tds/current/tutorial/DAP.html
- DAP Version 2: http://opendap.org/pdf/ESE-RFC-004v1.2.pdf
- DAP Version 2: DAP Version 4: http://docs.opendap.org/index.php/OPULS_Development#DAP4_Specification

RESTful (Restful in general)

- REST - REpresentational State Transfer
- An architectural style for distributed hypermedia systems
- Compose systems as a set of small loosely coupled components
- REST principles:
  - Client–server - separate GUI from data storage; simplify server side
  - Stateless - each request must contain all needed information; no relying on server state
  - Cacheable - client can keep results from earlier requests; reuse for similar requests
  - Uniform interface - acess to and modification of resources through unified interface
  - Layered system - hierarchical layers where a components can only interact with next layer
  - Code on demand (optional) - client functionality can be extended by downloading applets/scripts
- Resources:
  - https://restfulapi.net/ ; https://restfulapi.net/rest-architectural-constraints/
  - Roy Fielding PhD thesis (2000)

NorDataNet
Norwegian Scientific Data Network

Table Of Contents

REST

GeoServer provides a RESTful interface through which clients can retrieve information about an instance and make configuration changes. Using the REST interface's simple HTTP calls, clients can configure GeoServer without needing to use the Web administration interface.

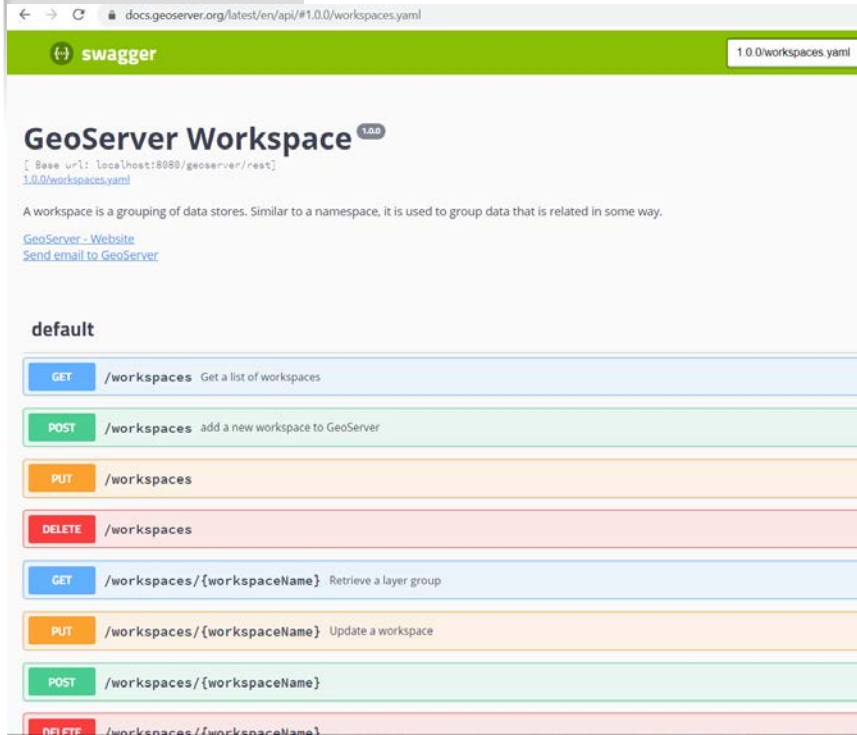
REST is an acronym for "REpresentational State Transfer". REST adopts a fixed set of operations on named resources, where the representation of each resource is the same for retrieving and setting information. In other words, you can retrieve (read) data in an XML format and also send data back to the server in similar XML format in order to set (write) changes to the system.

Operations on resources are implemented with the standard primitives of HTTP: GET to read; and PUT, POST, and DELETE to write changes. Each resource is represented as a URL, such as `http://GEOSERVER_HOME/rest/workspaces/topp`.

API

The following links provide direct access to the GeoServer REST API documentation, including definitions and examples of each endpoint:

- /about/manifests
- /about/system-status
- /datastores
- /coverages
- /coveragestores
- /featuretypes
- /fonts



OpenAPI Specification (OAS)

- REST has led to a wealth of different APIs with different flavours
- Need for standardisation of how these APIs are described
- The OpenAPI Initiative (OAI) was thus created by a group of industry experts
- Aim to standardize on how APIs are described, making vendor neutral
- Open governance structure under the Linux Foundation
- Based on earlier work (Swagger Specification, by SmartBear Software)
- OpenAPI Specification Version 3.0.3 http://spec.openapis.org/oas/v3.0.3
- Spec is online https://github.com/OAI/OpenAPI-Specification
- Resources: https://www.openapis.org/; https://swagger.io/specifi

