

Cryosphere Virtual Laboratory (CVL)

E.Malnes, D. Stødle, Ø. Godøy, A. Korosov, R. Skahjem-Eriksen, M.Itkin and O. Gråbak



POLAR SCIENCE CLUSTER MEETING
17. SEPTEMBER 2021

Team



Eirik, NORCE



Øystein, Met.no



Anton, NERSC



Mikhail, NPI



Daniel, NORCE



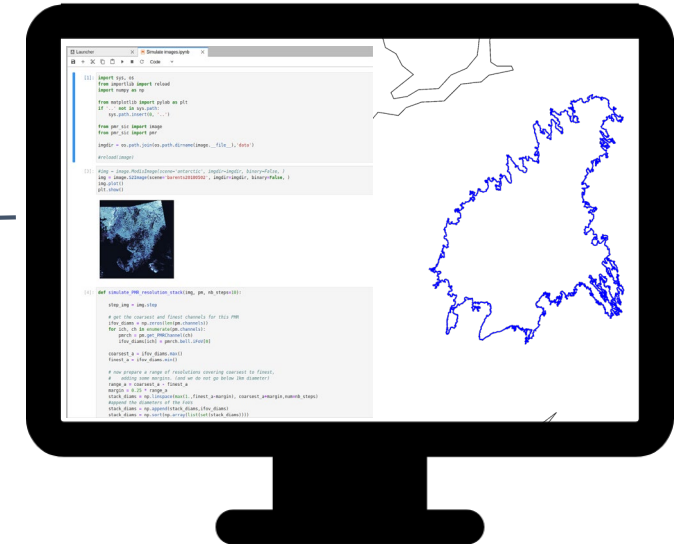
Robin, S&T

NORCE

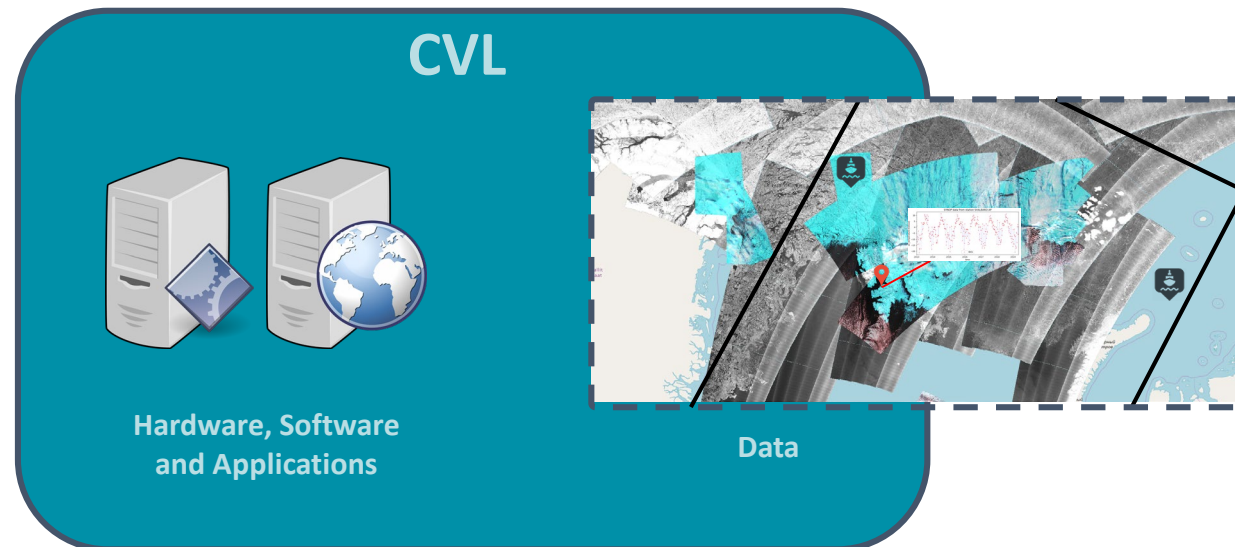
CVL as an Open Science Component



- a wide variety of data relevant for the Cryosphere
 - In-situ, model, EO
- software tools for processing, visualization and discovery of data.
- five predefined workflows for data access, visualization and processing.
- a generic platform supporting scientific work in an interdisciplinary context.
- champion science use cases



3

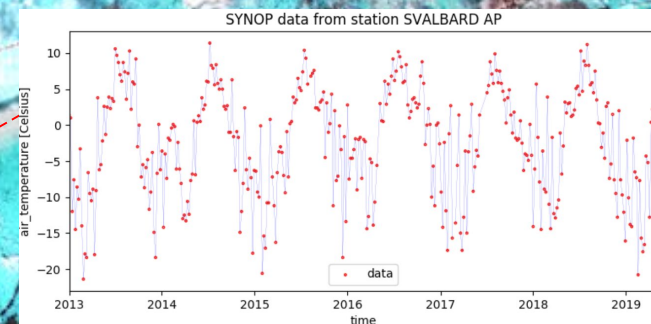
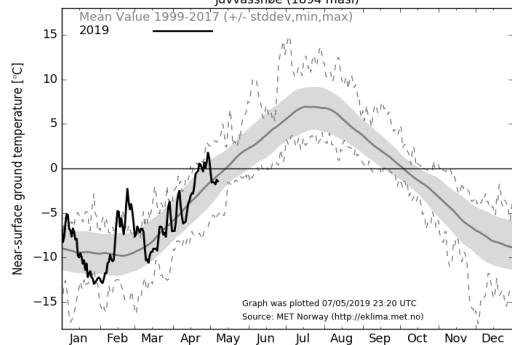


1 of 5 workflows

In order to understand and quantify the ongoing processes in the Cryosphere, the CVL aims to “Facilitate the exploitation, analysis, sharing, mining and visualization of massive EO data sets and high-level products within Europe and beyond.”



