DIVAnd full analysis

This notebook presents the different steps necessary for the creation of a climatology:

1. ODV data reading.
2. Extraction of bathymetry and creation of mask
3. Data download from other sources and duplicate removal.
4. Quality control.
5. Parameter optimisation.
6. Spatio-temporal interpolation with `DIVAnd`.

In [1]:
```
using DIVAnd
using PyPlot
using NCDatasets
using PhysOcean
using DataStructures
using DIVAnd
using PyPlot
if VERSION >= v"0.7.0-beta.0"
    using Dates
    using Statistics
    using Random
    using Printf
else
    using Compat: @info, @warn, @debug
end
using Compat
```

This notebook uses the module DIVAnd

(https://doi.org/10.5281/zenodo.1466985)

Configuration

- Define the horizontal, vertical and temporal resolutions.
- Select the variable

In [2]:
```
dx, dy = 0.125, 0.125
lonr = 11.5:dx:20
latr = 39:dy:46
timerange = [Date(1950,1,1),Date(2017,12,31)];

depthr = [0.,5., 10., 15., 20., 25., 30., 40., 50., 66,
    75, 85, 100, 112, 125, 135, 150, 175, 200, 225, 250,
    275, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750,
    800, 850, 900, 950, 1000, 1050, 1100, 1150, 1200, 1250,
    1300, 1350, 1400, 1450, 1500, 1600, 1750, 1850, 2000];
depthr = [0.,20.,50.];
```

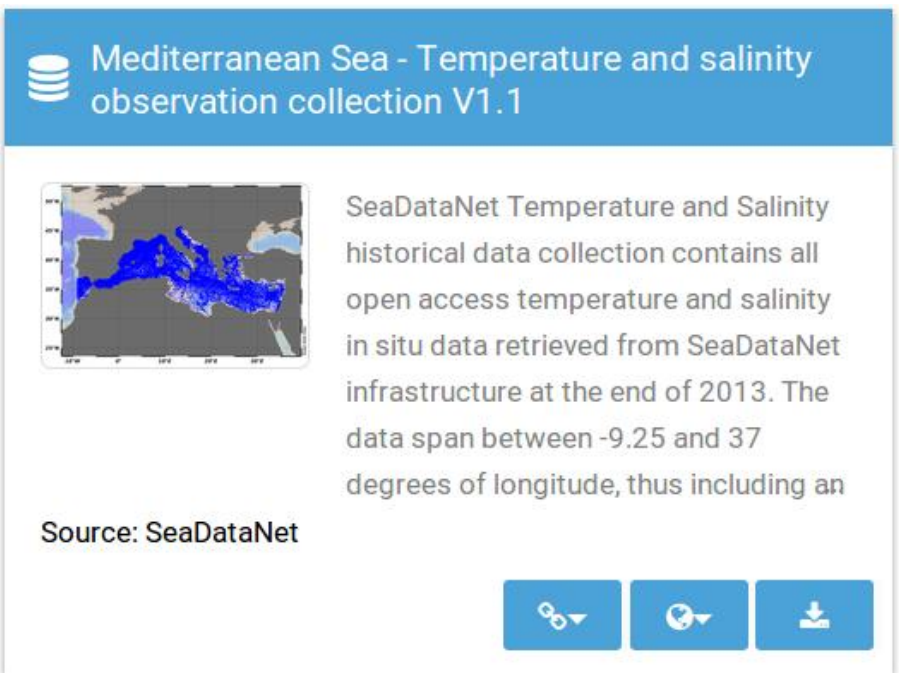
In [3]:
```
varname = "Salinity"
yearlist = [1900:2017];
monthlist = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]];
```

In [4]:
```
TS = DIVAnd.TimeSelectorYearListMonthList(yearlist,monthlist);
@show TS;
```

```
TS = TimeSelectorYearListMonthList{Array{UnitRange{Int64},1},Array{Array{
Int64,1},1}}(UnitRange{Int64}[1900:2017], Array{Int64,1}[[1, 2, 3], [4, 5
, 6], [7, 8, 9], [10, 11, 12]])
```

1. Read your ODV file

Adapt the `datadir` and `datafile` values.
The example is based on a sub-setting of the Mediterranean Sea (https://www.seadatanet.org/Products#/metadata/cd552057-b604-4004-b838-a4f73cc98fcf) aggregated dataset.



