Author: Heike Müller

Biogeochemical Parameters Oxygen and Chlorophyll of the Baltic Sea

Table of contents

1. Introduction

Go back to the "Table of contents"

This notebook is dedicated to the Baltic Sea.
The Baltic Sea is suffering by eutrophication.
One main reason is the mass of nutrients from the agriculture ending in the Baltic Sea through the rivers.
The consequence is the increase of phytoplankton biomass and the *Chlorophyll a (chl a)* concentration.
Chlorophyll a is an important indicator of water quality.

In relation to this environmental problem is the *Oxygen Depletion*, in particular in deeper layers.
The consequences are in worst case 'dead zones', where no marine life is possible.

The aims of this notebook are

- to visualize the *Dissolved Oxygen concentration* and the *Chlorophyll a concentration* over a time period.
- to show you the programming techniques to access the WEkEO data, work with the data and programm the visualization

Learning outcomes

- How to access the data from the WEkEO-Platform.
- How to create maps of one product for one date.
- How to create maps for a serie of dates.
- How to create maps on a regional zoom.
- How to create subplots.
- How to create maps for 2 parameters with subplots.
- How to create a vertical profile.
- How to use functions and loops in Python.

2. Explore the Data

Go back to the "Table of contents"

This notebook is using the **Baltic Sea Biogeochemistry Reanalysis model**.
We are downloading the monthly means data of the biogeochemical parameters **Dissolved Oxygen (o2)** and **Chlorophyll (chl)**.
The downloaded time period is from 2016-01-16 to 2020-12-01.
The downloaded depths are approx. 1.5 - 93.2 m.

Product Description

Baltic Sea Biogeochemistry Reanalysis

Product ID

EO:MO:DAT:BALTICSEA_REANALYSIS_BIO_003_012:dataset-reanalysis-scobi-monthlymeans

Product Links

BALTICSEA_REANALYSIS_BIO_003_012

WEkEO Data

Temporal Resolution

Monthly Means

Downloaded Time Period

from 2016-01-16 to 2020-12-01

Downloaded Depths

1.5013654232025146 - 93.28632354736328 m

Variables used

o2: Mole concentration of dissolved molecular oxygen in sea water mmol/m3

chl: Mass concentration of chlorophyll a in sea water mg/m3

Import of required Python libraries

Module name	Description
NumPy	*NumPy* is a library for scientific computing with Python. It provides a multidimensional array object and functions for working with these arrays.
xarray	*xarray* is a library for working with labelled multi-dimensional arrays. It introduces labels in the form of dimensions, coordinates and attributes on top of raw NumPy-like arrays.
Matplotlib	*Matplotlib* is the basic plotting library for creating static, animated and interactive visualization.
Cartopy	*Cartopy* is a library for processing geospatial data, geospatial data analyses and producing geospatial maps.
Panda	*Panda* is a library for data analsyis and manipulation.
os	*os* is a library, that gives us access on operating system functionality. For example: path manipulation.
glob	*glob* is a library, that allows us to discover files/pathnames matching a specified pattern.

Module name	Description
warnings	*warnings* is a library to manage warnings

In [1]:
```
# To avoid warning messages
import warnings
warnings.filterwarnings('ignore')

# Import libraries
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import matplotlib.patches as mpatches
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import pandas as pd
import glob
import datetime
import os
```

Install the WEkEO HDA client

The WEkEO HDA Client is a Python based library.

In [ ]:
```
pip install -U hda
```

Configure the WEkEO API Authentication

You need to create a configuration file in your root directory with your WEkEO Login data.

- Go to your home directory
- Click: File -> New -> Text File

Copy the following data:

url: https://wekeo-broker.apps.mercator.dpi.wekeo.eu/databroker user: your_wekeo_username password: your_wekeo_password

- Fill in your WEkEO Login data

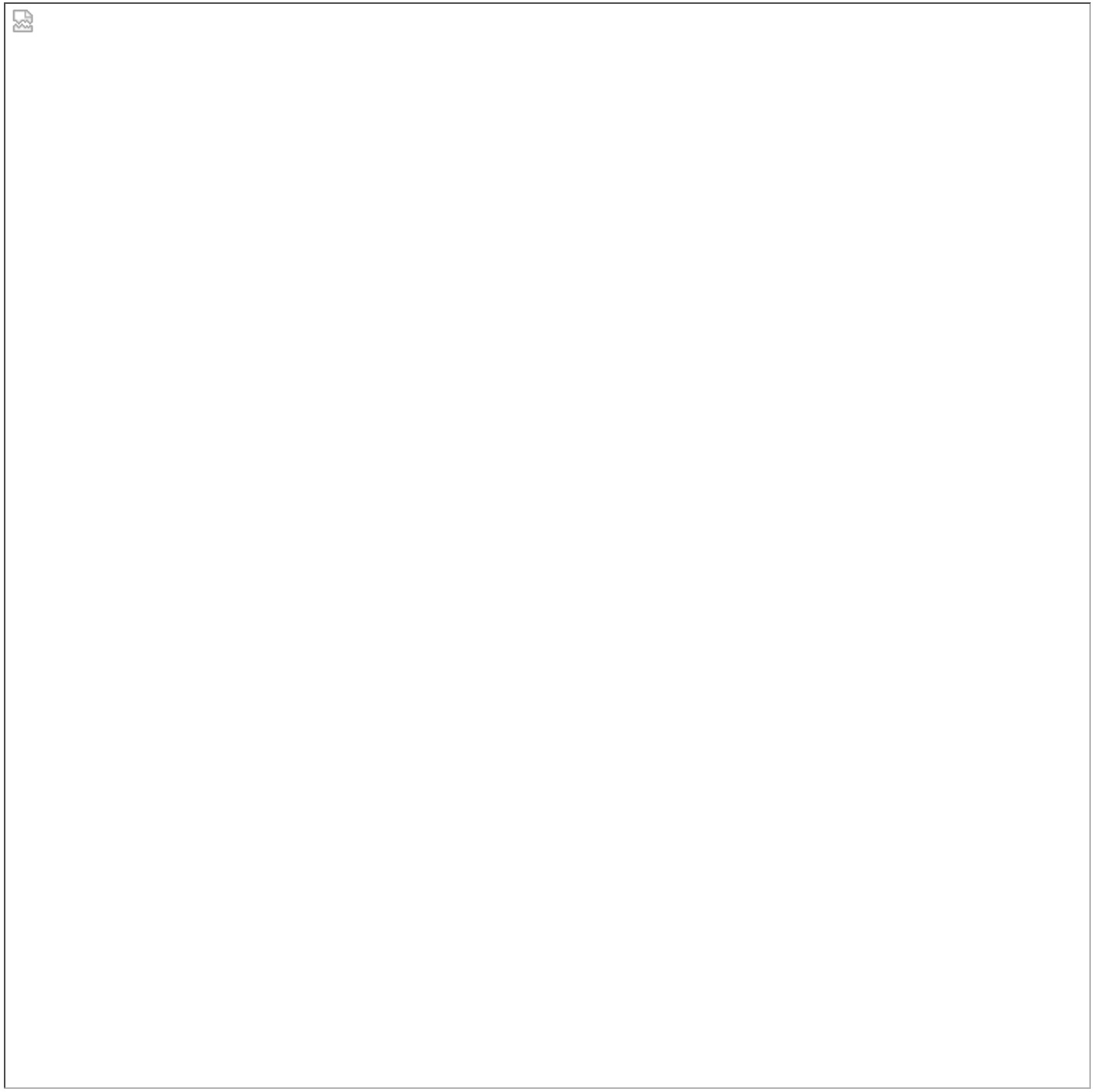
- Click: File -> Save File As -> .hdarc
- Move the file .hdarc to /home/jovyan :
  - Start a terminal in the home directory.
  - If you enter *ls -a* you can see the file in the directory.
  - Enter the following comand line: *cp .hdarc /home/jovyan*

After ending the WEkEO JupyterLab session, the .hdarc file ist deleted from /home/jovyan. The next comand copies it back. You have to fill in your user name.

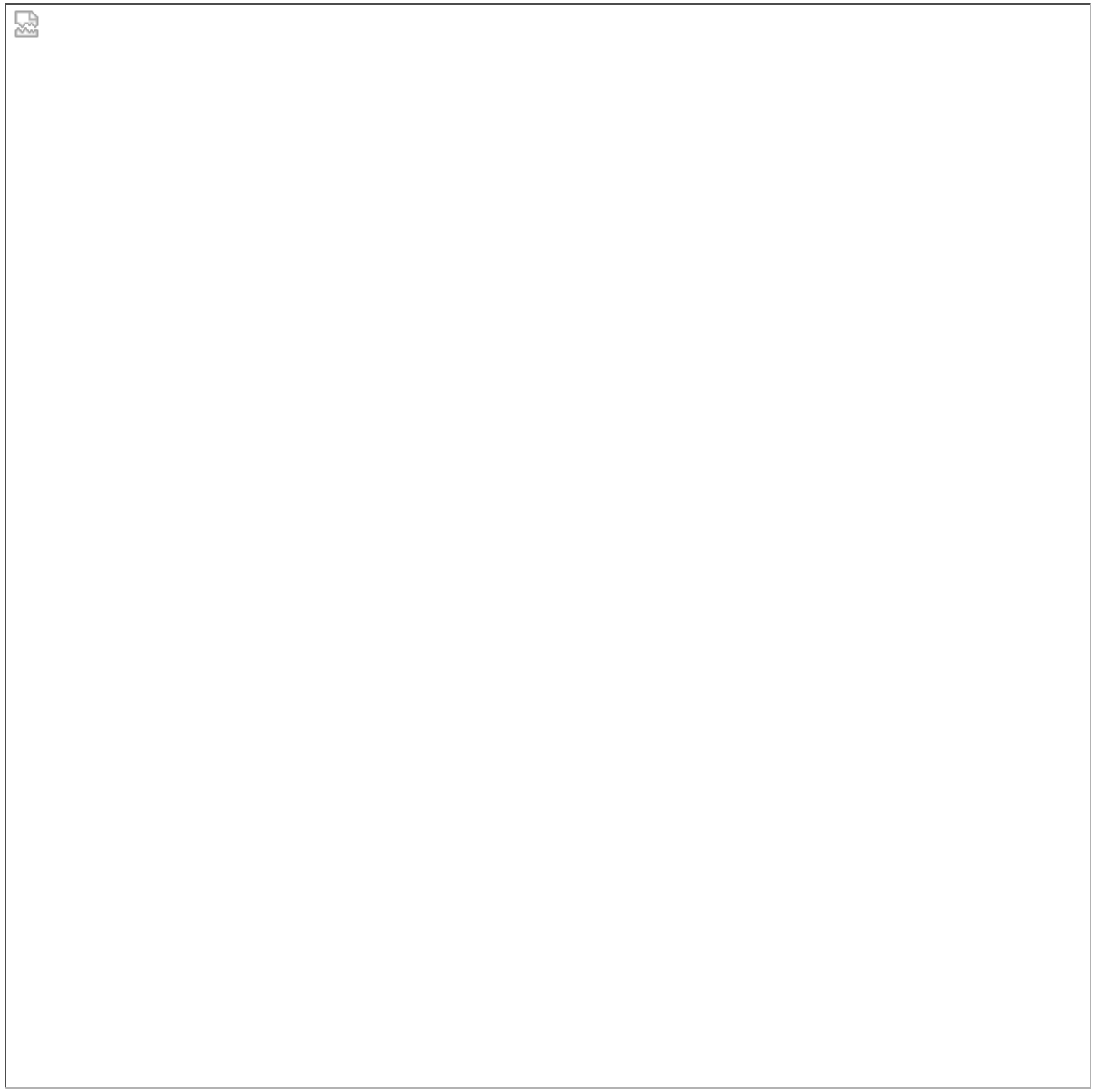
In [ ]:
```
os.system('cp /home/username/.hdarc /home/jovyan')
```

Get the WEkEO API Request

- Open WEkEO DATA
- CLick on + to add a layer
- Select the data set *Baltic Sea Biogeochemistry Reanalysis* in the Catalogue:

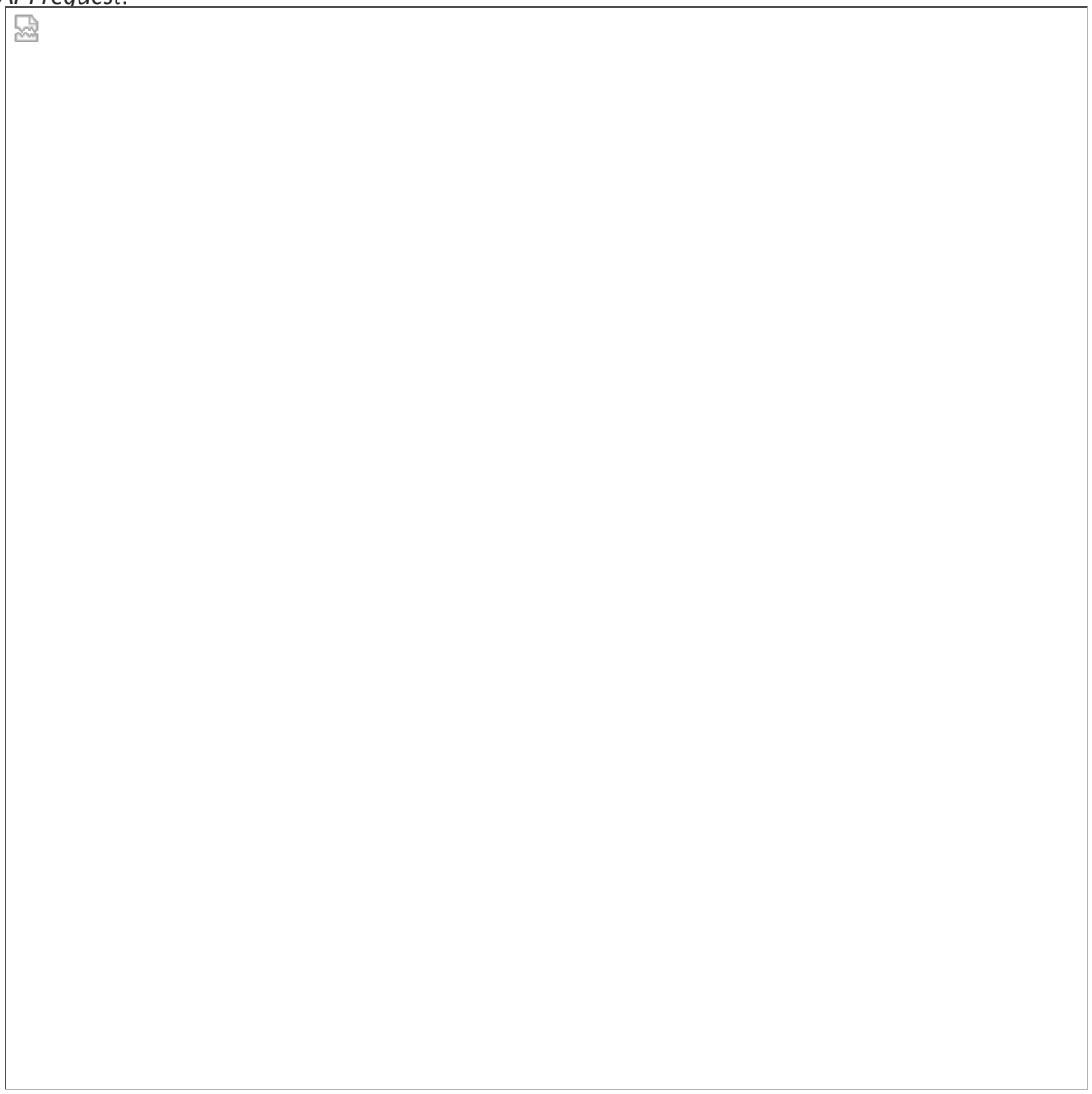


- Click: *Add to map...*
- Select: *Mole concentration of dissolved molecular oxygen in sea water* and click *Add to map*.



- You have to login on WEkEO to download the data.

- Enter the longitude (W: 8, E: 31), lattitude (S: 53, N:66), start date (16.01.2016), end date (01.12.2020), the start depth (1.5014) and the end depth (93.2863).
- Klick on *Show API request*.



Here you can copy the API request, It is already in the code of this notebook.

- We are downloading the Parameter **Dissolved Oxygen** and **Chlorophyll a**.
- You saw already the API-Request for the Dissolved Oxygen.
- To download the Chlorophyll a data, it is the same procedure besides you have to select *Mass concentration of chlorophyll a in sea water* in the second picture.
- We have two API Requests and 2 downloading processes.
- To download both parameters in one step, we have to change the API Request manually. We are not doing this here, because we can only download 1 GB.

Change the download path

In this notebook, we are changing the working directory for the download.
This is, because there were sometimes problems when the downloaded data were moved in another directory by the os-module.
After the download is finished and the datasets are opened, we will change the working directory back.

In [2]:
```
# We define a function to check, if the folder 'dir_path' exists. Otherwise it is created.

def dir_exist(dir_path):
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)
```

In [ ]:
```
# Get the working directory
os.getcwd()
```

In [4]:
```
# Check if the folder './Data' exists. Otherwise it is created.
dir_exist('./Data')
```

In [ ]:

```
# Change the working directory to the Folder \Data
os.chdir('Data')

# Get the new working directory
data_dir = os.getcwd()

# show data_dir
data_dir
```

Download the Data

Temporary Download-Problem

I contacted the user support and have been provided of the confirmation that this Copernicus Marine product are temporary not fetchable using the HDA since July, 22th 2022.
You can download the data from the product site: BALTICSEA_REANALYSIS_BIO_003_012

The download of data from WEkEO ist limited to 1 GB.
If you need more, you have to split it.

In the next two cells you see the queries of the API Requests for **Dissolved Oxygen** and **Chlorophyll a**.

In [6]:

```
data_o2 = {
  "datasetId": "EO:MO:DAT:BALTICSEA_REANALYSIS_BIO_003_012:dataset-reanalysis-scobi-monthlymeans",
  "boundingBoxValues": [
    {
      "name": "bbox",
      "bbox": [
        8,
        53,
        31,
        66
      ]
    }
  ],
  "dateRangeSelectValues": [
    {
      "name": "position",
      "start": "2016-01-16T00:00:00.000Z",
      "end": "2020-12-01T00:00:00.000Z"
    }
  ],
  "multiStringSelectValues": [
    {
      "name": "variable",
      "value": [
        "o2"
      ]
    }
  ],
  "stringChoiceValues": [
    {
      "name": "service",
      "value": "BALTICSEA_REANALYSIS_BIO_003_012-TDS"
    },
    {
      "name": "product",
      "value": "dataset-reanalysis-scobi-monthlymeans"
    },
    {
      "name": "startDepth",
      "value": "1.5013654232025146"
    },
    {
      "name": "endDepth",
```

```
      "value": "93.28632354736328"
    }
  ]
}
```


In [7]:

```
data_chl = {
  "datasetId": "EO:MO:DAT:BALTICSEA_REANALYSIS_BIO_003_012:dataset-reanalysis-scobi-monthlymeans",
  "boundingBoxValues": [
    {
      "name": "bbox",
      "bbox": [
        8,
        53,
        31,
        66
      ]
    }
  ],
  "dateRangeSelectValues": [
    {
      "name": "position",
      "start": "2016-01-16T00:00:00.000Z",
      "end": "2020-12-01T00:00:00.000Z"
    }
  ],
  "multiStringSelectValues": [
    {
      "name": "variable",
      "value": [
        "chl"
      ]
    }
  ],
  "stringChoiceValues": [
    {
      "name": "service",
      "value": "BALTICSEA_REANALYSIS_BIO_003_012-TDS"
    },
    {
      "name": "product",
      "value": "dataset-reanalysis-scobi-monthlymeans"
    },
    {
```

```
      "name": "startDepth",
      "value": "1.5013654232025146"
    },
    {
      "name": "endDepth",
      "value": "93.28632354736328"
```

In [ ]:
```
# The following line imports the HDA Python library
# We need this to download the data

from hda import Client
c = Client(debug=True)
```

In [ ]:
```
# Download the Dissolved Oxygen data

matches = c.search(data_o2)
print(matches)
matches.download()
```

In [ ]:
```
# Download the Chlorophyll data

matches = c.search(data_chl)
print(matches)
matches.download()
```

In [6]:
```
# Read in the downloaded nc-files sorted in descending order
# data_dir is the Data-Directory

files = [file for file in os.listdir(data_dir) if (file.lower().endswith('.nc'))]
files_list = []

for file in sorted(files,key=os.path.getmtime):
    files_list.append(file)
```

In [7]:
```
# Show the downloaded files

files_list
```

Out[7]:
```
['dataset-reanalysis-scobi-monthlymeans_1658565390466.nc',
```

In [8]:
```
# Length of files_list
# This is the number of downloaded files

length = len(files_list)
length
```

Out[8]: 2

In [9]:
```
# Open the o2-Dataset

data_baltic_o2=xr.open_dataset(files_list[0])

# Show data_baltic_o2
# Check, if there is a data variable o2
data_baltic_o2
```

Out[9]: xarray.Dataset

► Dimensions: (**time**: 60, **depth**: 21, **latitude**: 523, **longitude**: 383)

▼ Coordinates:

depth	(depth)	float32	1.501 4.513 7.54 ... 72.31 78.61
latitude	(latitude)	float32	48.49 48.53 48.56 ... 65.86 65...
time	(time)	datetime64[ns]	2016-01-16T12:00:00 ... 2020...
longitude	(longitude)	float32	9.014 9.069 9.125 ... 30.18 30...

▼ Data variables:

o2	(time, depth, latitude, longitude)	float32	...

► Attributes: (24)

In [10]:
```
# Open the chl-Dataset

data_baltic_chl=xr.open_dataset(files_list[1])

# Show data_baltic_chl
# Check, if there is a data variable chl
data_baltic_chl
```

Out[10]:

xarray.Dataset

► Dimensions: (**depth**: 21, **time**: 60, **latitude**: 523, **longitude**: 383)

▼ Coordinates:

depth	(depth)	float32	1.501 4.513 7.54 ... 72.31 78.61
latitude	(latitude)	float32	48.49 48.53 48.56 ... 65.86 65...
time	(time)	datetime64[ns]	2016-01-16T12:00:00 ... 2020...
longitude	(longitude)	float32	9.014 9.069 9.125 ... 30.18 30...

▼ Data variables:

chl	(time, depth, latitude, longitude)	float32	...

