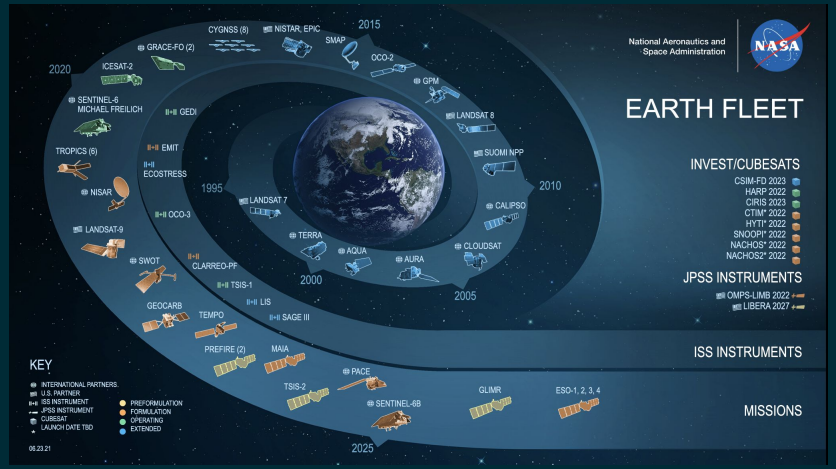
Processing EO Satellite Data with

Xin Zhang

What is Pytroll?



Reading

```
>>> from satpy import available_readers

# list all available readers in Satpy
>>> sorted(available_readers())
```

ABI L1B/L2	AHI HSD/HRIT	AGRI L1	AMI L1B	GLM L2
AVHRR L1B	SEVIRI L1B	CALIPO L2	GEOCAT	GPM IMERG
Sentinel-1 A/B SAR-CS	Sentinel 2A/B MSI	Sentinel 3 A/B SLSTR	**TROPOMI L2**	VIIRS

Full list at https://satpy.readthedocs.io/en/stable/index.html#reader-table

Reading

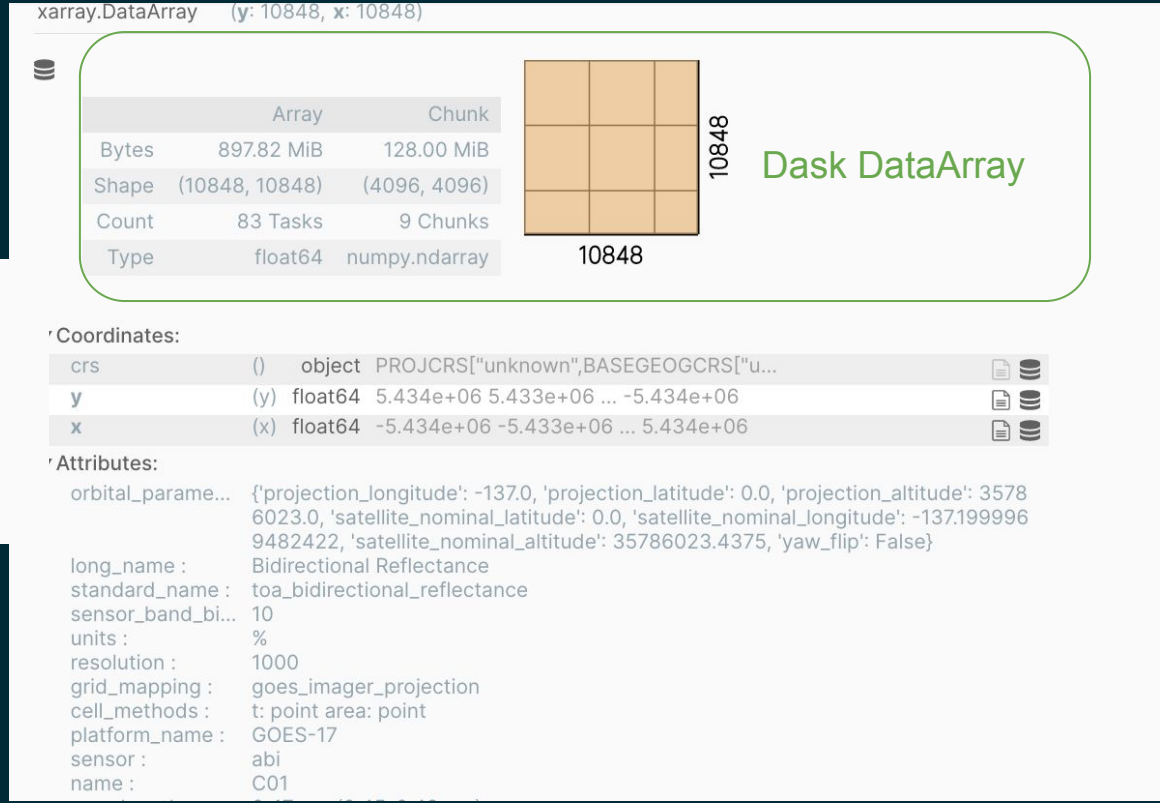
GEO

```
>>> from glob import glob
>>> from satpy import Scene

>>> f_abi = glob('../data/abi/OR_ABI-L1b*s20221822100*.nc')
>>> scn_abi = Scene(f_abi, reader='abi_l1b')

>>> scn_abi.available_dataset_names()
    ['C01', 'C02', …, 'C15', 'C16']
```