Near-surface ground temperature (0.2m)
Juvvasshøe (1894 masl)



NWP

What about the data

Challenges

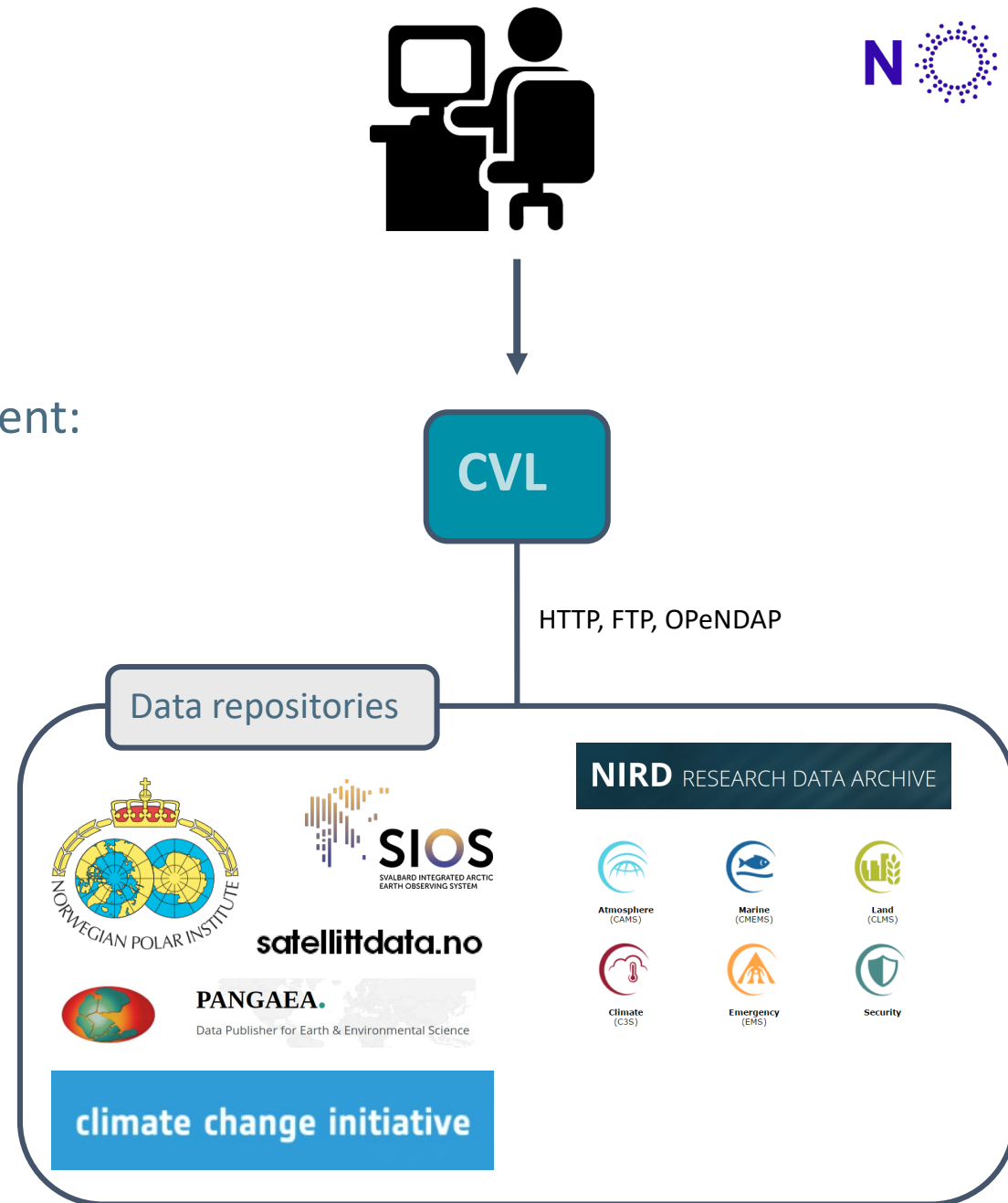
variety, velocity, volume, heterogeneity, access -> Big Data

FAIR guiding principles of Data Management:

- Findable
- Accessible
- Interoperable
- Reusable

Distributed data management

- remote access utilizing OPeNDAP
- allows for subsetting and aggregation
- supports streaming of data over the internet
- Harvesting of metadata



Execution Environments and Workflows



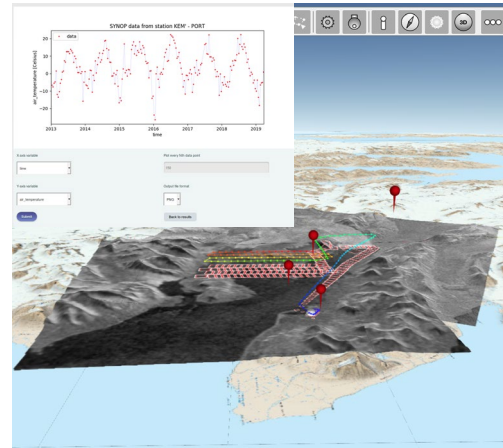
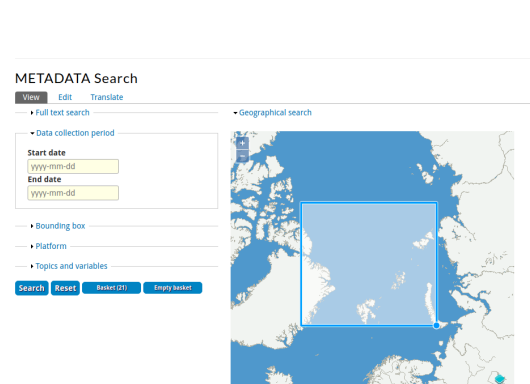
Search for Data

Visualizing data

Visualizing / Publishing
results

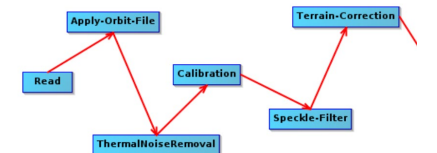
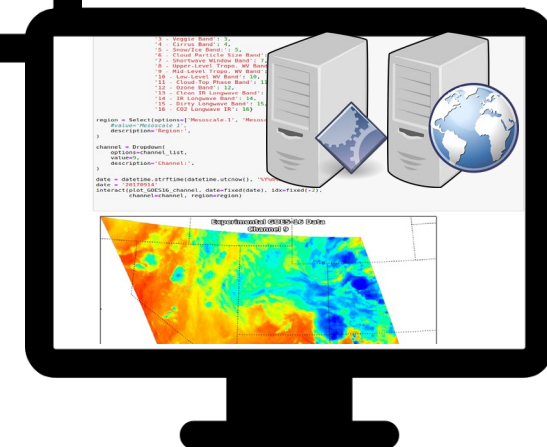
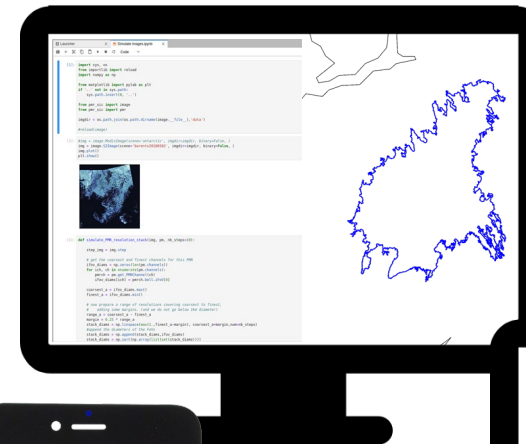
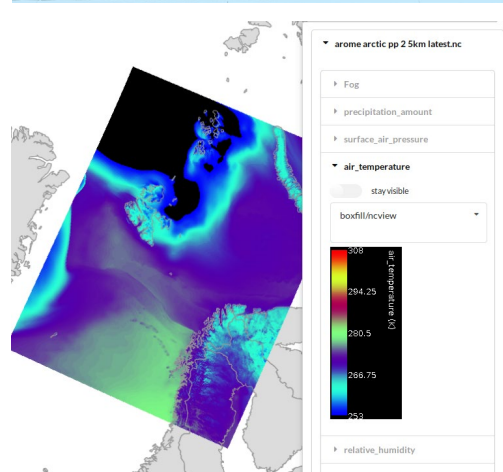
Interactive analysis on
hosted VM