► Attributes: (24)

In [ ]:
```
# Set the working directory back to the folder BalticSea
# This is necessary because of the maps-directories

dirname = os.path.dirname(os.getcwd())
os.chdir(dirname)

# Check the working directory
os.getcwd()
```

In [12]:
```
# Check the dimensions of data_baltic_o2

data_baltic_o2.dims
```

Out[12]:
```
Frozen({'time': 60, 'depth': 21, 'latitude': 523, 'longitude': 383})
```

In [13]:
```
# Check the variables of data_baltic_o2

data_baltic_o2.variables
```

Out[13]:
```
Frozen({'o2': <xarray.Variable (time: 60, depth: 21, latitude: 523, longitude: 383)>
[252389340 values with dtype=float32]
Attributes:
    standard_name:  mole_concentration_of_dissolved_molecular_oxygen_in_sea_w...
    long_name:      Dissolved_Oxygen_Concentration
    units:          mmol m-3
    _ChunkSizes:    [  1  19 175 128], 'depth': <xarray.IndexVariable 'depth' (depth: 21)>
array([ 1.501365,  4.513265,  7.539884, 10.584852, 13.652688, 16.749002,
       19.880766, 23.056614, 26.287226, 29.58579 , 32.96853 , 36.455338,
       40.070496, 43.84348 , 47.809814, 52.011955, 56.50011 , 61.332855,
       66.57746 , 72.30962 , 78.612465], dtype=float32)
Attributes:
    axis:                     Z
    positive:                 down
    standard_name:            depth
    long_name:                depth
    units:                    m
    _CoordinateAxisType:      Height
    _CoordinateZisPositive:   down
    valid_min:                1.5013654
    valid_max:                78.612465, 'latitude': <xarray.IndexVariable 'latitude' (latitude: 523)>
array([48.4917  , 48.525032, 48.558365, ..., 65.82475 , 65.858086, 65.89142 ],
      dtype=float32)
Attributes:
    axis:                 Y
    standard_name:        latitude
    long_name:            latitude
    units:                degrees_north
    _CoordinateAxisType:  Lat
    valid_min:            48.4917
    valid_max:            65.89142, 'time': <xarray.IndexVariable 'time' (time: 60)>
array(['2016-01-16T12:00:00.000000000', '2016-02-15T12:00:00.000000000',
       '2016-03-16T12:00:00.000000000', '2016-04-16T00:00:00.000000000',
       '2016-05-16T12:00:00.000000000', '2016-06-16T00:00:00.000000000',
       '2016-07-16T12:00:00.000000000', '2016-08-16T12:00:00.000000000',
       '2016-09-16T00:00:00.000000000', '2016-10-16T12:00:00.000000000',
       '2016-11-16T00:00:00.000000000', '2016-12-16T12:00:00.000000000',
       '2017-01-16T12:00:00.000000000', '2017-02-15T00:00:00.000000000',
```

```
       '2017-03-16T12:00:00.000000000', '2017-04-16T00:00:00.000000000',
       '2017-05-16T12:00:00.000000000', '2017-06-16T00:00:00.000000000',
       '2017-07-16T12:00:00.000000000', '2017-08-16T12:00:00.000000000',
       '2017-09-16T00:00:00.000000000', '2017-10-16T12:00:00.000000000',
       '2017-11-16T00:00:00.000000000', '2017-12-16T12:00:00.000000000',
       '2018-01-16T12:00:00.000000000', '2018-02-15T00:00:00.000000000',
       '2018-03-16T12:00:00.000000000', '2018-04-16T00:00:00.000000000',
       '2018-05-16T12:00:00.000000000', '2018-06-16T00:00:00.000000000',
       '2018-07-16T12:00:00.000000000', '2018-08-16T12:00:00.000000000',
       '2018-09-16T00:00:00.000000000', '2018-10-16T12:00:00.000000000',
       '2018-11-16T00:00:00.000000000', '2018-12-16T12:00:00.000000000',
       '2019-01-16T12:00:00.000000000', '2019-02-15T00:00:00.000000000',
       '2019-03-16T12:00:00.000000000', '2019-04-16T00:00:00.000000000',
       '2019-05-16T12:00:00.000000000', '2019-06-16T00:00:00.000000000',
       '2019-07-16T12:00:00.000000000', '2019-08-16T12:00:00.000000000',
       '2019-09-16T00:00:00.000000000', '2019-10-16T12:00:00.000000000',
       '2019-11-16T00:00:00.000000000', '2019-12-16T12:00:00.000000000',
       '2020-01-16T12:00:00.000000000', '2020-02-15T12:00:00.000000000',
       '2020-03-16T12:00:00.000000000', '2020-04-16T00:00:00.000000000',
       '2020-05-16T12:00:00.000000000', '2020-06-16T00:00:00.000000000',
       '2020-07-16T12:00:00.000000000', '2020-08-16T12:00:00.000000000',
       '2020-09-16T00:00:00.000000000', '2020-10-16T12:00:00.000000000',
       '2020-11-16T00:00:00.000000000', '2020-12-16T12:00:00.000000000'],
      dtype='datetime64[ns]')
Attributes:
    axis:                 T
    long_name:            Validity time
    standard_name:        time
    _ChunkSizes:          512
    _CoordinateAxisType:  Time
    valid_min:            24121.5
    valid_max:            25917.5, 'longitude': <xarray.IndexVariable 'longitude' (longitude: 383)>
array([ 9.013755,  9.06931 ,  9.124865, ..., 30.124655, 30.18021 , 30.235765],
      dtype=float32)
Attributes:
    standard_name:        longitude
    long_name:            longitude
    units:                degrees_east
    axis:                 X
    _CoordinateAxisType:  Lon
    valid_min:            9.013755
```

Exercise

Check the dimensions and variables of data_baltic_chl.

In [14]:
```
# Get the o2 long_name variable
# We want to use this name for the titles of the maps

o2_name = data_baltic_o2.o2.long_name
o2_name
```

Out[14]: 'Dissolved_Oxygen_Concentration'

In [15]:
```
# Get the chl long_name variable
# We want to use this name for the titles of the maps

chl_name = data_baltic_chl.chl.long_name
chl_name
```

Out[15]: 'Chlorophyll_a_Concentration'

In [16]:
```
# Get the minimum depth of the o2 datasets
depth_min_o2 = data_baltic_o2.depth.valid_min
depth_min_o2
```

Out[16]: 1.5013654

In [17]:
```
# Get the minimum depth of the chl datasets
depth_min_chl = data_baltic_chl.depth.valid_min
depth_min_chl
```

Out[17]: 1.5013654

In [18]:
```
# In this notebook only the Chlorophyll data with the minimum depth are considered
# Select this data
chl_data = data_baltic_chl.sel(depth=depth_min_chl, method='nearest')
```

3. Dissolved Oxygen

Go back to the "Table of contents"

In this section we are plotting the **Dissolved Oxygen** for one date, for a serie of dates and for a serie of depths.

3.1 Dissolved Oxygen of the Baltic Sea for one date

In [19]:
```
# Get the minimum and maximum longitude and latitude of the o2 dataset

lon_min = data_baltic_o2.longitude.valid_min
print(lon_min)
lon_max = data_baltic_o2.longitude.valid_max
print(lon_max)
lat_min = data_baltic_o2.latitude.valid_min
print(lat_min)
lat_max = data_baltic_o2.latitude.valid_max
print(lat_max)
```

```
9.013755
30.235765
48.4917
65.89142
```

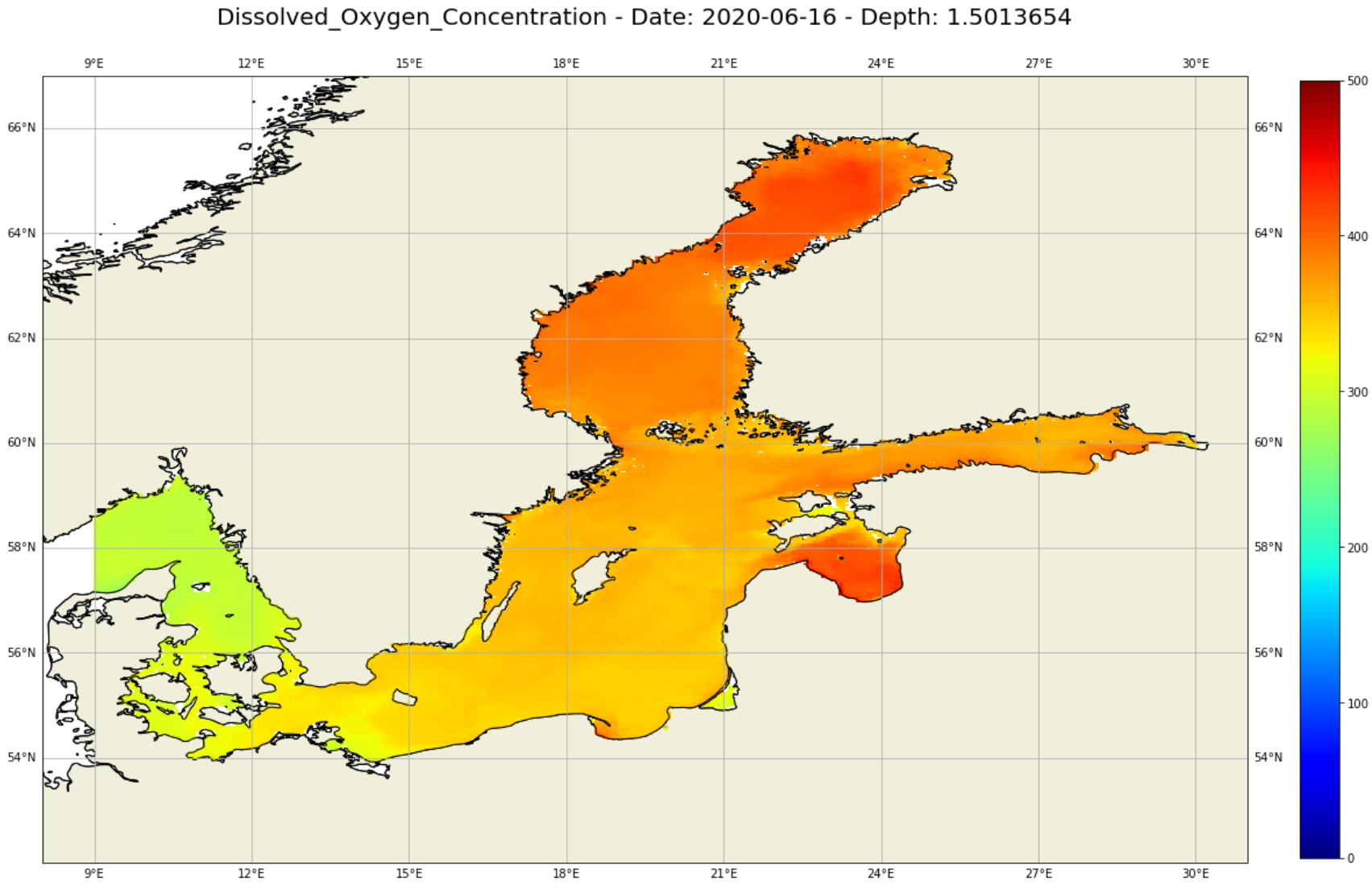
In [20]:
```
# We set the longitude and latitude for the maps (plot window) we want to generate

long_lat = [8, 31, 52, 67]
long_lat
```

Out[20]: [8, 31, 52, 67]

In [22]:
```
# Definition of a function

# We are defining a function for plotting a map of Dissolved Oxygen
# Input parameter: date, depth_x (depth), map_dir (directory to save the map)

def diso2_map(date, depth_x, map_dir):

    # Select the o2-data with the input parameter depth_x and date
    o2_baltic = data_baltic_o2.sel(depth=depth_x, method='nearest').squeeze()
    o2_baltic_map = o2_baltic['o2'].sel(time=date, method = 'nearest').squeeze()

    # Define the settings of the plot
    f = plt.figure(figsize=(20, 15))                                        # define the size of the plot
    ax = plt.axes(projection=ccrs.PlateCarree())                            # create an ax and select the p
    ax.coastlines()                                                         # add coastlines
    gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True)             # add map gridlines
    ax.add_feature(cfeature.LAND, zorder=1, edgecolor='k')                  # add land mask
    ax.set_extent(long_lat,crs=ccrs.PlateCarree())                          # define the extent of the map

    # Plot the o2-data, set the minimum and maximum values of the colorbar and the colormap to use
    im = ax.pcolor(o2_baltic_map['longitude'].data, o2_baltic_map['latitude'].data,o2_baltic_map,vmin=0,vmax=500,cmap=

    # Add a title
    Title = o2_name + ' - Date: {}'.format(date) + ' - Depth: ' + str(depth_x)
    plt.title(Title + '\n', fontsize=20)

    # Add color scale
    f.colorbar(im,ax=ax,fraction=0.03, pad=0.04)

    # Save the figure in the directory (parameter map_dir)
    plt.savefig(map_dir + '/o2_map_' + date + '_' + str(int(depth_x)))
```

In [23]:
```
# Check if the folders './maps' and './maps/o2_maps' exists. Otherwise they are created.
dir_exist('./maps')
dir_exist('./maps/o2_maps')
```

In [24]:
```
# Call the funktion disso2_map with a date
# The map is saved in the folder maps/o2_maps
diso2_map('2020-06-16', depth_min_o2, 'maps/o2_maps' )
```



Exercise

Call the funktion disso2_map with another date.

3.2 Dissolved Oxygen of the Baltic Sea for a serie of dates

Go back to the "Table of contents"

In this section we are plotting maps of a serie of dates of the **Dissolved Oxygen**.

We are selecting every second month of the year 2020.

In [25]:
```
# Length of the date array of the o2-data
time_len_o2 = len(data_baltic_o2.time)
time_len_o2
```

Out[25]: 60

In [26]:
```
# Variable for the list of the selected dates
Date_list_o2 = []

# 1. Parameter: First index of interested date. We start with the index 48, that is the date 16.01.2020.
# 2. Parameter: length of the date array
# 3. Parameter: intervall of interested dates. We choose index steps 2.
# The selected dates with the indices are printed.

for i in range(48, time_len_o2, 2):
    print(i)
    dt = data_baltic_o2.time[i].values
    dt_str = str(dt)[:10]
    print(dt_str)
    Date_list_o2.append(dt_str)

print(Date_list_o2)

# Print the number of the selected dates
len(Date_list_o2)
```

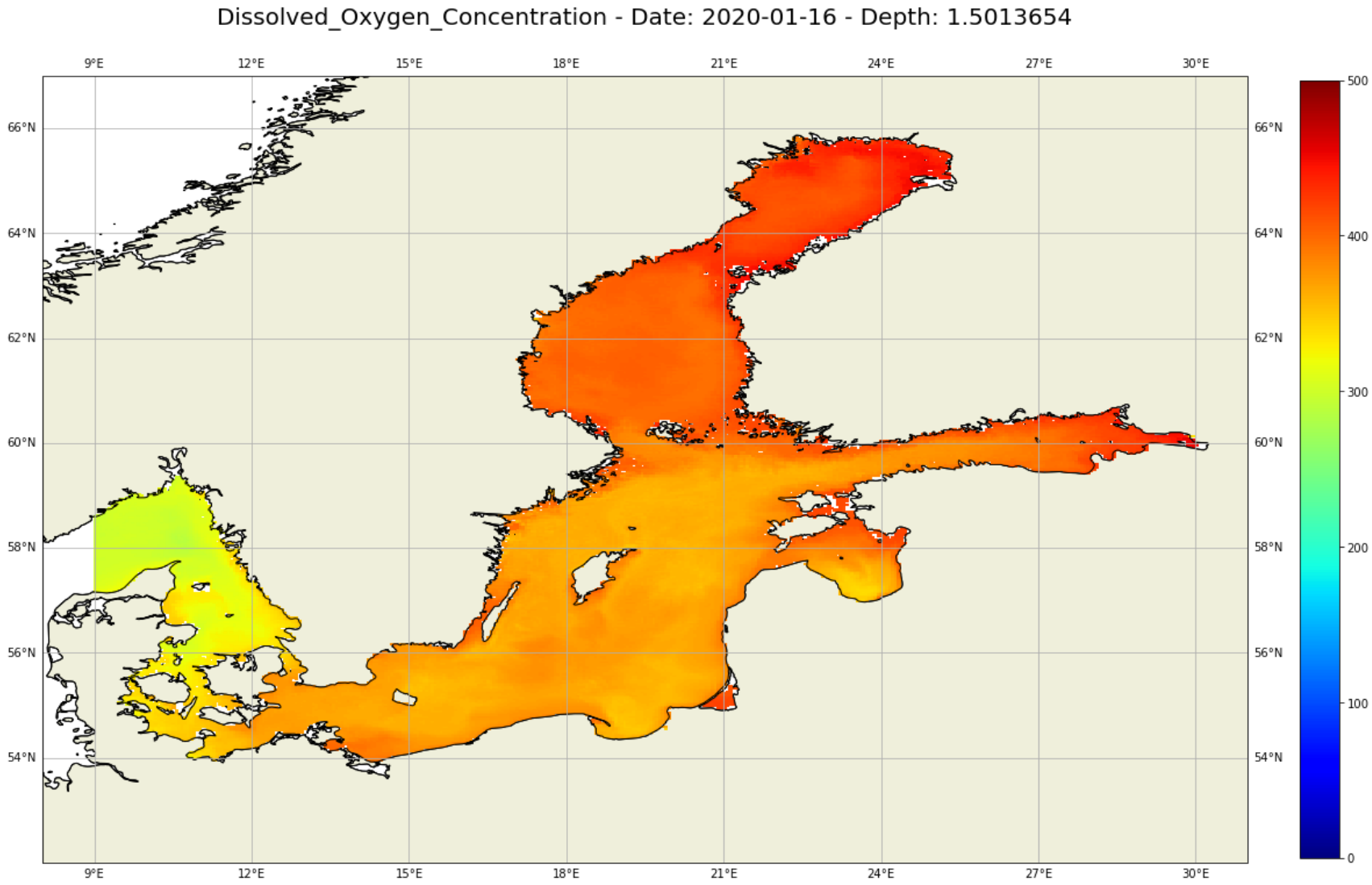
```
48
2020-01-16
50
2020-03-16
52
2020-05-16
54
2020-07-16
56
2020-09-16
58
```