Reading

GEO

```
# load one channel data
>>> scn_abi.load(['C01'])

>>> scn_abi['C01']
```



Reading

GEO

```
# access attributes
>>> scn_abi['C01'].attrs['area']
    Area ID: GOES-West
    Description: 1km at nadir
    Projection ID: abi_fixed_grid
    Projection: {'ellps': 'GRS80', 'h': '35786023', 'lon_0': '-137', 'no_defs': 'None', 'proj': 'geos',
    'sweep': 'x', 'type': 'crs', 'units': 'm', 'x_0': '0', 'y_0': '0'}
    Number of columns: 10848
    Number of rows: 10848
    Area extent: (-5434894.8851, -5434894.8851, 5434894.8851, 5434894.8851)

# get lon and lat
>>> lons, lats = scn_abi['C01'].attrs['area'].get_lonlats()
```

Reading

LEO

```
>>> f_tropomi = glob('../data/tropomi/S5P*L2__NO2____20220701T1226*')

>>> scn_tropomi = Scene(f_tropomi, reader='tropomi_l2')
>>> scn_tropomi.load(['assembled_lon_bounds', 'assembled_lat_bounds',
'cloud_radiance_fraction_nitrogendioxide_window']
```

Satpy

v. s.

xarray

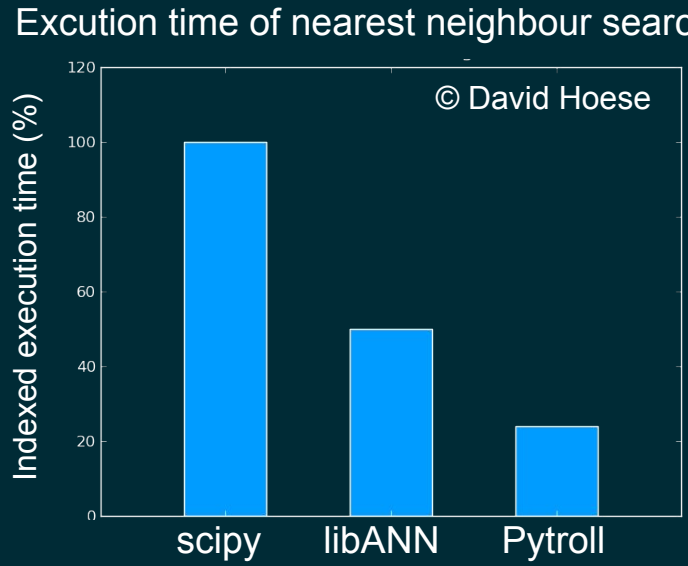
```
>>> ds_detail = xr.open_dataset(f_tropomi, group='PRODUCT/SUPPORT_DATA/DETAILED_RESULTS/')
>>> ds_geo = xr.open_dataset(f_tropomi, group='PRODUCT/SUPPORT_DATA/GEOLOCATIONS/')

>>> ds['cloud_radiance_fraction_nitrogendioxide_window'], ds_geo['longitude_bounds']
```

