Web-based workflows to produce ocean climatologies using DIVA (Data-Interpolating Variational Analysis) and Jupyter notebooks

Alexander Barth, Charles Troupin, Sylvain Watelet, Aida Alvera-Azcárate, and Jean-Marie Beckers

GHER, University of Liège, Belgium

Traditional scientific workflow

- **Explore** initial idea (often in Matlab)
- **Collaborate** with colleagues (emails)
- **Production** with large data set
- **Publication** of results
- **Education** and possibly outreach

Traditional scientific workflow

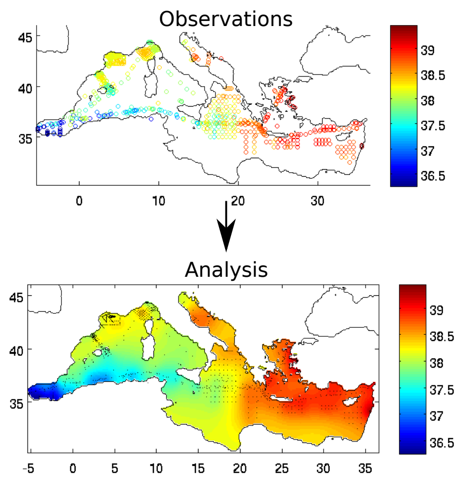
- **Explore** initial idea (often in Matlab)
- **Collaborate** with colleagues (emails)
- **Production** with large data set
- **Publication** of results
- **Education** and possibly outreach
- Can this be improved?
  - Reduce dependency on proprietary software
  - Better reproducibility
  - Avoid rewriting the code by using the same tools at different stages

Traditional scientific workflow

- **Explore** initial idea (often in Matlab)
- **Collaborate** with colleagues (emails)
- **Production** with large data set
- **Publication** of results
- **Education** and possibly outreach
- Can this be improved?
  - Reduce dependency on proprietary software
  - Better reproducibility
  - Avoid rewriting the code by using the same tools at different stages
- Let's try to answer these questions in the context of **generating ocean climatologies** with DIVA

What is DIVA?

- DIVA: Data Interpolating Variational Analysis
- Objective: **derive a gridded climatology from in situ observations**
- The variational inverse methods aim to derive a continuous field which is:
    - **close to the observations** (it should not necessarily pass through all observations because observations have errors)
    - "**smooth**"



What is DIVA?

- DIVA is a complex software
- Written in **Fortran** and **shell scripts**
- It has a large number of dependencies (NetCDF, bash, various shell utilities)
- Not so easy to install

What is DIVA?

- DIVA is a complex software
- Written in **Fortran** and **shell scripts**
- It has a large number of dependencies (NetCDF, bash, various shell utilities)
- Not so easy to install
- We aim to fully rewrite DIVA in **Julia** (divand.jl)
- Julia: good trade-off between **efficiency** of a compiled language and **flexibility** of a dynamic language
- Facilitate the installation:
    - Use **Jupyter notebooks** fully configured environment for divand.jl
    - **Docker container** allows one to easily replicate these environments