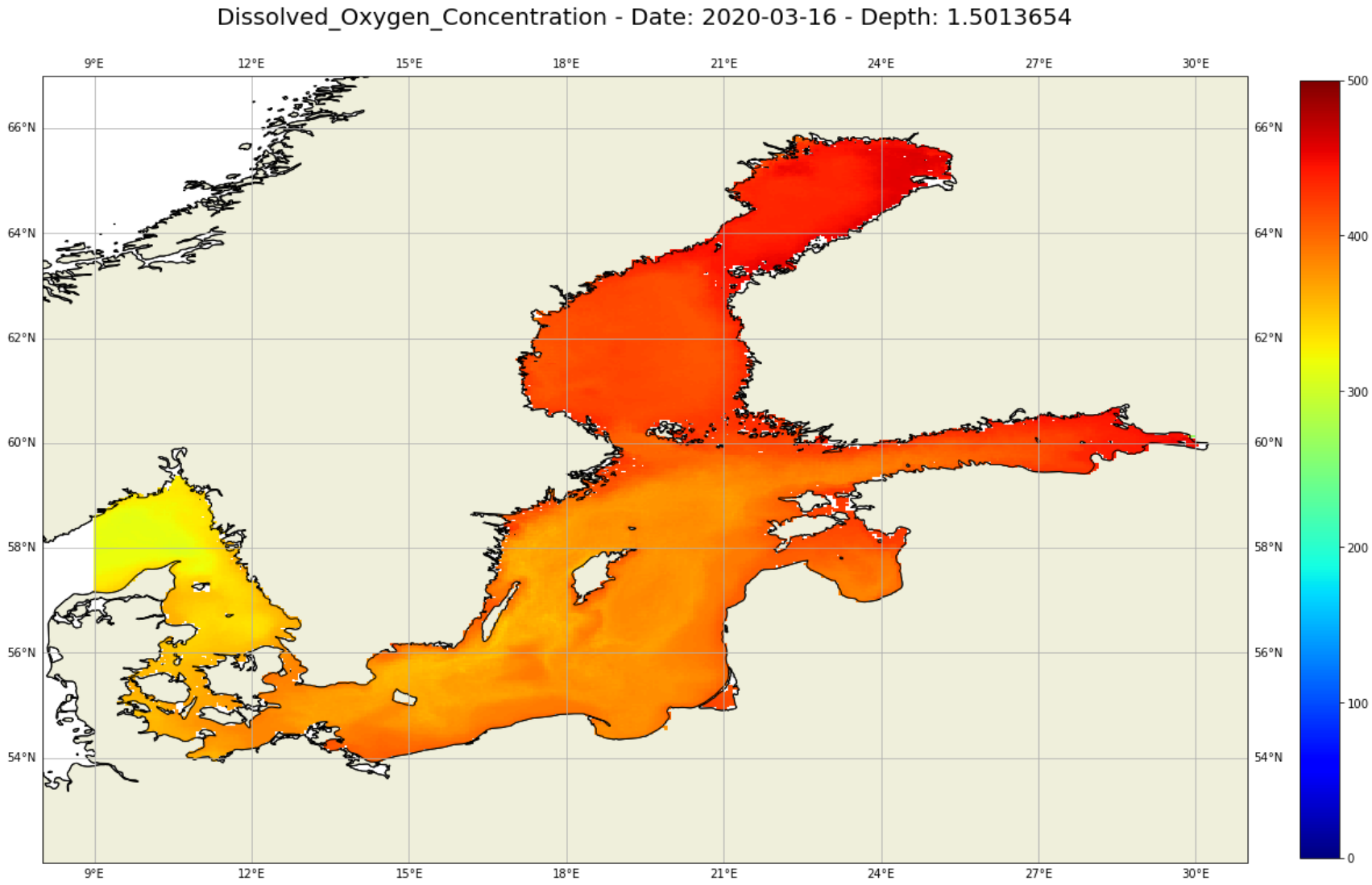
```
2020-11-16
```

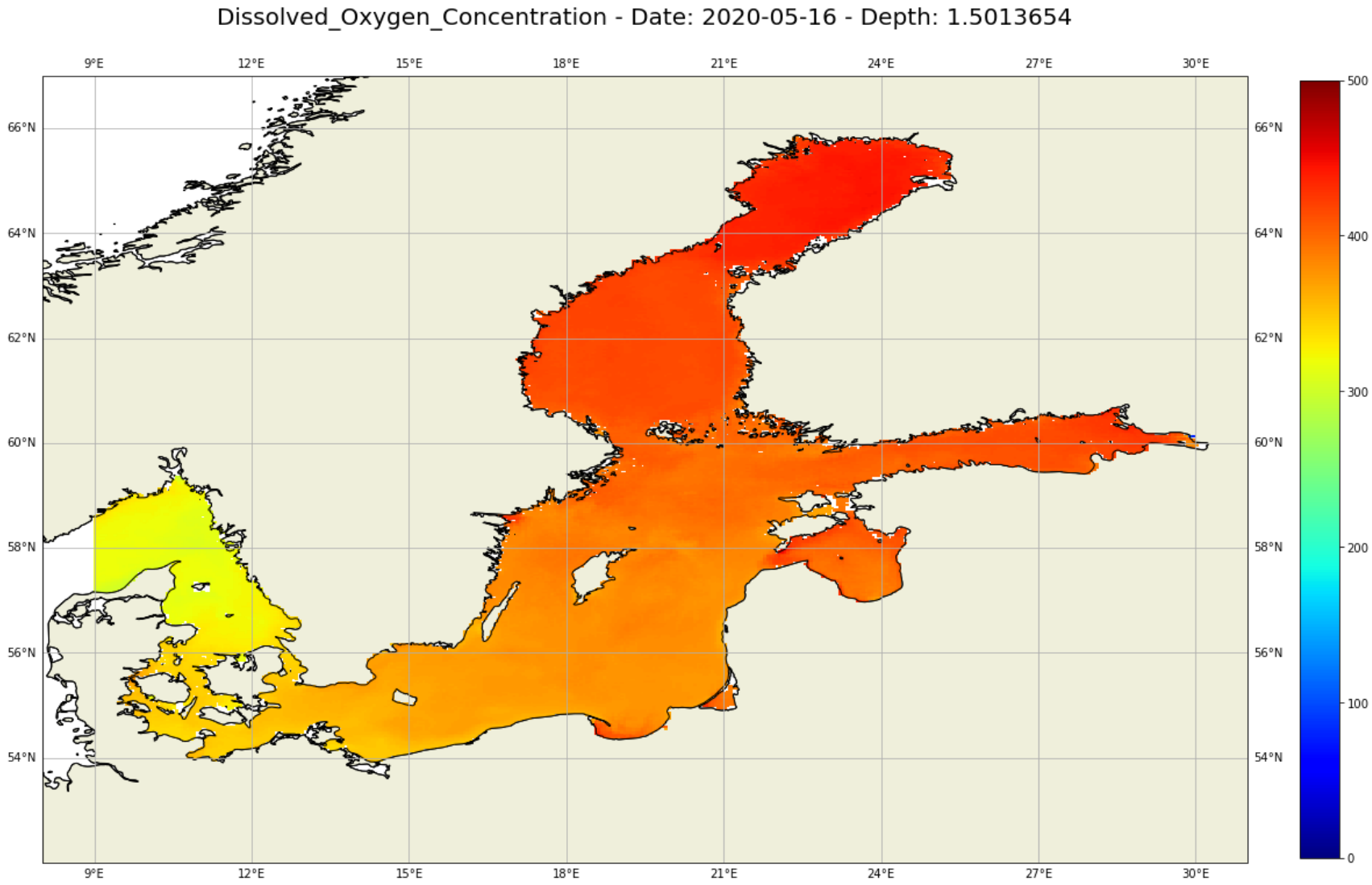
Out[26]:
```
6
```

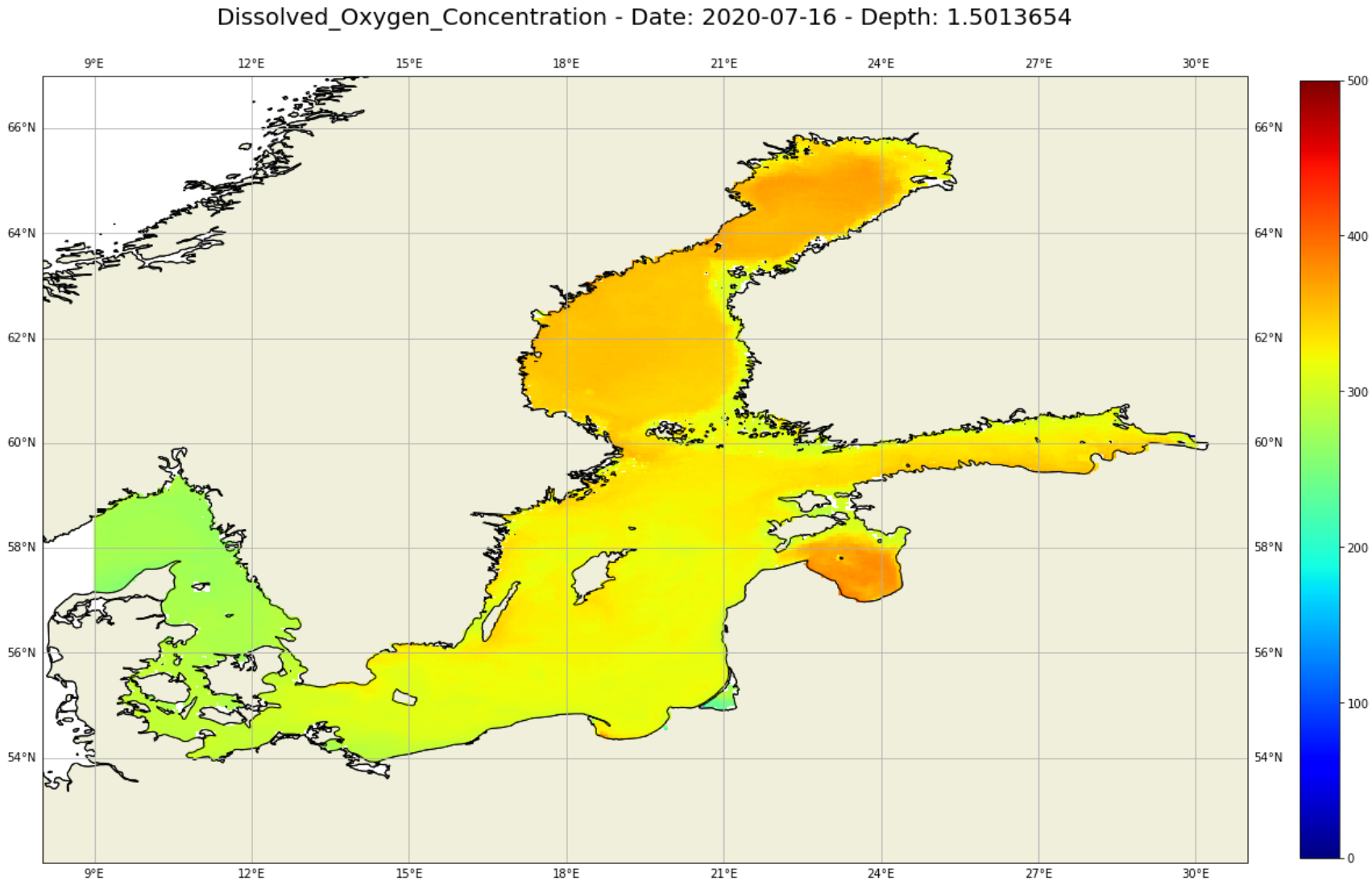
In [27]:
```
# Check if the folder './maps/o2_maps/date' exists. Otherwise it is created.
dir_exist('./maps/o2_maps/date')
```

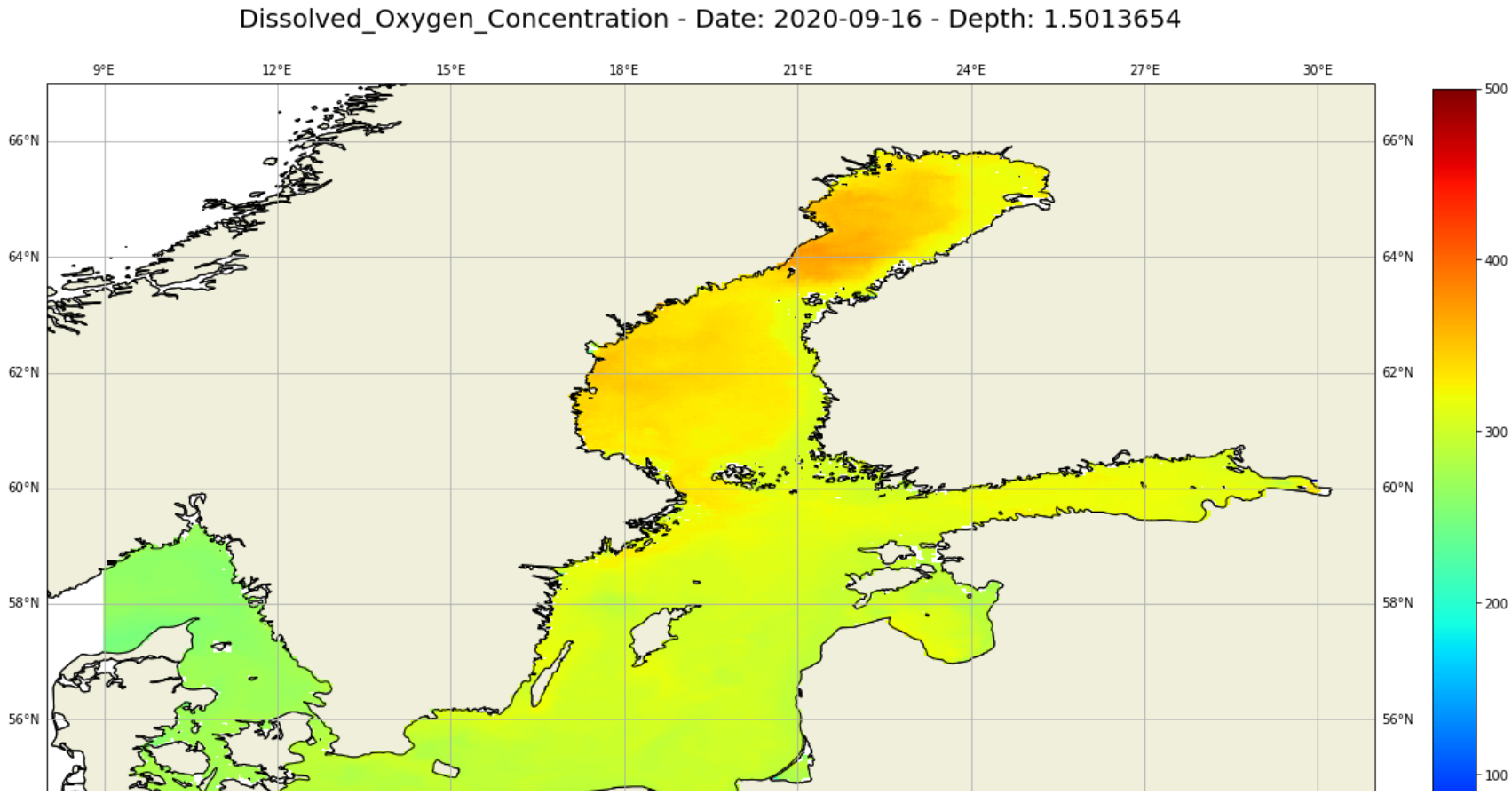
In [28]:
```
# Now we are generating the maps for Date_list_o2 and save the maps in the directory maps/o2_maps/date

for date in Date_list_o2 :
   diso2_map(date, depth_min_o2, 'maps/o2_maps/date')
```


Dissolved_Oxygen_Concentration - Date: 2020-01-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0


Dissolved_Oxygen_Concentration - Date: 2020-03-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0


Dissolved_Oxygen_Concentration - Date: 2020-05-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0

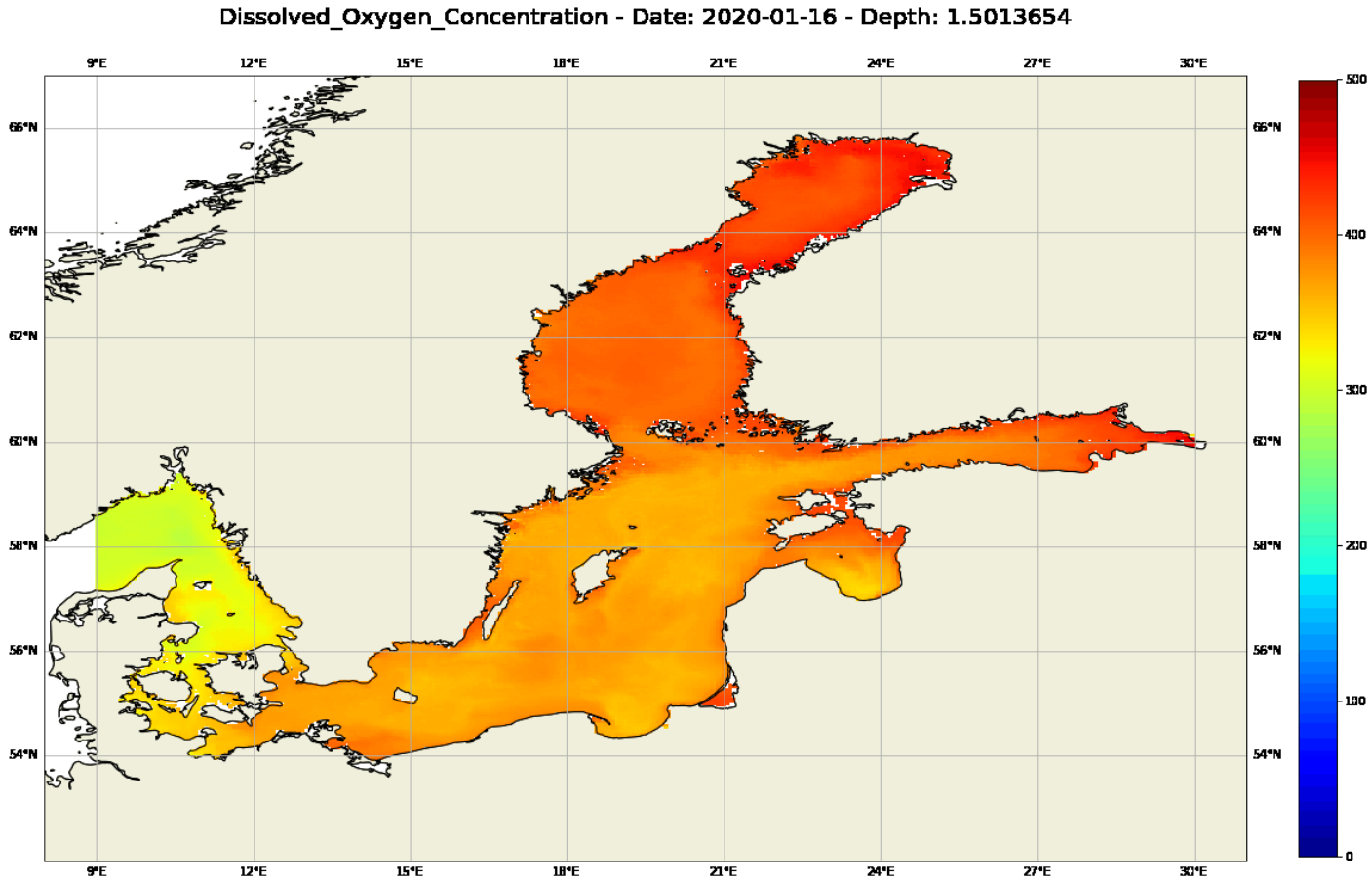

Dissolved_Oxygen_Concentration - Date: 2020-07-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0



These maps are now shown in an animated slide show.

In [29]:

```
# Import the modules for the animation

import imageio
from pathlib import Path
from IPython.display import Image

# Search the maps
image_path = Path('maps/o2_maps/date')
images = sorted(image_path.glob('o2_map_*.png'))
image_list = []
for file_name in images:
    image_list.append(imageio.imread(file_name))


# Save the animated gif-file
imageio.mimwrite('maps/o2_maps/date/o2_map.gif',image_list,duration=2.5,loop=1)

# Show the animated gif-file
with open('maps/o2_maps/date/o2_map.gif','rb') as f:
    display(Image(data=f.read(),format='png'))
```


Dissolved_Oxygen_Concentration - Date: 2020-01-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0

Exercise

Delete the maps in maps/o2_maps/date/.
Change the date list. For example choose every April and October of the years 2016-2020.
Plot the animated maps.

3.3 Dissolved Oxygen of the Baltic Sea for a serie of depths

Go back to the "Table of contents"

In this section we are plotting maps of the **Dissolved Oxygen** for a serie of depths.

In [30]:
```
# We ar selecting the depth array of the o2 dataset and print the length
depth_o2 = data_baltic_o2.depth
depth_len_o2 = len(depth_o2.depth)
depth_len_o2
```

Out[30]: 21

In [31]:
```
# At least 4 depths are needed, because in a following section we want to print 4 suplots
# Variable for the list of the selected depths

depth_list = []

# 1. Parameter: First index of interested depth. We start with the index 0.
# 2. Parameter: length of o2 depth array
# 3. Parameter: intervall of interested depths. We choose index steps 4.
# The selected depths with the indices are printed.

for i in range(0, depth_len_o2, 4):
    print(i)
    dv = depth_o2.depth.values[i]
    print(dv)
    depth_list.append(dv)

print(depth_list)
```

```
0
1.5013654
4
13.652688
8
```

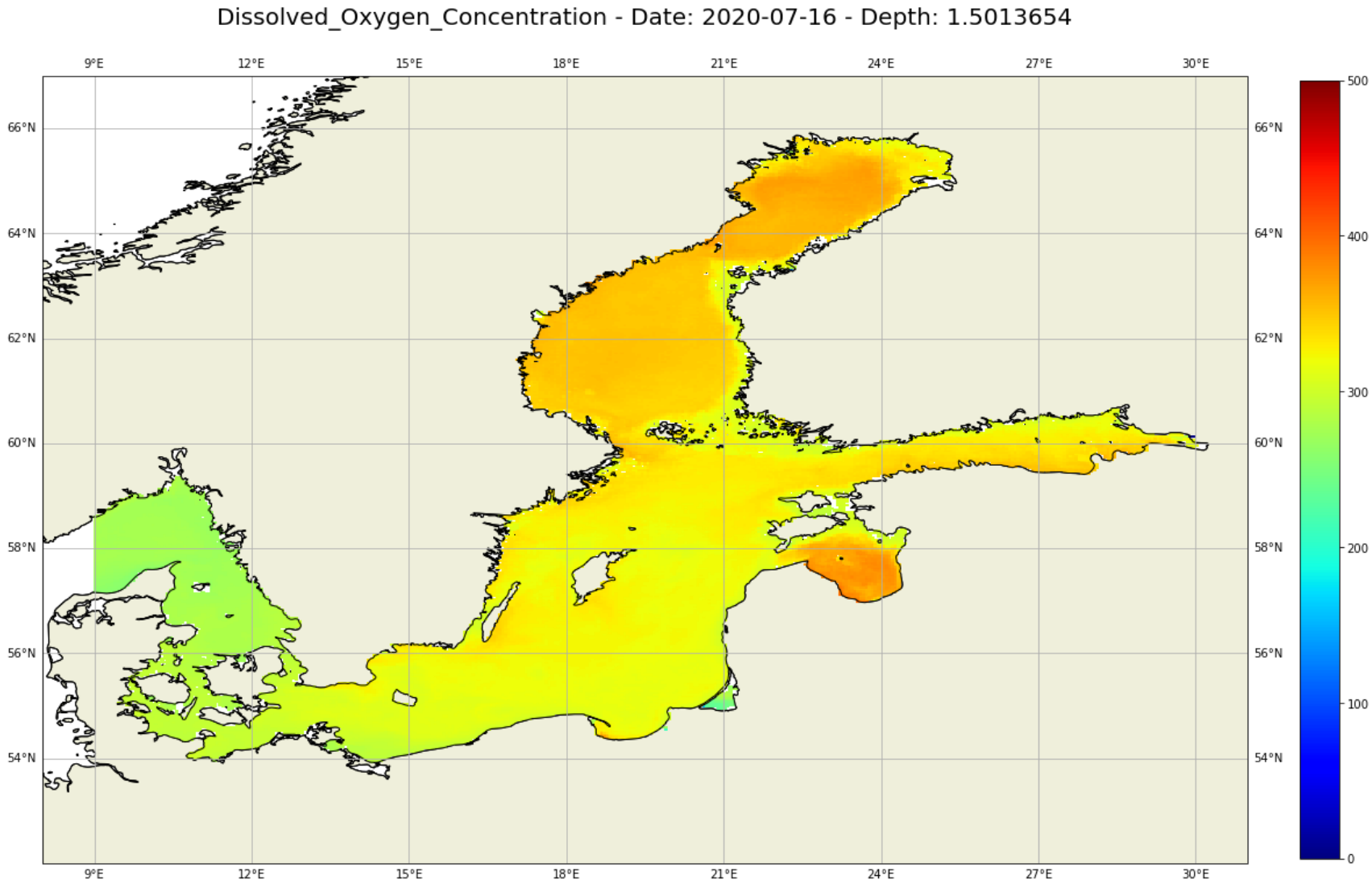
```
26.287226
12
40.070496
16
56.50011
20
78.612465
[1.5013654  13.652688   26.287226   40.070496   56.50011    78.612465]
```

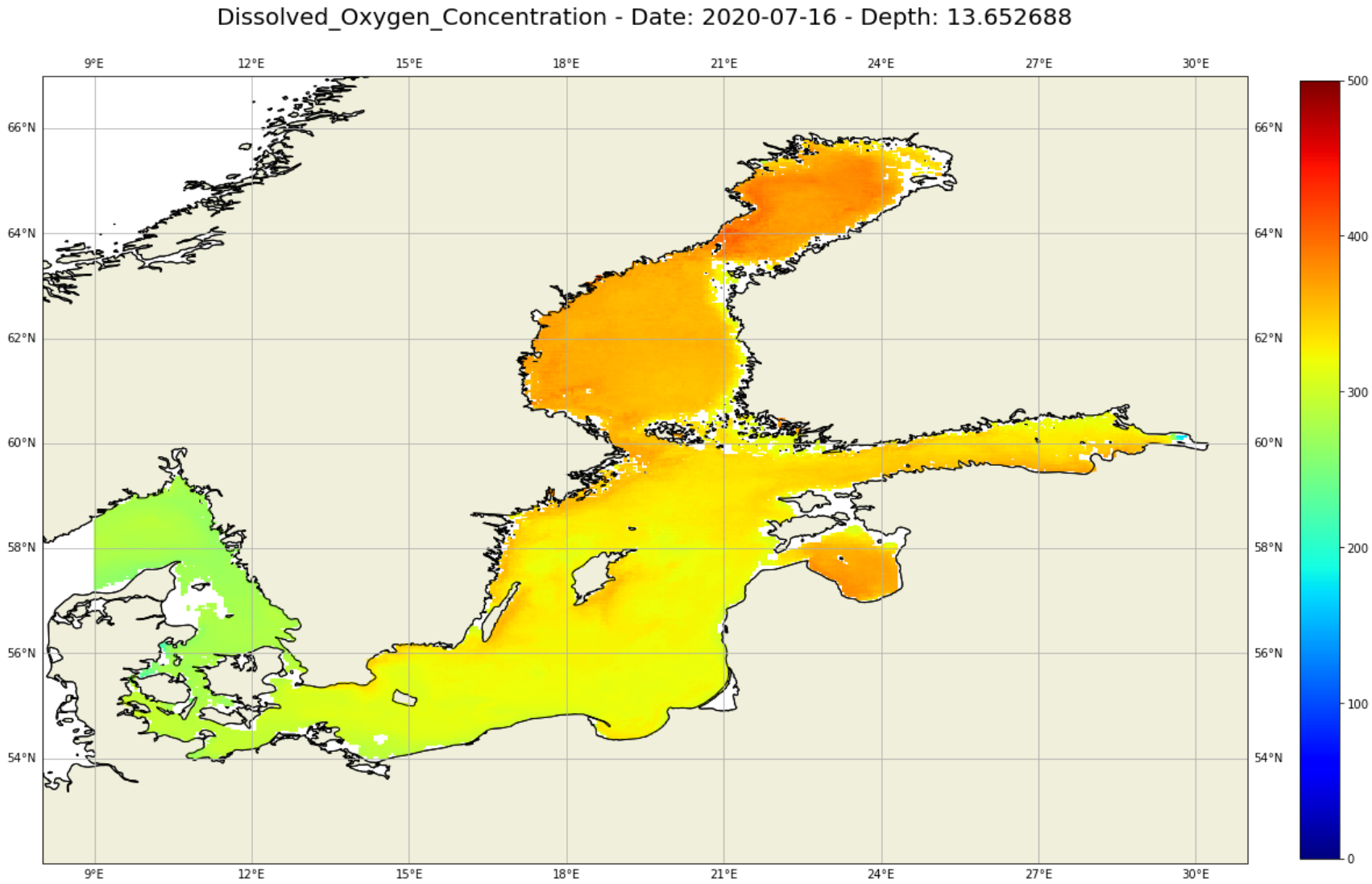
In [32]:

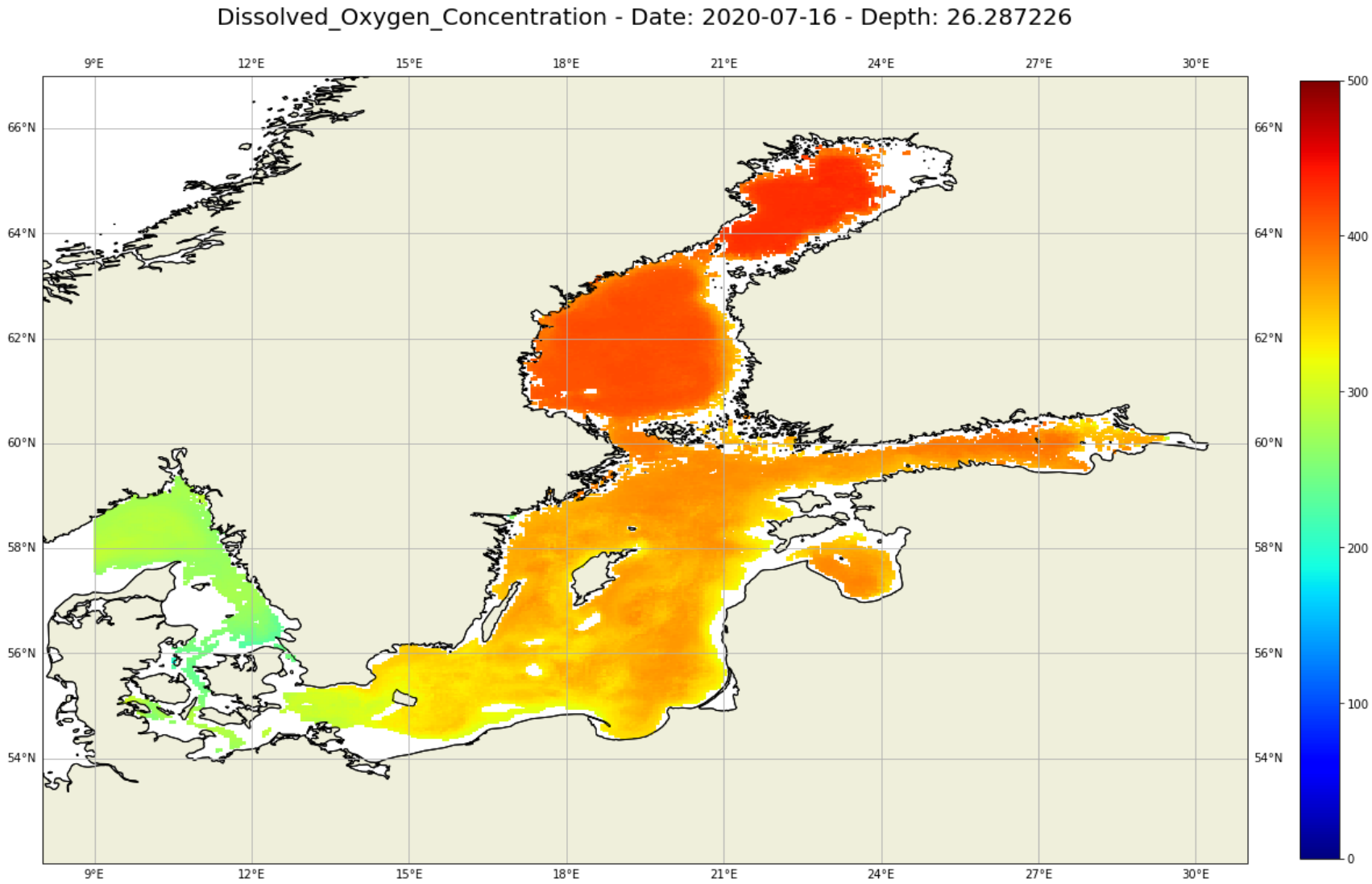
```
# Check if the folder './maps/o2_maps/depth' exists. Otherwise it is created.
dir_exist('./maps/o2_maps/depth')
```

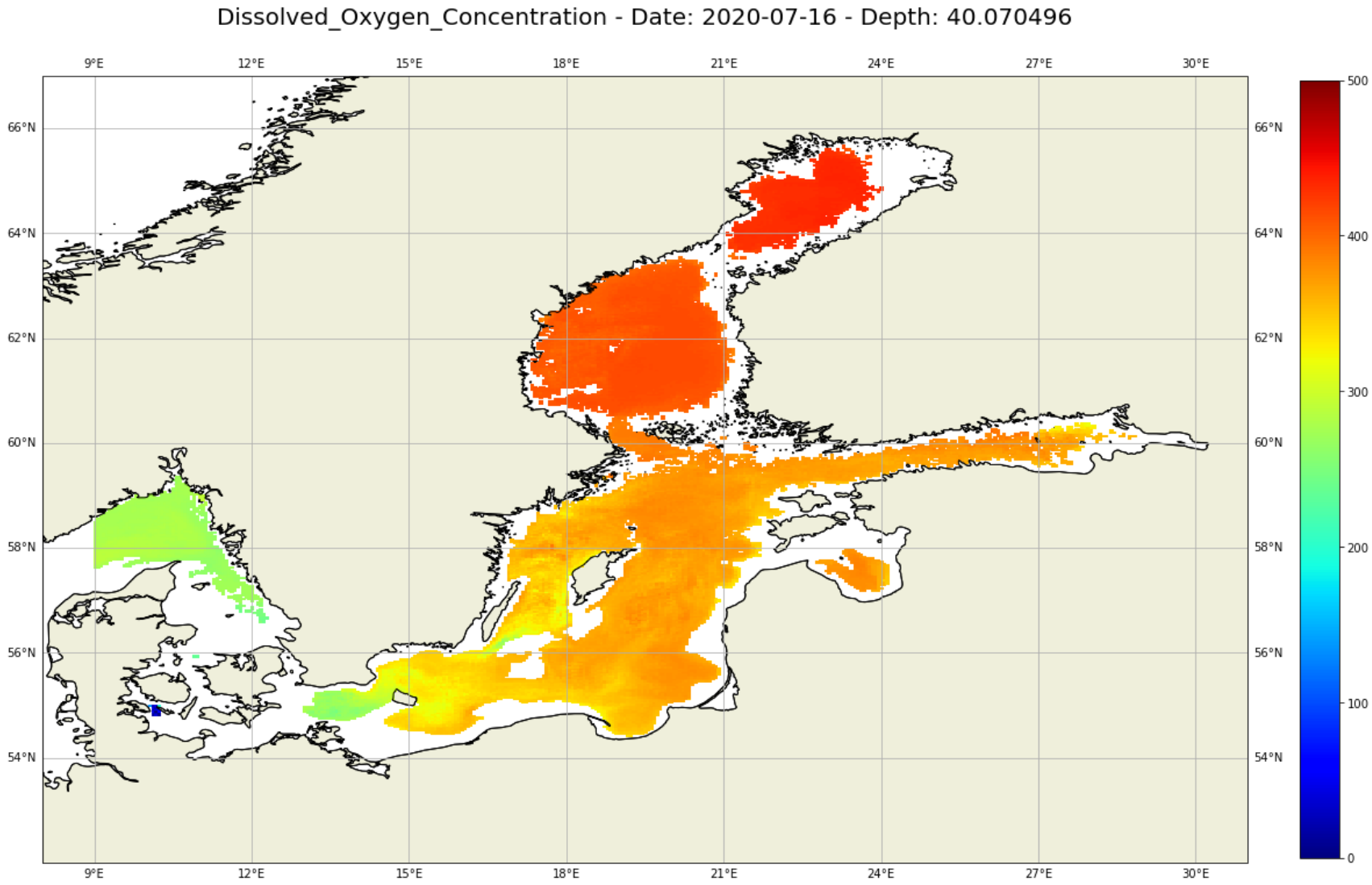
In [33]:

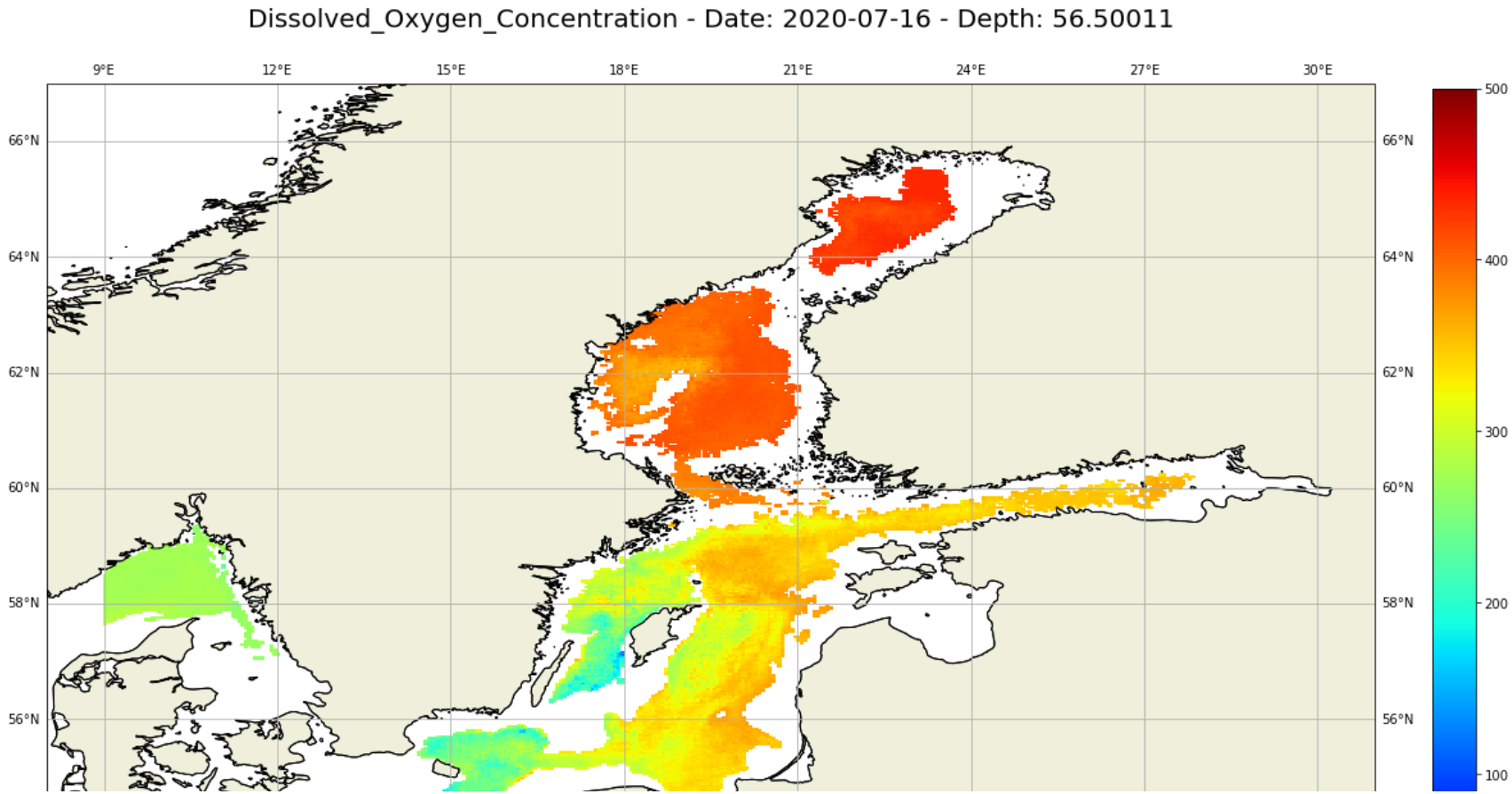
```
# We choose the date 16.07.2020
# Loop over the depth_list
# The maps are saved in maps/o2_maps/depth

for i in depth_list :
    diso2_map('2020-07-16', i, 'maps/o2_maps/depth')
```


Dissolved_Oxygen_Concentration - Date: 2020-07-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0


Dissolved_Oxygen_Concentration - Date: 2020-07-16 - Depth: 13.652688
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0


Dissolved_Oxygen_Concentration - Date: 2020-07-16 - Depth: 26.287226
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0

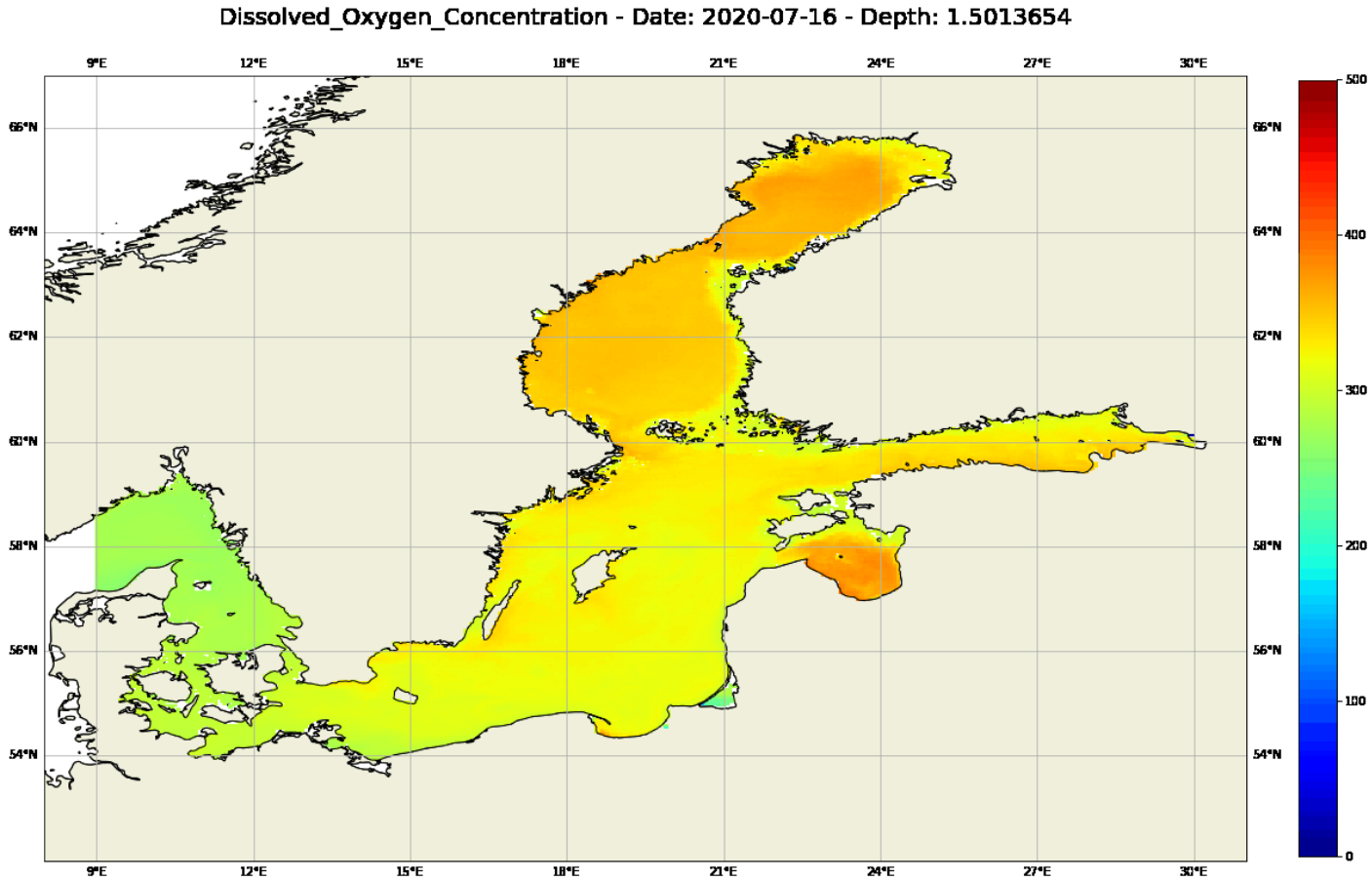

Dissolved_Oxygen_Concentration - Date: 2020-07-16 - Depth: 40.070496
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0



These maps are now shown in an animated slide show.

In [34]:

```
# Import the modules for the animation

import imageio
from pathlib import Path

# Search the maps
image_path = Path('maps/o2_maps/depth')
images = sorted(image_path.glob('o2_map_*.png'))
image_list = []
for file_name in images:
    image_list.append(imageio.imread(file_name))


# Save the animated gif-file
imageio.mimwrite('maps/o2_maps/depth/o2_map.gif',image_list,duration=2.5,loop=1)

# Show the animated gif-file
with open('maps/o2_maps/depth/o2_map.gif','rb') as f:
    display(Image(data=f.read(),format='png'))
```


Dissolved_Oxygen_Concentration - Date: 2020-07-16 - Depth: 1.5013654
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
500
400
300
200
100
0

Exercise

> Delete the maps in maps/o2_maps/depth/.
> Choose other dates, for example 16.04.2020 and 16.09.2020.
> Plot the animated maps.

4. Chlorophyll

Go back to the "Table of contents"

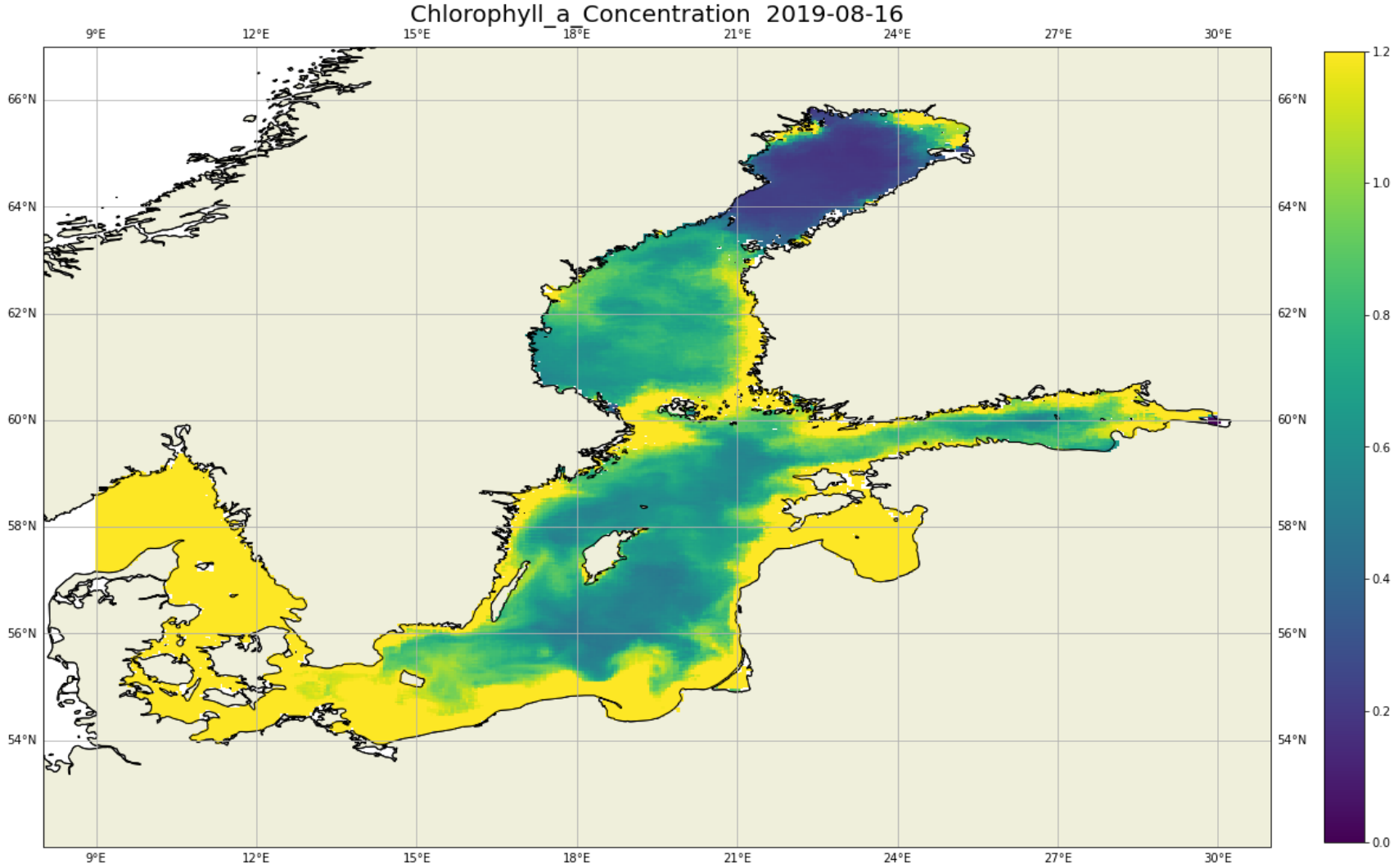
In this section we explore the **Chlorophyll a concentration** of the Baltic Sea.
We are plotting the Chlorophyll for one date and for a serie of dates.

4.1 Chlorophyll of the Baltic Sea for one date

In [35]:
```
# Definition of a function

# We are defining a function for plotting a map of Chlorophyll at the minimum depth
# Input parameter: date, map_dir (directory to save the map), long_lat_in (longitude and latitude)

def chl_map(date, map_dir, long_lat_in):

    # Select the chl-data with the input parameter date
    chl_map = chl_data['chl'].sel(time=date, method = 'nearest').squeeze()

    # Define the settings of the plot
    f = plt.figure(figsize=(20, 15))                                        # define the size of the plot
    ax = plt.axes(projection=ccrs.PlateCarree())                            # create an ax and select the p
    ax.coastlines()                                                         # add coastlines
    gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True)             # add map gridlines
    ax.add_feature(cfeature.LAND, zorder=1, edgecolor='k')                  # add land mask
    ax.set_extent(long_lat_in,crs=ccrs.PlateCarree())                       # define the extent of the map

    # Plot the chl-data, set the minimum and maximum values of the colorbar and the colormap to use
    im = ax.pcolor(chl_map['longitude'].data, chl_map['latitude'].data,chl_map,vmin=0,vmax=1.2,cmap='viridis')

    # Add a title
    Title = chl_name + '  {}'.format(date)
    plt.title(Title, fontsize=20)

    # Add color scale
    f.colorbar(im,ax=ax,fraction=0.03, pad=0.04)

    # Save the figure in the directory map_dir
    plt.savefig(map_dir + '/chl_map_' + date)
```

In [36]:
```
# Check if the folder './maps/chl_maps' exists. Otherwise it is created.
dir_exist('./maps/chl_maps')
```

In [37]:
```
# Call the Funktion chl_map with a date
# The map is saved in the folder maps/chl_maps
chl_map('2019-08-16', 'maps/chl_maps', long_lat)
```



Exercise

Call the funktion chl_map with another date.

4.2 Chlorophyll of the Baltic Sea for a serie of dates

In this section we are plotting maps of a serie of dates of **Chlorophyll**. We are selecting every second month of the year 2020.

