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

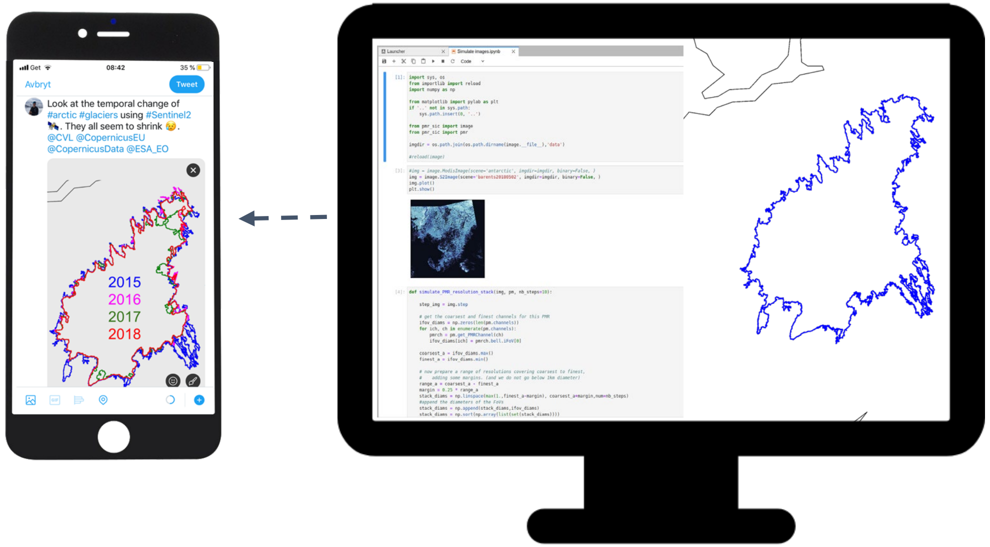
Mikhail, NPI

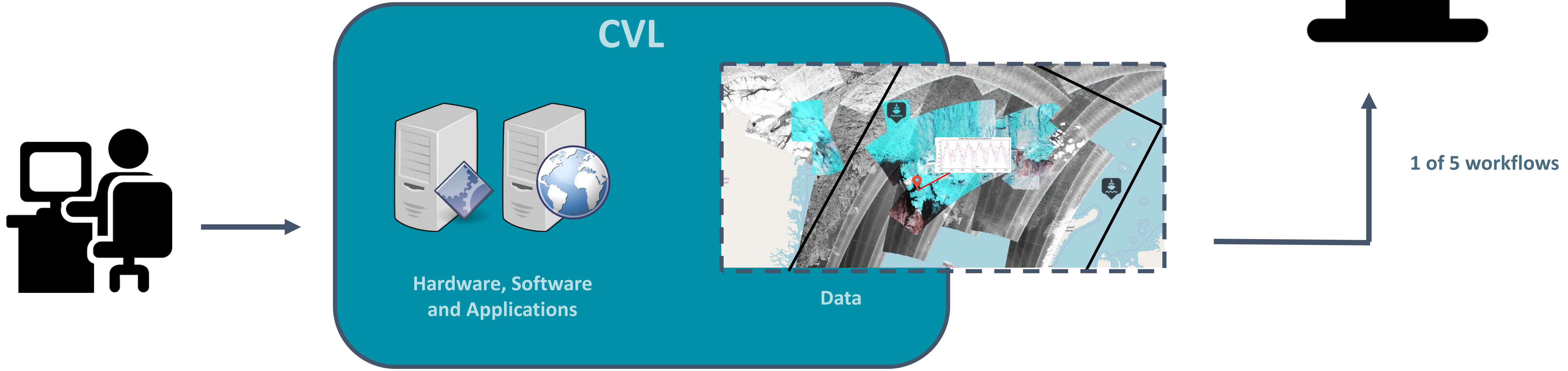
Daniel, NORCE

Robin, S&T

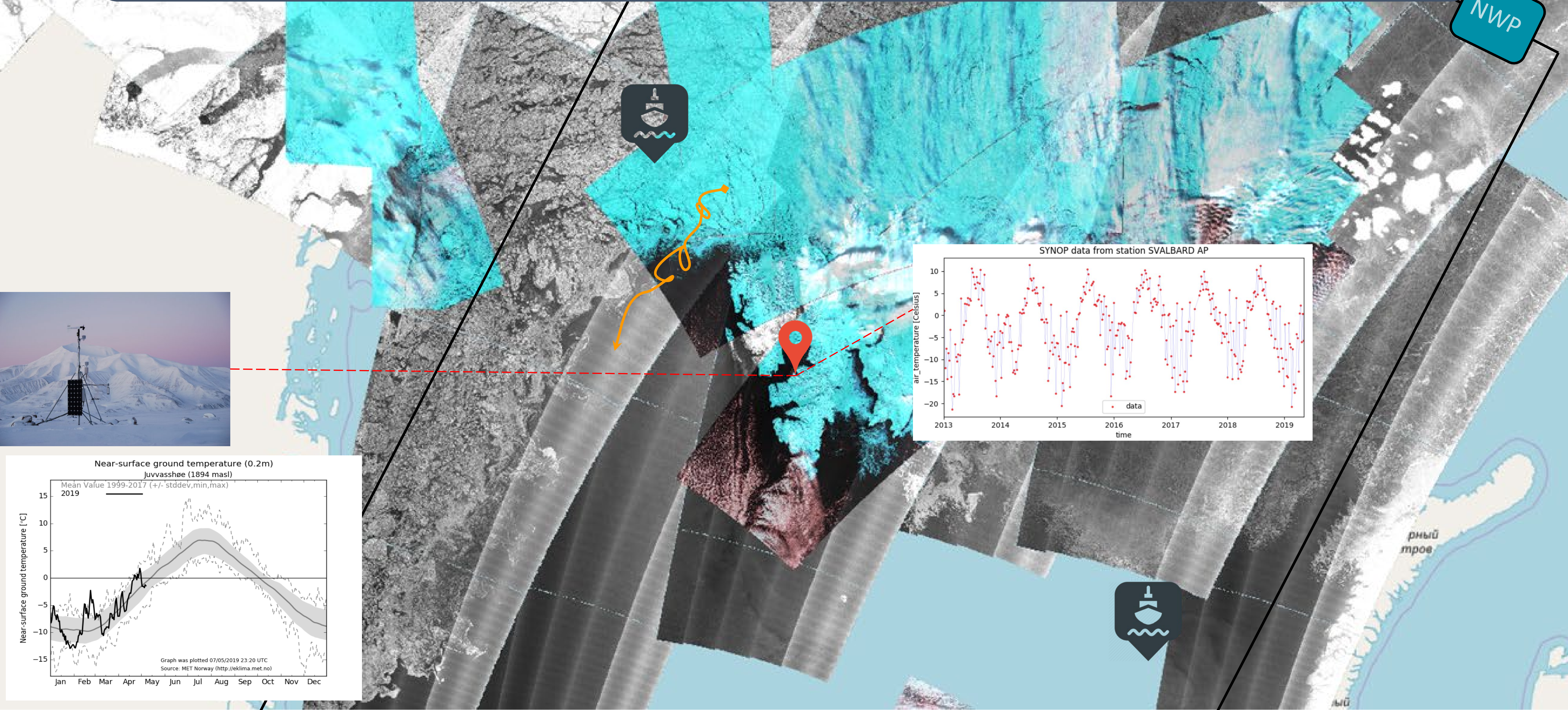
CVL as an Open Science Component

- 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





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.”*



What about the data

Challenges

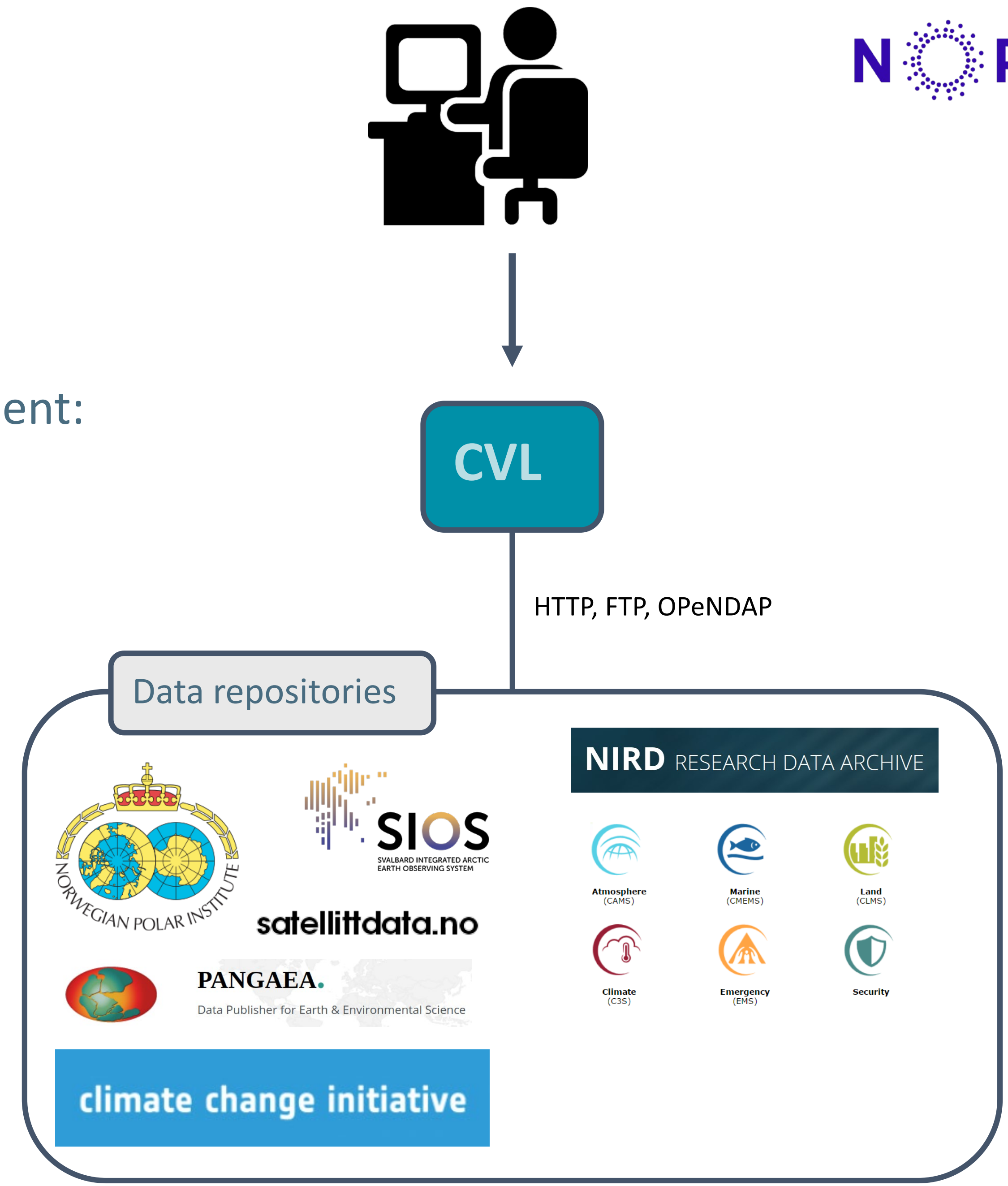
variety, velocity, volume, heterogeneity, access -> Big Data

FAIR guiding principles of Data Management:

- **F**indable
- **A**ccessible
- **I**nteroperable
- **R**eusable

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

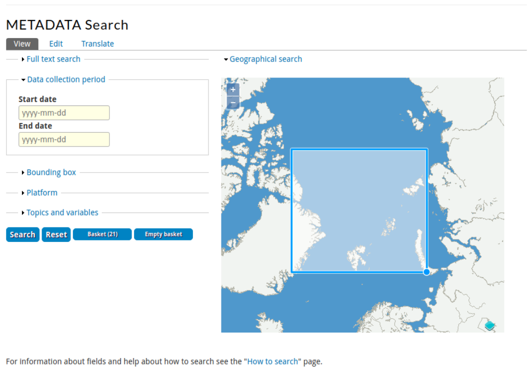
Search for Data

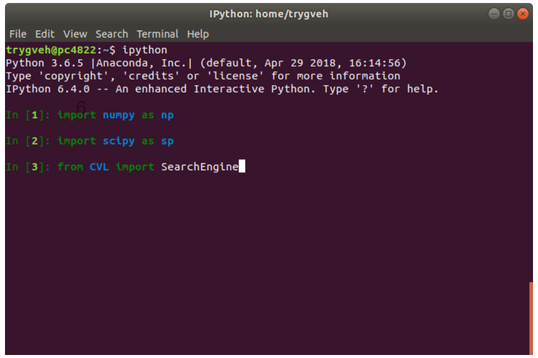
Visualizing data

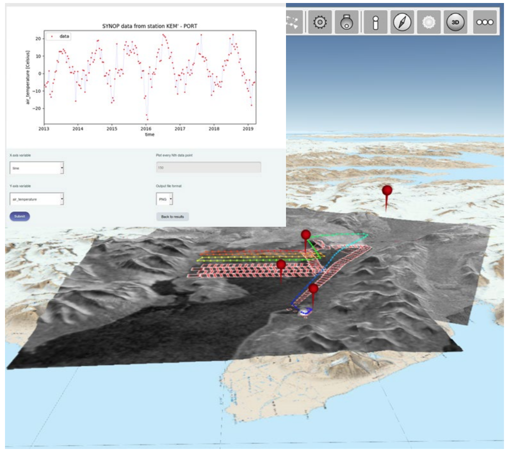
Visualizing / Publishing results