In [5]:
```
datadir = "./Adriatic/"
datafile = joinpath(datadir, "AdriaticSea_SDC.txt")
```

Out[5]: "./Adriatic/AdriaticSea_SDC.txt"

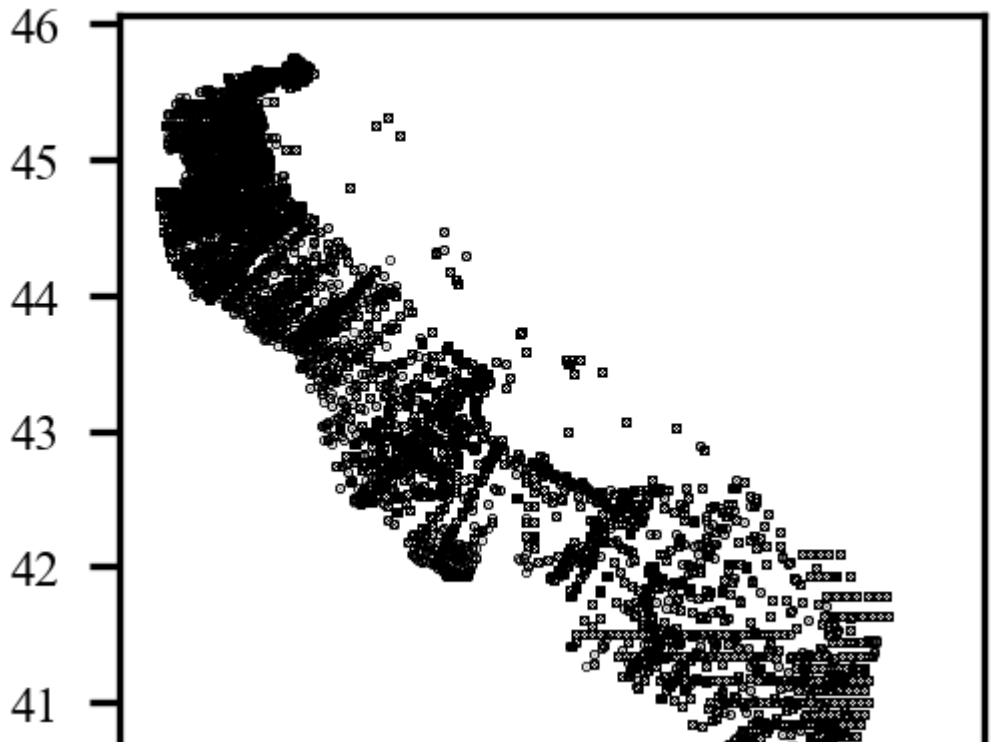
⏳⏳

Depending on the size of your file, this step can take up to a few minutes.

In [6]:
```
@time obsval,obslon,obslat,obsdepth,obstime,obsid = ODVspreadsheet.load(Floa
t64,[datafile],
                              ["Water body salinity"]; nametype = :localname );
```

```
┌ Info: Reading data from file ./Adriatic/AdriaticSea_SDC.txt
└ @ DIVAnd.ODVspreadsheet /home/ctroupin/.julia/dev/DIVAnd/src/ODVspreads
heet.jl:81
┌ Info: No. of profiles in the file: 34709
└ @ DIVAnd.ODVspreadsheet /home/ctroupin/.julia/dev/DIVAnd/src/ODVspreads
heet.jl:250

109.463845 seconds (887.75 M allocations: 40.568 GiB, 46.16% gc time)
```

In [7]:
```
figure("Adriatic-Data", figsize=(2,2))
ax = subplot(1,1,1)
plot(obslon, obslat, "ko", markersize=.1)
aspect_ratio = 1/cos(mean(latr) * pi/180)
ax[:tick_params]("both",labelsize=6)
gca()[:set_aspect](aspect_ratio)
```



Check the extremal values of the observations

In [8]:
```
checkobs((obslon,obslat,obsdepth,obstime),obsval,obsid)
```

```
              minimum and maximum of obs. dimension 1: (

┌ Info: Checking ranges for dimensions and observations
└ @ DIVAnd /home/ctroupin/.julia/dev/DIVAnd/src/obsstat.jl:75

12.25017, 20.11)
              minimum and maximum of obs. dimension 2: (39.55676, 45.755)
              minimum and maximum of obs. dimension 3: (0.0, 1484.203)
              minimum and maximum of obs. dimension 4: (1911-08-17T09:24:
00, 2015-02-10T10:43:00)
                           minimum and maximum of data: (4.07, 40.880001)
```

2. Extract the bathymetry

It is used to delimit the domain where the interpolation is performed.

2.1 Choice of bathymetry

Modify `bathname` according to the resolution required.