NorDataNet
Norwegian Scientific Data Network

Benefits of using interoperable data

- Benefits
  - Many open source tools and libraries available that support these standards
  - Support in many programming languages (C, …, Python, R, …)
  - Online tutorials for writing your own tools
  - Same tool can access data from multiple providers
  - Many free clients (GUI tools) that allows you to quickly investigate data

NorDataNet
Norwegian Scientific Data Network

Integration in Python

- Tutorial on OPeNDAP access with Python
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+access+with+python
- Access one timestep of a gridded dataset, and one timeseries of point data
- Follow the steps of the tutorial
  - Install needed packages (PythonXY, NetCDF4 package, pydap, …)
  - Select NetCDF file from OPeNDAP server (http://opendap.deltares.nl)
  - Get the URL to files with grid and timeseries data
  - Download the desired parts of the data
  - Plot the downloaded data

NorDataNet
Norwegian Scientific Data Network

Integration in Python

- Access http://opendap.deltares.nl and select the two files

OpenEarthTools provides a module `opendap.py` that makes pydap quack like netCDF4 (repos, manual download) so you can talk directly to opendap data via the web. Now execute the following Python lines, or download the full example code (repos,manual download):

```
#!/usr/bin/env python
# Read data from an opendap server
import netCDF4, pydap,     urllib
import pylab,   matplotlib
import numpy as np
from opendap import opendap # OpenEarthTools module, see above that makes pypdap quack like netCDF4
```

1. Go to an OPeNDAP server (e.g. http://opendap.deltares.nl) and pick a netCDF file by copying the contents of the Data URL box.
2. Define the associated url you just copied.

```
url_grid = r'http://opendap.deltares.nl/thredds/dodsC/opendap/rijkswaterstaat/vaklodingen/vaklodingenKB116_4
url_time = r'http://opendap.deltares.nl/thredds/dodsC/opendap/rijkswaterstaat/waterbase/concentration_of_sus
```

3. Extract the data.

```
# Get grid data
grid   = opendap(url_grid)
G_x    = grid.variables['x']
G_y    = grid.variables['y']
G_z    = grid.variables['z']

G      = {} # dictionary ~ Matlab struct
G['x'] = G_x[:].squeeze()
G['y'] = G_y[:].squeeze()
G['z'] = G_z[1,:,:].squeeze() # download only one temporal slice
```

Hint: double-click to select code

```
# represent fillValue from data as Masked Array
# the next release of netcdf4 will return a masked array by default, handling NaNs
# correctly too (http://code.google.com/p/netcdf4-python/issues/detail?id=168)
G['z'] = np.ma.masked_invalid(G['z'])
```

```
# Get time series data
time   = opendap(url_time)
T_t    = time.variables['time']
T_z    = time.variables['concentration_of_suspended_matter_in_water']

T      = {} # dictionary ~ Matlab struct
T['t'] = netCDF4.num2date(T_t[:], units=T_t.units)
T['z'] = T_z[:].squeeze()
```

Integration in Python

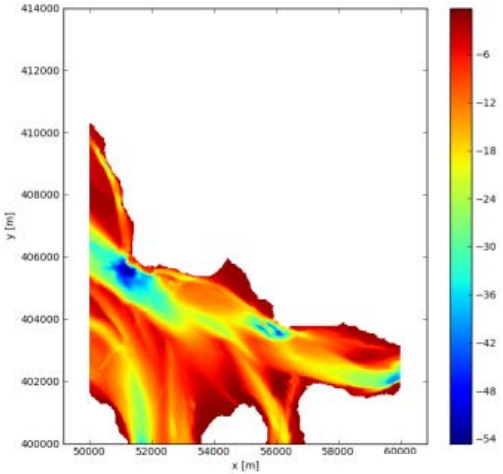
– Plot the downloaded data

```
# plot grid data
matplotlib.pyplot.pcolormesh(G['x'],G['y'],G['z'])
pylab.xlabel('x [m]')
pylab.ylabel('y [m]')
matplotlib.pyplot.colorbar()
matplotlib.pyplot.axis('tight')
matplotlib.pyplot.axis('equal')
matplotlib.pyplot.title(url_grid)
pylab.savefig('vaklodingenKB116_4544')

# plot time series data
pylab.clf()
matplotlib.pyplot.plot_date(T['t'], T['z'], fmt='b-', xdate=True, ydate=False)
pylab.ylabel('SPM [kg/m3]')
matplotlib.pyplot.title(url_time)
pylab.savefig('DELFZBTHVN')
```

