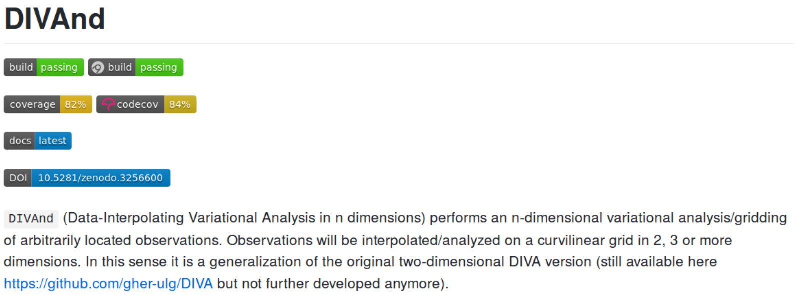
DIVA developments and next training: suggestions for harmonization and improvements of results

C. Troupin, A. Barth & J.-M. Beckers (GHER–ULiège)

3rd Product Meeting, Plouzané (France), 15 October, 2019

Thanks for using DIVAnd

Thanks for using DIVAnd



DIVAnd

build passing build passing

coverage 82% codecov 84%

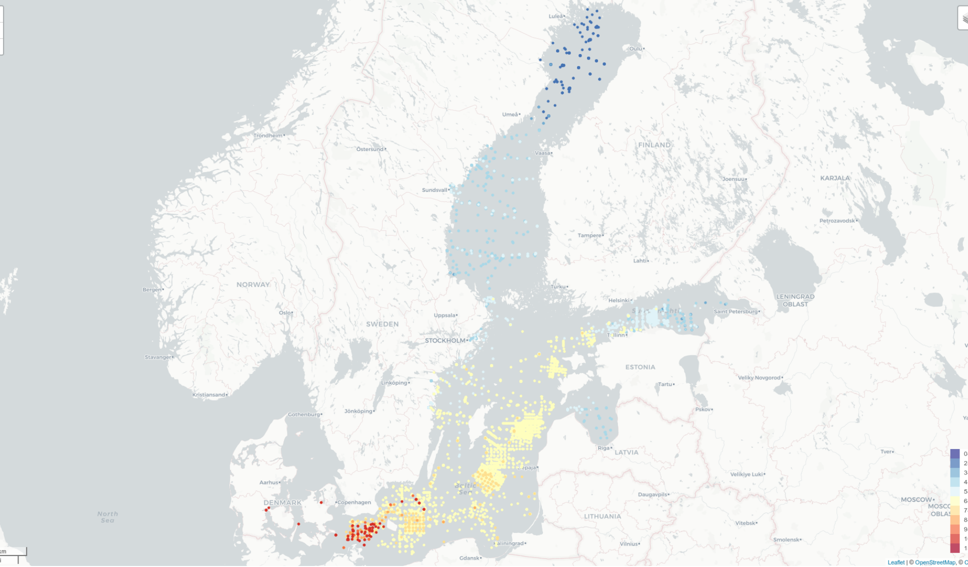
docs latest

DOI 10.5281/zenodo.3256600

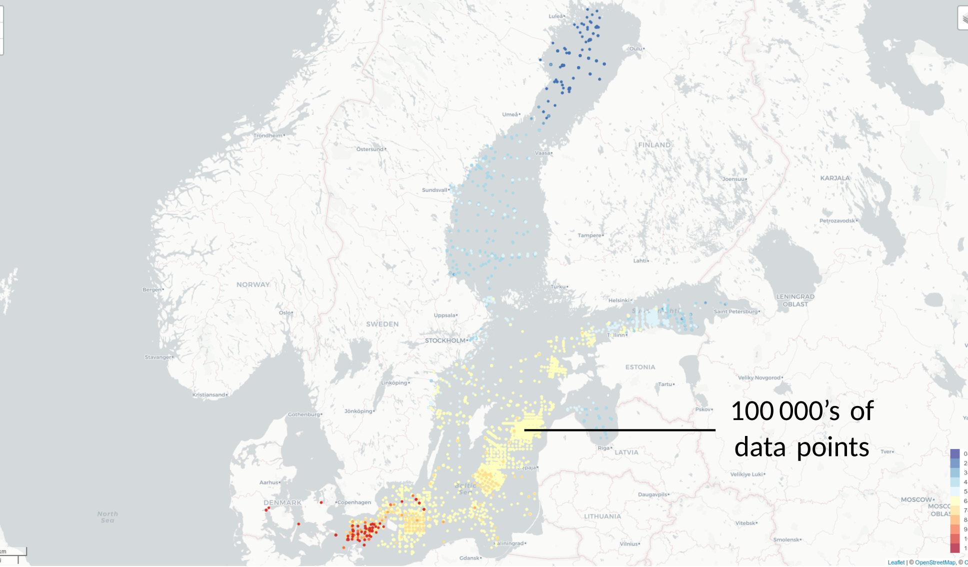