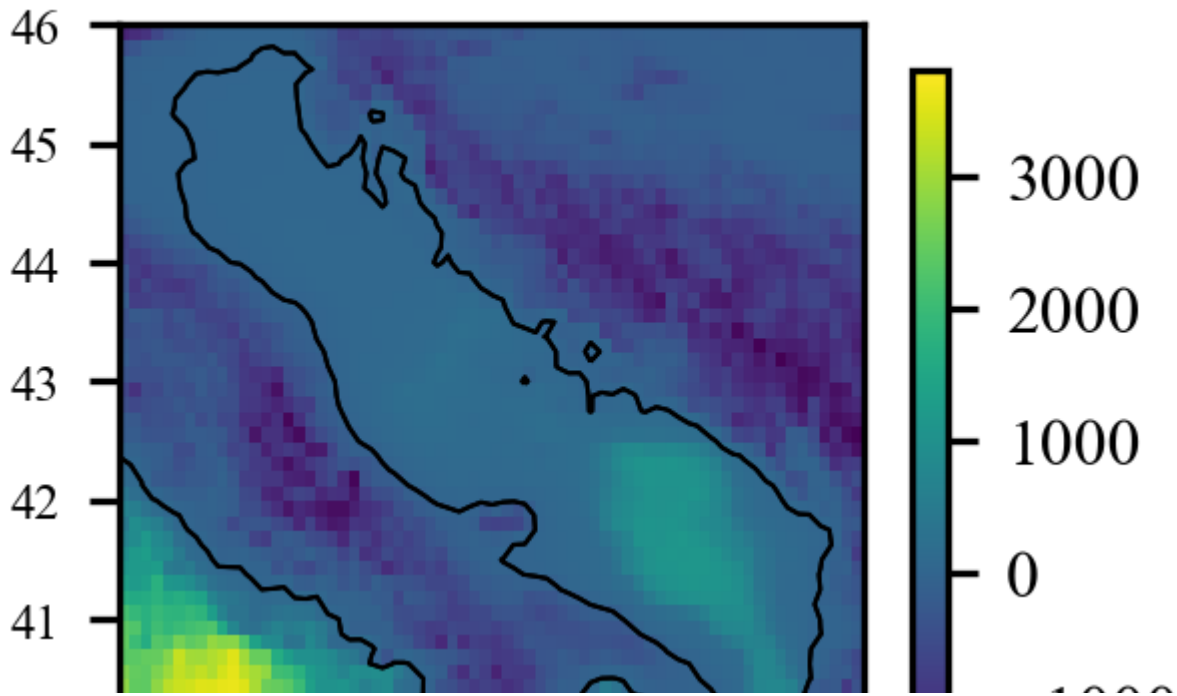
In [9]:
```
bathname = "./data/gebco_30sec_8.nc"
if !isfile(bathname)
    download("https://b2drop.eudat.eu/s/o0vinoQutAC7eb0/download",bathname)
else
    @info("Bathymetry file already downloaded")
end
```

```
┌ Info: Bathymetry file already downloaded
└ @ Main In[9]:5
```

In [10]:
```
@time bx,by,b = load_bath(bathname,true,lonr,latr);
```

```
  3.449754 seconds (9.09 M allocations: 459.119 MiB, 11.90% gc time)
```

In [11]:
```
figure("Adriatic-Bathymetry", figsize=(2,2))
ax = subplot(1,1,1)
pcolor(bx,by,permutedims(b, [2,1]));
colorbar(orientation="vertical", shrink=0.8)[:ax][:tick_params](labelsize=8)
contour(bx,by,permutedims(b, [2,1]), [0, 0.1], colors="k", linewidths=.5)
gca()[:set_aspect](aspect_ratio)
ax[:tick_params]("both",labelsize=6)
```