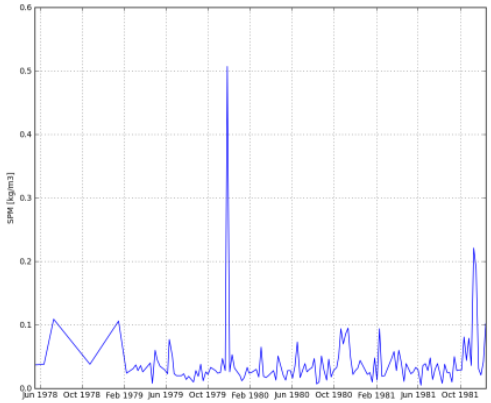
NorDataNet
Norwegian Scientific Data Network

Integration in Python

– Plot the downloaded data




```
# plot grid data
matplotlib.pyplot.pcolormesh(G['x'],G['y'],G['z'])
pylab.xlabel('x [m]')
pylab.ylabel('y [m]')
matplotlib.pyplot.colorbar()
matplotlib.pyplot.axis('tight')
matplotlib.pyplot.axis('equal')
matplotlib.pyplot.title(url_grid)
pylab.savefig('vaklodingenKB116_4544')

# plot time series data
pylab.clf()
matplotlib.pyplot.plot_date(T['t'], T['z'], fmt='b-', xdate=True, ydate=False)
pylab.ylabel('SPM [kg/m3]')
matplotlib.pyplot.title(url_time)
pylab.savefig('DELFZBTHVN')
```



NorDataNet
Norwegian Scientific Data Network

Integration in R

- Tutorial on OPeNDAP access with R
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+access+with+R
- Access one timestep of a gridded dataset, and one timeseries of point data
- Follow the steps of the tutorial
  - Load package for NetCDF (w/OPeNDAP access)
  - Select NetCDF file from OPeNDAP server (http://opendap.deltares.nl)
  - Get the URL to files with grid and timeseries data
  - Download the desired parts of the data
  - Plot the downloaded data

NorDataNet
Norwegian Scientific Data Network

Integration in R

Access http://opendap.deltares.nl and select the two files

```
require(ncdf)
```

1. Go to an OPeNDAP server (e.g. http://opendap.deltares.nl) and pick a netCDF file by copying the contents of the Data URL box. Because the netcdf packages for windows are not yet opendap-enabled, download them.
2. Define the associated url you just copied.

```
url_grid <-
"http://opendap.deltares.nl/thredds/fileServer/opendap/rijkswaterstaat/vaklodingen/vaklodingenKB116_4544.nc" # note: netcdf4 does not work on w

url_time <-
"http://opendap.deltares.nl/thredds/fileServer/opendap/rijkswaterstaat/waterbase/concentration_of_suspended_matter_in_sea_water/id410-DELFZBTHV

download.file(url_grid, "vaklodingenKB116_4544.nc", method = "auto",
quiet = FALSE, mode="wb", cacheOK = TRUE)

download.file(url_time, "id410-DELFZBTHVN.nc", method = "auto",
quiet = FALSE, mode="wb", cacheOK = TRUE)
```

NorDataNet
Norwegian Scientific Data Network

```
grid.nc <- open.ncdf("vaklodingenKB116_4544.nc")

# look what's in there...
grid.nc

# Get grid data
G.x <- get.var.ncdf(grid.nc,'x')
G.y <- get.var.ncdf(grid.nc,'y')

# get only first timestep
G.z <- get.var.ncdf(grid.nc,'z')[,,1]

# to get a black background, and set the scale of depth values to start from 0
G.z[G.z == -9999] <- 0

# image.plot needs sorted x- and y-values;
# as y-values are descending, the order is reversed here...
G.y <- rev(G.y)
G.z <- G.z[,length(G.y):1]
```

Hint: double-click to select code

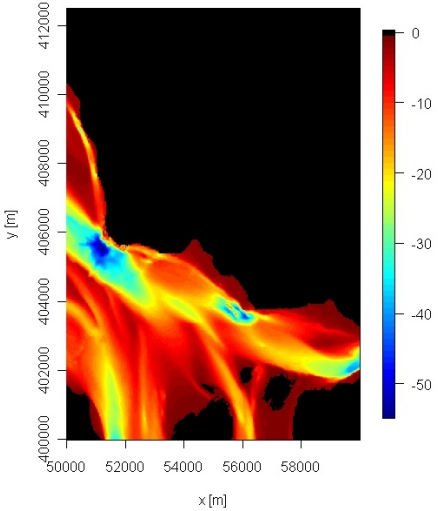
```
time.nc <- open.ncdf("id410-DELFZBTHVN.nc")
# look what's in there...
time.nc

T.t <- get.var.ncdf(time.nc,'time')
T.eta <-
get.var.ncdf(time.nc,'concentration_of_suspended_matter_in_sea_water')
```