`DIVAnd` (Data-Interpolating Variational Analysis in n dimensions) performs an n-dimensional variational analysis/gridding of arbitrarily located observations. Observations will be interpolated/analyzed on a curvilinear grid in 2, 3 or more dimensions. In this sense it is a generalization of the original two-dimensional DIVA version (still available here https://github.com/gher-ulg/DIVA but not further developed anymore).

1. Well tested
2. Always improvable

Why interpolation is complex?

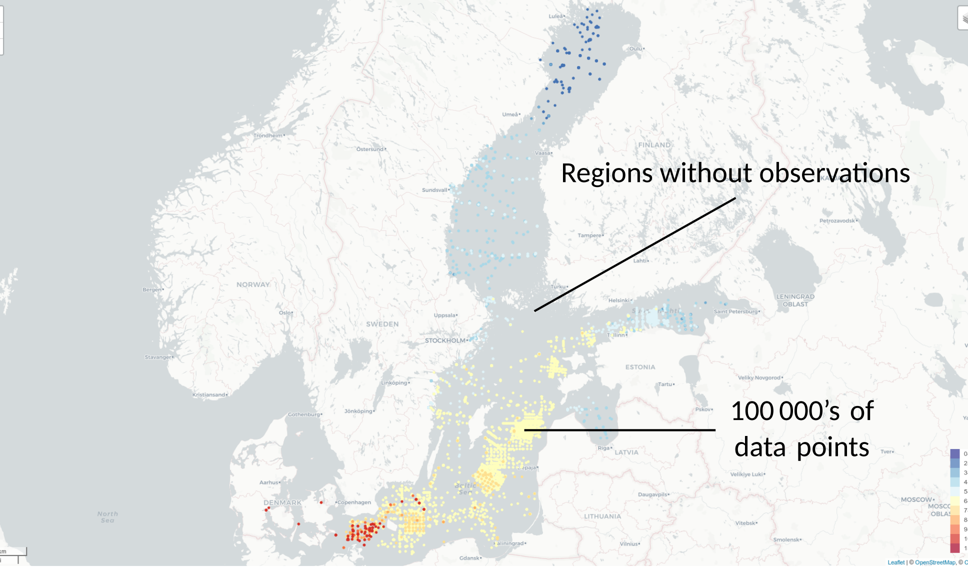


Why interpolation is complex?