In [38]:
```
# Length of the date array of the chl-data
time_len_chl = len(chl_data.time)
time_len_chl
```

Out[38]: 60

In [39]:
```
# Variable for the list of the selected dates
Date_list = []

# 1. Parameter : First index of interested date. We start with the index 48, that is the date 16.01.2020.
# 2. Parameter : length of the date array
# 3. Parameter: intervall of interested dates. We choose index steps 2.
# The selected dates with the indices are printed.

for i in range(48, time_len_chl, 2):
    # print(i)
    dt = chl_data.time[i].values
    dt_str = str(dt)[:10]
    print(dt_str)
    Date_list.append(dt_str)

print(Date_list)

# Print the number of selected dates
len(Date_list)
```

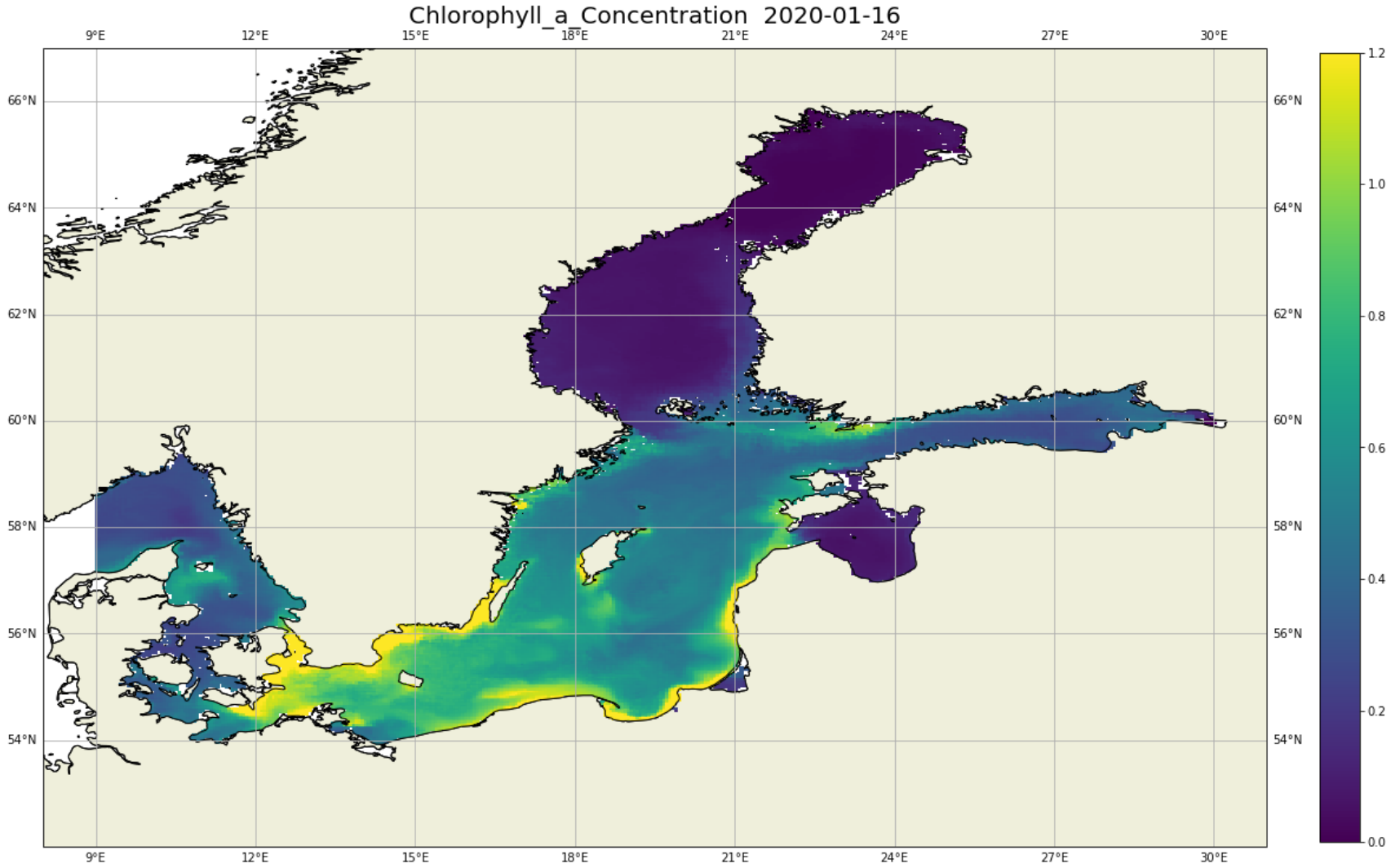
```
2020-01-16
2020-03-16
2020-05-16
2020-07-16
2020-09-16
2020-11-16
['2020-01-16', '2020-03-16', '2020-05-16', '2020-07-16', '2020-09-16', '2020-11-16']
```

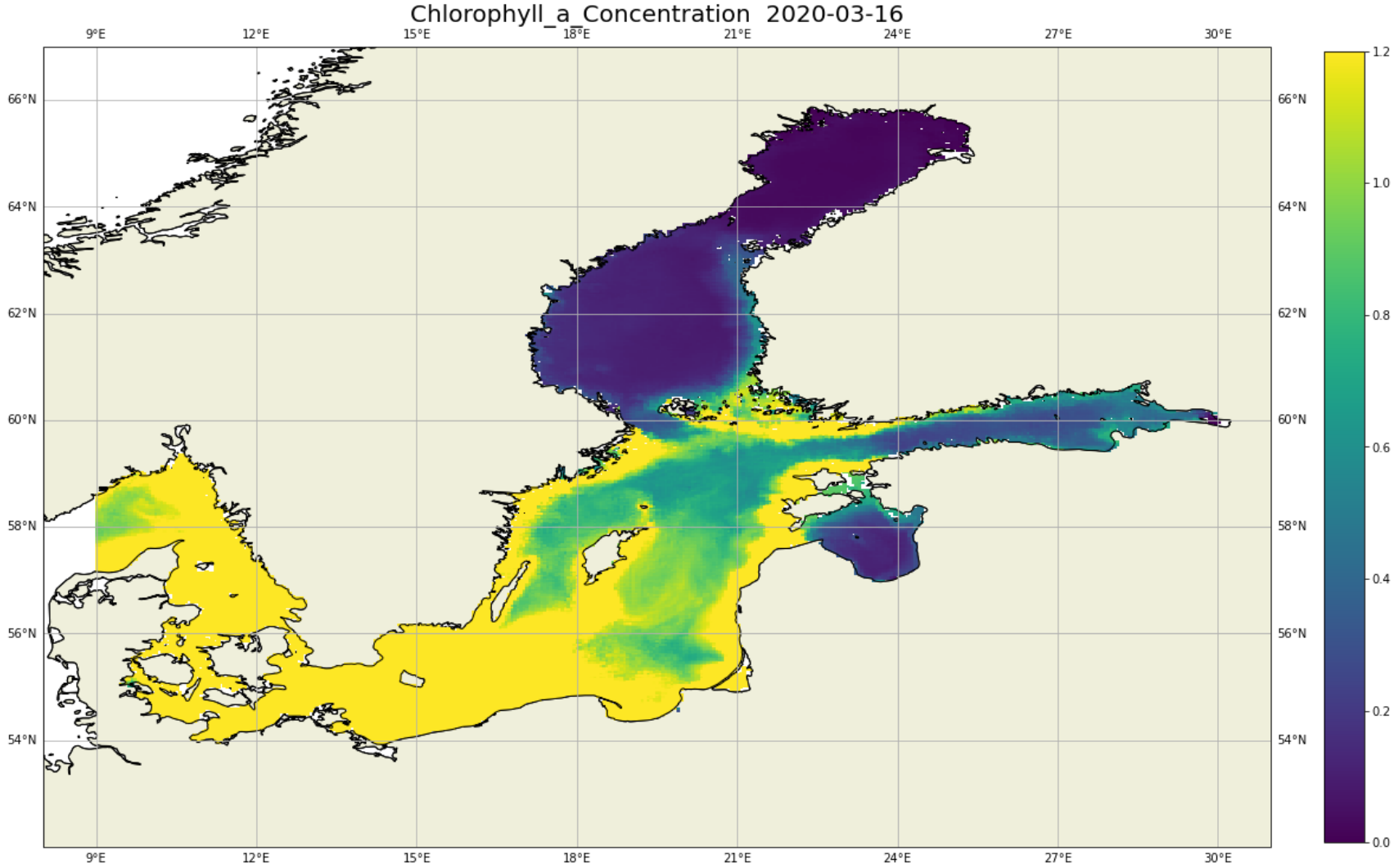
Out[39]: 6

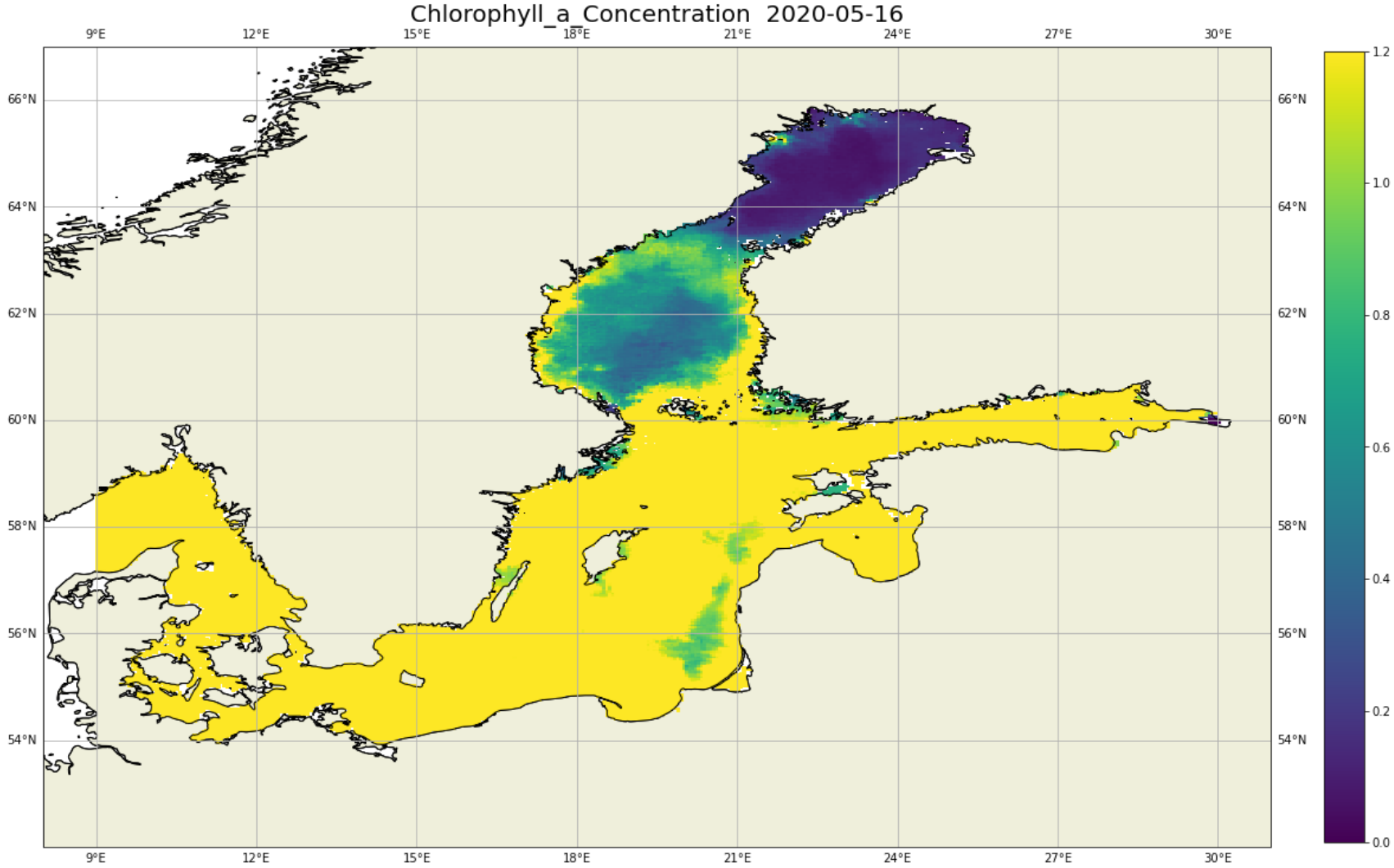
In [40]:
```
# Check if the folder './maps/chl_maps/date' exists. Otherwise it is created.
dir_exist('./maps/chl_maps/date')
```

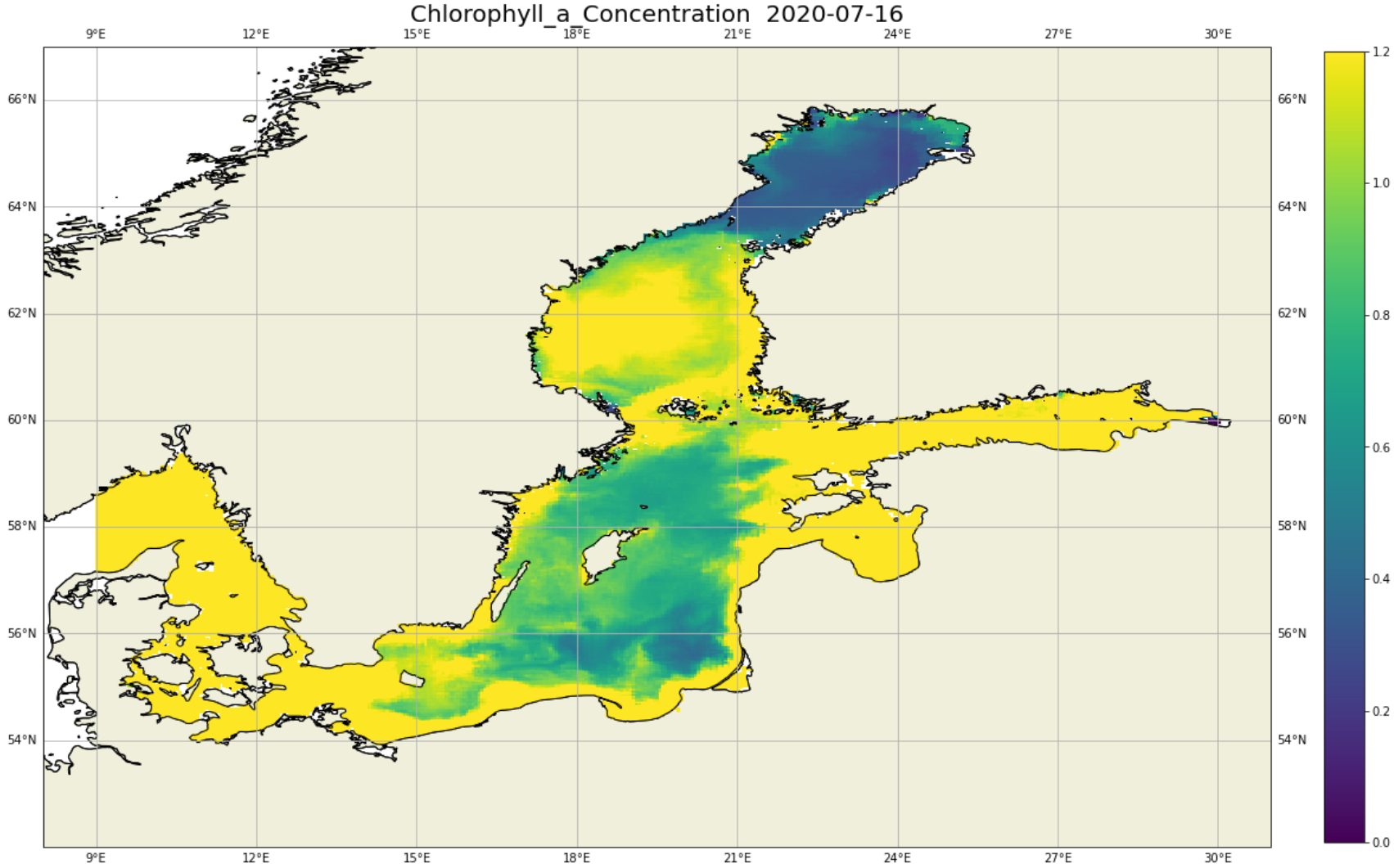
In [41]:

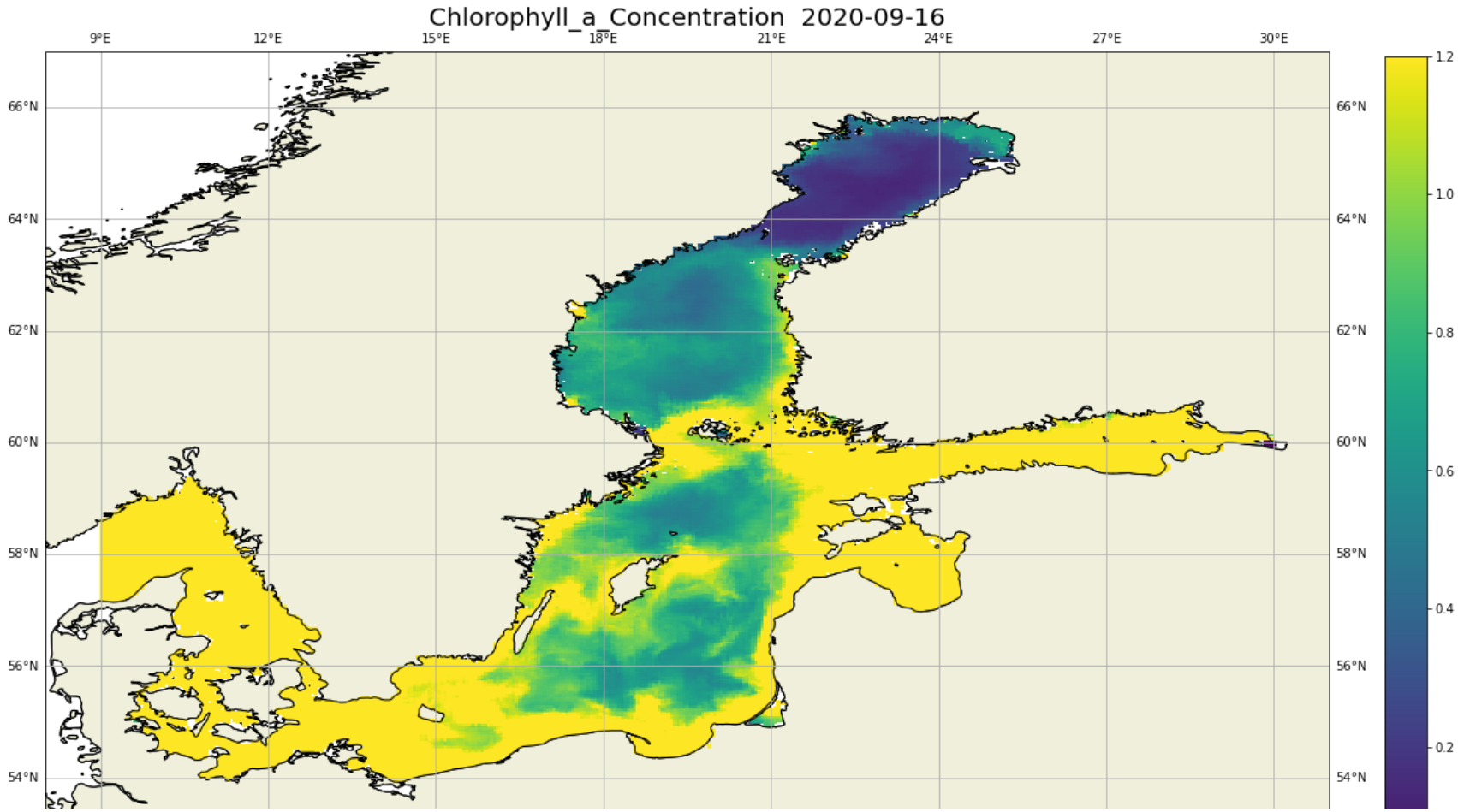
```
# Now we are generating the maps for the Date_list
# The maps are saved in the folder maps/chl_maps/date

for date in Date_list :
    chl_map(date, 'maps/chl_maps/date', long_lat)
```




Chlorophyll_a_Concentration 2020-03-16
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
1.2
1.0
0.8
0.6
0.4
0.2
0.0


Chlorophyll_a_Concentration 2020-05-16
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
1.2
1.0
0.8
0.6
0.4
0.2
0.0


Chlorophyll_a_Concentration 2020-07-16
9°E
12°E
15°E
18°E
21°E
24°E
27°E
30°E
66°N
64°N
62°N
60°N
58°N
56°N
54°N
1.2
1.0
0.8
0.6
0.4
0.2
0.0



Exercise

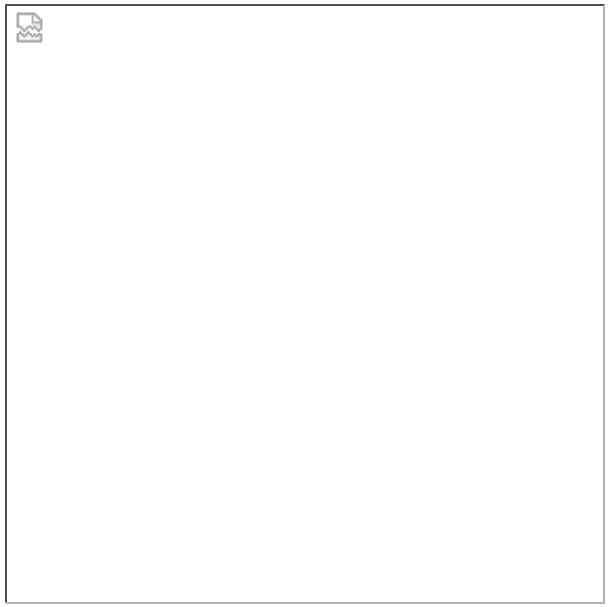
Delete the maps in maps/chl_maps/date/.

Change the date list. For example choose every April and October of the years 2016-2020.

Plot the maps.

5. Chlorophyll-Oxygen-Relation at the coast from the Bay of Kiel to the Bay of Pomeranian

Now we zoom on the coast of the Baltic Sea from the Bay of Kiel to the Bay of Pomeranian:



We are plotting maps to show the relation between the **Chlorophyll a** and the **Dissolved Oxygen** in a serie of depths.