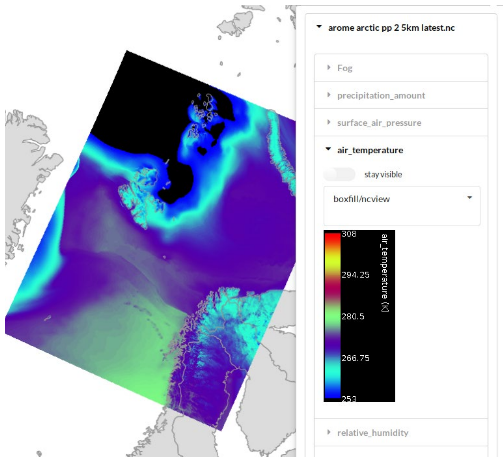
Interactive analysis on hosted VM

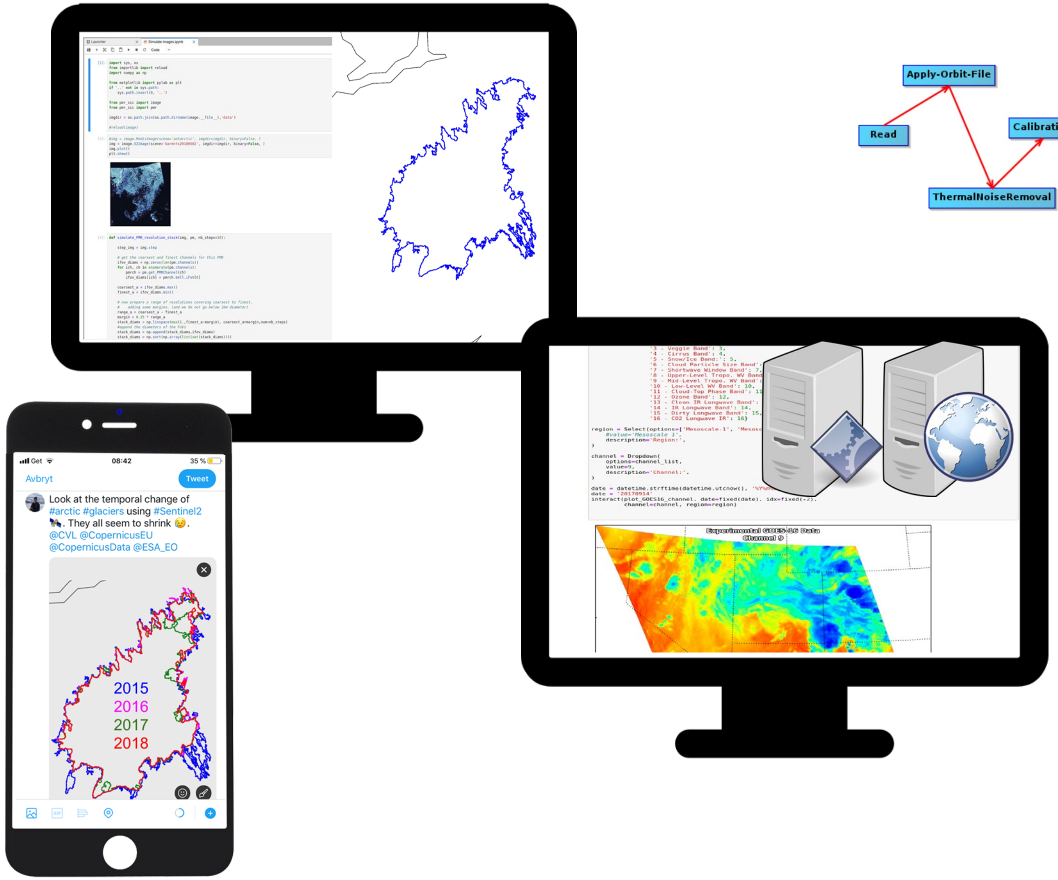
Processing chain

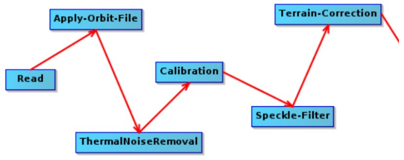








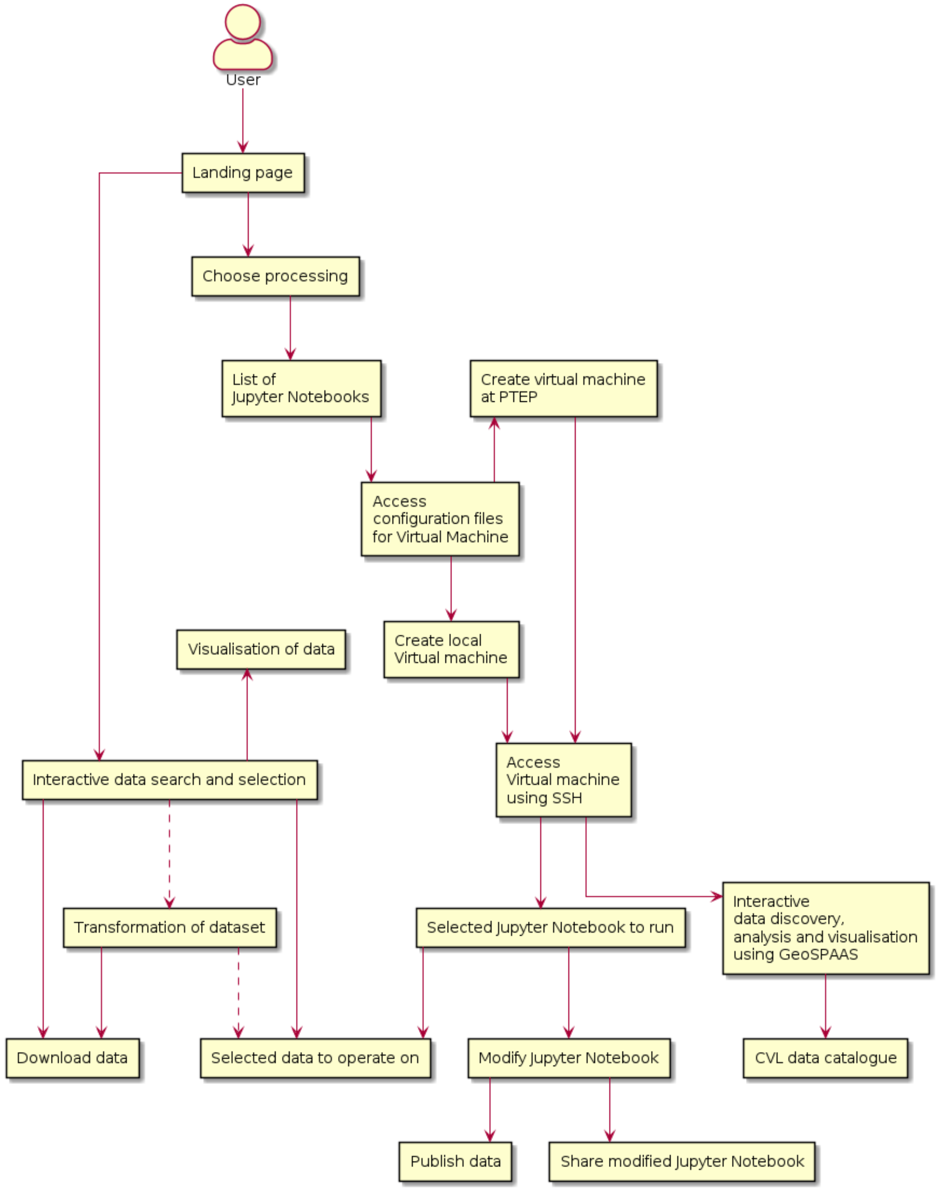




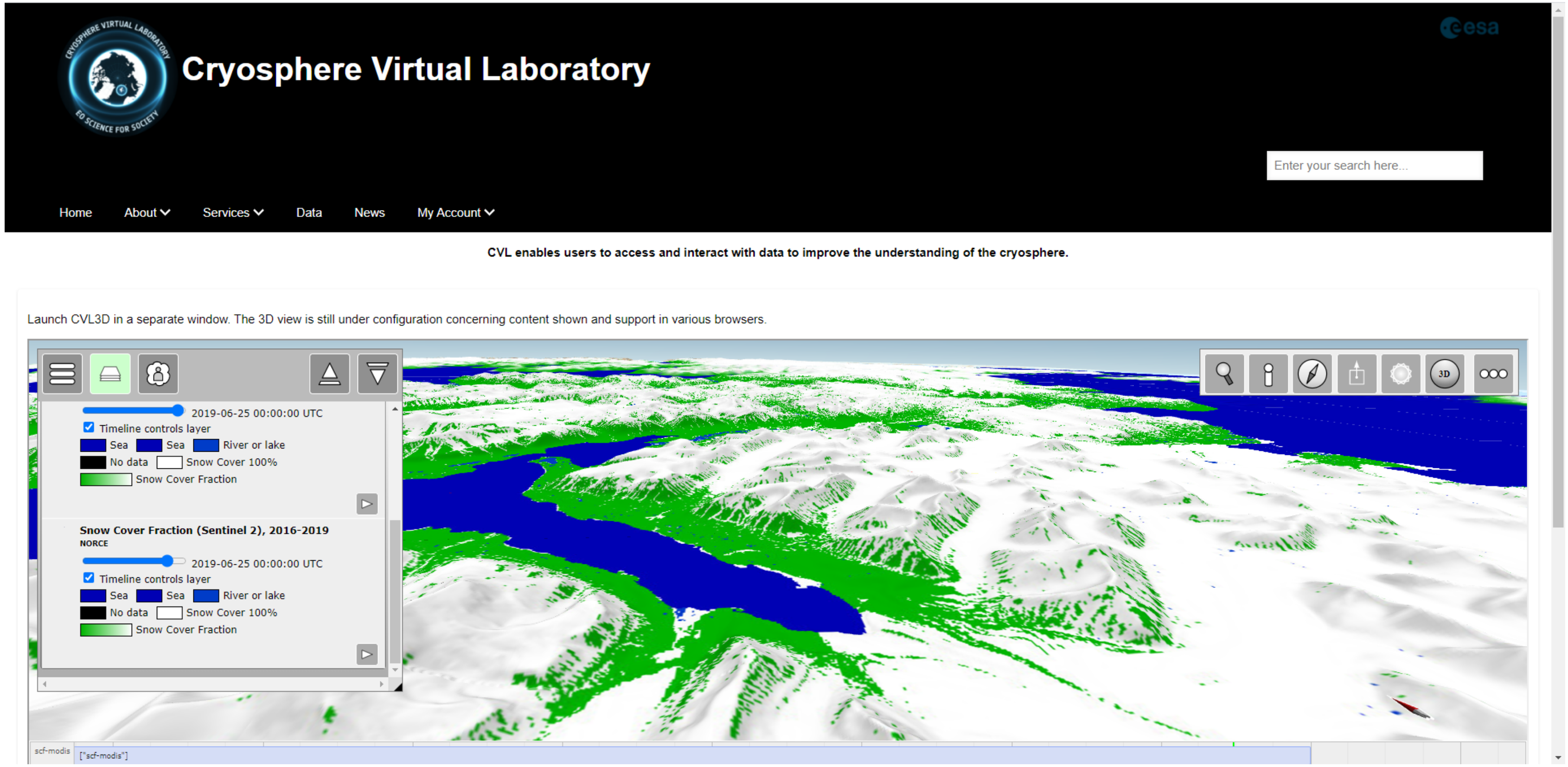
Work flows

- 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