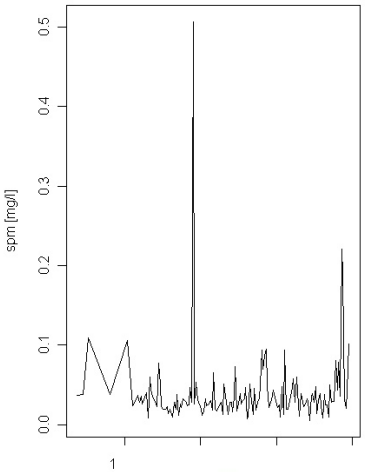
Integration in R

- Plot the data

```
# R-package fields provides nice image facilities and color s
par (mfrow = c(1,2))
library(fields)
image.plot(G.x,G.y,as.matrix(G.z),
        col = c(tim.colors(),"black"),
        xlab = "x [m]", ylab = "y [m]")
```

```
plot(as.Date(T.t, origin="1970-01-01"), T.eta, type = "l", yl
[mg/l]")
```





NorDataNet
Norwegian Scientific Data Network

Integration in Panoply

Panoply is a Data Viewer for NetCDF, HDF, GRIB

NetCDF

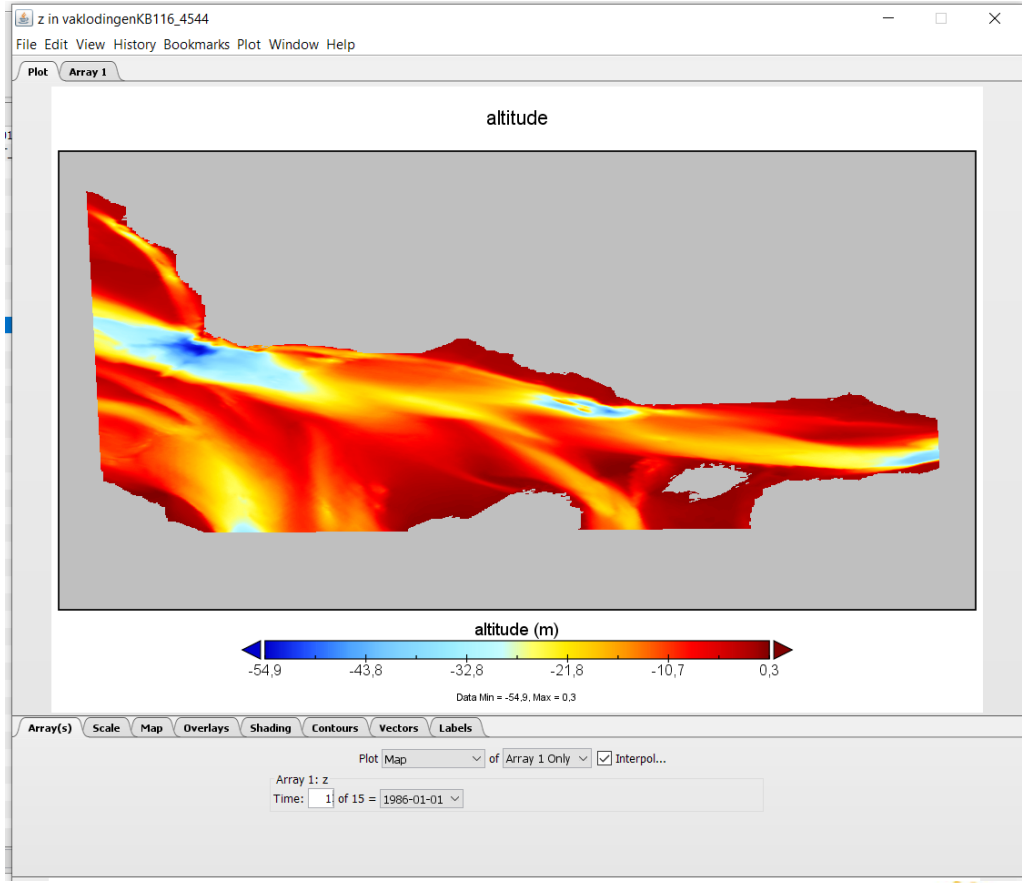
- File input
- OPeNDAP access (e.g TDS)