Matlab

Python

R

Linear Algebra

Modern programming concepts

Data frames

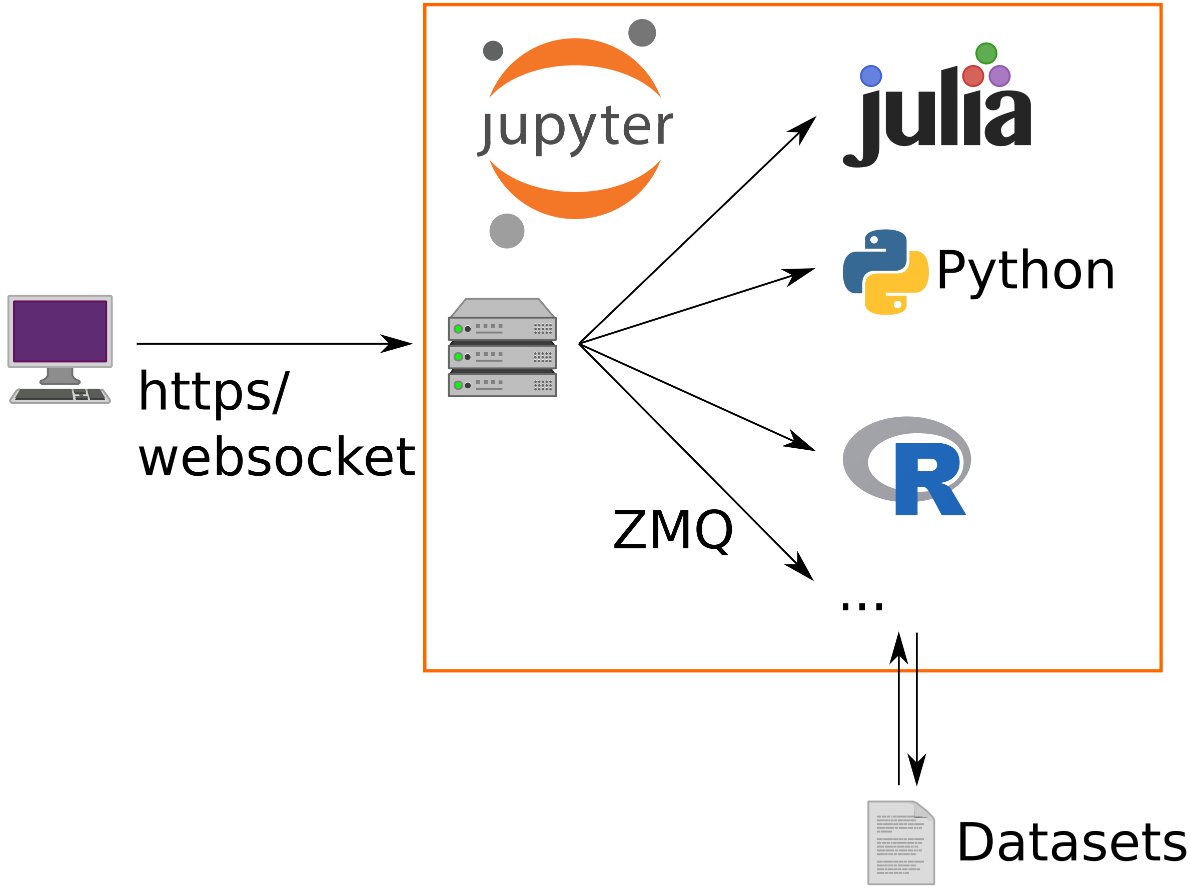
julia

+ Compilation to machine code
+ Performance approaching C
+ Multiple dispatch
+ Type system
+ Lisp-like macros and Metaprogramming