2.2 Create mask

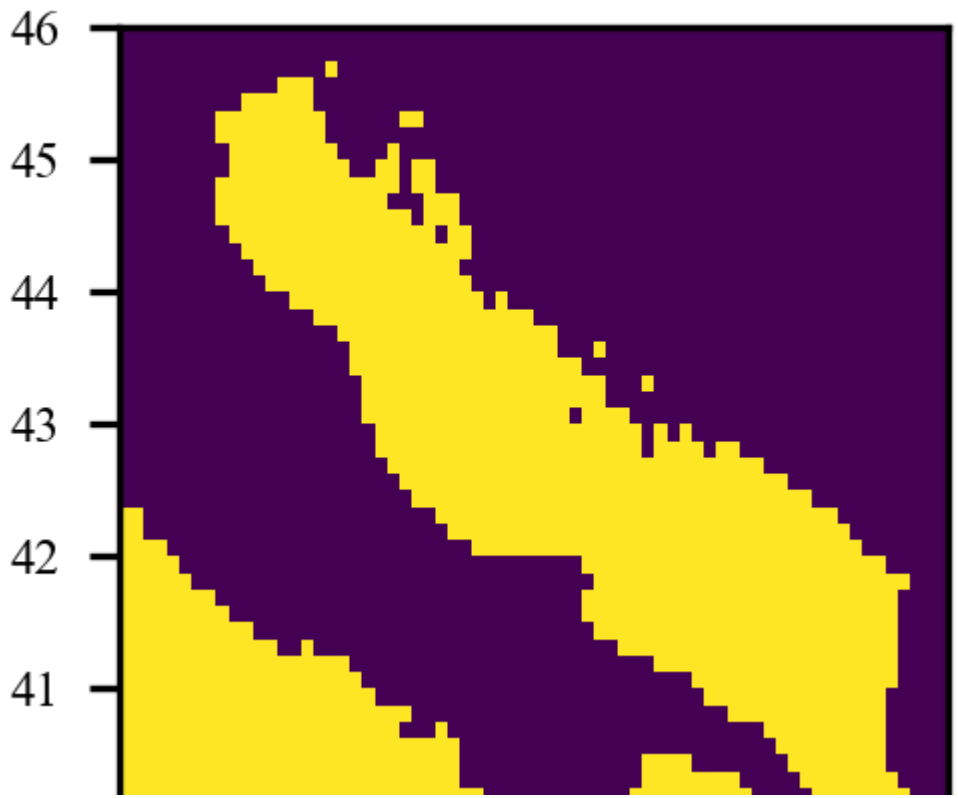
- False for sea
- True for land

In [12]:
```
mask = falses(size(b,1),size(b,2),length(depthr))
for k = 1:length(depthr)
    for j = 1:size(b,2)
        for i = 1:size(b,1)
            mask[i,j,k] = b[i,j] >= depthr[k]
        end
    end
end
@show size(mask)
```

```
size(mask) = (69, 57, 3)
```

Out[12]: (69, 57, 3)

In [13]:
```
figure("Adriatic-Mask", figsize=(2,2))
ax = subplot(1,1,1)
gca()[:set_aspect](aspect_ratio)
ax[:tick_params]("both",labelsize=6)
pcolor(bx,by,permutedims(Float64.(mask[:,:,2]), [2,1]));
```



2.3 Edit the mask

As an example we will remove the Mediterranean Sea from the domain.

In [14]:
```
grid_bx = [i for i in bx, j in by];
grid_by = [j for i in bx, j in by];
```

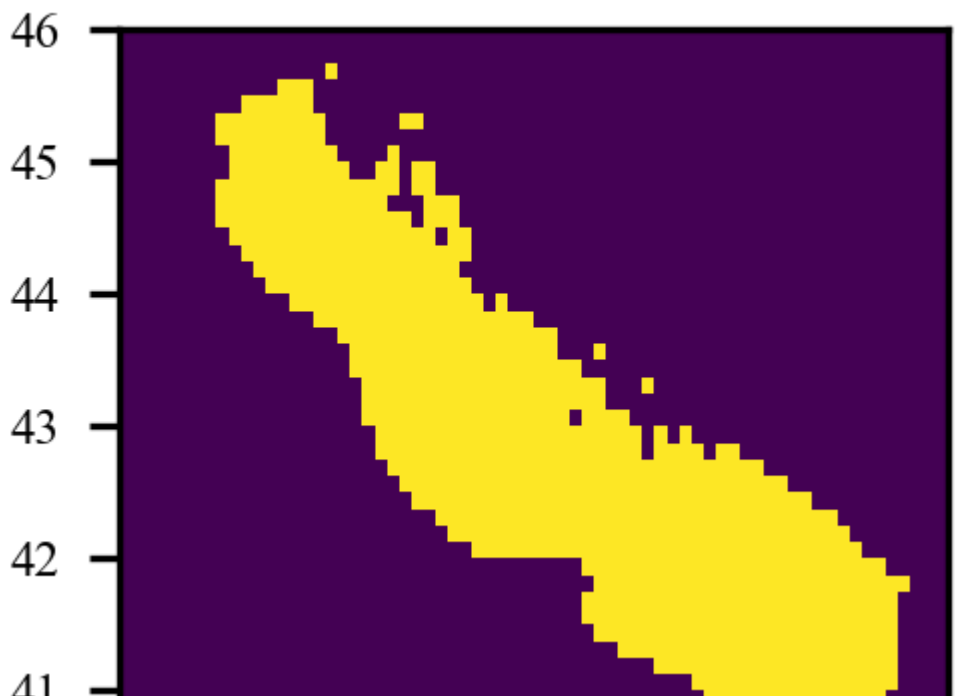
In [15]:
```
mask_edit = copy(mask);
sel_mask1 = (grid_by .<= 42.6) .& (grid_bx .<= 14.);
sel_mask2 = (grid_by .<= 41.2) .& (grid_bx .<= 16.2);
mask_edit = mask_edit .* .!sel_mask1 .* .!sel_mask2;
@show size(mask_edit)
```

```
size(mask_edit) = (69, 57, 3)
```

Out[15]:
```
(69, 57, 3)
```

The edited mask now looks like this:

In [16]:
```
figure("Adriatic-Mask-Edited", figsize=(2,2))
ax = subplot(1,1,1)
gca()[:set_aspect](aspect_ratio)
ax[:tick_params]("both",labelsize=6)
pcolor(bx, by, permutedims(Float64.(mask_edit[:,:,2]), [2,1]));
gca()[:set_aspect](aspect_ratio)
xlim()
```



3. Extract data from other sources

As an illustration we use the World Ocean Database, among other possibilities.