In [42]:
```
# The longitude and latitude of the region to explore.
long_lat_reg = [8, 15, 53, 56]
```

In [43]:
```
# Show the depth_list
depth_list
```

Out[43]: [1.5013654, 13.652688, 26.287226, 40.070496, 56.50011, 78.612465]

In [44]:
```
# Check if the folder './maps/chl_o2_maps' exists. Otherwise it is created.
dir_exist('./maps/chl_o2_maps')
```

In [46]:

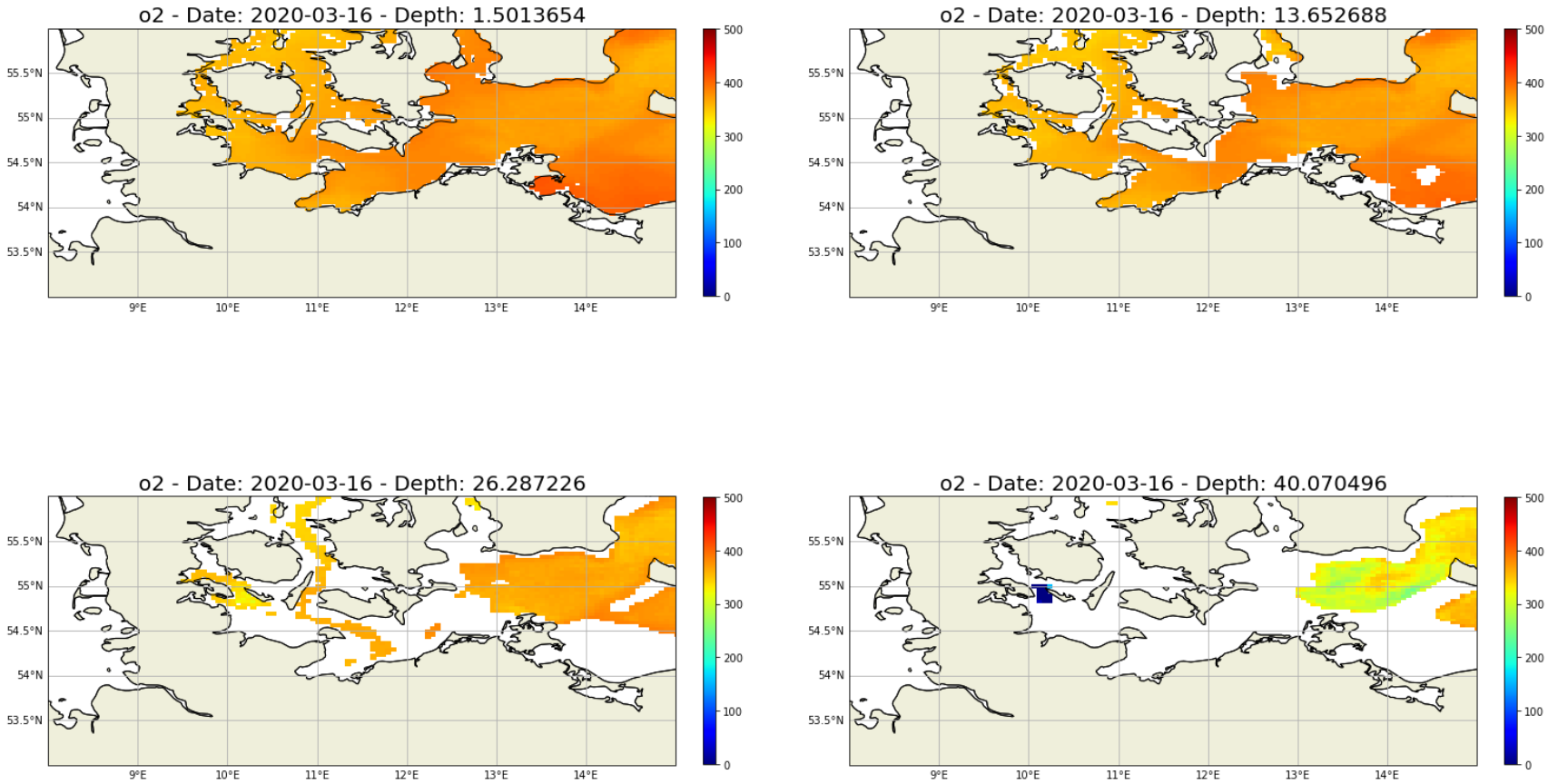
```
# Definition of a function

# This function is generating 4 suplots of Dissolved Oxygen on the first 4 depths selected above in the variable depth
# The parameter is a date.

def o2_4subp(date):

    # Generating 4 subplots
    f, axs = plt.subplots(2, 2, figsize = (25,15), subplot_kw=dict(projection=ccrs.PlateCarree()))

    # First depth
    # Select the o2-data of the first depth and the date
    data_o2_depth = data_baltic_o2.sel(depth=depth_list[0], method='nearest')
    o2_map = data_o2_depth['o2'].sel(time=date, method = 'nearest').squeeze()

    # Define the settings of the first subplot
    axs[0,0].coastlines()                                                    # add coastlines
    gl = axs[0,0].gridlines(crs=ccrs.PlateCarree(), draw_labels=True)        # add map gridlines
    gl.right_labels = False                                                  # remove labels on the right
    gl.top_labels = False                                                    # remove labels on the top
    axs[0,0].add_feature(cfeature.LAND, zorder=1, edgecolor='k')             # add land mask
    axs[0,0].set_extent(long_lat_reg,crs=ccrs.PlateCarree())                 # define the extent of the map

    # Plot the o2-data, set the minimum and maximum values of the colorbar and the colormap to use
    im1 = axs[0,0].pcolor(o2_map['longitude'].data, o2_map['latitude'].data,o2_map,vmin=0,vmax=500,cmap='jet')

    # Add color scale
    f.colorbar(im1,ax=axs[0,0],fraction=0.02, pad=0.04)

    # Add a title
    axs[0,0].set_title( 'o2 - Date: {}'.format(date) + ' - Depth: ' + str(depth_list[0]),fontsize=20)


    # Second depth
    # Select the o2-data of the second depth and the date
    data_o2_depth = data_baltic_o2.sel(depth=depth_list[1], method='nearest')
    o2_map = data_o2_depth['o2'].sel(time=date, method = 'nearest').squeeze()

    # Define the settings of the second subplot
    axs[0,1].coastlines()                                                    # add coastlines
    gl2 = axs[0,1].gridlines(crs=ccrs.PlateCarree(), draw_labels=True)       # add map gridlines
    gl2.right_labels = False                                                 # remove labels on the right
    gl2.top_labels = False                                                   # remove labels on the top
```

```
axs[0,1].add_feature(cfeature.LAND, zorder=1, edgecolor='k')                    # add land mask
axs[0,1].set_extent(long_lat_reg,crs=ccrs.PlateCarree())                        # define the extent of the map

# Plot the o2-data, set the minimum and maximum values of the colorbar and the colormap to use
im2 = axs[0,1].pcolor(o2_map['longitude'].data, o2_map['latitude'].data,o2_map,vmin=0,vmax=500,cmap='jet')

# Add color scale
f.colorbar(im2,ax=axs[0,1],fraction=0.02, pad=0.04)

# Add a title
axs[0,1].set_title( 'o2 - Date: {}'.format(date) + ' - Depth: ' + str(depth_list[1]),fontsize=20)


# Third depth
# Select the o2-data of the third depth and the date
data_o2_depth = data_baltic_o2.sel(depth=depth_list[2], method='nearest')
o2_map = data_o2_depth['o2'].sel(time=date, method = 'nearest').squeeze()

# Define the settings of the third subplot
axs[1,0].coastlines()                                                           # add coastlines
gl2 = axs[1,0].gridlines(crs=ccrs.PlateCarree(), draw_labels=True)              # add map gridlines
gl2.right_labels = False                                                        # remove labels on the right
gl2.top_labels = False                                                          # remove labels on the top
axs[1,0].add_feature(cfeature.LAND, zorder=1, edgecolor='k')                    # add land mask
axs[1,0].set_extent(long_lat_reg,crs=ccrs.PlateCarree())                        # define the extent of the map

# Plot the o2-data, set the minimum and maximum values of the colorbar and the colormap to use
im3 = axs[1,0].pcolor(o2_map['longitude'].data, o2_map['latitude'].data,o2_map,vmin=0,vmax=500,cmap='jet')

# Add color scale
f.colorbar(im3,ax=axs[1,0],fraction=0.02, pad=0.04)

# Add a title
axs[1,0].set_title( 'o2 - Date: {}'.format(date) + ' - Depth: ' + str(depth_list[2]),fontsize=20)


# Fourth depth
# Select the o2-data of the fourth depth and the date
data_o2_depth = data_baltic_o2.sel(depth=depth_list[3], method='nearest')
o2_map = data_o2_depth['o2'].sel(time=date, method = 'nearest').squeeze()

# Define the settings of the fourth subplot
axs[1,1].coastlines()                                                           # add coastlines
gl2 = axs[1,1].gridlines(crs=ccrs.PlateCarree(), draw_labels=True)              # add map gridlines
gl2.right_labels = False                                                        # remove labels on the right
```

```
gl2.right_labels = False                                    # remove labels on the right
gl2.top_labels = False                                      # remove labels on the top
axs[1,1].add_feature(cfeature.LAND, zorder=1, edgecolor='k') # add land mask
axs[1,1].set_extent(long_lat_reg,crs=ccrs.PlateCarree())    # define the extent of the map

# Plot the o2-data, set the minimum and maximum values of the colorbar and the colormap to use
im4 = axs[1,1].pcolor(o2_map['longitude'].data, o2_map['latitude'].data,o2_map,vmin=0,vmax=500,cmap='jet')

# Add color scale
f.colorbar(im4,ax=axs[1,1],fraction=0.02, pad=0.04)

# Add a title
axs[1,1].set_title( 'o2 - Date: {}'.format(date) + ' - Depth: ' + str(depth_list[3]),fontsize=20)

# Save the image of the subplots
plt.savefig('maps/chl_o2_maps/' + 'o2_map_' + date)

# Show the subplots
plt.show()
```

In [48]:

```
# Generate the supplots for the 16.03.2020
# The map is saved in the directory maps/chl_o2_maps
o2_4subp('2020-03-16')
```



Exercise

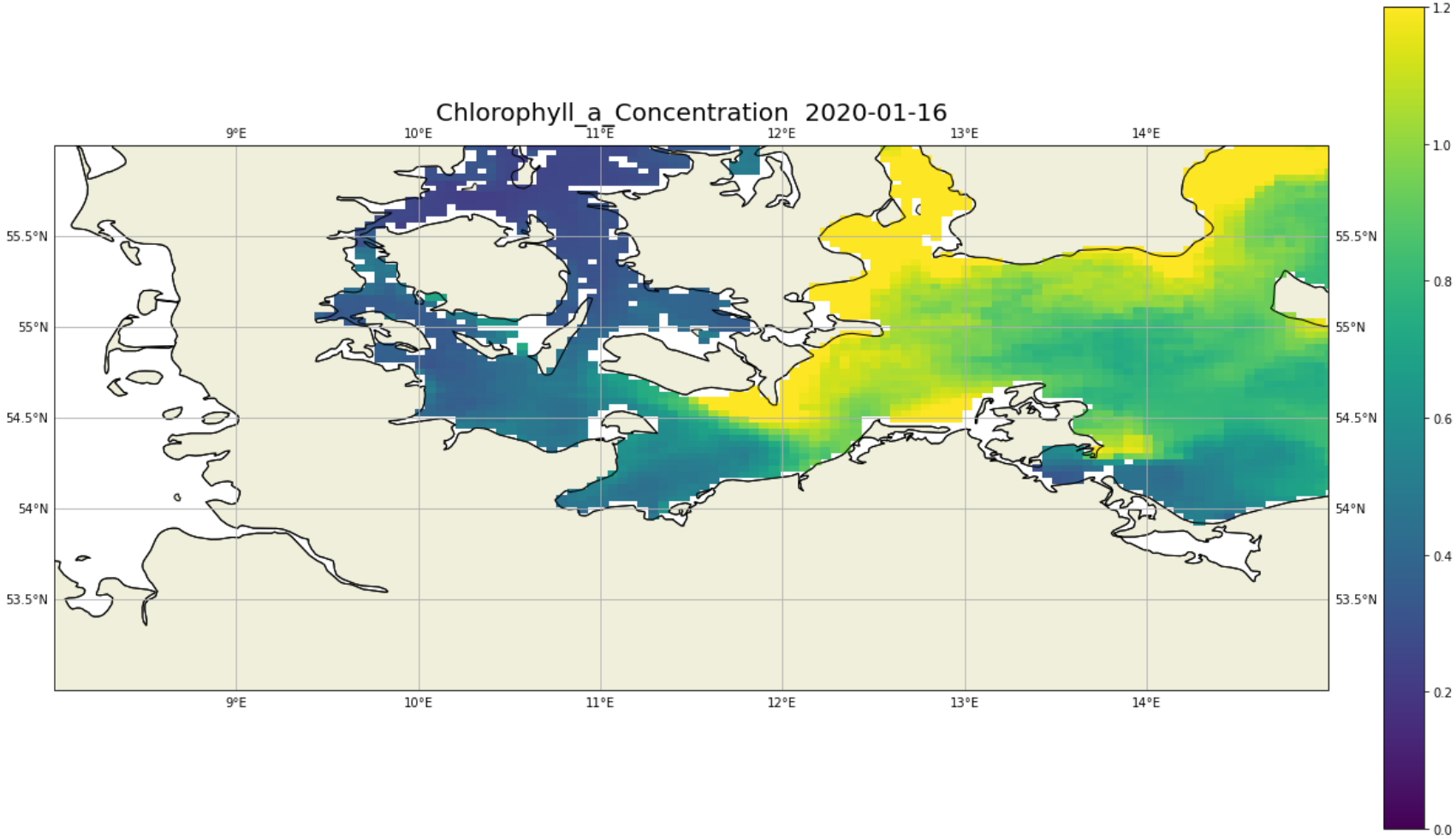
Call the funktion o2_4subp with another date.

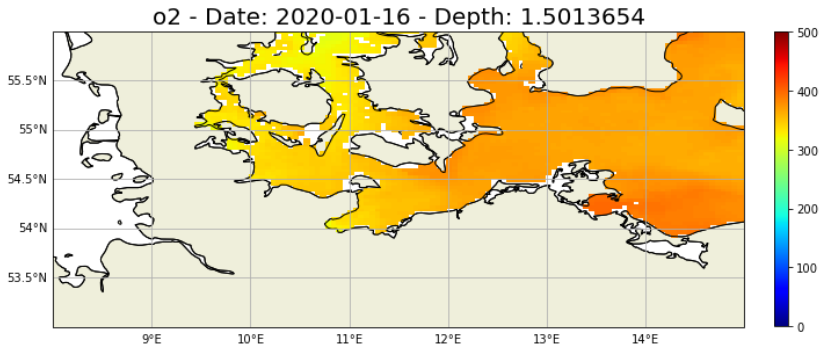
In the next step the notebook is generating the relation between the **Chlorophyll** and the **Dissolved Oxygen** for 4 depths over a serie of dates.

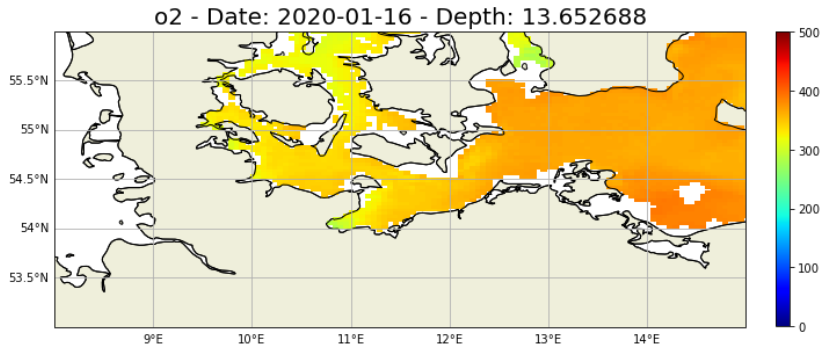
In [49]:
```
# Check again the date and depth lists
print('Date List: ', Date_list)
print('Depth List: ',depth_list)
```

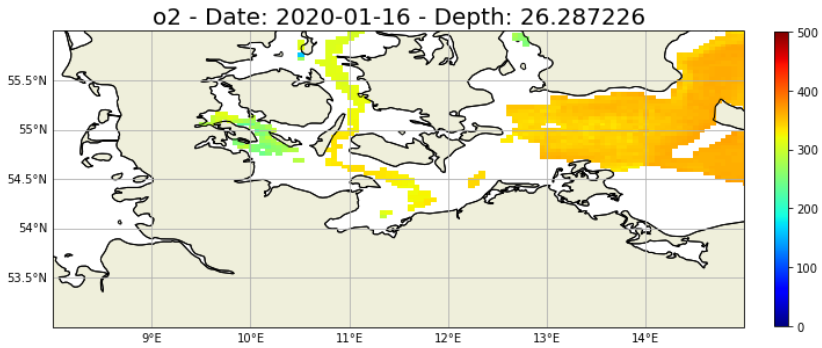
```
Date List:  ['2020-01-16', '2020-03-16', '2020-05-16', '2020-07-16', '2020-09-16', '2020-11-16']
Depth List:  [1.5013654, 13.652688, 26.287226, 40.070496, 56.50011, 78.612465]
```

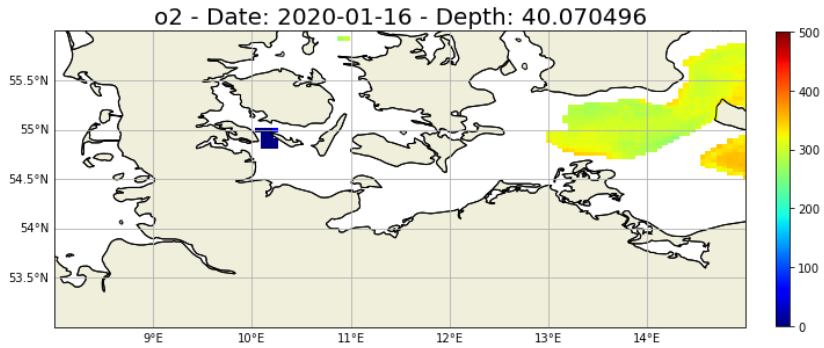
In [50]:
```
# Generate maps for the Chlorophyll concentration and the Oxygen concentration of the first 4 depths

for date in Date_list :
    # Call the function to generate the chl maps
    chl_map(date, 'maps/chl_o2_maps', long_lat_reg)
    # Call the function to generate the o2 subplots
    o2_4subp(date)
```

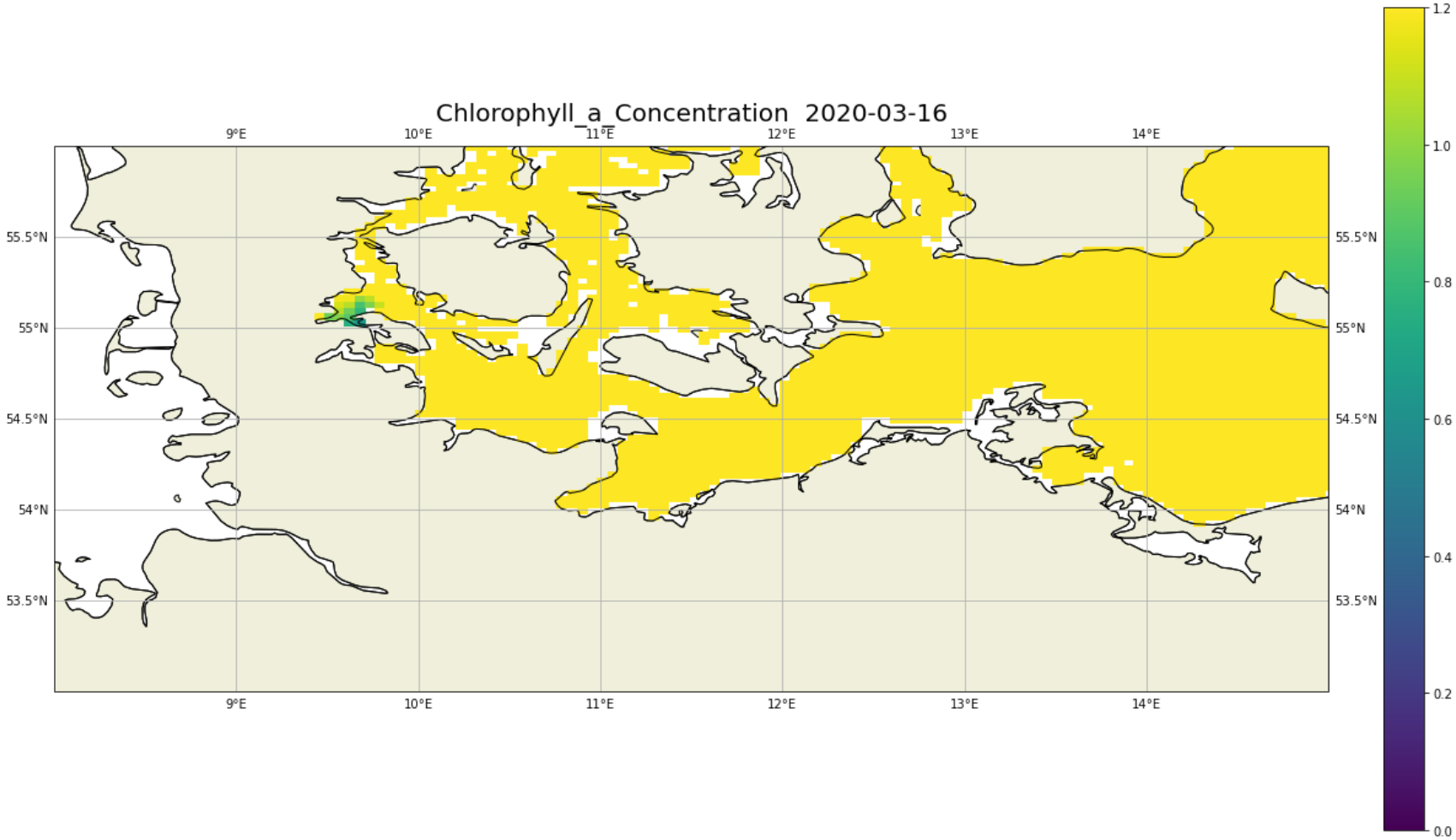


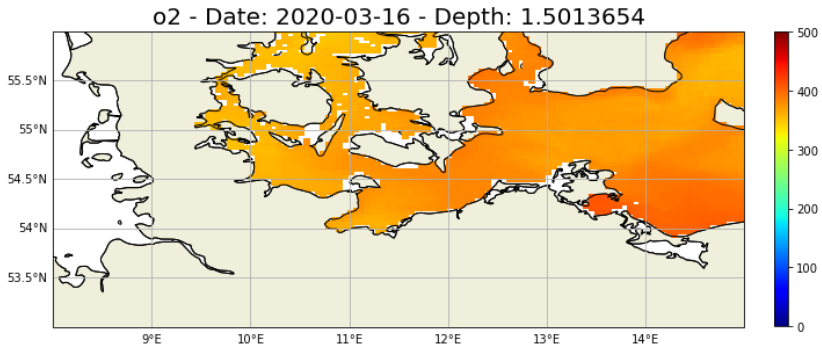

o2 - Date: 2020-01-16 - Depth: 1.5013654
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

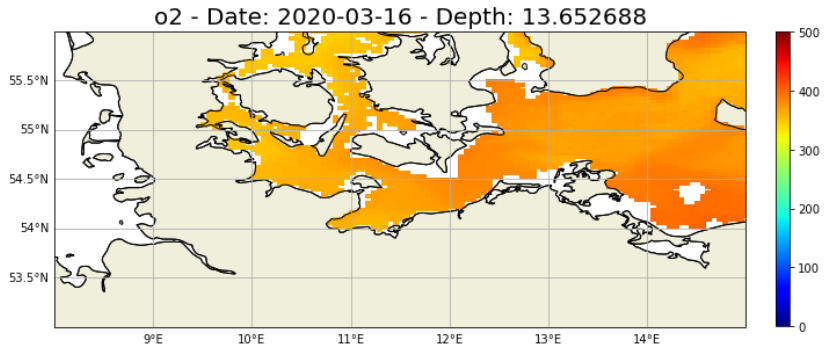

o2 - Date: 2020-01-16 - Depth: 13.652688
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

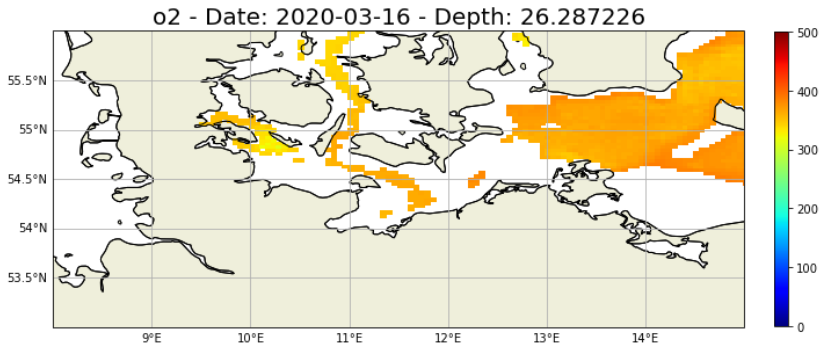

o2 - Date: 2020-01-16 - Depth: 26.287226
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