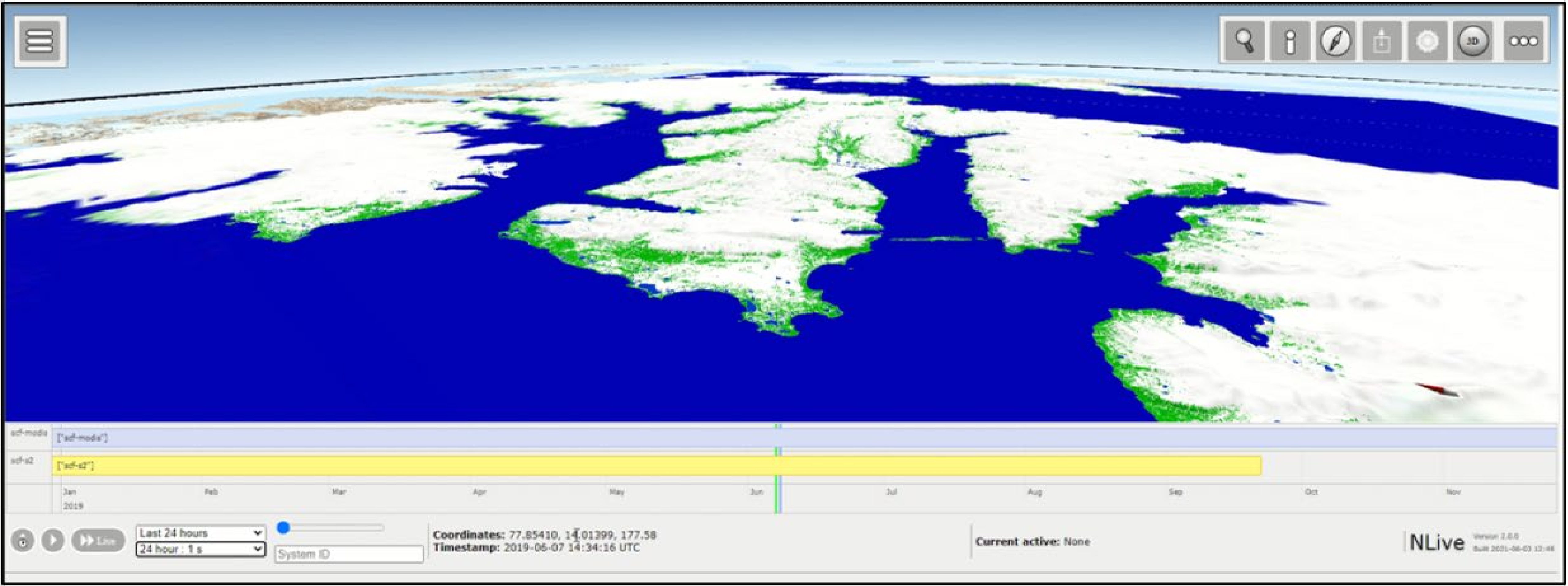
Visualization



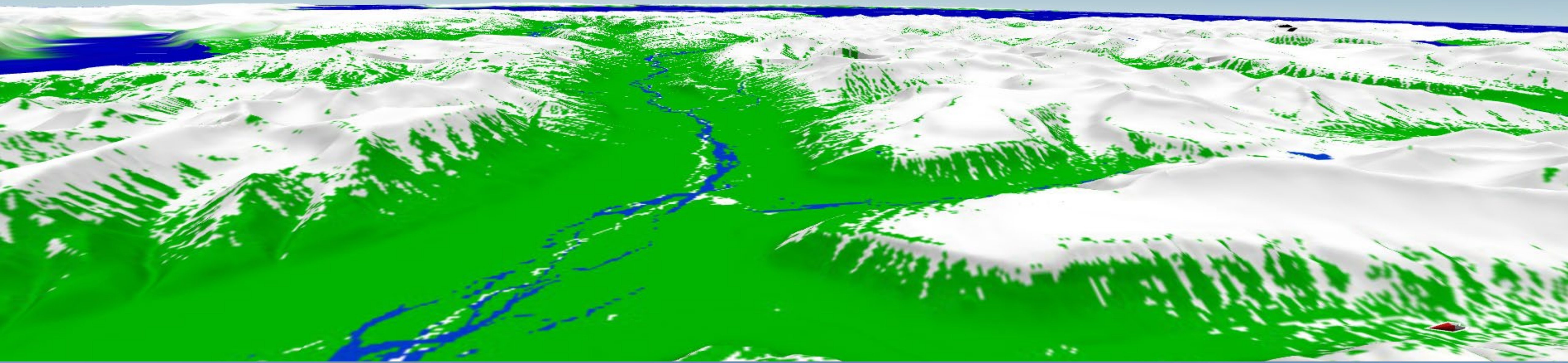
- CVL have implemented a dedicated 3D visualisation tool where examples are available on the Landing page cvl.eo.esa.int
- 3Dviz can be embedded in Jupyter notebooks for vizualization of results