Jupyter notebooks

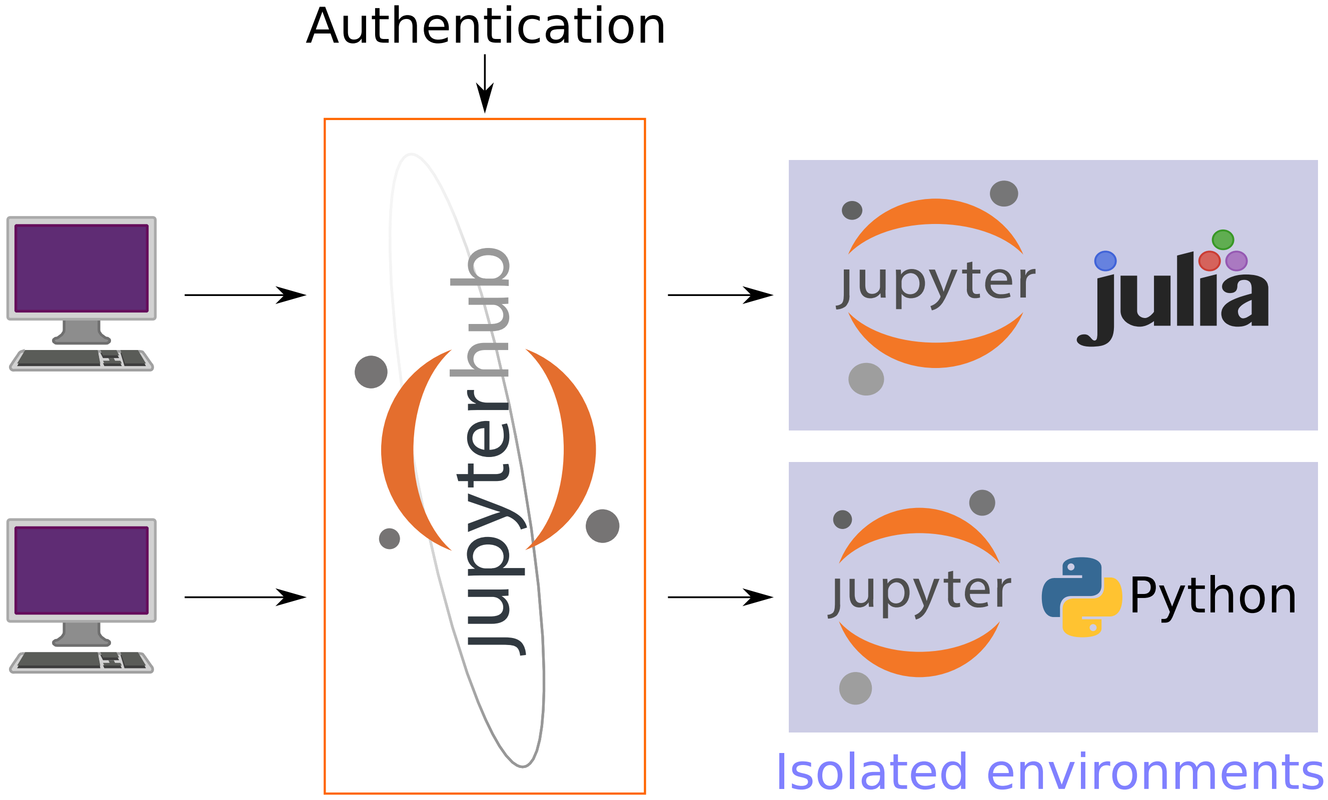
- Integrated web environment
  - **Computing**
    - Interactive
    - *Ju*lia, *Py*thon, *R*,...
  - **Visualization**
  - **Documentation**
    - High-quality type setting and equations (Latex)
    - Export to HTML and PDF (among others)
- Easy to **share**, on e.g. nbviewer.jupyter.org and github.com
- Facilitate **reproducibility** and peer-review (of DIVA climatologies in particular)
- Significant community around Jupyter notebooks
- Also involvement of players outside of the scientific community (Google, Microsoft with Azure ML)
- Jupyter notebooks: **single** user

Jupyter architecture



Jupyterhub architecture

Jupyterhub: **multiple** users



Adaptation for SeaDataCloud

- Make **Docker containers**, preinstalled with Julia and various Julia packages:
  - Plotting library (PyPlot) and a more specialized library for ocean data
  - ZMQ
  - DIVAnd
  - ...
- Julia packages are precompiled
- Integration in **SeaDataCloud authentication**:
  - Implementation CAS authentication
  - Marine ID can be used to login into jupyterhub
  - EUDATs B2Access might be considered as an alternative
- Transfer files via **WebDAV** in Julia:
  - Either transparently mounted or using explicit download and upload requests

Example notebook

DIVA in Jupyter Notebook

- Interpolate in situ observations of the Black Sea on a regular grid
- The first step is to load modules

In [1]:
```
using divand
using NetCDF
using OceanPlot
using PyPlot
```

- Get data from EUDAT's B2DROP

In [2]:
```
fname = "B2DROP/Data/divand-example-data/BlackSea/Salinity.bigfile"
bathname = "B2DROP/Data/divand-example-data/Global/Bathymetry/gebco_30sec_

obsvalue,obslon,obslat,obsdepth,obstime,obsid = loadbigfile(fname);
```

- setup the domain
  - define resolution
  - geographical bounding box
  - the depth range

Example notebook

- setup the domain
  - define resolution
  - geographical bounding box
  - the depth range
  - time time range

In [3]:
```
dx = dy = 0.1
lonr = 27:dx:42
latr = 40:dy:47
depthr = [0.]
timer = 1:1:12
```
Out[3]:
```
1:1:12
```

- the relative error variance on the observations

In [4]:
```
epsilon2 = 0.1
```
Out[4]: 0.1

- correlation length

In [5]:
```
# sz is the size of the domain
sz = (length(lonr),length(latr),length(depthr),length(timer))

# horizontal correlation length in meters
```