100 000's of data points

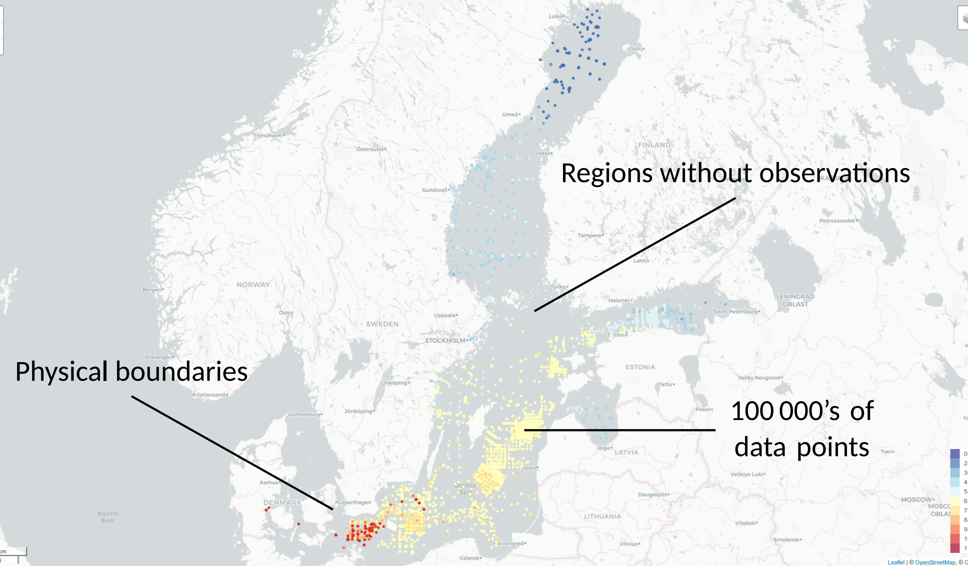
Why interpolation is complex?



Regions without observations

100 000's of data points

Why interpolation is complex?



Regions without observations

Physical boundaries

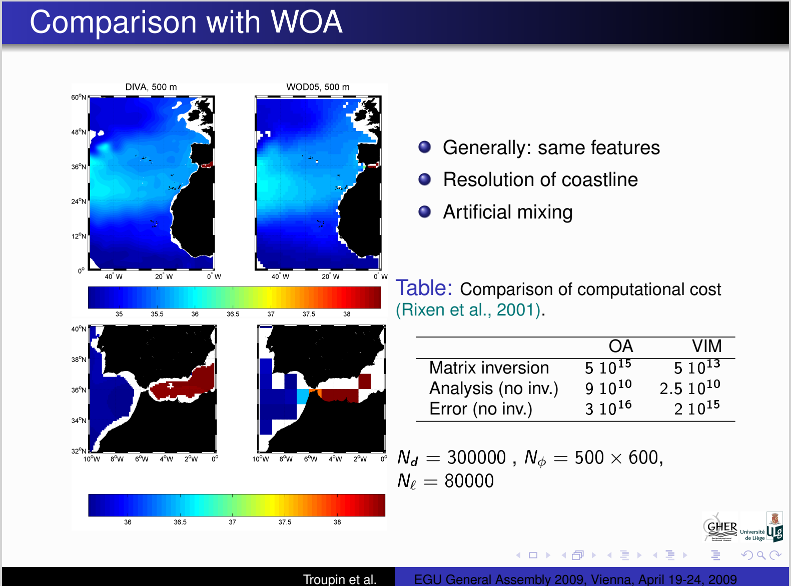
100 000's of data points

Developing cutting-edge interpolation products

Back in 2009 at EGU...

Developing cutting-edge interpolation products

Back in 2009 at EGU...



Comparison with WOA

- Generally: same features
- Resolution of coastline
- Artificial mixing

Table: Comparison of computational cost (Rixen et al., 2001).

	OA	VIM
Matrix inversion	$5\ 10^{15}$	$5\ 10^{13}$
Analysis (no inv.)	$9\ 10^{10}$	$2.5\ 10^{10}$
Error (no inv.)	$3\ 10^{16}$	$2\ 10^{15}$