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

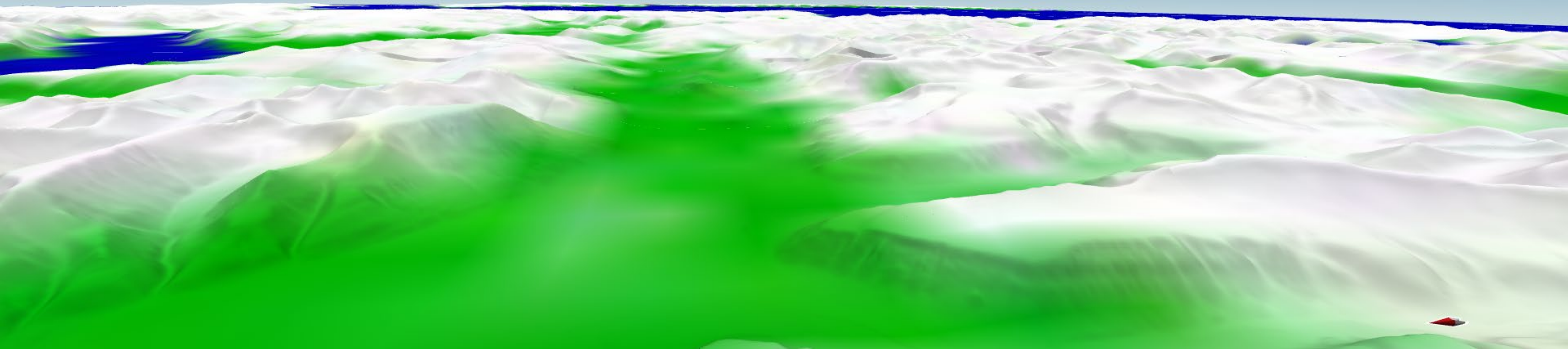
NORCE



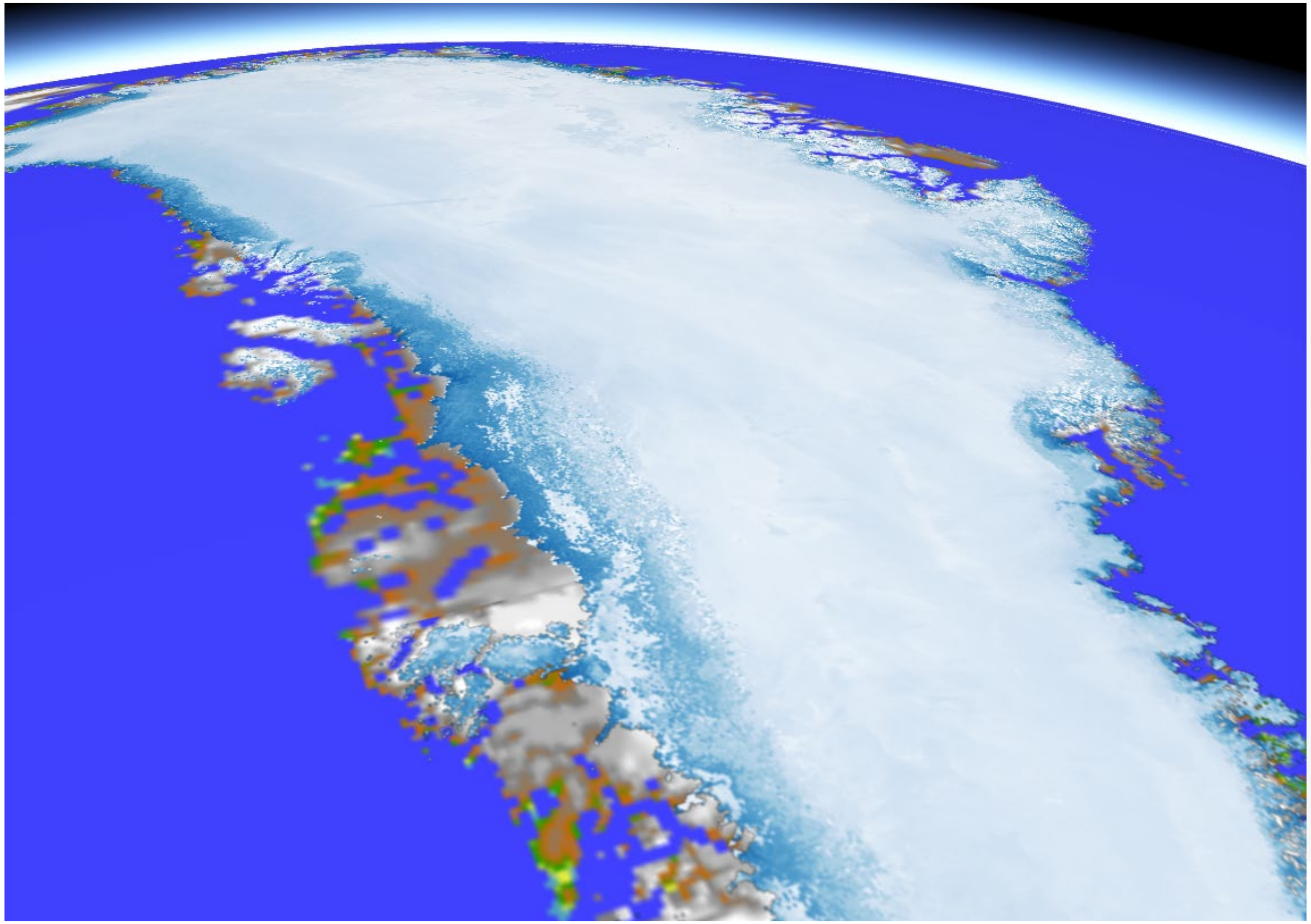
S2 20180618



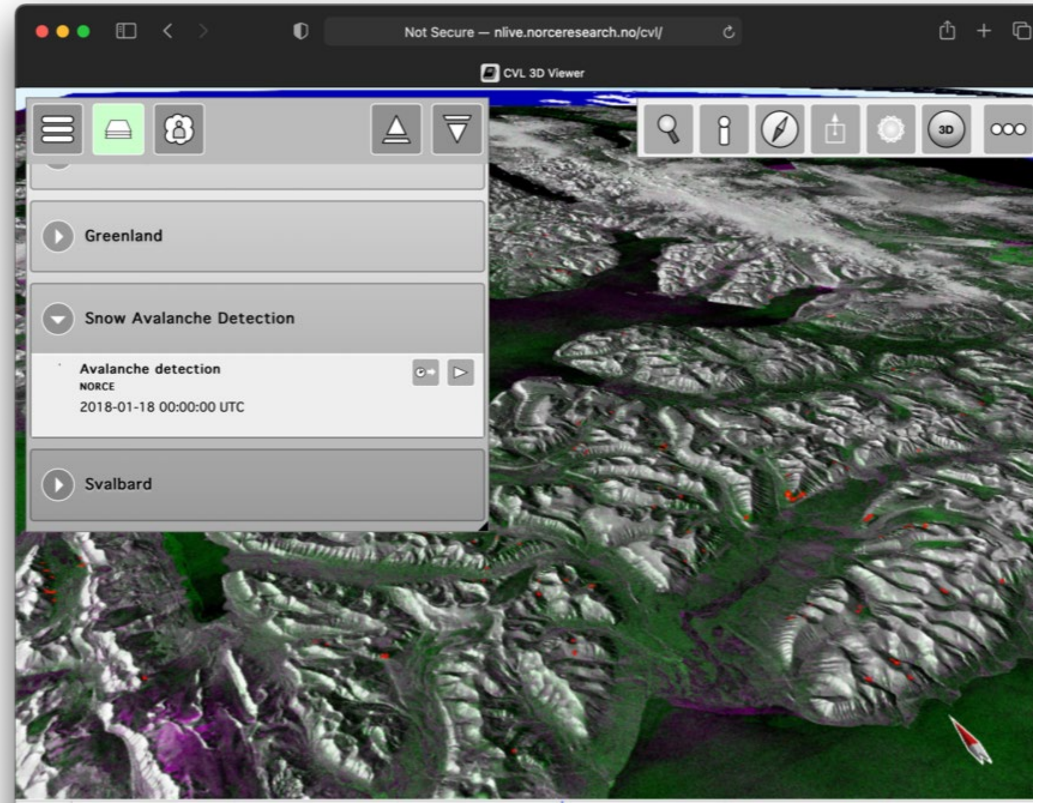
MODIS 20180618



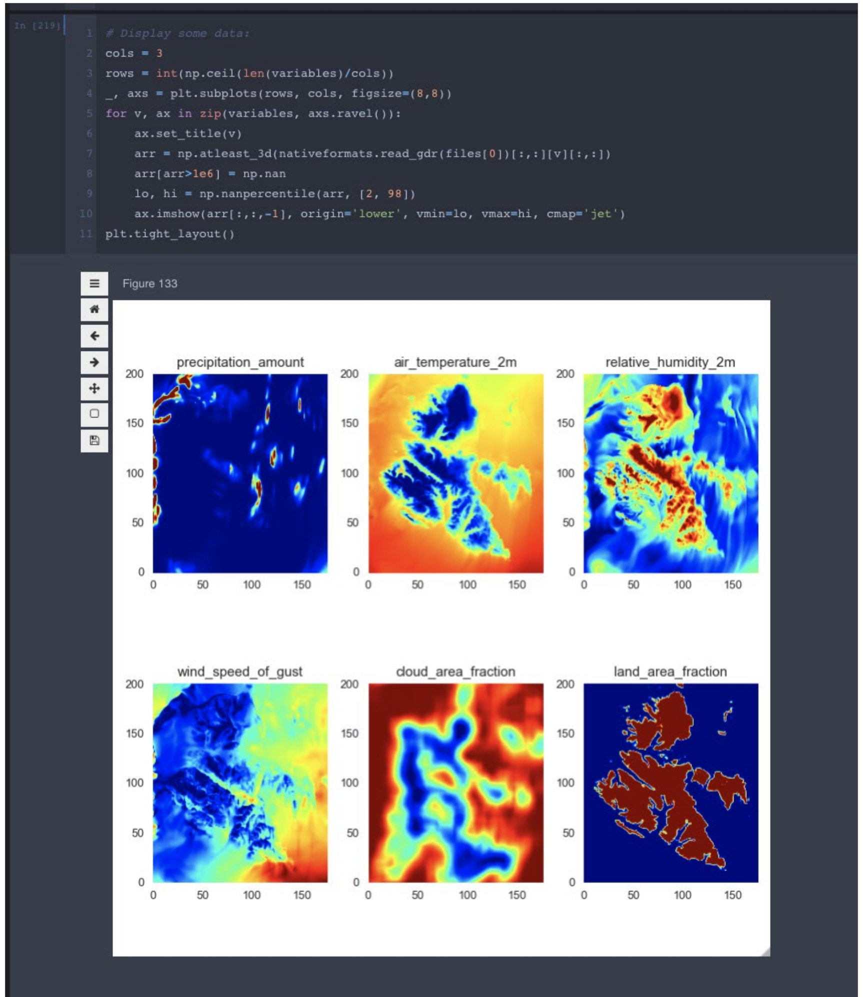
Ex. Albedo on Greenland from GEUS

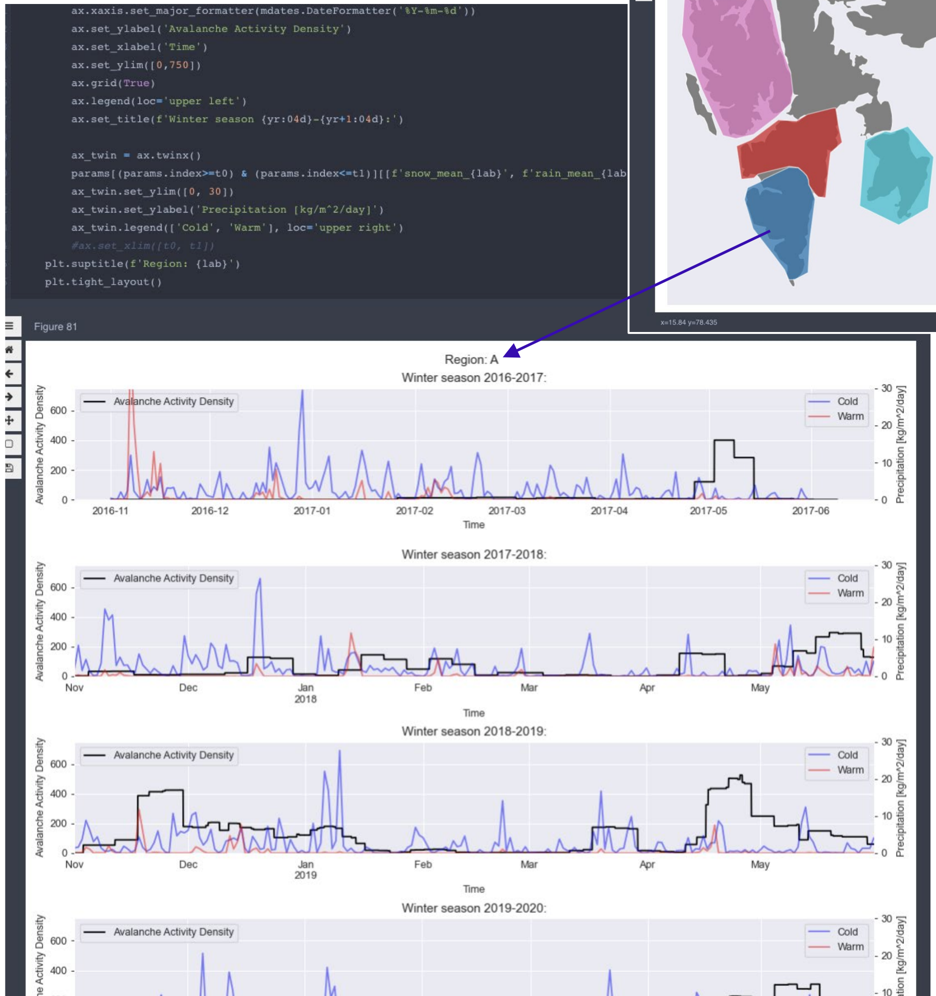


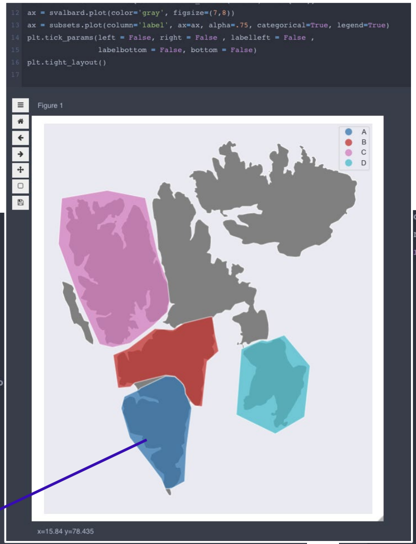
Ex. Avalanche detection on Svalbard

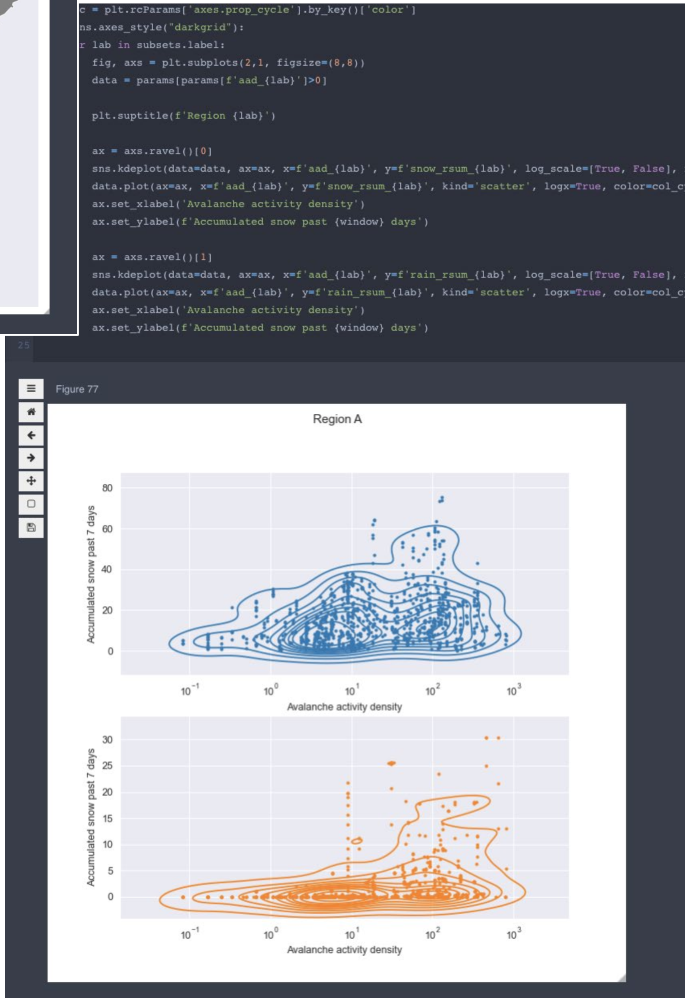


Ex. Avalanche detection on Svalbard





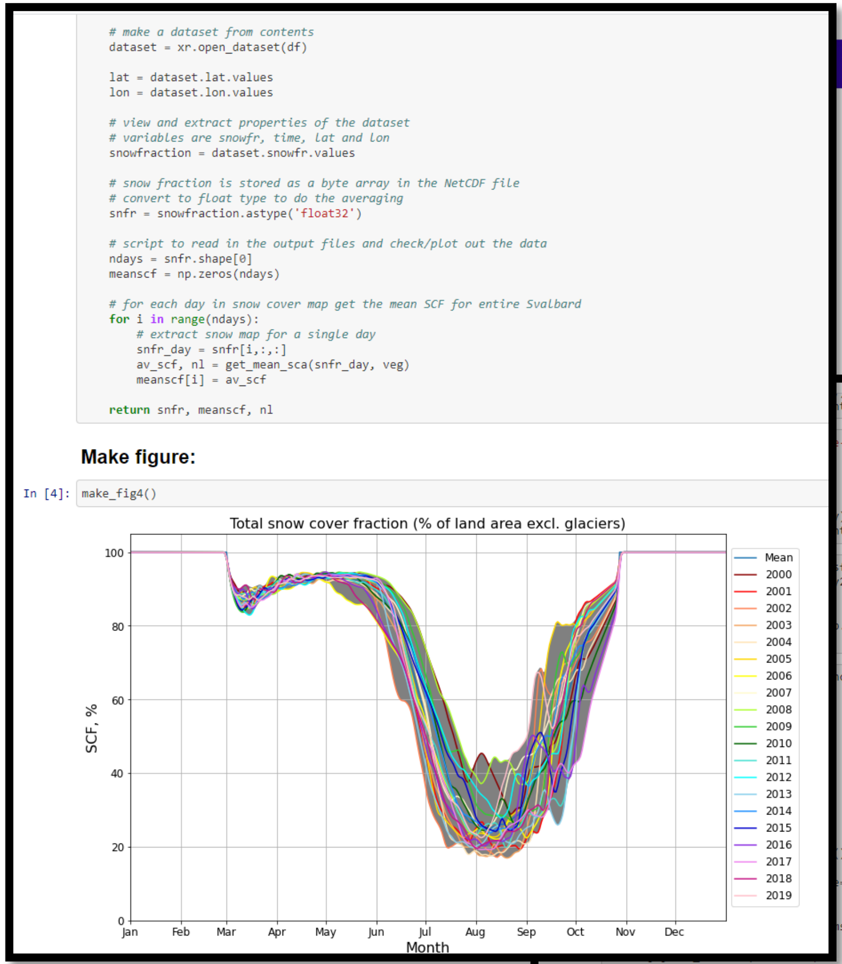




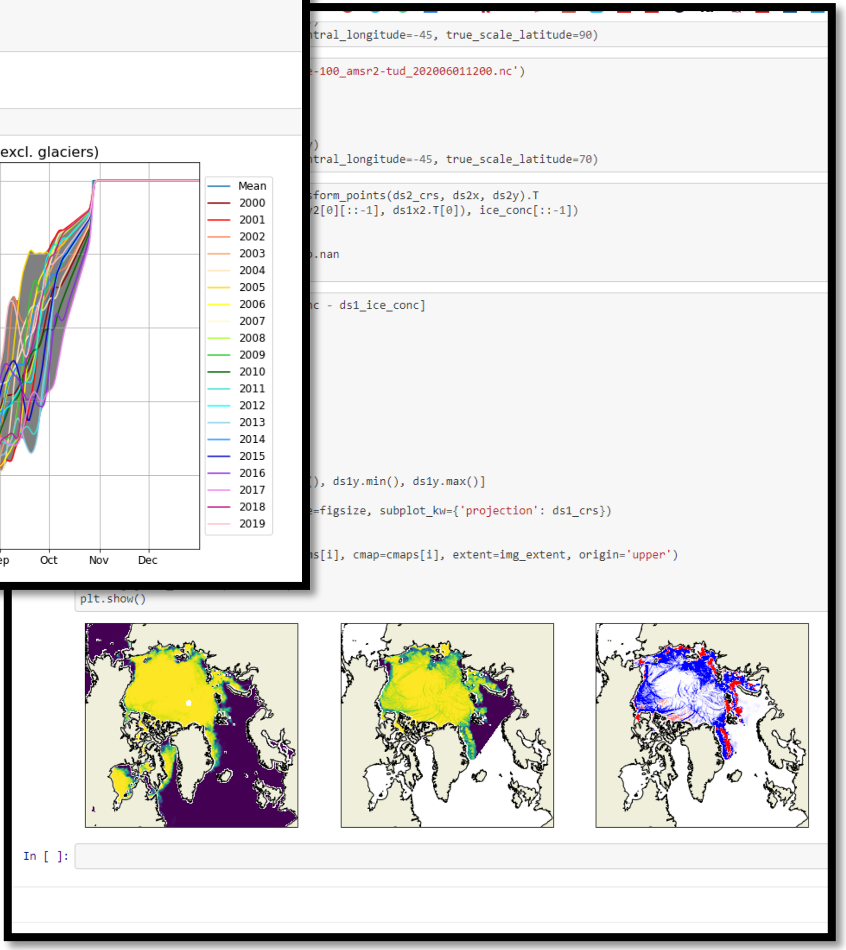
Jupyter notebooks