Example notebook

- correlation length

In [5]:

```julia
# sz is the size of the domain
sz = (length(lonr),length(latr),length(depthr),length(timer))

# horizontal correlation length in meters
lenx = 200_000
leny = 200_000

# vertical correlation length in meters
lenz = Array{Float64}(sz)
for n = 1:sz[4]
    for k = 1:sz[3]
        for j = 1:sz[2]
            for i = 1:sz[1]
                lenz[i,j,k,n] = 10 + depthr[k]/5
            end
        end
    end
end

# correlation time-scale in month
lent = 1.
```

Out[5]: 1.0

- run DIVA

Example notebook

- run DIVA
- Determine the field φ close to the observations d_j for $j = 1, N_d$

$$J[\varphi] = \sum_{j=1}^{N_d} \mu_j [d_j - \varphi(x_j, y_j)]^2 + \|\varphi - \varphi_b\|^2$$

where the regularization constrain is given by

$$\|\varphi\|^2 = \int_D (\alpha_2 \nabla\nabla\varphi : \nabla\nabla\varphi + \alpha_1 \nabla\varphi \cdot \nabla\varphi + \alpha_0 \varphi^2) dD$$

The parameters μ, α_0, α_1, α_2 are coefficients related to the accuracy of the observations and to the correlation length. φ_b is a background estimate.

In [6]:
```
@time fi = diva(("longitude","latitude","depth","time"),
          (lonr,latr,depthr,timer),
          (obslon,obslat,obsdepth,obstime),
          obsvalue, epsilon2,
          (lenx,leny,lenz,lent),
          divand.aggregation_monthly;
          bathname = bathname
);
```

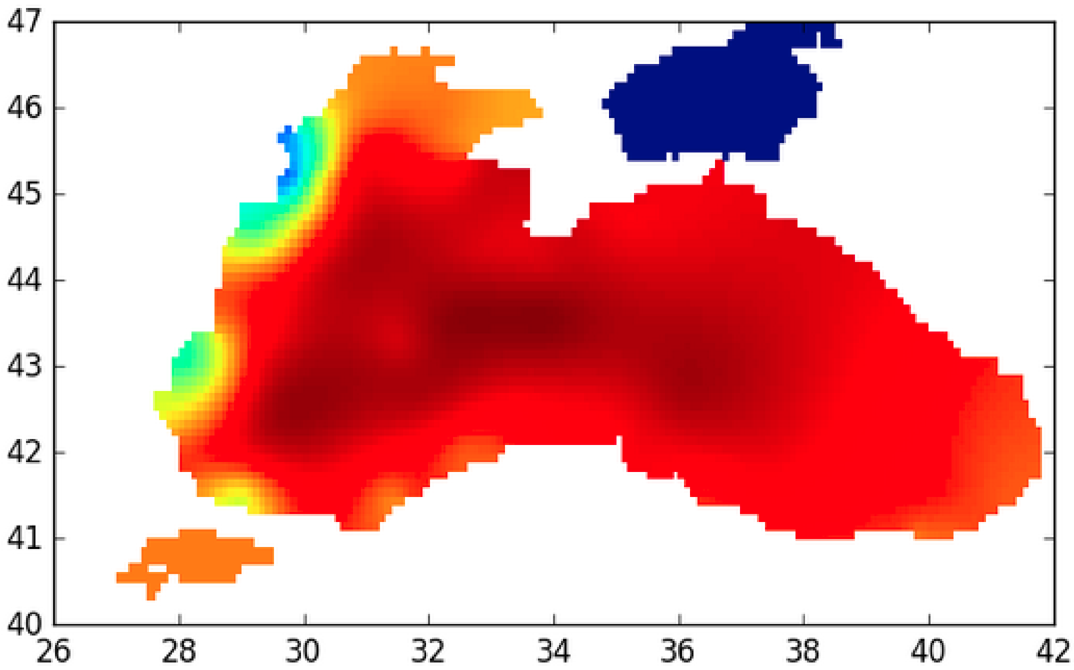
```
32.735618 seconds (29.08 M allocations: 2.214 GB, 3.12% gc time)
```