Processing chain



```
File Edit View Search Terminal Help
trygveh@pc4822:~$ ipython
Python 3.6.5 [Anaconda, Inc.] (default, Apr 29 2018, 16:14:56)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.

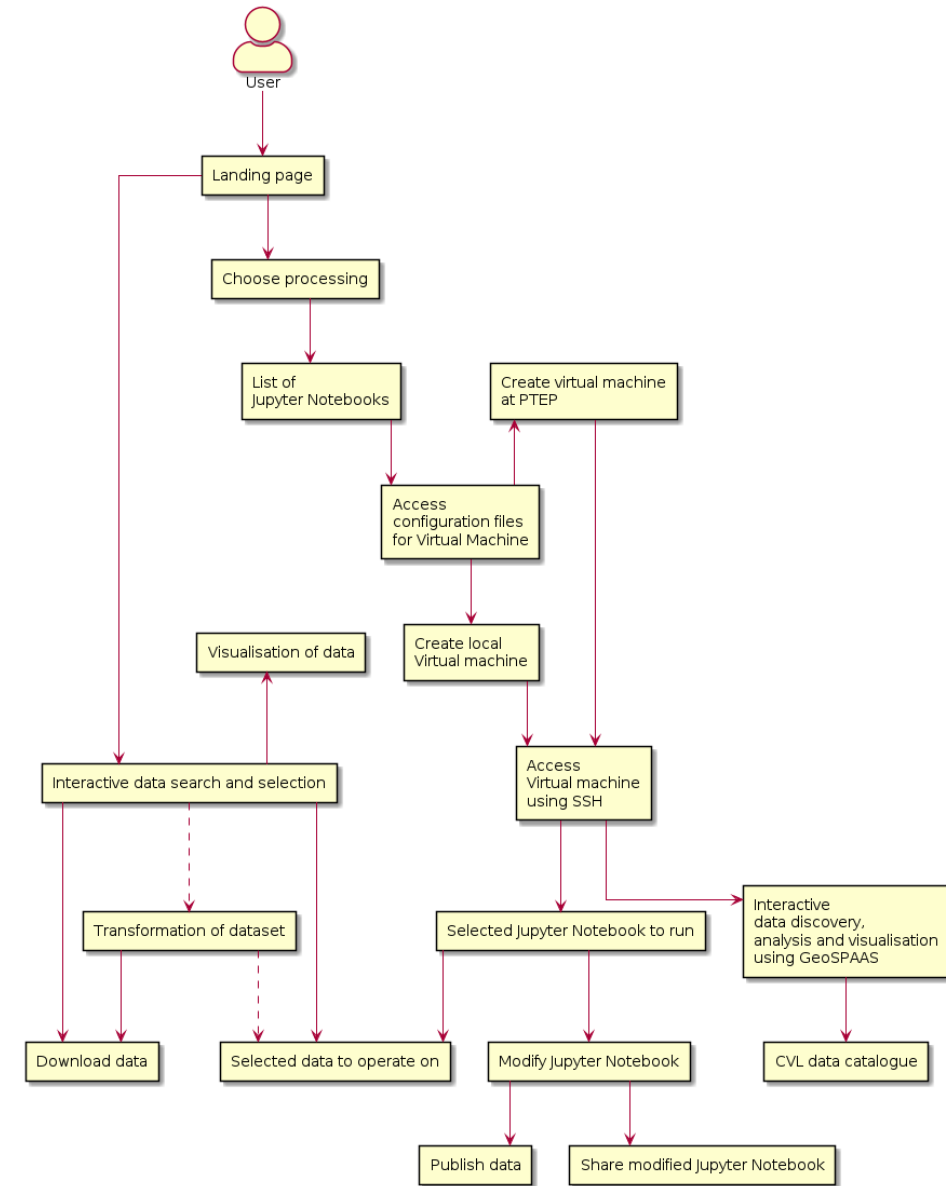
In [1]: import numpy as np
In [2]: import scipy as sp
In [3]: from CVL import SearchEngine
```