$N_d = 300000$, $N_\phi = 500 \times 600$, $N_\ell = 80000$

Troupin et al. EGU General Assembly 2009, Vienna, April 19-24, 2009

1 Harmonisation

Harmonisation

Ensure consistency…
taking into account the regional specificities

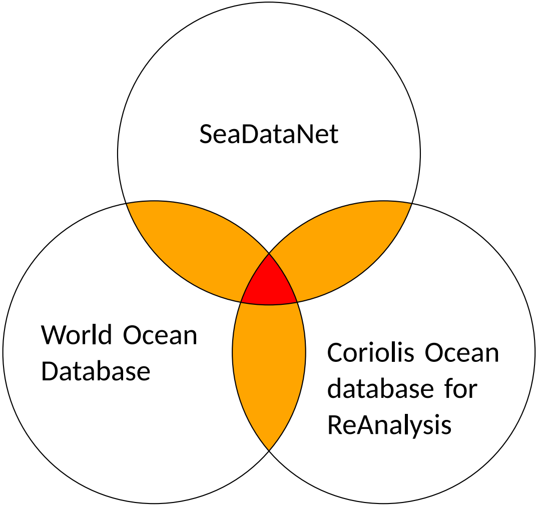
3 axes of harmonization

Data sources

Interpolation tool

Procedure & parameters

Data sources

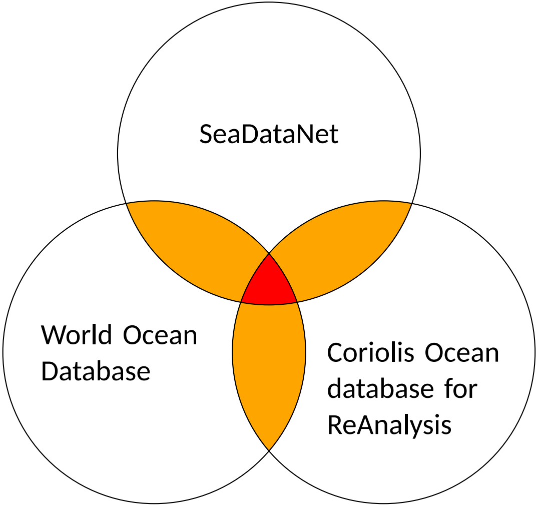


SeaDataNet

World Ocean Database

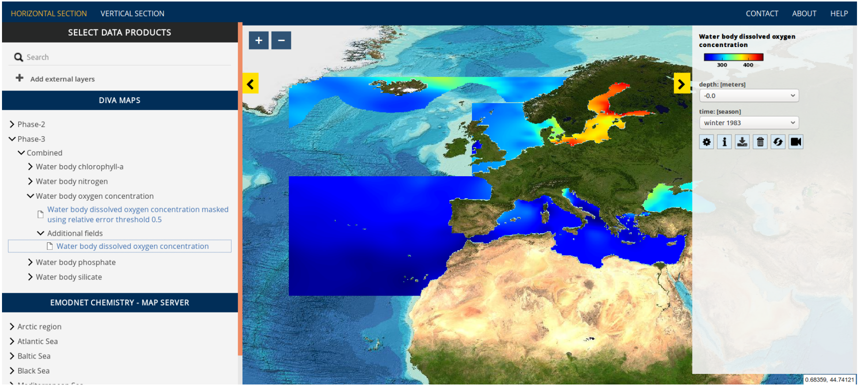
Coriolis Ocean database for ReAnalysis

Data sources



1. Which source(s)?
2. Which version?
3. How to eliminate **duplicates**?

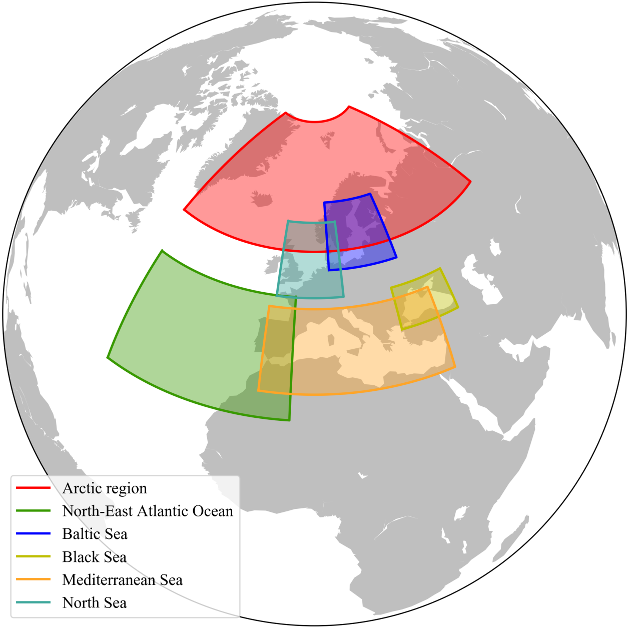
Taking care of the domains...