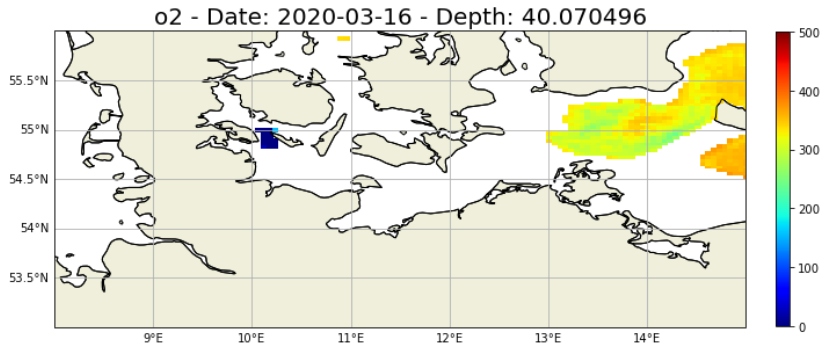

o2 - Date: 2020-01-16 - Depth: 40.070496
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

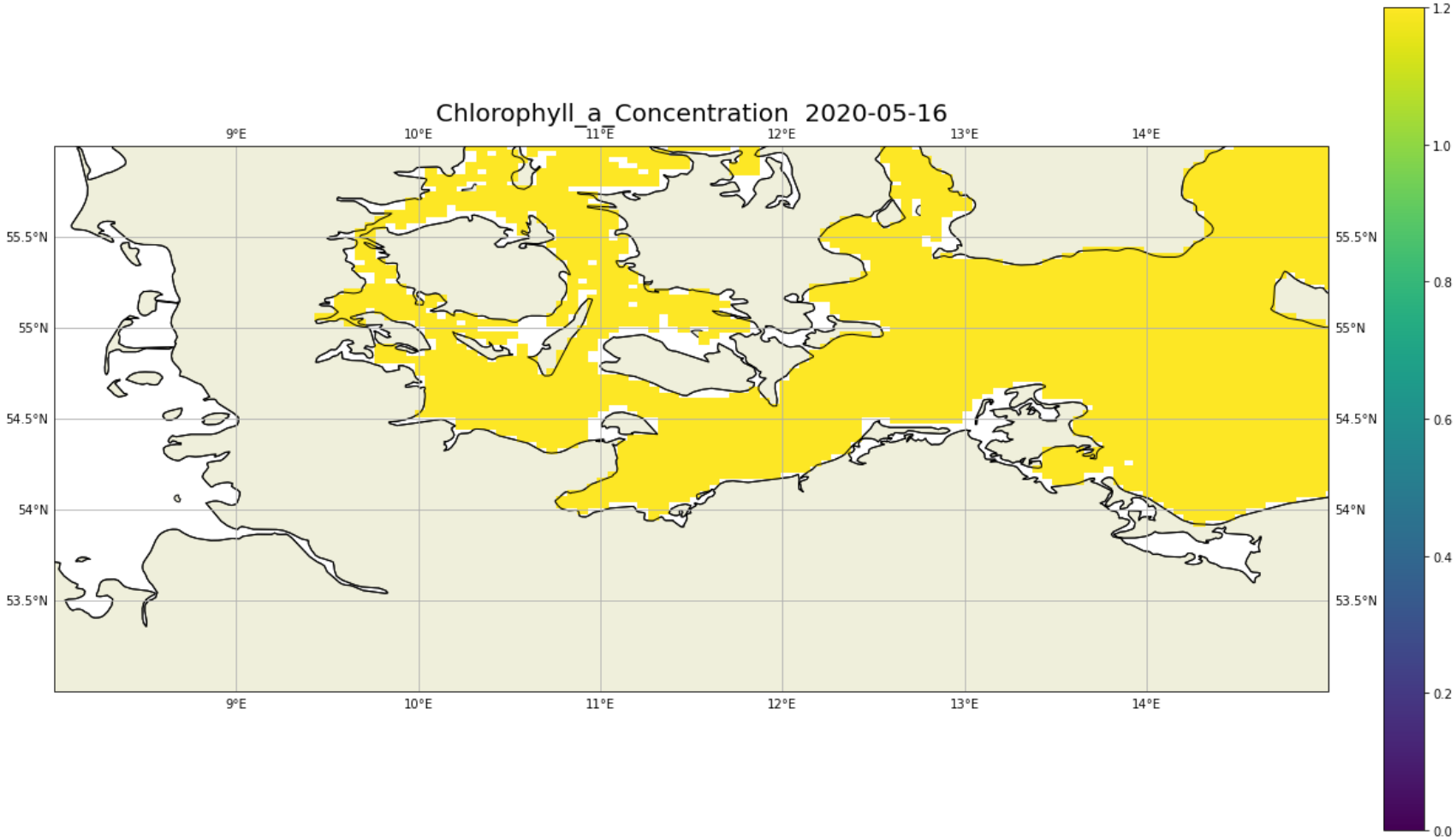

Chlorophyll_a_Concentration 2020-03-16
9°E
10°E
11°E
12°E
13°E
14°E
55.5°N
55°N
54.5°N
54°N
53.5°N
1.2
1.0
0.8
0.6
0.4
0.2
0.0

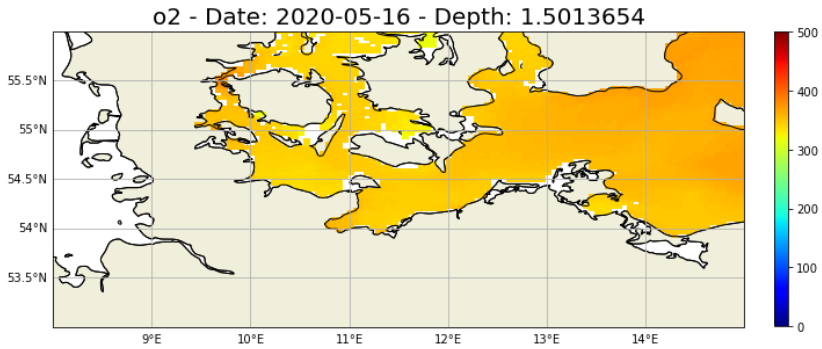

o2 - Date: 2020-03-16 - Depth: 1.5013654
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

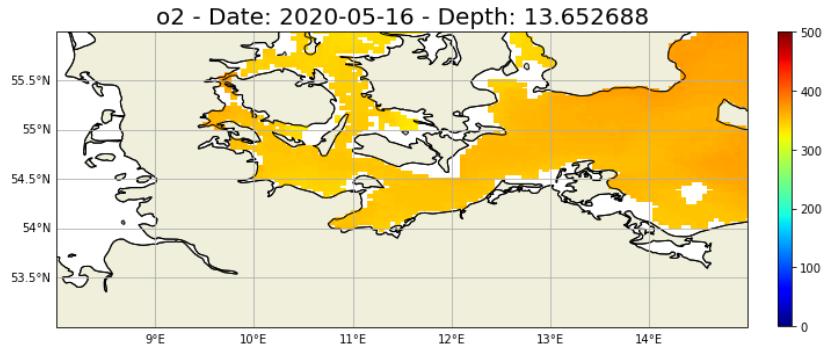

o2 - Date: 2020-03-16 - Depth: 13.652688
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

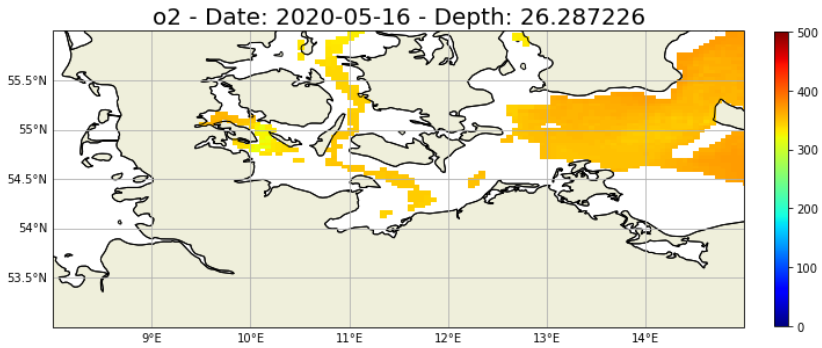

o2 - Date: 2020-03-16 - Depth: 26.287226
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

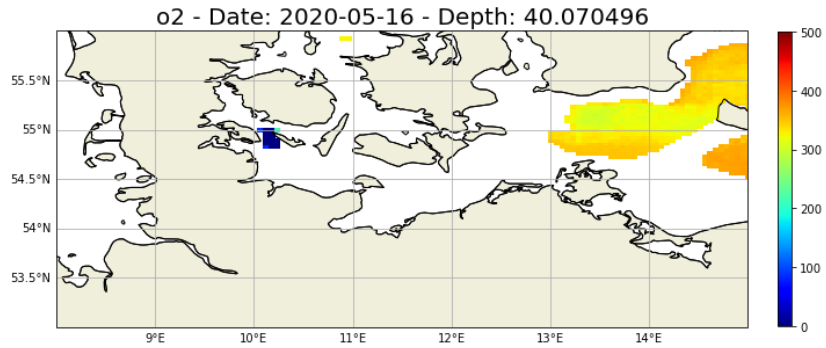

o2 - Date: 2020-03-16 - Depth: 40.070496
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

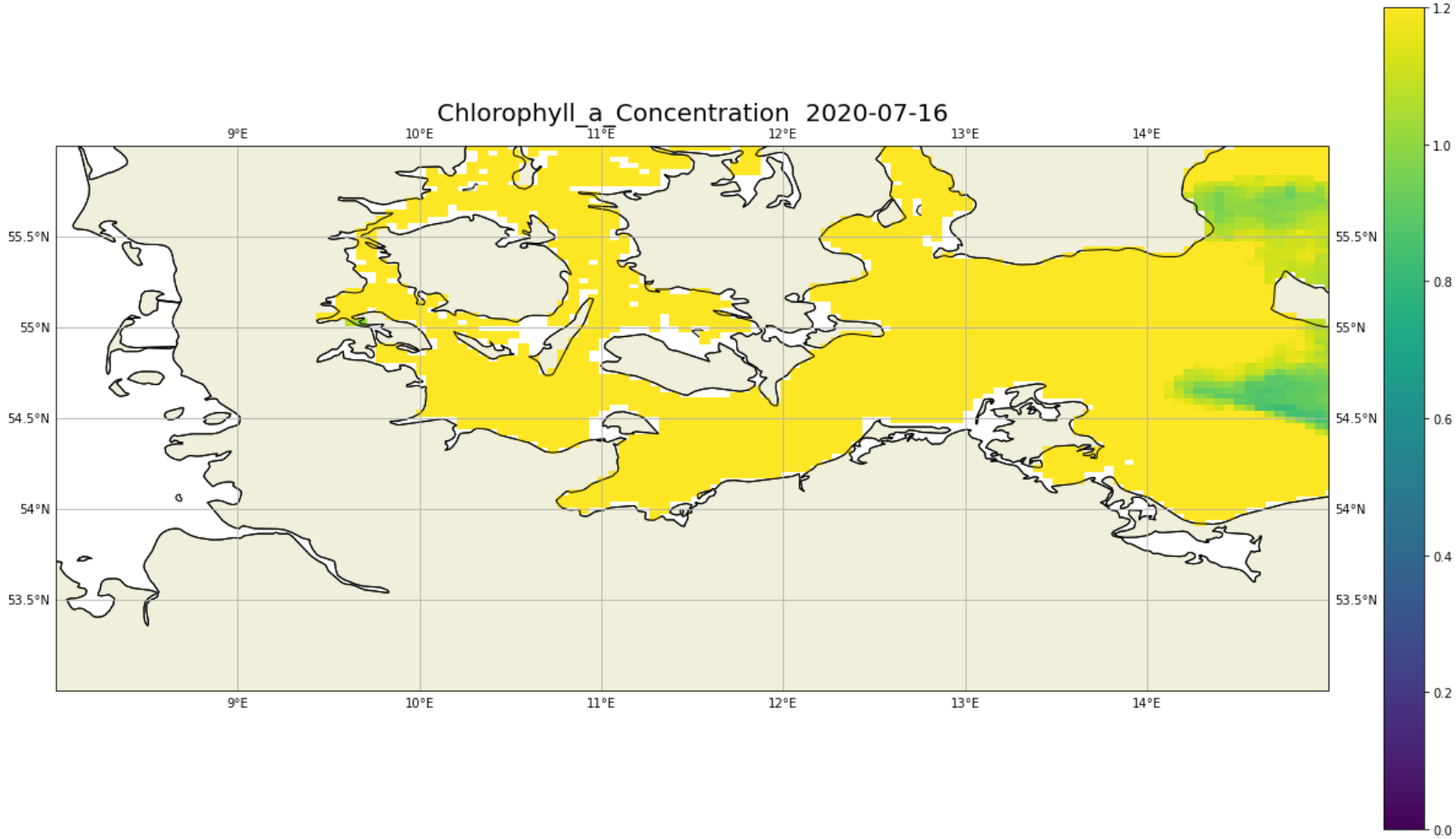

Chlorophyll_a_Concentration 2020-05-16
9°E
10°E
11°E
12°E
13°E
14°E
55.5°N
55°N
54.5°N
54°N
53.5°N
1.2
1.0
0.8
0.6
0.4
0.2
0.0

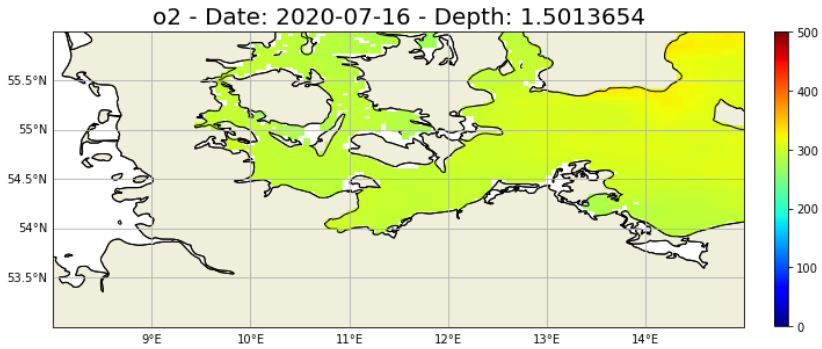

o2 - Date: 2020-05-16 - Depth: 1.5013654
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

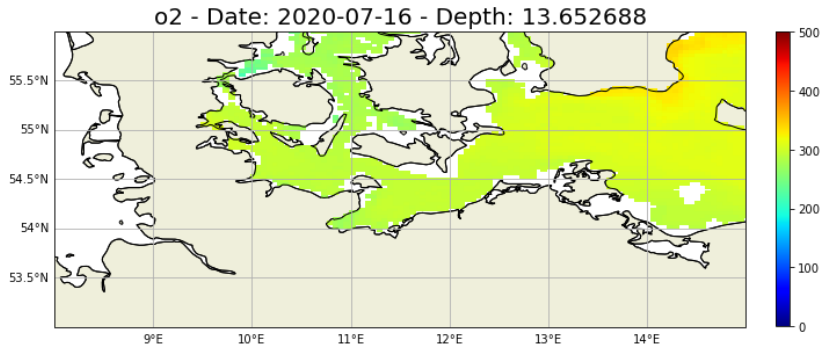

o2 - Date: 2020-05-16 - Depth: 13.652688
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

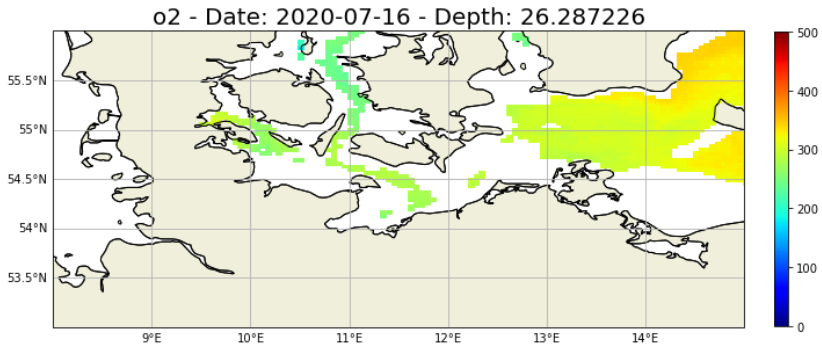

o2 - Date: 2020-05-16 - Depth: 26.287226
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

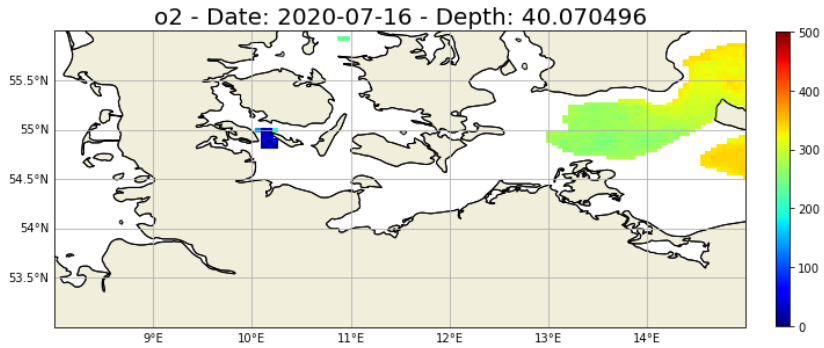

o2 - Date: 2020-05-16 - Depth: 40.070496
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0


Chlorophyll_a_Concentration 2020-07-16
9°E
10°E
11°E
12°E
13°E
14°E
55.5°N
55°N
54.5°N
54°N
53.5°N
1.2
1.0
0.8
0.6
0.4
0.2
0.0


o2 - Date: 2020-07-16 - Depth: 1.5013654
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0


o2 - Date: 2020-07-16 - Depth: 13.652688
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0


o2 - Date: 2020-07-16 - Depth: 26.287226
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0


o2 - Date: 2020-07-16 - Depth: 40.070496
55.5°N
55°N
54.5°N
54°N
53.5°N
9°E
10°E
11°E
12°E
13°E
14°E
500
400
300
200
100
0

Exercise

Change the date list and/or depth list and show the relation between the Chlorophyll and Oxygen concentrations.

Exercise

Try to change the region and explore the relation between the Chlorophyll and Oxygen concentrations.

6. Vertical profile - Arkona Sea

Go back to the "Table of contents"

Now we want to plot a vertical profile of a section in the Arkona Sea.

In [51]:
```
# Set the coordinates of a line in the Arkona Sea

lon_min,lon_max = 13,14.5
lat = 55
```

In [52]:

```
# Plot a minimap with a red section line

# Define the settings of the plot
f = plt.figure(figsize=(8, 8))                                  # define the size of the plot
ax = plt.axes(projection=ccrs.PlateCarree())                    # create an ax and select the proje
ax.coastlines()                                                 # add coastlines
gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True)     # add map gridlines
ax.add_feature(cfeature.LAND, zorder=1, edgecolor='k')          # add land mask
ax.add_feature(cfeature.OCEAN)                                  # add ocean mask
ax.add_feature(cfeature.BORDERS)                                # add borders mask
ax.add_feature(cfeature.RIVERS)                                 # add rivers mask
ax.set_extent(long_lat_reg,crs=ccrs.PlateCarree())              # define the extent of the map

# plot red section line on the map
plt.plot([lon_min, lon_max], [lat, lat], color='red', linewidth=3, transform=ccrs.PlateCarree())
plt.show()
```

In [53]:

```
# Check if the folder './maps/section' exists. Otherwise it is created.
dir_exist('./maps/section')
```

In [54]:
```
# Definition of a function

# We are defining a function for plotting vertical profiles of Dissolved Oxygen and Chlorophyll along a sectin line
# Input parameter: date and lat_in, lon1_in, lon2_in: coordinates of the section line

def vert_sec(date, lat_in, lon1_in, lon2_in):
    # Select the o2-data and chl-data along the section line
    o2_mod = data_baltic_o2.sel(latitude=lat_in,method='nearest').sel(longitude=slice(lon1_in,lon2_in)).sel(time=date)
    chl_mod = data_baltic_chl.sel(latitude=lat_in,method='nearest').sel(longitude=slice(lon1_in,lon2_in)).sel(time=dat

    # Define the settings of the plot
    f,axs = plt.subplots(1,2, figsize=(20,10))                                    # define 2 subplots and
    for ax in axs :                                                               # apply to the two subp
        ax.set_ylabel("Depth (m)",fontsize=18)                                    # set the label of the
        ax.set_xlabel("Longitude",fontsize=18)                                    # set the label of the
        ax.invert_yaxis()                                                         # reverse the y axis

    # Plot the o2-data and chl-data, set the minimum and maximum values of the colorbar and the colormap to use
    im0 = axs[0].pcolor(o2_mod['longitude'],o2_mod['depth'],o2_mod['o2'],vmin = 0,vmax=500,cmap='jet')
    im1 = axs[1].pcolor(chl_mod['longitude'],chl_mod['depth'],chl_mod['chl'],vmin = 0,vmax=1.2,cmap='viridis')

    # Add the legends
    cbar0 = f.colorbar(im0,ax=axs[0],fraction=0.03,pad=0.04)
    cbar1 = f.colorbar(im1,ax=axs[1],fraction=0.03,pad=0.04)

    # Add the titles
    axs[0].set_title("o2 - Date: " + date,fontsize=20)
    axs[1].set_title("chl - Date: " + date,fontsize=20)

    # Save the image of the subplots
    plt.savefig('maps/section/' + 'o2_chl_section_' + date)
```

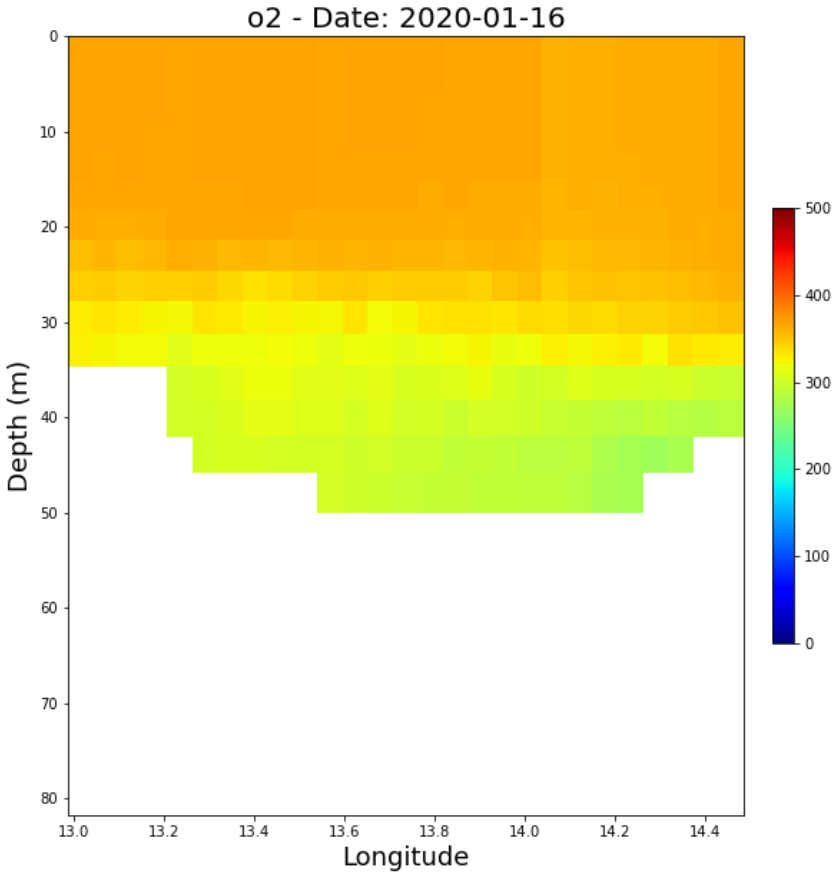
In [55]:
```
# Check the Date-list

Date_list_o2
```