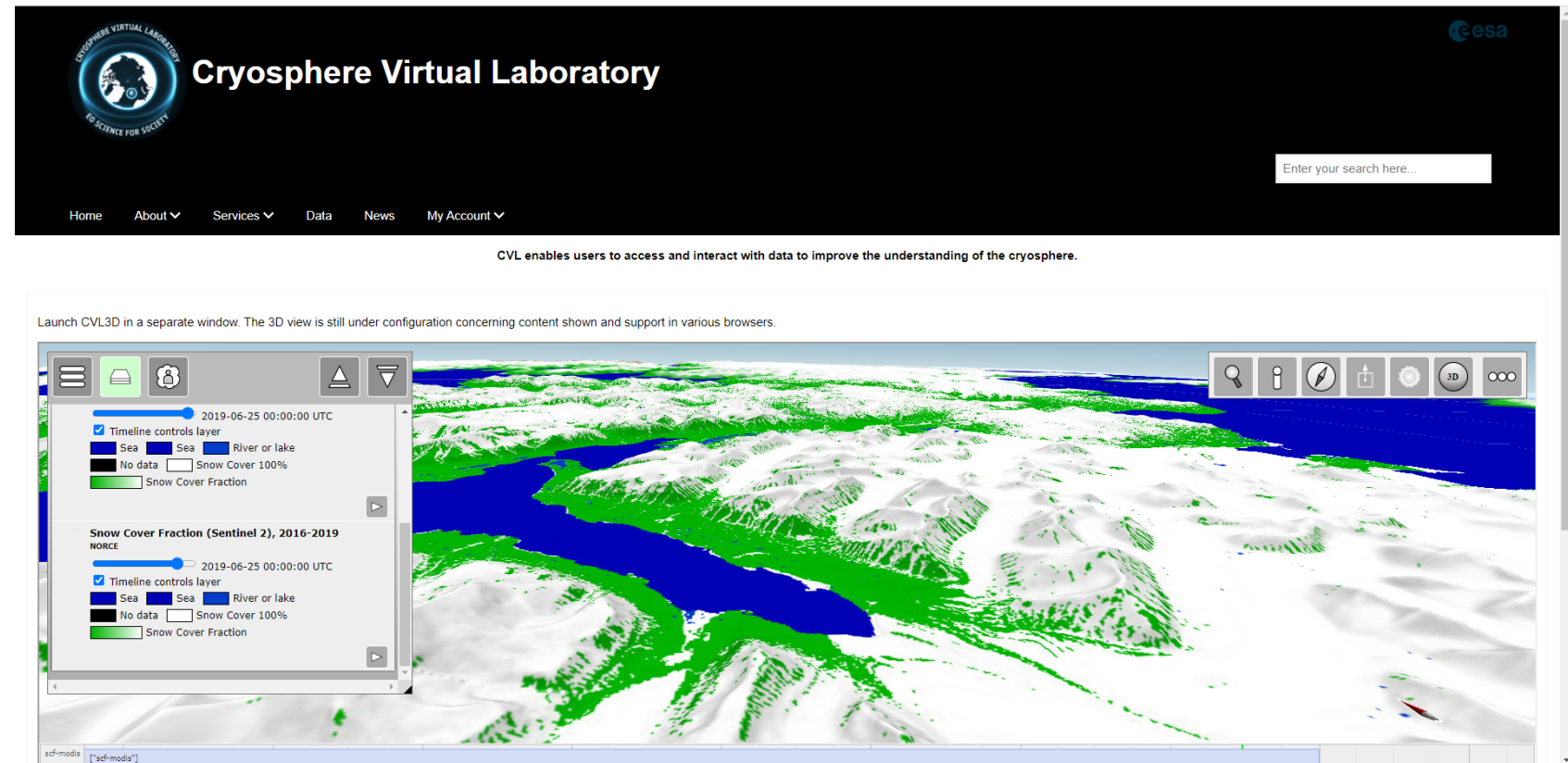
Work flows



- CVL allows a wide variety of work flows for the implementation of scientific code
 - Interactive or batch processing
 - Local /virtual processing
 - Integration with VRE (Virtual Research Environments) i.e. PTEP
 - Local virtual machines
 - Virtual processing (Jupyter scripts)
- Examples will be available on GitHub

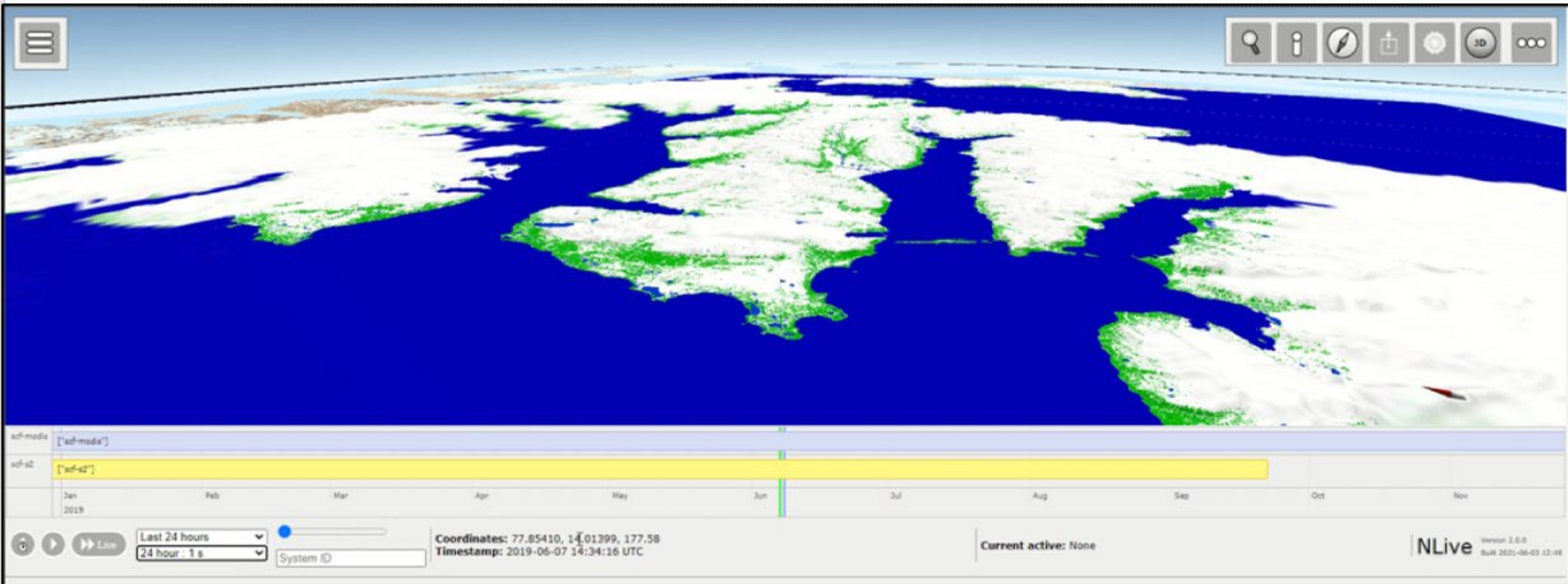


Visualization

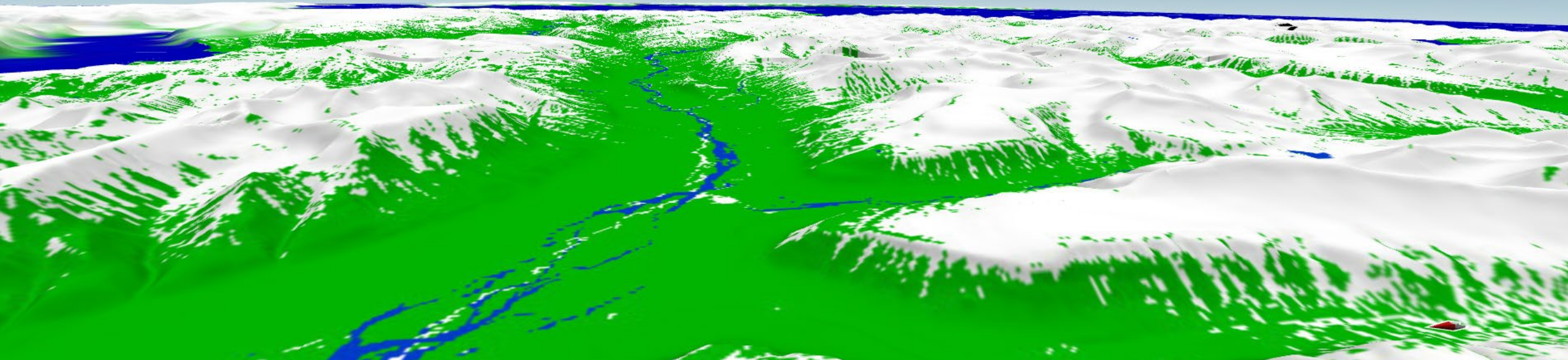


- CVL have implemented a dedicated 3D visualisation tool where examples are available on the Landing page cvl.eo.esa.int
- 3Dviz can be embedded in Jupyter notebooks for vizualization of results