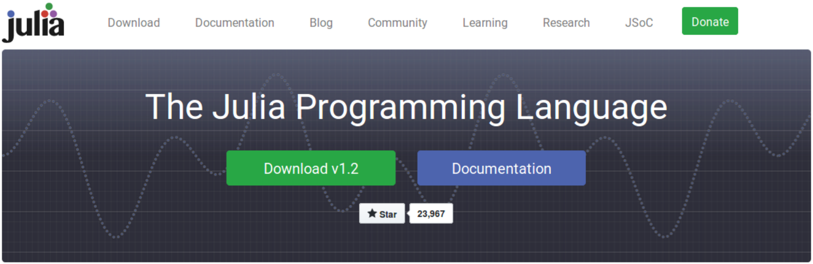
depth: [meters] -0.0

time: [season] winter 1983

Taking care of the domains...



Interpolation tool



Julia version: $\geq$ 1.0 (now at 1.2)

DIVAnd version: $\geq$ v2.1.0 (now at v2.4.0)

Procedure

Domain

Spatial resolution: depending on region

Total time coverage: depending on region

Decade definitions: consistency (merging)

Vertical levels: consistency across regions (merging)

Procedure

Basic parameters

Bathymetry: GEBCO or EMODnet Bathymetry, resolution depending on domain

Correlation length: optimized (if coverage allows) + good

Noise-to-signal ratio: optimized (if coverage allows)

Data weights: optional (check sensitivity on a few levels)

Background field: to discuss (case by case)

Procedure

Other parameters

`surfextend` = true (vertical extension at surface)

`coeff_derivative2` = [0., 0., 10^{-8}] (sensitivity test)

2

Improvement of DIVAnd

Continuous improvement using user feedback

Event		Number of commits	
		DIVAnd code	Notebooks
1st workshop	(6 April 2018)	773	265
2nd training course	(26 June 2019)	73	7

Examples of what we’ve changed

V2.0.2 (Aug 21, 2018): matrix allocation in ODV reading

Examples of what we’ve changed

V2.0.2 (Aug 21, 2018): matrix allocation in ODV reading

V2.1.0 (Sep 3, 2018):

- Support Julia 1.0
- Optimization in interpolation routines
- Improved type-stability

Examples of what we've changed

V2.0.2 (Aug 21, 2018): matrix allocation in ODV reading

V2.1.0 (Sep 3, 2018):

- Support Julia 1.0
- Optimization in interpolation routines
- Improved type-stability

V2.1.1 (Oct 19, 2018): allow one to force the direct solver

Examples of what we've changed

V2.0.2 (Aug 21, 2018): matrix allocation in ODV reading

V2.1.0 (Sep 3, 2018):

- Support Julia 1.0
- Optimization in interpolation routines
- Improved type-stability

V2.1.1 (Oct 19, 2018): allow one to force the direct solver

V2.2.0 (Dec 20, 2018): NetCDF import for ODV

Examples of what we've changed

V2.3.0 (Jan 24, 2019):

- time varying background
- edit mask with flood-fill
- optimization

Examples of what we've changed

V2.3.0 (Jan 24, 2019):

- time varying background
- edit mask with flood-fill
- optimization

V2.3.1 (Feb 11, 2019): fix interpolation issue for time series near masked grid points (coast line)

Examples of what we've changed

V2.3.0 (Jan 24, 2019):

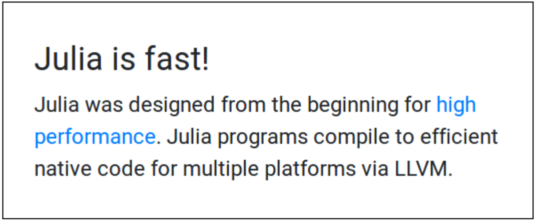
- time varying background
- edit mask with flood-fill
- optimization

V2.3.1 (Feb 11, 2019): fix interpolation issue for time series near masked grid points (coast line)

V2.4.0 (Jun 25, 2019):

- Fixed issue on `DIVAnd.fit`
- `diva3d` correlation length fitting using an empty tuple for len
- `fit_isotropic` arguments

Speeding up things



Julia is fast!