3.1 World Ocean Database

In [17]:
```
# Configuration
email = "your-mail@domain"
woddatadir = "./Adriatic/WOD/"
mkpath(woddatadir);
```

In [18]:
```
# Uncomment the next line if you have to download the data
# WorldOceanDatabase.download(lonr,latr,timerange,varname,email,woddatadir);
```

⏳⏳

Read the data. This can also take up to a few minutes, depending on the size of the domain.

In [19]:
```
@time obsvalwod,obslonwod,obslatwod,obsdepthwod,obstimewod,obsidwod =
WorldOceanDatabase.load(Float64,woddatadir,varname);
```

```
 20.525629 seconds (13.24 M allocations: 1.164 GiB, 11.58% gc time)
```

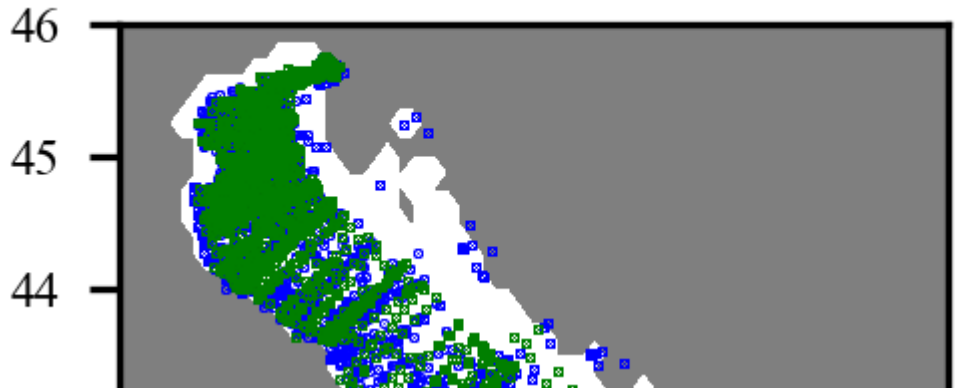
Remove the data outside Adriatic (similar to mask editing)

In [20]:
```
sel_data1 = (obslatwod .<= 42.6) .& (obslonwod .<= 14.);
sel_data2 = (obslatwod .<= 41.2) .& (obslonwod .<= 16.2);
ndataremove = sum((sel_data1) .| (sel_data2))
sel_data = .~((sel_data1) .| (sel_data2));

obslatwod = obslatwod[sel_data];
obslonwod = obslonwod[sel_data];
obsdepthwod = obsdepthwod[sel_data];
obstimewod = obstimewod[sel_data];
obsvalwod = obsvalwod[sel_data];
obsidwod = obsidwod[sel_data];

@info("Number of removed WOD data: $ndataremove");
obsidwod[1:5]
```

```
┌ Info: Number of removed WOD data: 96294
└ @ Main In[20]:13
```

Out[20]:
```
5-element Array{String,1}:
 "wod_007663161O"
 "wod_007663161O"
 "wod_007663161O"
 "wod_007663161O"
```

In [21]:
```
figure("Adriatic-Mask-Edited", figsize=(2,2))
ax = subplot(1,1,1)
ax[:tick_params]("both",labelsize=6)
contourf(bx, by, permutedims(Float64.(mask_edit[:,:,1]),[2,1]),
    levels=[-1e5,0],cmap="binary");
plot(obslon, obslat, "bo", markersize=.2,
    label="SeaDataNet")
plot(obslonwod, obslatwod, "go", markersize=.2,
    label="World Ocean Database");
legend(fontsize=6);
ylim(39.0, 46.0);
xlim(11.5, 20.0);
gca()[:set_aspect](aspect_ratio)
```



3.2 Extract from another source (optional)

Add here the code to read data from another file.

3.3 Remove duplicates

⏳

Criteria (can be adapted according to the application):

- Horizontal distance: 0.01 degree (about 1km)
- Vertical separation: 0.01 m depth
- Time separation: 1 minute.
- Salinity difference: 0.01 psu.

In [22]:
```
@time dupl = DIVAnd.Quadtrees.checkduplicates(
    (obslon,obslat,obsdepth,obstime), obsval,
    (obslonwod,obslatwod, obsdepthwod, obstimewod), obsvalwod,
    (0.01,0.01,0.01,1/(24*60)),0.01);
```

```
 43.028832 seconds (96.82 M allocations: 19.914 GiB, 17.27% gc time)
```

Find the indices of the possible duplicates:

In [23]:
```
index = findall.(.!isempty.(dupl));
ndupl = length(index);
pcdupl = round(ndupl / length(obslon) * 100; digits=2);
@info("Number of possible duplicates: $ndupl")
@info("Percentage of duplicates: $pcdupl%")
```

```
┌ Info: Number of possible duplicates: 1178109
└ @ Main In[23]:4
┌ Info: Percentage of duplicates: 82.34%
└ @ Main In[23]:5
```

If you decide to combine the 2 (or more) datasets:

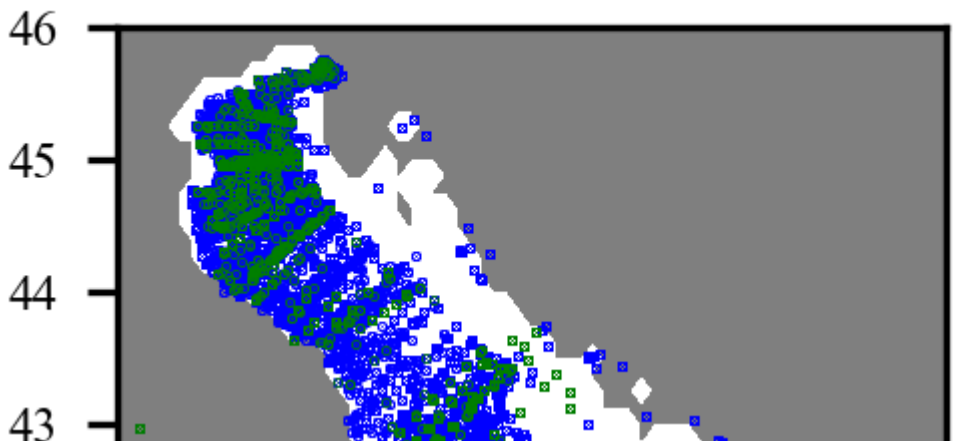
In [24]:
```
newpoints = isempty.(dupl);
@info("Number of new points: " * string(sum(newpoints)))
```

```
┌ Info: Number of new points: 375062
└ @ Main In[24]:2
```

In [25]:
```
obslon = [obslon; obslonwod[newpoints]];
obslat = [obslat; obslatwod[newpoints]];
obsdepth = [obsdepth; obsdepthwod[newpoints]];
obstime = [obstime; obstimewod[newpoints]];
obsval = [obsval; obsvalwod[newpoints]];
obsid = [obsid; obsidwod[newpoints]];
```

Create a plot showing the additional data points:

In [26]:
```
figure("Adriatic-Additional-Data", figsize=(2,2))
ax = subplot(1,1,1)
ax[:tick_params]("both",labelsize=6)
ylim(39.0, 46.0);
xlim(11.5, 20.0);
contourf(bx, by, permutedims(Float64.(mask_edit[:,:,1]),[2,1]),
    levels=[-1e5,0],cmap="binary");
plot(obslon, obslat, "bo", markersize=.2, label="SeaDataNet")
plot(obslonwod[newpoints], obslatwod[newpoints], "go",
    markersize=.2, label="Additional data\nfrom World Ocean Database")
legend(loc=3, fontsize=4)
gca()[:set_aspect](aspect_ratio)
```



4. Quality control

We check the salinity value.
Adapt the criteria to your region and variable.

In [27]:
```
sel = (obsval .<= 40) .& (obsval .>= 25);
```

In [28]:
```
obsval = obsval[sel]
obslon = obslon[sel]
obslat = obslat[sel]
obsdepth = obsdepth[sel]
obstime = obstime[sel]
obsids = obsid[sel];
```

5. Analysis parameters

Modify data weight

⏳⏳⏳

Compute the new weights that takes into account close points.
If the dataset is large, this can take a few minutes.
The maximal and mean values provide an indication of the spatial proximity between the data.
If you apply this technique, you need to adapt `epsilon2`:

In [29]:
```
#@time rdiag=1.0./DIVAnd.weight_RtimesOne((obslon,obslat),(0.03,0.03));
#@show maximum(rdiag),mean(rdiag)
```

Correlation lengths and noise-to-signal ratio

We will use the function `diva3D` for the calculations.
With this function, the correlation length has to be defined in meters, not in degrees.

In [30]:
```
sz = (length(lonr),length(latr),length(depthr));
lenx = fill(100_000.,sz)   # 100 km
leny = fill(100_000.,sz)   # 100 km
lenz = fill(25.,sz);      # 25 m
len = (lenx, leny, lenz);
epsilon2 = 0.1;
#epsilon2 = epsilon2 * rdiag;
```

Output file name

In [31]:
```
filename = "Water_body_$(replace(varname," "=>"_"))_Adriatic.4Danl.nc"
```

Out[31]: "Water_body_Salinity_Adriatic.4Danl.nc"

6. Metadata and attributes

Edit the different fields according to the project, the authors etc.
This is used for the netCDF file but also for the XML needed for the Sextant catalog.