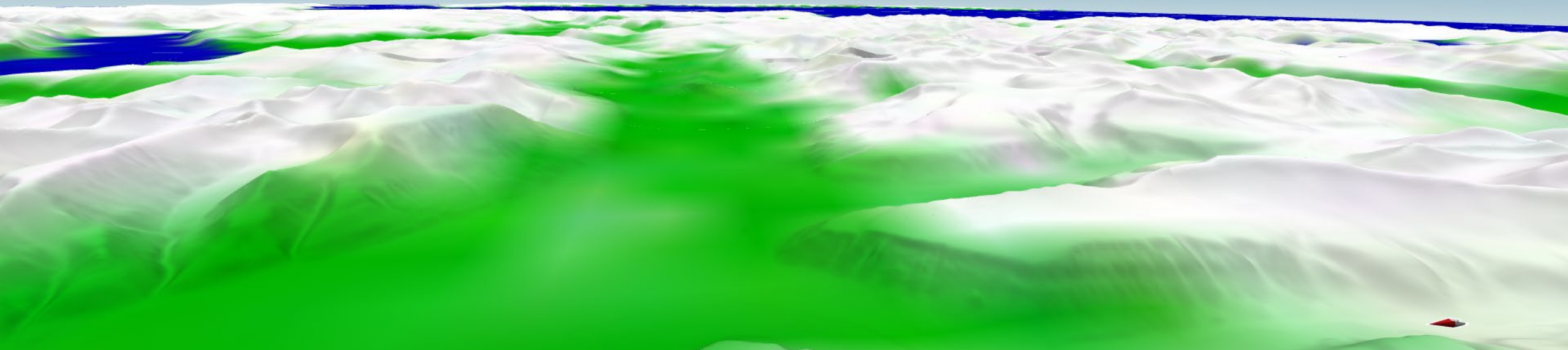
Comparison of snow cover fraction with Sentinel-2 (20m) and MODIS (500m)