Install Panoply

- macOS, Windows, Linux

View dataset from OPeNDAP

- Open Remote Dataset, enter URL



Plot, view metadata, overlay borders, export maps as graphics

NorDataNet
Norwegian Scientific Data Network

Panoply: Panoply — Sources

File Edit View History Bookmarks Plot Window Help

Create Plot | Combine Plot | Open Dataset | Remove | Remove All | Hide Info

Datasets | Catalogs | Bookmarks

Name	Long Name	Type
vaklodingen.nc	vaklodingen.nc	Remote File
id	id	—
lat_from	lat from	1D
lat_to	lat to	1D
lon_from	lon from	1D
lon_to	lon to	1D
rectangle	rectangle	2D
url	url	—
x_from	x from	1D
x_to	x to	1D
y_from	y from	1D
y_to	y to	1D

Show: All variables

File "vaklodingen.nc"

File type: NetCDF-3/CDM

```
netcdf http://opendap.deltares.nl/thredds/fileServer/opendap/deltares/vaklodingen/vaklodingen.nc {
  dimensions:
    id = 177;
    n = 4;
    s1 = 10;
    s2 = 98;
  variables:
    char id(id=177, s1=10);
      :long_name = "id";
      :units = "-";
      :comment = " ";

    char url(id=177, s2=98);
      :long_name = "url";
      :units = "-";
      :comment = " ";

    double x_from(id=177);
      :long_name = "x_from";
      :units = "m";
      :comment = " ";

    double x_to(id=177);
      :long_name = "x_to";
      :units = "m";
      :comment = " ";

    double y_from(id=177);
      :long_name = "y_from";
      :units = "m";
      :comment = " ";
```

Resources for integration in Python, R, Matlab

Some resources to try out

- OPeNDAP access with Python
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+access+with+python
- OPeNDAP access with R
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+access+with+R
- OPeNDAP subsetting with R
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+subsetting+with+R
- OPeNDAP access with Matlab
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+access+with+Matlab
- OPeNDAP access with Excel
  https://publicwiki.deltares.nl/display/OET/OPeNDAP+access+with+Excel

NorDataNet
Norwegian Scientific Data Network

Jupyter Notebook

https://jupyter.org/

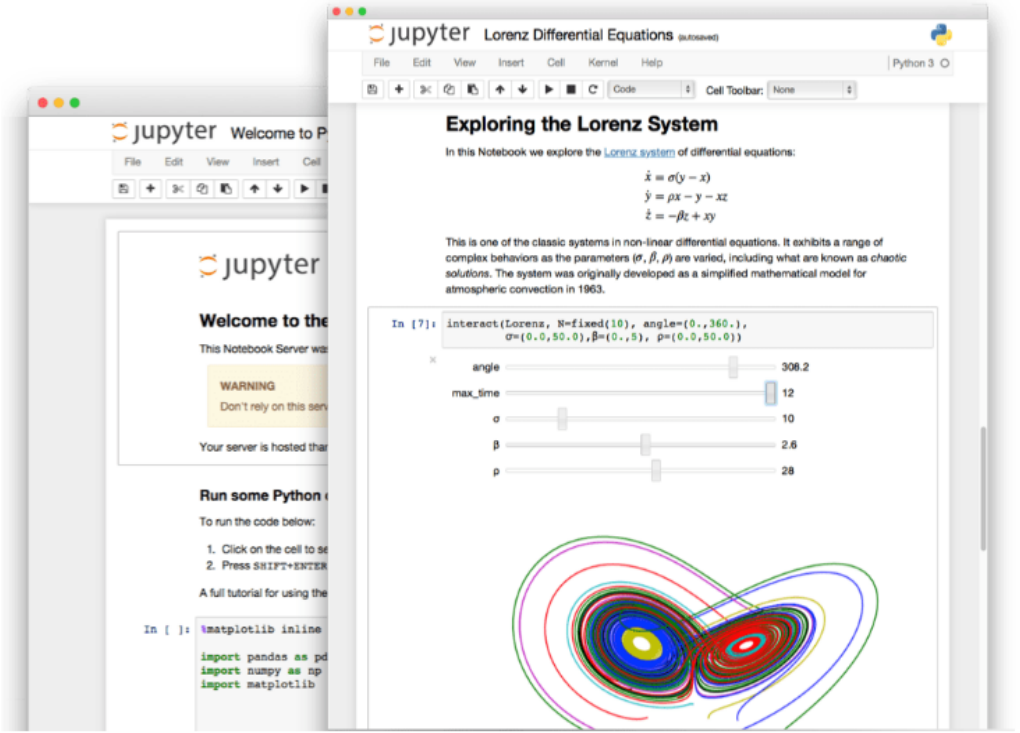
A web-based tool for developing, documenting and testing your code

Supports:

- Python
- R
- Matlab
- Julia
- Scala
- ...