In [32]:

```
metadata = OrderedDict(
    # Name of the project (SeaDataCloud, SeaDataNet, EMODNET-chemistry, ...)
    "project" => "SeaDataCloud",

    # URN code for the institution EDMO registry,
    # e.g. SDN:EDMO::1579
    "institution_urn" => "SDN:EDMO::1579",

    # Production group
    #"production" => "Diva group",

    # Name and emails from authors
    "Author_e-mail" => ["Your Name1 <name1@example.com>", "Other Name <name2
@example.com>"],

    # Source of the observation
    "source" => "observational data from SeaDataNet and World Ocean Atlas",

    # Additional comment
    "comment" => "Duplicate removal applied to the merged dataset",

    # SeaDataNet Vocabulary P35 URN
    # http://seadatanet.maris2.nl/v_bodc_vocab_v2/search.asp?lib=p35
```

SeaDataNet global attributes:

In [33]:
```
ncglobalattrib,ncvarattrib = SDNMetadata(metadata,filename,varname,lonr,latr
)
```

Out[33]:
```
(OrderedDict("project"=>"SeaDataCloud","institution"=>"University of Lieg
e, GeoHydrodynamics and Environment Research","institution_urn"=>"SDN:EDM
O::1579","Author_e-mail"=>"Your Name1 <name1@example.com>, Other Name <na
me2@example.com>","source"=>"observational data from SeaDataNet and World
Ocean Atlas","comment"=>"Duplicate removal applied to the merged dataset"
,"parameter_keyword"=>"Water body salinity","parameter_keyword_urn"=>"SDN
:P35::EPC00001","search_keywords"=>"Salinity of the water column","search
_keywords_urn"=>"SDN:P02::PSAL"…), OrderedDict("units"=>"1e-3","standard_
name"=>"sea_water_salinity","long_name"=>"sea water salinity"))
```

7. Analysis

Remove the result file before running the analysis, otherwise you'll get the message

```
NCDatasets.NetCDFError(13, "Permission denied")
```

In [34]:

```
if isfile(filename)
    rm(filename) # delete the previous analysis
    @info "Removing file $filename"
end
```

```
┌ Info: Removing file Water_body_Salinity_Adriatic.4Danl.nc
└ @ Main In[34]:3
```

7.1 Plotting function

Define a plotting function that will be applied for each time index and depth level. All the figures will be saved in a selected directory.

In [35]:

```
figdir = "./Adriatic/figures/"
if ~(isdir(figdir))
    mkdir(figdir)
else
    @info("Figure directory already exists")
end
```

```
┌ Info: Figure directory already exists
└ @ Main In[35]:5
```

In [36]:

```
function plotres(timeindex,sel,fit,erri)
    tmp = copy(fit)
    nx,ny,nz = size(tmp)
    for i in 1:nz
        figure("Adriatic-Additional-Data", figsize=(2,2))
        ax = subplot(1,1,1)
        ax[:tick_params]("both",labelsize=6)
        ylim(39.0, 46.0);
        xlim(11.5, 20.0);
        title("Depth: $(depthr[i]) \n Time index: $(timeindex)", fontsize=6)
        pcolor(lonr.-dx/2.,latr.-dy/2, permutedims(tmp[:,:,i], [2,1]);
               vmin = 33, vmax = 40)
        colorbar(extend="both", orientation="vertical", shrink=0.8)[:ax][:ti
ck_params](labelsize=8)

        contourf(bx,by,permutedims(b,[2,1]), levels = [-1e5,0],colors = [[.5
,.5,.5]])
        gca()[:set_aspect](aspect_ratio)

        figname = varname * @sprintf("_%02d",i) * @sprintf("_%03d.png",timei
ndex)
        PyPlot.savefig(joinpath(figdir, figname), dpi=600, bbox_inches="tigh
+"\).
```

7.2 Create the gridded fields using `diva3d`

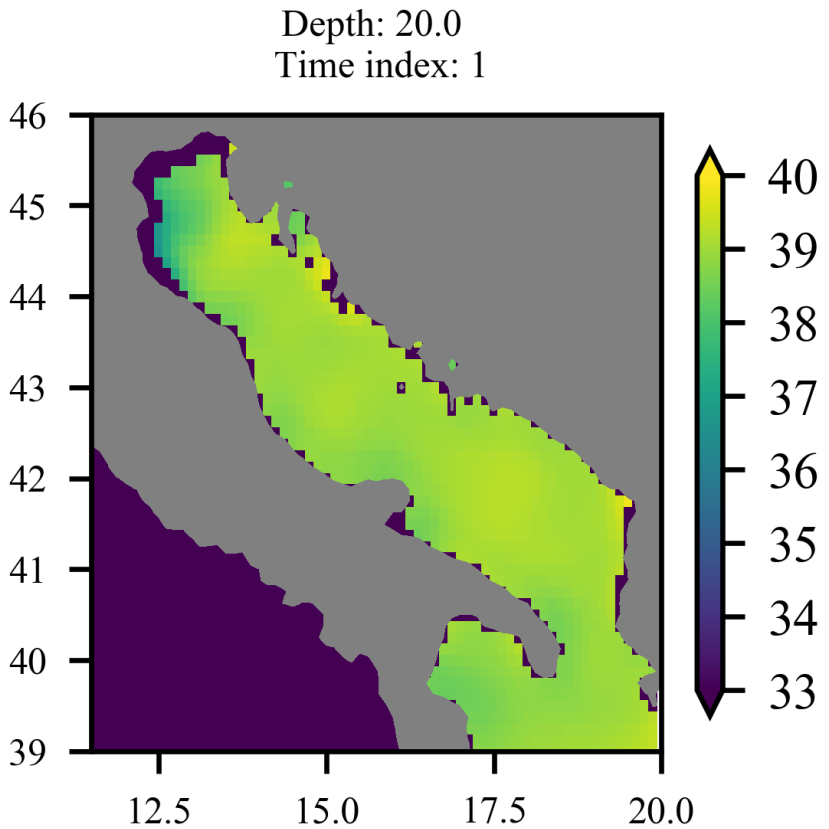
Here only the noise-to-signal ratio is estimated.
Set `fitcorrlen` to `true` to also optimise the correlation length.