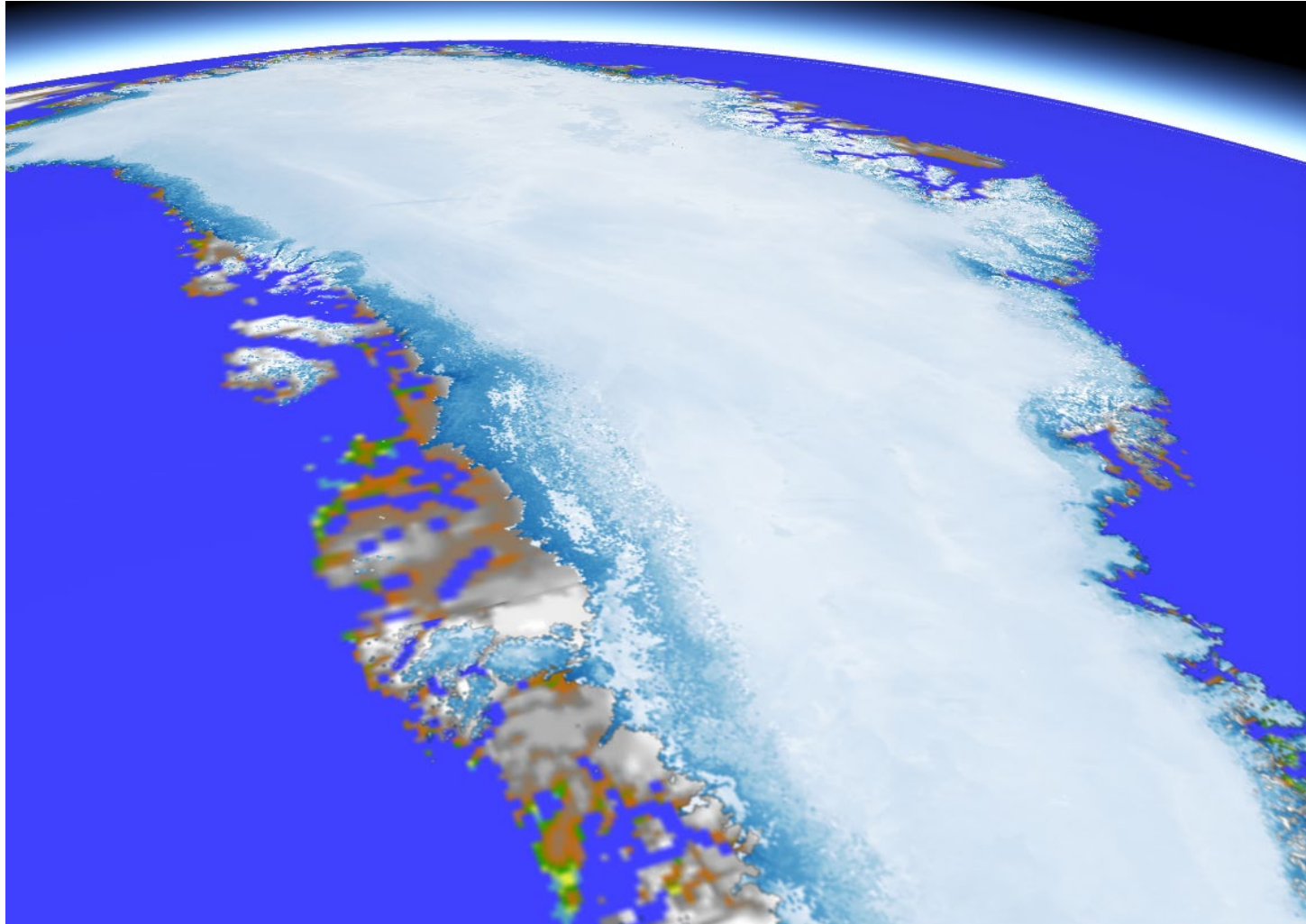
S2 20180618



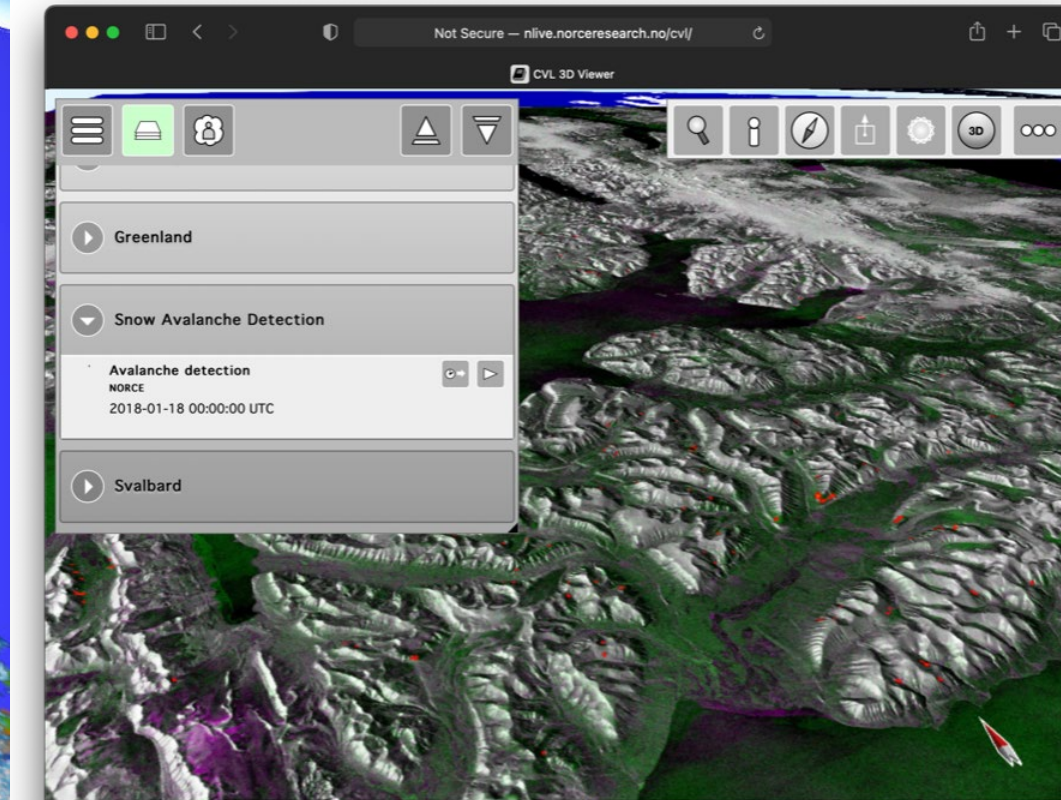
MODIS 20180618



Ex. Albedo on Greenland from GEUS



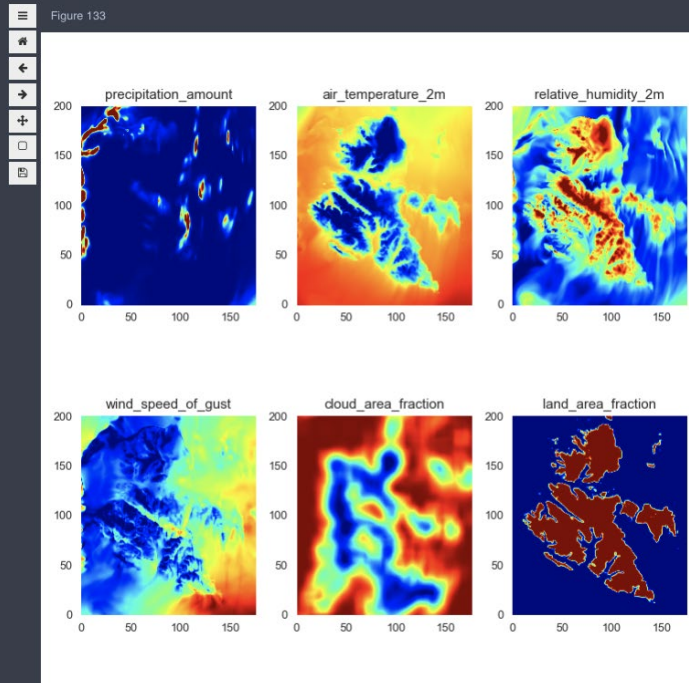
NORCE
Ex. Avalanche detection
on Svalbard



Ex. Avalanche detection on Svalbard

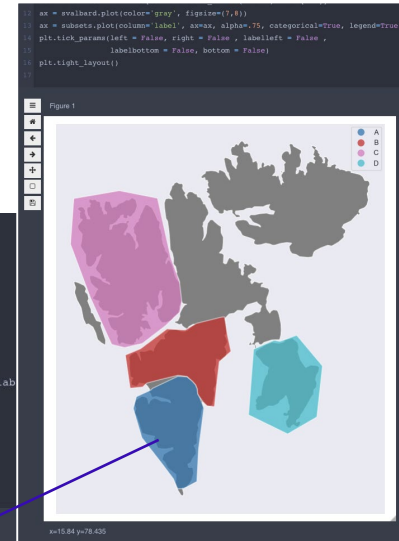


```
In [219]: # Display some data:
1 cols = 3
2 rows = int(np.ceil(len(variables)/cols))
3 _, axs = plt.subplots(rows, cols, figsize=(8,8))
4 for v, ax in zip(variables, axs.ravel()):
5     ax.set_title(v)
6     arr = np.atleast_3d(nativeformats.read_gdr(files[0])[0,:][v][:,:])
7     arr[arr>1e6] = np.nan
8     lo, hi = np.nanpercentile(arr, [2, 98])
9     ax.imshow(arr[:, :, -1], origin='lower', vmin=lo, vmax=hi, cmap='jet')
10
11 plt.tight_layout()
```



```
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
ax.set_ylabel('Avalanche Activity Density')
ax.set_xlabel('Time')
ax.set_ylim([0, 750])
ax.grid(True)
ax.legend(loc='upper left')
ax.set_title(f'Winter season (yr:04d)-(yr+1:04d):')

ax_twin = ax.twinx()
params[(params.index>=t0) & (params.index<=t1)][f'snow_mean_{lab}', f'rain_mean_{lab}']
ax_twin.set_ylim([0, 30])
ax_twin.set_ylabel('Precipitation (kg/m^2/day)')
ax_twin.legend(['Cold', 'Warm'], loc='upper right')
#ax.set_xllim([t0, t1])
plt.suptitle(f'Region: {lab}')
plt.tight_layout()
```