Julia was designed from the beginning for high performance. Julia programs compile to efficient native code for multiple platforms via LLVM.

Don’t forget:

- Packages are pre-compiled when a kernel is started
- Functions gets compiled during the 1st execution
- Some operations can be executed once and their results stored

Speeding up things

Data reading

1. Read the original netCDF ODV file . . . . 30”

```
obsval, obslon, obslat, obsdepth, obstime, obsid =
    NCODV.load(Float64, ODVfile1, ”Water body salinity”);
```

2. Re-write the data . . . . 15”

```
DIVAnd.saveobs(obsfile, ”Water body salinity”, obsval,
   (obslon, obslat, obsdepth, obstime), obsid)
```

3. Use the newly written files for the climatologies . . . . 10”

```
DIVAnd.saveobs(obsfile, ”Water body salinity”, obsval,
   (obslon, obslat, obsdepth, obstime), obsid)
```

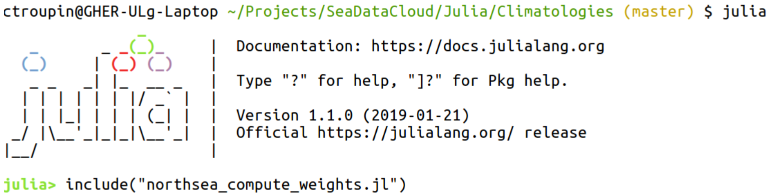
Computing weights ”offline”

```
using DIVAnd
using JLD

datadir = ”/data/SeaDataCloud/NorthSea/”
varname = ”Salinity”
obsfile = joinpath(datadir, ”NorthSea_obs.nc”)
netcdfODV = joinpath(datadir, ”data_from_SDC_NS_DATA_DISCRETE_TS_V1b.nc”
isfile(netcdfODV)
@info(”Reading data from the observation file”)
@time obsval,obslon, obslat, obsdepth, obstime,obsid = DIVAnd.loadobs(F
@info(”Total number of data points: $(length(obsval))”);

@time rdiag=1.0./DIVAnd.weight_RtimesOne((obslon,obslat),(0.03,0.03));
@show maximum(rdiag),mean(rdiag)
save(”northsea_weights.jld”, ”rdiag”, rdiag);
```

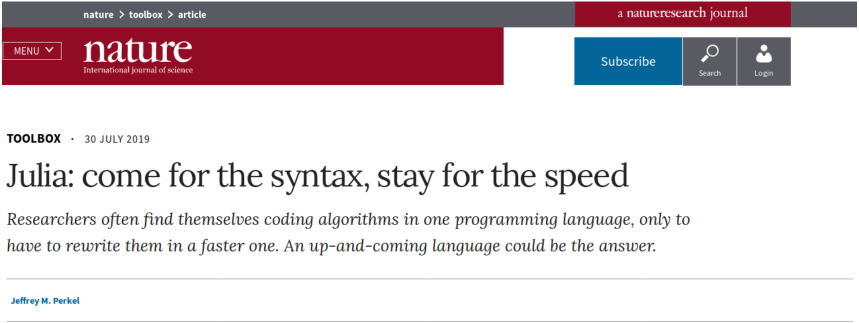
Export the notebooks as `.jl` files



```
ctroupin@GHER-ULg-Laptop ~/Projects/SeaDataCloud/Julia/Climatologies (master) $ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.1.0 (2019-01-21)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> include("northsea_compute_weights.jl")
```

Julia is becoming more famous!



TOOLBOX · 30 JULY 2019

Julia: come for the syntax, stay for the speed

Researchers often find themselves coding algorithms in one programming language, only to have to rewrite them in a faster one. An up-and-coming language could be the answer.

Jeffrey M. Perkel

https://www.nature.com/articles/d41586-019-02310-3

Thanks
for your attention

1. Shall we upload the products to OceanBrowser?
2. Would you agree to publish the notebooks used for the products? (FAI**R** principes) doi