Output in browser (HTML)

Notebooks can be shared



The Jupy

The Jupyter N
to create and
visualizations
transformatio
visualization,

Try it in you

localhost

jupyter Welcome to Python (unsaved changes)

File Edit View Insert Cell Kernel Help **Menubar** Python 3

Markdown CellToolbar **Toolbar** **Cell Mode Indicator** | **Kernel Indicator**

jupyter rackspace

Welcome to the Temporary Notebook (tmpnb) service!

This Notebook Server was **launched just for you**. It's a temporary way for you to try out a recent development version of the IPython/Jupyter notebook.

WARNING

Don't rely on this server for anything you want to last - your server will be *deleted after 10 minutes of inactivity*.

Your server is hosted thanks to Rackspace, on their on-demand bare metal servers, OnMetal.

Cell In Command Mode

Run some Python code!

To run the code below:

1. Click on the cell to select it.
2. Press `SHIFT+ENTER` on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available here.

In [ ]:
```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib
```

https://jupyter-notebook.readthedocs.io/en/stable/ui_components.html

localhost

jupyter Welcome to Python Last Checkpoint: Last Tuesday at 2:34 PM (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Markdown Cell Toolbar: None

Edit Mode Indicator

```
<div class="clearfix" style="padding: 10px; padding-left: 0px">
<img src="https://raw.githubusercontent.com/jupyter/nature-demo/master/images/jupyter-logo.png" width="150px"
style="display: inline-block; margin-top: 5px;">
<a href="http://bit.ly/tmpnbdevrax"><img src="https://cloud.githubusercontent.com/assets/836375/4916141/2732892e-64d6-
11e4-980f-11afcf03ca31.png" width="150px" class="pull-right" style="display: inline-block; margin: 0px;"></a>
</div>

## Welcome to the Temporary Notebook (tmpnb) service!

This Notebook Server was **launched just for you**. It's a temporary way for you to try out a recent development
version of the IPython/Jupyter notebook.

<div class="alert alert-warning" role="alert" style="margin: 10px">
<p>**WARNING**</p>

<p>Don't rely on this server for anything you want to last - your server will be *deleted after 10 minutes of
inactivity*.</p>
</div>

Your server is hosted thanks to [Rackspace](http://bit.ly/tmpnbdevrax), on their on-demand bare metal servers,
[OnMetal](http://bit.ly/onmetal).
```

Cell In Edit Mode

Run some Python code!

To run the code below:

1. Click on the cell to select it.
2. Press SHIFT+ENTER on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available here

https://jupyter-notebook.readthedocs.io/en/stable/ui_components.html

Example - Aggregation

Monthly sea ice concentration in Svalbard region

Input:

- Data: Daily sea ice concentration from met.no
- Time range: Jan 2010 – present (through CMEMS)
- Format: NetCDF/CF

Output:

- Data: monthly sea ice concentration, plot
- Time range: 1 month
- Format: NetCDF/CF, GIF

Example - Aggregation

Jupyter Notebook

Main steps:

- Setup (import needed packages)
- Get info on dataset variables and dimensions
- Determine time period (1 month)
- Calculate statistics (average)
- Store generated monthly field (NetCDF)
- Plot the mean SIC field

Code: https://github.com/ec-intaros/enb-sea-ice-concentration

NorDataNet
Norwegian Scientific Data Network

localhost:8888/notebooks/Downloads/tssic_cmems.ipynb

jupyter tssic_cmems (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 2

Run Markdown

Setup

This service requires the following libraries installed in your notebook environment:

- netcdf4, https://github.com/Unidata/netcdf4-python
- matplotlib, https://matplotlib.org/

In [2]:
```
# Import the libraries needed
from netCDF4 import Dataset
from datetime import datetime
import sys
import numpy as np
import matplotlib.pyplot as plt
```

Preparing for output

The notebook will generate output in NetCDF format, using the same map projection and coordinates as the CMEMS "Svalbard ice chart" product. Hence, we start with obtaining information about an instance of this CMEMS product.