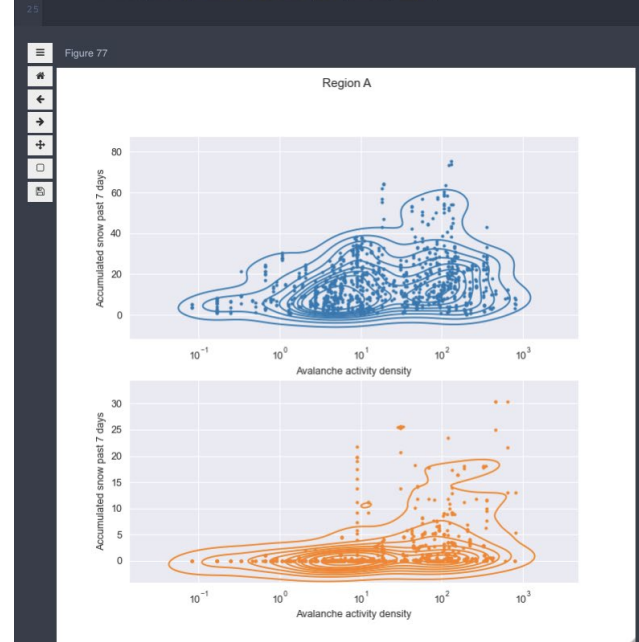


```
c = plt.rcParams['axes.prop_cycle'].by_key()['color']
ns.axes_style('darkgrid'):
r lab in subsets.label:
fig, axs = plt.subplots(2,1, figsize=(8,8))
data = params[params[f'and_{lab}']>0]

plt.suptitle(f'Region {lab}')

ax = axs.ravel()[0]
sns.kdeplot(data=data, ax=ax, x=f'aad_{lab}', y=f'snow_rsum_{lab}', log_scale=[True, False],
            data.plot(ax=ax, x=f'aad_{lab}', y=f'snow_rsum_{lab}', kind='scatter', logx=True, color=col_c
ax.set_xlabel('Avalanche activity density')
ax.set_ylabel(f'Accumulated snow past {window} days')

ax = axs.ravel()[1]
sns.kdeplot(data=data, ax=ax, x=f'aad_{lab}', y=f'rain_rsum_{lab}', log_scale=[True, False],
            data.plot(ax=ax, x=f'aad_{lab}', y=f'rain_rsum_{lab}', kind='scatter', logx=True, color=col_c
ax.set_xlabel('Avalanche activity density')
ax.set_ylabel(f'Accumulated snow past {window} days')
```



Jupyter notebooks

- CVL utilize Jupyter notebooks for implementing scientific use cases
- 5 initial use cases
 - Marginal Ice zone –validation
 - Marginal ice zone –biology
 - Arctic amplification
 - Snow
 - Avalanches

Ice zone-validation

```
# make a dataset from contents
dataset = xr.open_dataset(df)

lat = dataset.lat.values
lon = dataset.lon.values

# view and extract properties of the dataset
# variables are snowfr, time, lat and lon
snowfraction = dataset.snowfr.values

# snow fraction is stored as a byte array in the NetCDF file
# convert to float type to do the averaging
snfr = snowfraction.astype('float32')

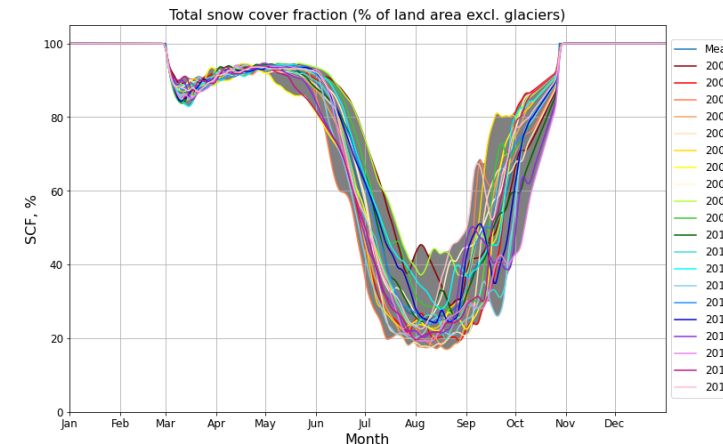
# script to read in the output files and check/plot out the data
ndays = snfr.shape[0]
meanscf = np.zeros(ndays)

# for each day in snow cover map get the mean SCF for entire Svalbard
for i in range(ndays):
    # extract snow map for a single day
    snfr_day = snfr[i,:,:]
    av_scf, n1 = get_mean_sca(snfr_day, veg)
    meanscf[i] = av_scf

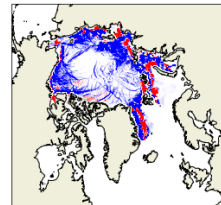
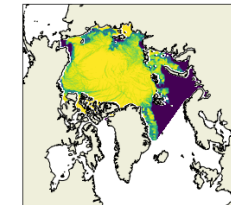
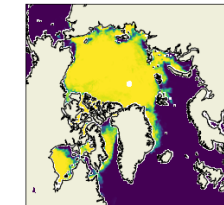
return snfr, meanscf, n1
```

Make figure:

In [4]: make_fig4()



plt.show()



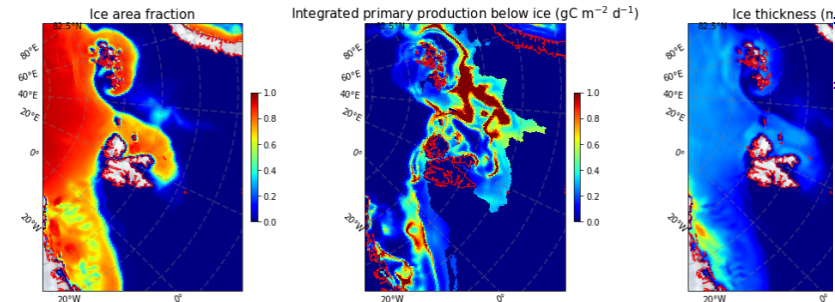
In []:

Marginal ice zone –biology

```
primary_production = nc.variables['npp'][0,:,:]
primary_production = primary_production/1000.
```

```
alice = ncice.variables['aice_d'][0,:,:]
thick = ncice.variables['hi_d'][0,:,:]
#create a masked array with zeros as filled values
primary_production = np.ma.masked_where(aice < 0.01, primary_production).filled(fill_value=0)
#store a dictionary for properties of plots
plotted = {}
plotted[0] = {'data':alice, 'max':1., 'title':'Ice area fraction'}
plotted[2] = {'data':thick, 'max':5., 'title':'Ice thickness (m)'}
plotted[1] = {'data':primary_production, 'max':1., 'title':'Integrated primary production below ice (gC m-2 d-1)'}
```

```
cmap = plt.get_cmap('jet')
figure=plt.figure(figsize=(18,6))
projection=ccrs.Stereographic(90, -45) # set the projection of plots
xyz_pro = projection.transform_points(ccrs.PlateCarree(), lon, lat) # convert Longitude and Latitude
for p in range(3):
    ax=figure.add_subplot(1,3,p + 1, projection=projection)
    # set the maximum and minimum of values of spectrum of plot(as well as colorbar)
    norm = colors.Normalize(vmin=0, vmax=plotted[p]['max'])
    # drawing the plot with pcolormesh
    img = ax.pcolormesh(xyz_pro[:, :, 0], xyz_pro[:, :, 1], plotted[p]['data'], cmap=cmap, transform=projection, norm=norm)
    # set extend of Svalbard
    ax.set_extent([500000, 2000000, -1500000, 500000], crs=projection)
    # make coastlines
    ax.coastlines(color='red', resolution='10m')
    ax.stock_img()
    # draw gridlines for better preceiving of projection view
    gl = ax.gridlines(transform=projection, draw_labels=True, linewidth=2, color='gray', alpha=0.3, linestyle='--')
    # remove some labels in order not to overlap with colorbar
    gl.right_labels = False
    ax.set_title(plotted[p]['title'], fontsize=15)
    figure.colorbar(img, ax = ax, fraction=0.03, pad=0.04)
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.5, hspace=None)
plt.show()
```