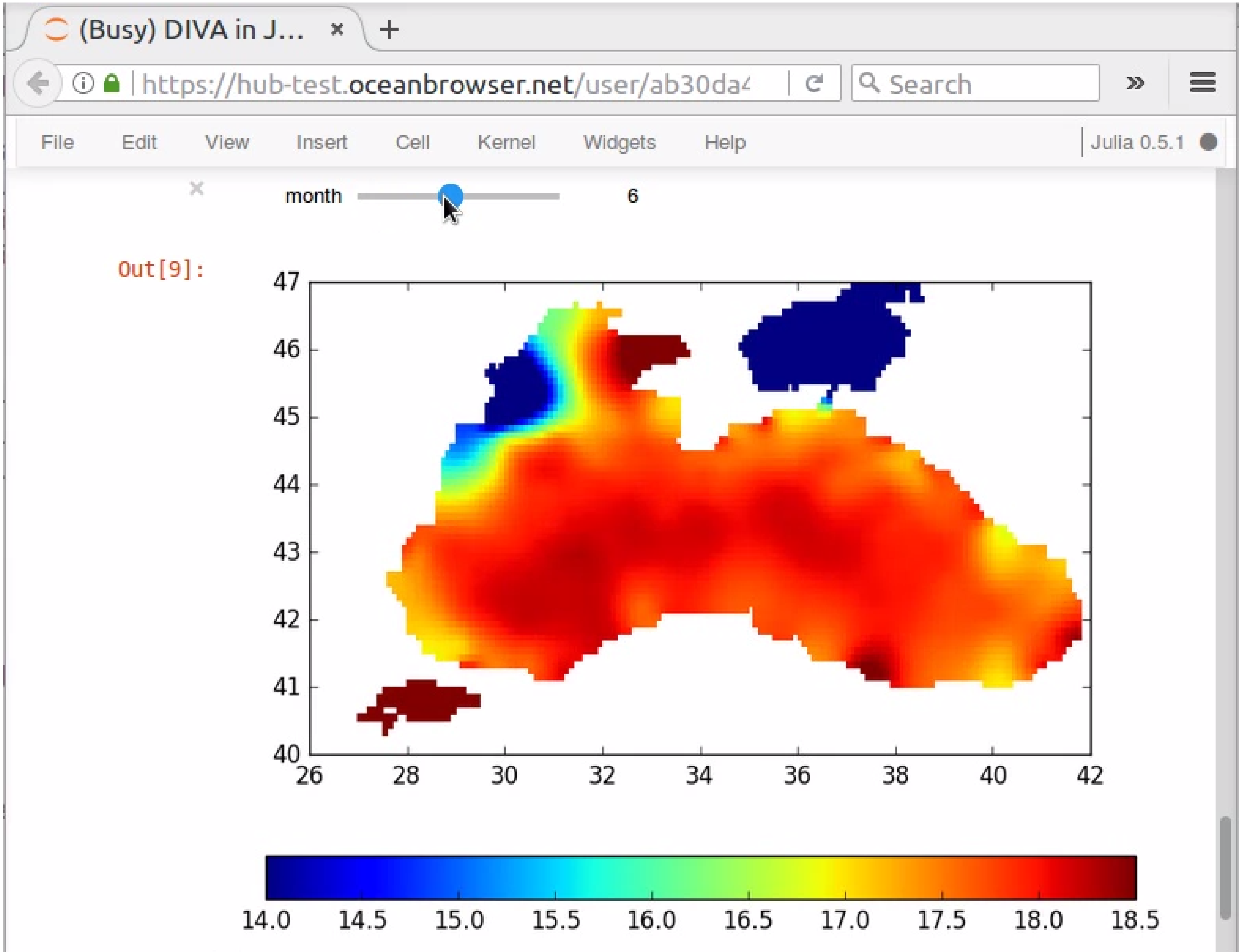
- view the result

Example notebook

In [7]:

```
OceanPlot.pcol(lonr,latr,fi[:,:,1,1]',vmin=15.); colorbar(orientation = "hor
set_aspect_ratio()
```




(Busy) DIVA in J...
https://hub-test.oceanbrowser.net/user/ab30da4
Search
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Julia 0.5.1
month
6
Out[9]:
47
46
45
44
43
42
41
40
26
28
30
32
34
36
38
40
42
14.0
14.5
15.0
15.5
16.0
16.5
17.0
17.5
18.0
18.5

Conclusions

- Jupyterhub provides interesting options for generation climatologies and data products:
  - A **fast access** to the data
  - Docker allows to provide a **standardized computing environment** to all users
  - The jupyter notebook can be used to fully **document the generation of the climatology**
  - **Straightforward to reproduce** the work of others and to try to improve it