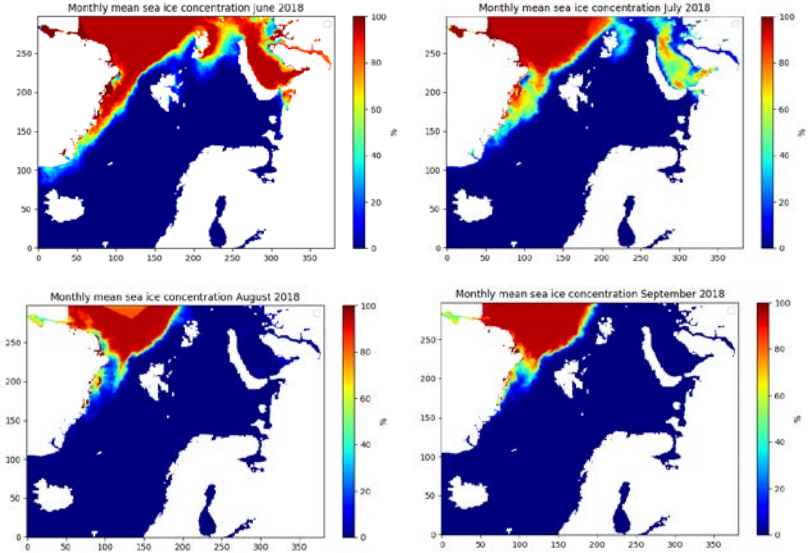
In [3]:
```
# Get dimensions and positions of CMEMS SIC data for Svalbard region
filename = 'https://thredds.met.no/thredds/dodsC/myocean/siw-tac/siw-metno-svalbard/2018/06/ice_conc_svalbard_201806011500.nc'
sic_dataset = Dataset(filename, 'r')
print('CMEMS product variables:', sic_dataset.variables)
print('Variable ice concentration shape:', sic_dataset.variables['ice_concentration'].shape)
yc = sic_dataset.variables['ice_concentration'].shape[1]
xc = sic_dataset.variables['ice_concentration'].shape[2]
print('Variable ice concentration has num lines, columns:', yc, xc)
```

```
CMEMS product variables: {'time': <class 'netCDF4._netCDF4.Variable'>
int32 time(time)
    long_name: reference time of sea ice file
    units: seconds since 1981-01-01 00:00:00
    calendar: standard
unlimited dimensions:
current shape = (1,)
filling off, 'yc': <class 'netCDF4._netCDF4.Variable'>
float32 yc(yc)
```