- 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

Snow-Svalbard



Ice zone-validation

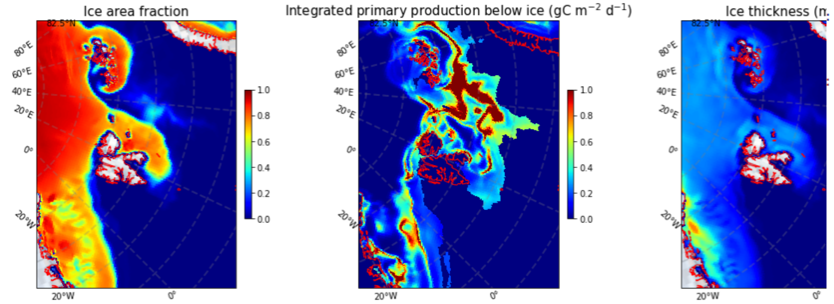


Marginal ice zone –biology

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

```
aice  = 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 dictionioray for properties of plots
plotted = {}
plotted[0] = {'data':aice, 'max':1., 'title':'Ice area fraction'}
plotted[2] = {'data':thick, 'max':5., 'title':'Ice thickness (m)'}
plotted[1] = {'data':primary_production, 'max':1., 'title':r'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'])
    # drowing 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.

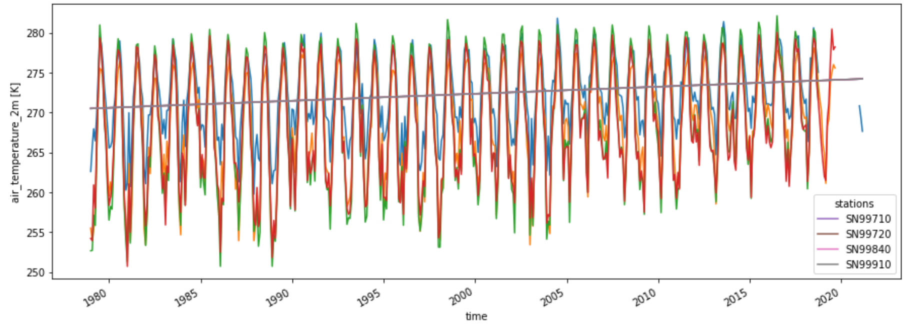
```
mydict = dict()
i = 0
for item in airtempmonthlymeans['stations'].values:
    print('Processing:', item)
    mytrend = estlincoef(airtempmonthlymeans['time'].values,airtempmonthlymeans[0].values)
    mydict[item] = xr.DataArray(mytrend['values'],dims=['time'],coords=[airtempmonthlymeans['time'].values],attrs={'s
emperature_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 0x7f9889713588>,
 <matplotlib.lines.Line2D at 0x7f9889713630>,
 <matplotlib.lines.Line2D at 0x7f98897136d8>]
```



Avalanches-Svalbard

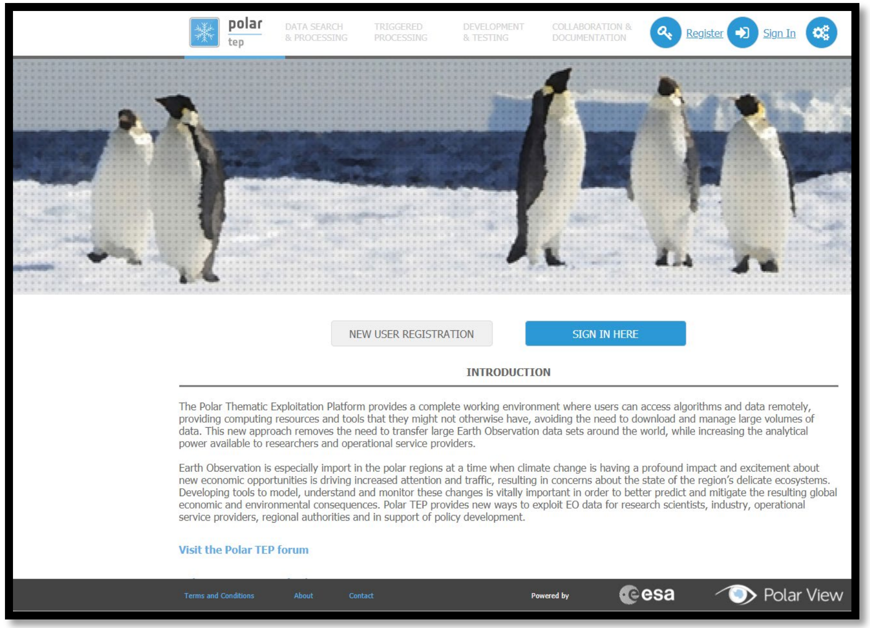
```
        ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
        ax.set_ylabel('Avalanche Activity Density')
        ax.set_xlabel('Time')
        ax.set_ylim([0,750])
        ax.grid(True)
        ax.legend(loc='upper left')
        ax.set_title(f'Winter season {yr:04d}-{yr+1:04d}:')

        ax_twin = ax.twinx()
        params[(params.index>=t0) & (params.index<=t1)][[f'snow_mean_{lab}', f'rain_mean_{lab}']].plot(ax=ax_twin, color=['b', 'r'], alpha=
        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_xlim([t0, t1])
    plt.suptitle(f'Region: {lab}')
    plt.tight_layout()
```

Figure 81

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