Arctic amplification

Plot monthly temperature series with linear trends.

Create a new DataArray with linear trends.

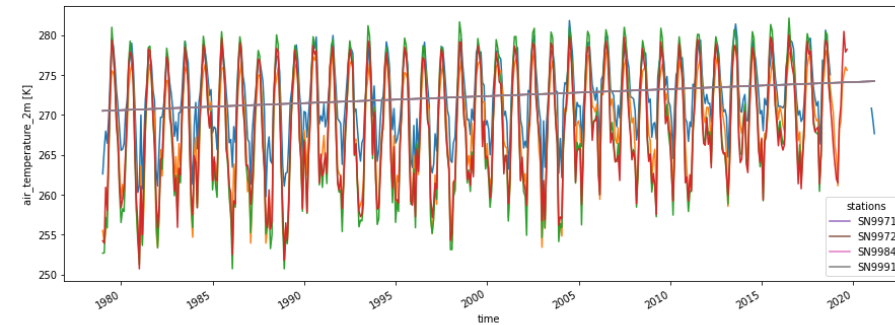
```
mydict = dict()
i = 0
for item in airtempmonthlymeans['stations'].values:
    print('Processing:', item)
    mytrend = estlincoef(airtempmonthlymeans['time'].values, airtempmonthlymeans[0].values)
    mydict[item] = xr.DataArray(mytrend['values'], dims=['time'], coords=[airtempmonthlymeans['time'].values], attrs={'temperature_2m', 'units': 'K'})
    i += 1
tmp = xr.concat([mydict[s] for s in mydict.keys()], dim='stations')
airtempmonthlymeanslintrends = tmp.assign_coords(stations=("stations", list(mydict.keys())))
```

Processing: SN99710
Processing: SN99720
Processing: SN99840
Processing: SN99910

Plot the monthly means and the linear trends for all the stations in one figure.

```
fig = plt.figure(figsize=(15,5))
airtempmonthlymeans.plot.line(x='time')
airtempmonthlymeanslintrends.plot.line(x='time')
```

```
[<matplotlib.lines.Line2D at 0x7f988876fdd8>,
 <matplotlib.lines.Line2D at 0x7f9888971358>,
 <matplotlib.lines.Line2D at 0x7f9888971360>,
 <matplotlib.lines.Line2D at 0x7f988897136d8>]
```



Avalanches-Svalbard

```
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
ax.set_ylabel('Avalanche Activity Density')
ax.set_xlabel('time')
ax.set_ylim(0, 750)
ax.grid(True)
ax.legend(loc='upper left')
ax.set_title(f'Winter season (yr104d)-(yr1104d):')

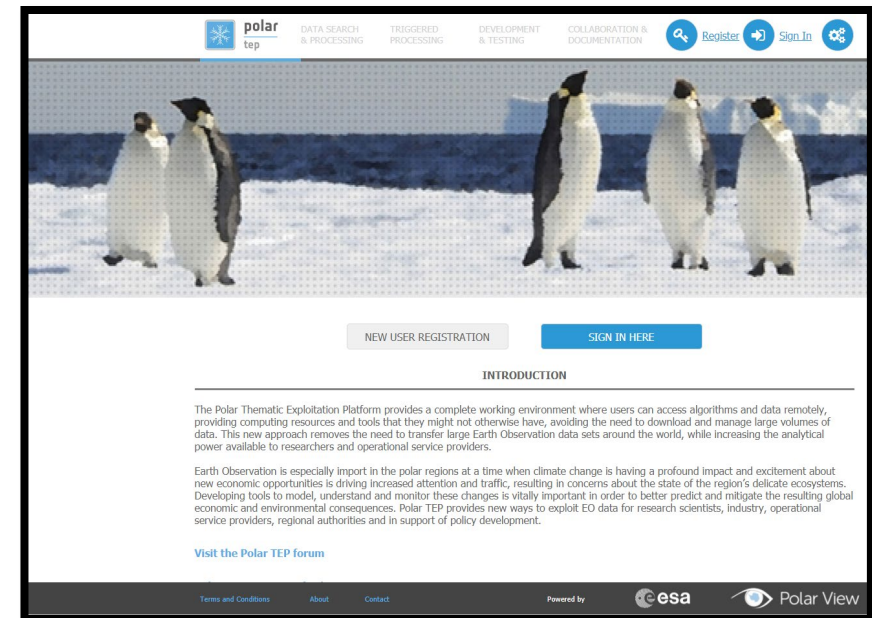
ax_twin = ax.twinx()
params[(params.index==t0) & (params.index==t1)][['snow_mean_lab'], ['rain_mean_lab']].plot(ax=ax_twin, color=['b', 'r'], alpha=0.5)
ax_twin.set_ylabel('Precipitation (kg/m²/day)')
ax_twin.legend(['Cold', 'Warm'], loc='upper right')
#ax.set_ylim(10, 50)
plt.suptitle(f'Region: {lab}')
plt.tight_layout()
```



Virtual machines



- CVL will utilize ESA's Polar Thematic Exploitation Platform (PTEP)
- PTEP will allow CVL users to have Virtual Machines where they can run Jupyter Notebooks
- <https://portal.polartep.io/>



Early adaptors call



- CVL will issue an early adopters open call (Sept 2021) where early carrier scientists at PhD/Msc –level can apply for grants
- 20 grants, 3000 €
- The grant receivers will be obliged to develop a Jupyter script that utilize CVL infrastructure/data/processing tools
- The Jupyter scripts will be stored on CVL GitHub for further utilization

Summary



- Cryosphere Virtual Laboratory - CVL (cvl.eo.esa.int) is an ESA project that provide access to ICT tools and data for easier access, processing and visualisation of cryosphere data