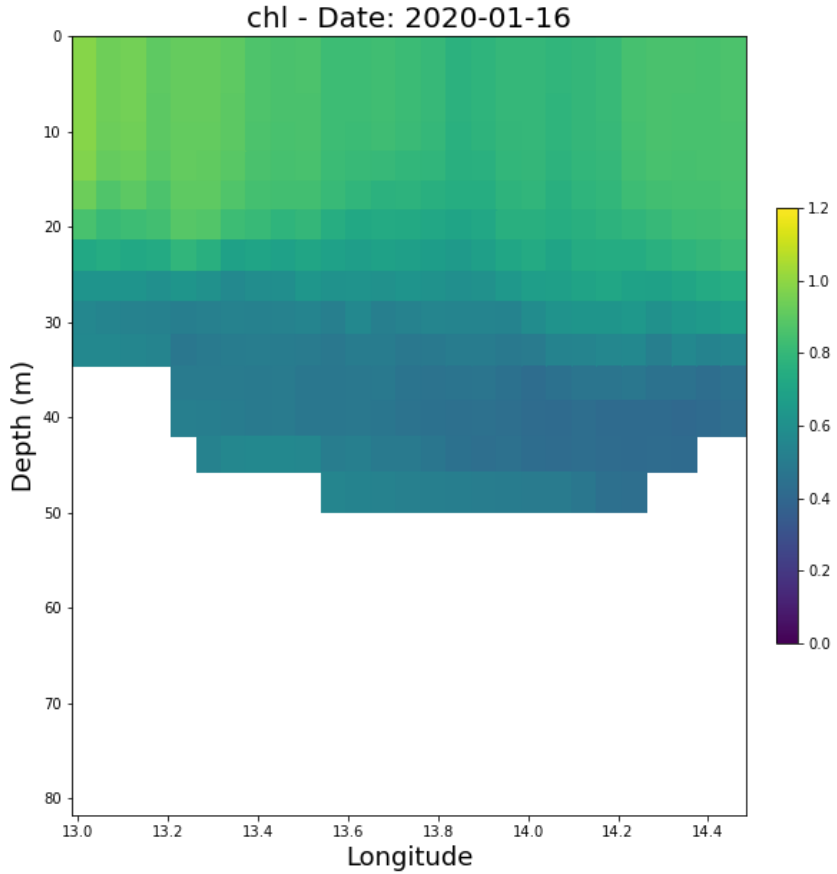
Out[55]:
```
['2020-01-16',
 '2020-03-16',
 '2020-05-16',
 '2020-07-16',
```

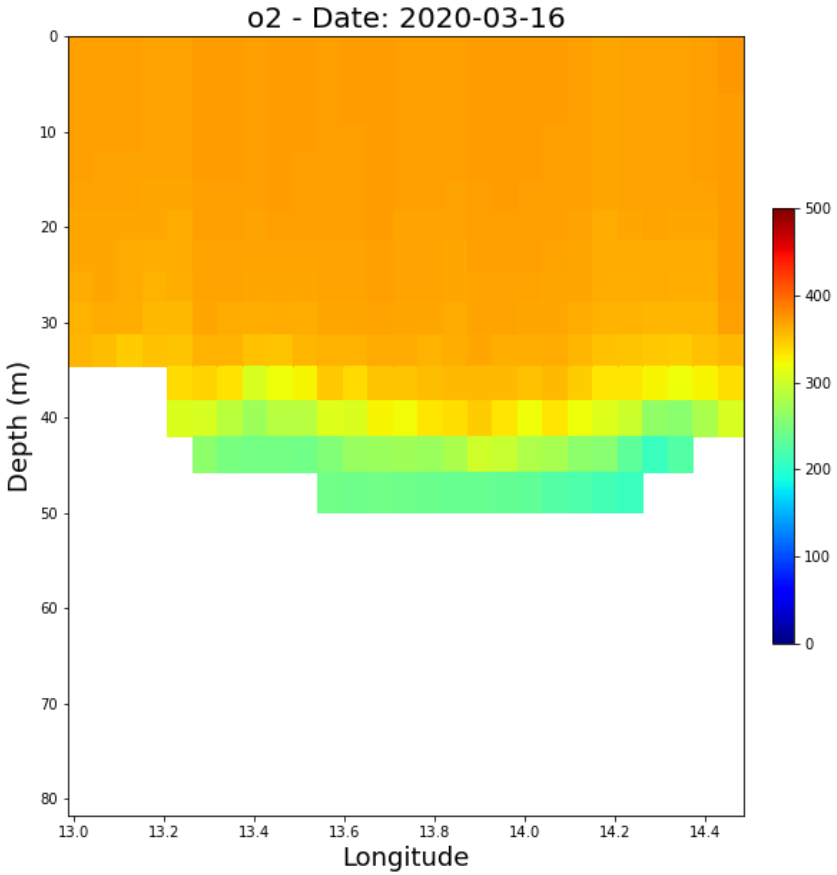
```
'2020-09-16',
```

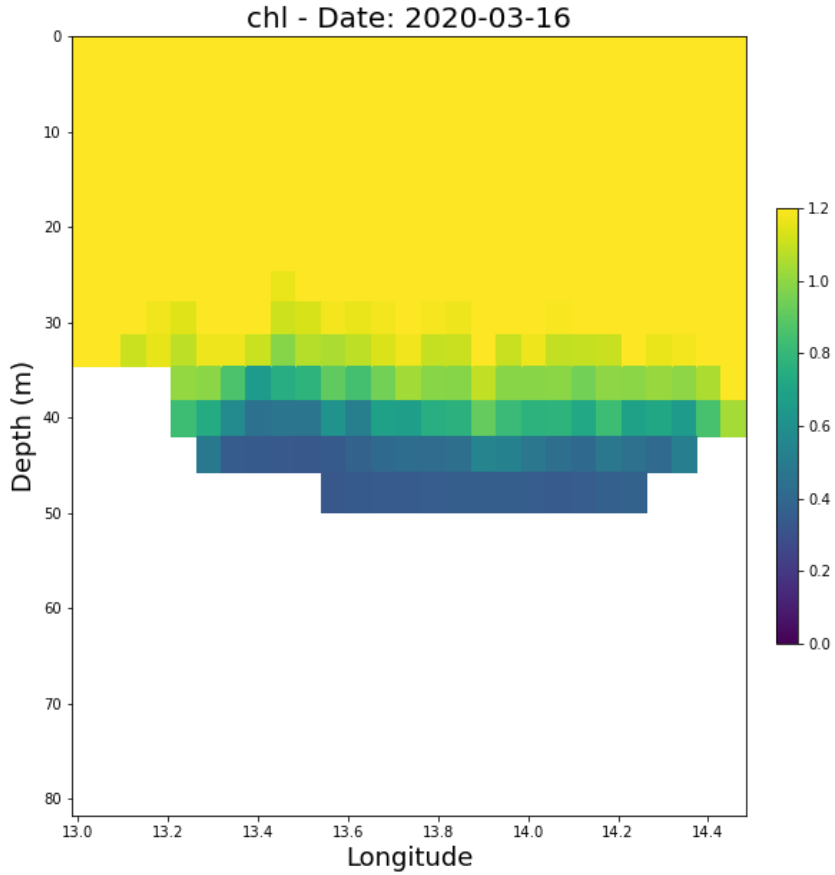
In [56]:

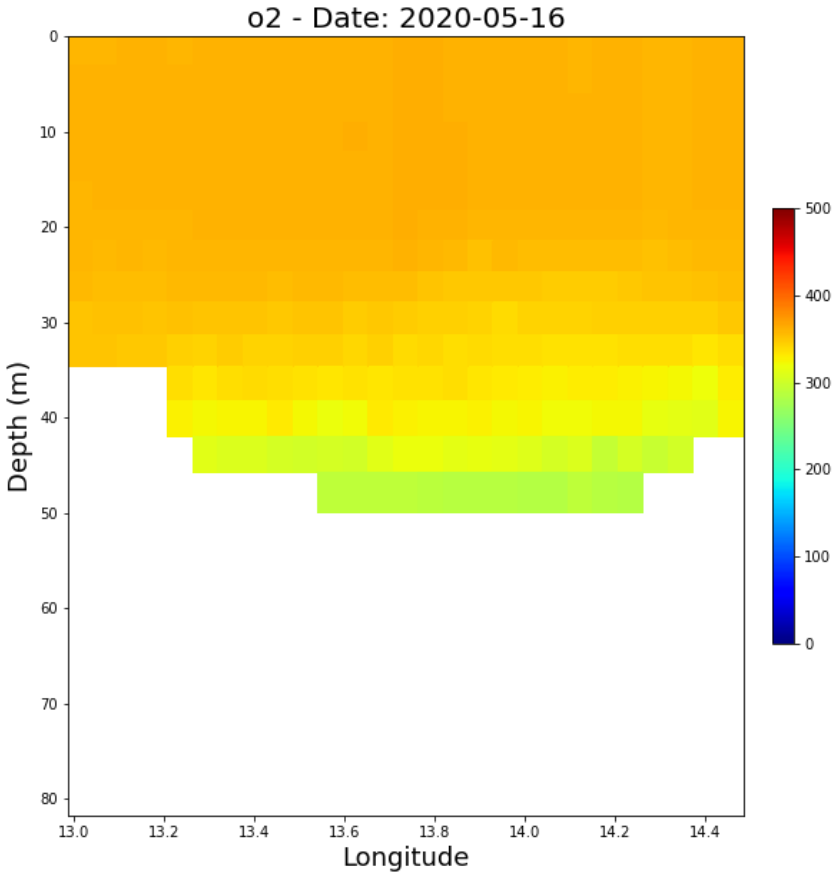
```
# Now we are generating the maps for Date_list_o2 and save the maps in the directory maps/section

for date in Date_list_o2 :
    vert_sec(date, lat, lon_min, lon_max)
```

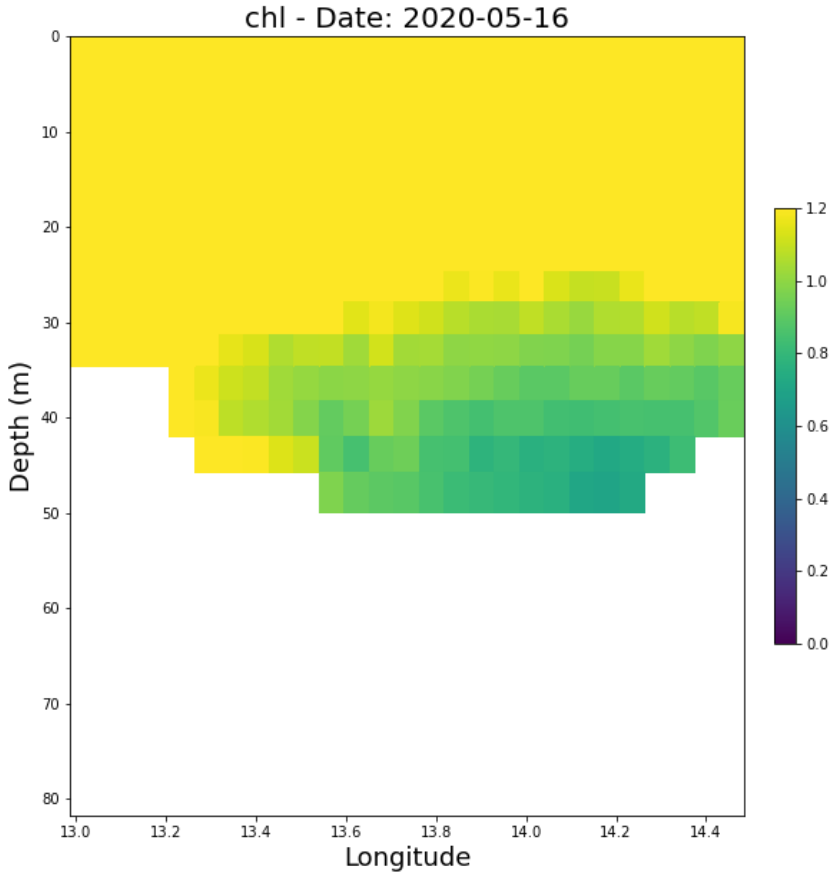


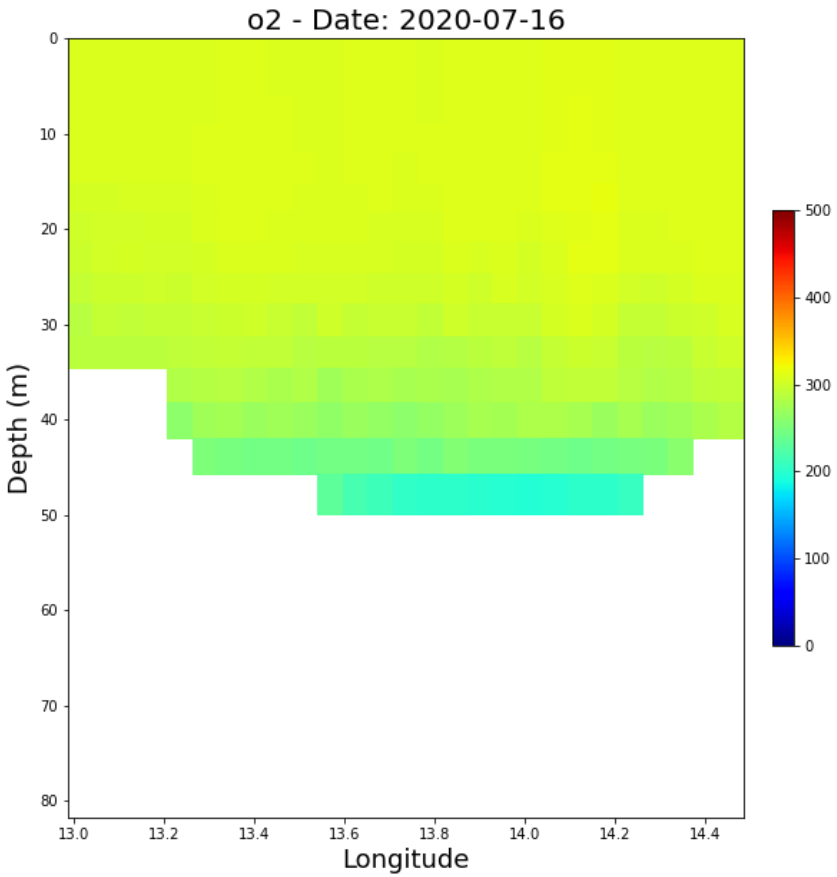


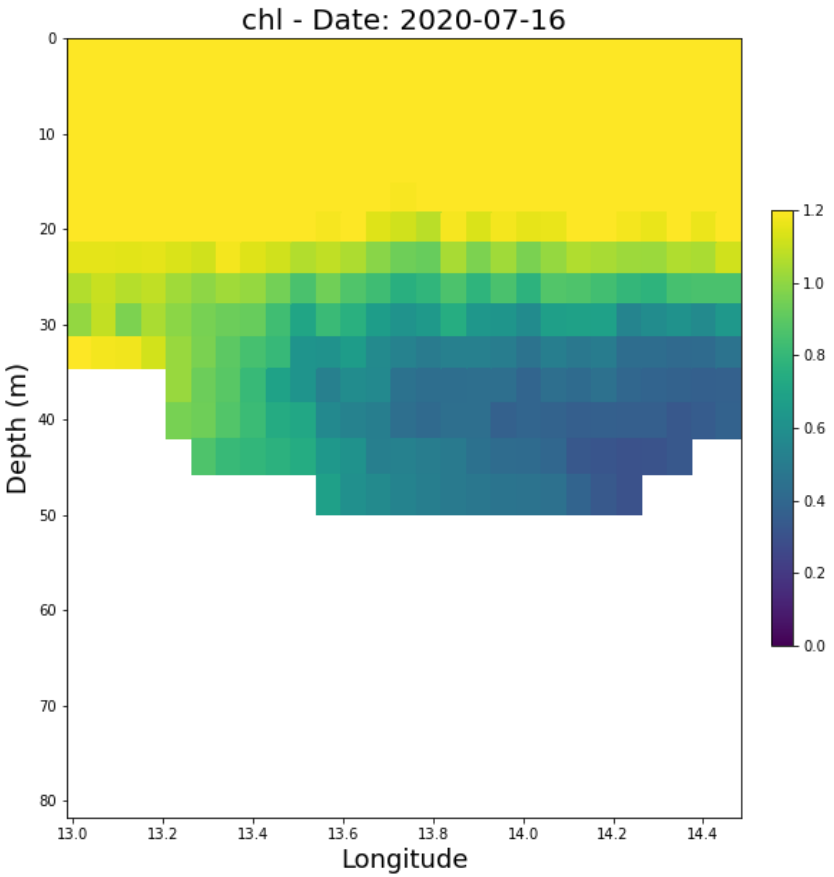

o2 - Date: 2020-03-16
Depth (m)
Longitude
0
10
20
30
40
50
60
70
80
13.0
13.2
13.4
13.6
13.8
14.0
14.2
14.4
500
400
300
200
100
0

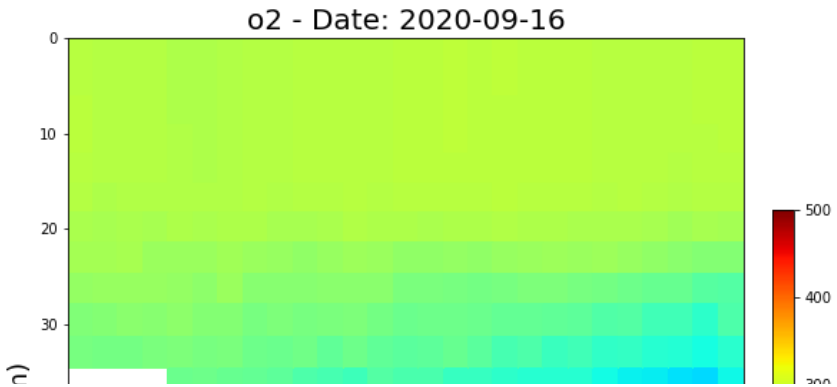

chl - Date: 2020-03-16
Depth (m)
Longitude
0
10
20
30
40
50
60
70
80
13.0
13.2
13.4
13.6
13.8
14.0
14.2
14.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0

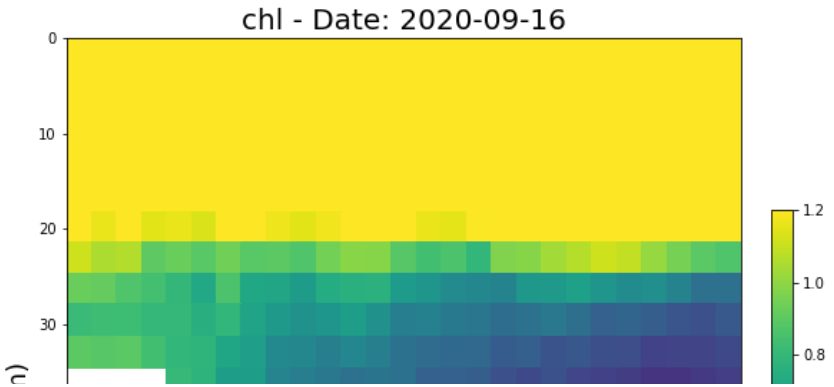

o2 - Date: 2020-05-16
Depth (m)
Longitude
0
10
20
30
40
50
60
70
80
13.0
13.2
13.4
13.6
13.8
14.0
14.2
14.4
500
400
300
200
100
0


chl - Date: 2020-05-16
Depth (m)
Longitude
0
10
20
30
40
50
60
70
80
13.0
13.2
13.4
13.6
13.8
14.0
14.2
14.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0





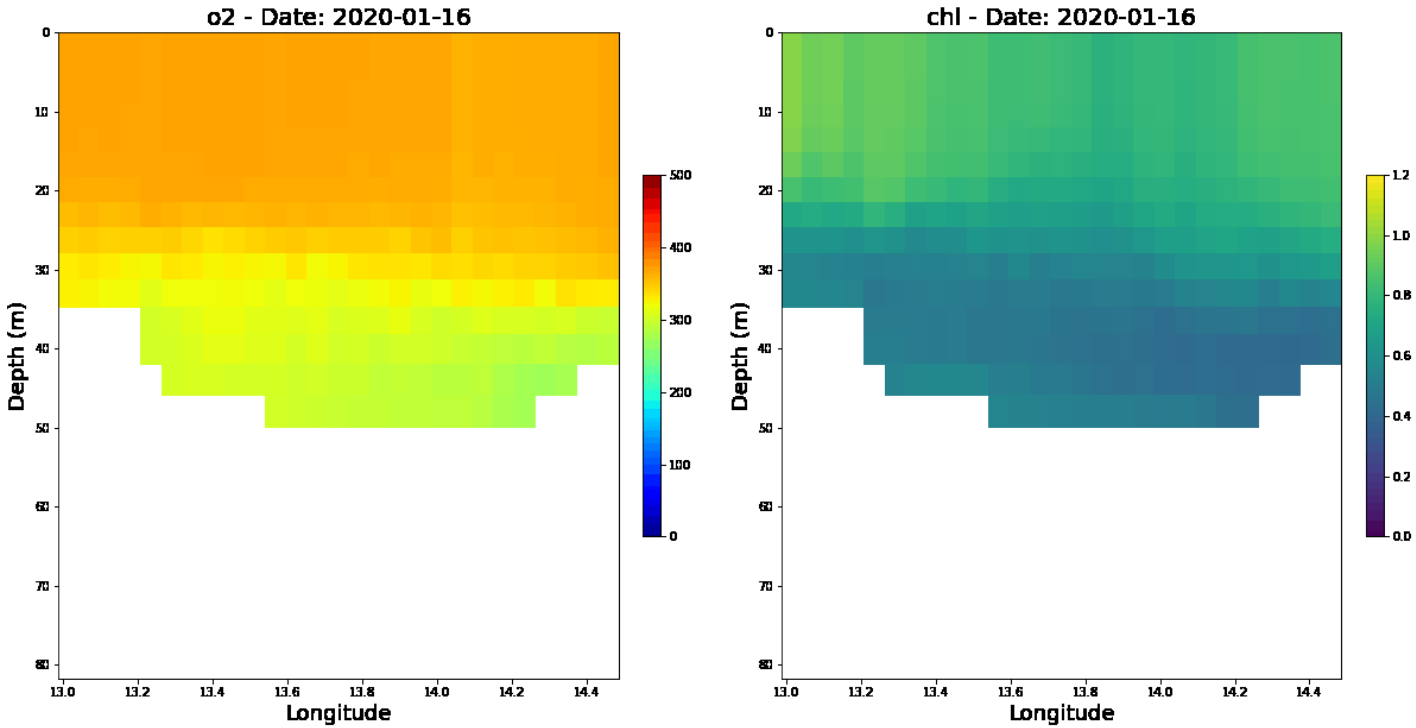




These maps are now shown in an animated slide show.

In [57]:

```
# Import the modules for the animation

import imageio
from pathlib import Path

# Search the maps
image_path = Path('maps/section')

images = sorted(image_path.glob('o2_chl_section_*.png'))
image_list = []
for file_name in images:
    image_list.append(imageio.imread(file_name))

# Save the animated gif-file
imageio.mimwrite('maps/section/o2_chl_section.gif',image_list,duration=2.5,loop=1)

# Show the animated gif-file
with open('maps/section/o2_chl_section.gif','rb') as f:
    display(Image(data=f.read(),format='png'))
```



Exercise

Explore the profiles for every month of the year 2020.

Exercise

Change the section line and plot the vertical profile.