In [37]:

```
@time dbinfo = diva3d((lonr,latr,depthr,TS),
    (obslon,obslat,obsdepth,obstime), obsval,
    len, epsilon2,
    filename,varname,
    bathname=bathname,
    plotres = plotres,
    mask = mask_edit,
    fitcorrlen = false,
    niter_e = 2,
    ncvarattrib = ncvarattrib,
    ncglobalattrib = ncglobalattrib
    );
```

```
┌ Info: Creating netCDF file
└ @ DIVAnd /home/ctroupin/.julia/dev/DIVAnd/src/diva.jl:201
┌ Info: Time step 1 / 4
└ @ DIVAnd /home/ctroupin/.julia/dev/DIVAnd/src/diva.jl:236
┌ Info: number of windows: 1
└ @ DIVAnd /home/ctroupin/.julia/dev/DIVAnd/src/DIVAndgo.jl:90
┌ Info: number of windows: 1
└ @ DIVAnd /home/ctroupin/.julia/dev/DIVAnd/src/DIVAndgo.jl:90
┌ Info: Time step 2 / 4
└ @ DIVAnd /home/ctroupin/.julia/dev/DIVAnd/src/diva.jl:236
```

Example of results: salinity at 20 meters for the January-Febrary-March period.



Save the observation metadata in the NetCDF file

In [38]:
```
DIVAnd.saveobs(filename,(obslon,obslat,obsdepth,obstime),obsid);
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

8. Apply a posteriori quality control

We can use the structure `dbinfo`. histogram for quality flags

9. XML metadata

For DIVAnd analysis using SeaDataCloud/EMODnet-Chemistry data, one can create a XML description for the product for Sextant

Name of the project:

- "SeaDataCloud" or
- "EMODNET-chemistry"

In [39]:
```
project = "SeaDataCloud";
```

Download CDI list

In [40]:
```
cdilist = "CDI-list-export.zip"

if !isfile(cdilist)
   download("http://emodnet-chemistry.maris2.nl/download/export.zip",cdilist
)
end
```

If `ignore_errors` is `false` (default), then a missing CDI will stop the creatation of the XML metadata.

In [41]:
```
ignore_errors = true

# File name based on the variable (but all spaces are replaced by _)
xmlfilename = "Water_body_$(replace(varname," "=>"_")).4Danl.xml"
```

Out[41]: "Water_body_Salinity.4Danl.xml"

Uncomment the following line of you are using SeaDataCloud or EMODnet-Chemistry data.

In [42]:
```
# generate a XML file for Sextant (only for )
divadoxml(filename,varname,project,cdilist,xmlfilename,
          ignore_errors = ignore_errors)
```

```
NCDatasets.NetCDFError(-49, "NetCDF: Variable not found")

Stacktrace:
 [1] check at /home/ctroupin/.julia/packages/NCDatasets/jb0hO/src/NCDatas
ets.jl:43 [inlined]
 [2] nc_inq_varid(::Int32, ::String) at /home/ctroupin/.julia/packages/NC
Datasets/jb0hO/src/netcdf_c.jl:1155
 [3] variable(::Dataset, ::String) at /home/ctroupin/.julia/packages/NCDa
tasets/jb0hO/src/NCDatasets.jl:530
 [4] getindex(::Dataset, ::String) at /home/ctroupin/.julia/packages/NCDa
tasets/jb0hO/src/NCDatasets.jl:615
 [5] #gettemplatevars#432(::String, ::Array{String,1}, ::Bool, ::Function
, ::Array{String,1}, ::String, ::String, ::String) at /home/ctroupin/.jul
ia/dev/DIVAnd/src/SDNMetadata.jl:417
 [6] #gettemplatevars at ./none:0 [inlined]
 [7] #divadoxml#442(::Bool, ::Dict{String,Any}, ::Array{String,1}, ::Func
tion, ::Array{String,1}, ::String, ::String, ::String, ::String) at /home
/ctroupin/.julia/dev/DIVAnd/src/SDNMetadata.jl:733
 [8] #divadoxml at ./none:0 [inlined]
```