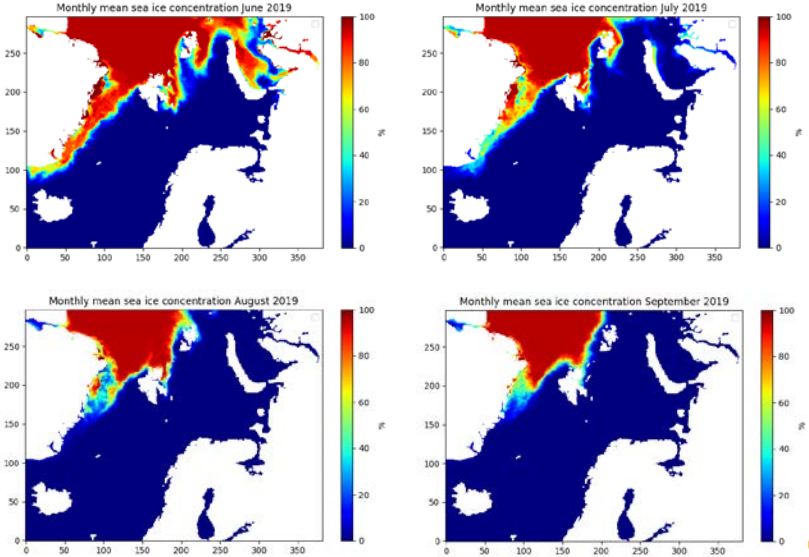
Example - Aggregation

Example output

June – September 2018



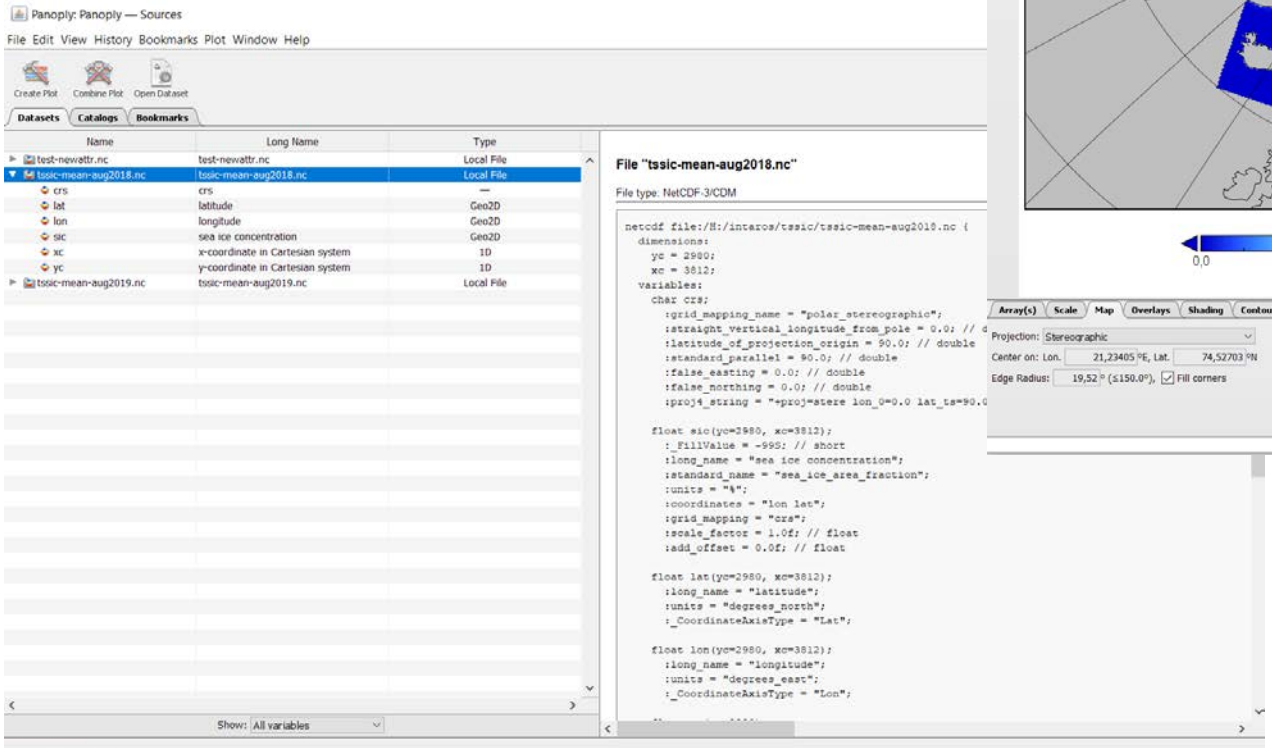
June – September 2019

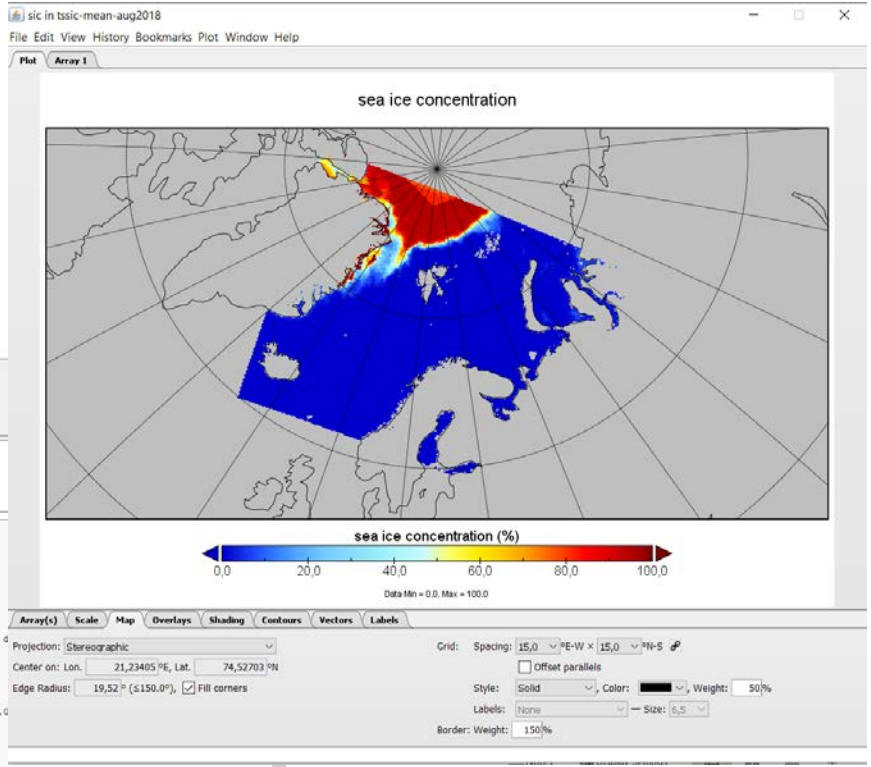


NorDataNet
Norwegian Scientific Data Network

Example - Aggregation

Inspecting output





NorDataNet
Norwegian Scientific Data Network

Resources

Rosetta: http://tomcat.nersc.no/rosetta/ ; https://github.com/Unidata/rosetta

CO/CDO: http://nco.sourceforge.net/

netCDF4: https://unidata.github.io/netcdf4-python/netCDF4/index.html

xarray: http://xarray.pydata.org/en/stable/

Jupyter Notebook: https://jupyter.org/

Jupyter Hub: https://jupyter.org/hub

DataOne: https://old.dataone.org/education-modules

ESIP: https://commons.esipfed.org/node/1422 ; Data management training