Processing

Dask for parallel computations

- Lazy loading/processing
- Chunked parallel processing

  -> Scale computations to multiple cores



Resampling

Pyresample

- Unprojected & Grid
- Proj.4, more than 130 different projections

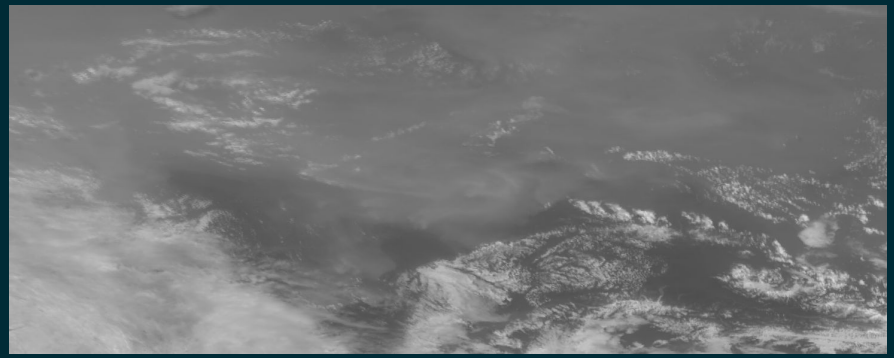
Resampler	Description	Related
nearest	Nearest Neighbor	`KDTreeResampler`
ewa	Elliptical Weighted Averaging	`DaskEWAResampler`
ewa_legacy	Elliptical Weighted Averaging (Legacy)	`LegacyDaskEWAResampler`
native	Native	`NativeResampler`
bilinear	Bilinear	`BilinearResampler`
bucket_avg	Average Bucket Resampling	`BucketAvg`
bucket_sum	Sum Bucket Resampling	`BucketSum`
bucket_count	Count Bucket Resampling	`BucketCount`
bucket_fraction	Fraction Bucket Resampling	`BucketFraction`
gradient_search	Gradient Search Resampling	`GradientSearchResampler`

Available Resampling Algorithms
(More resampling usage: https://pyresample.readthedocs.io/)

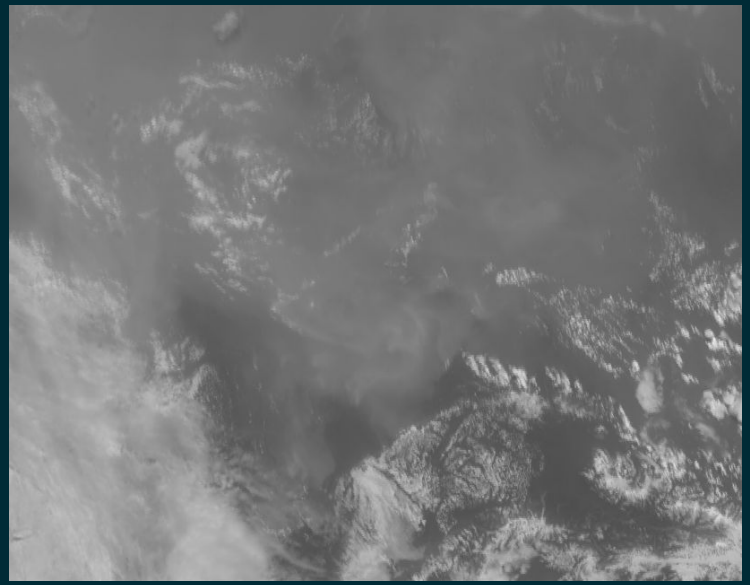
Resampling

Pyresample

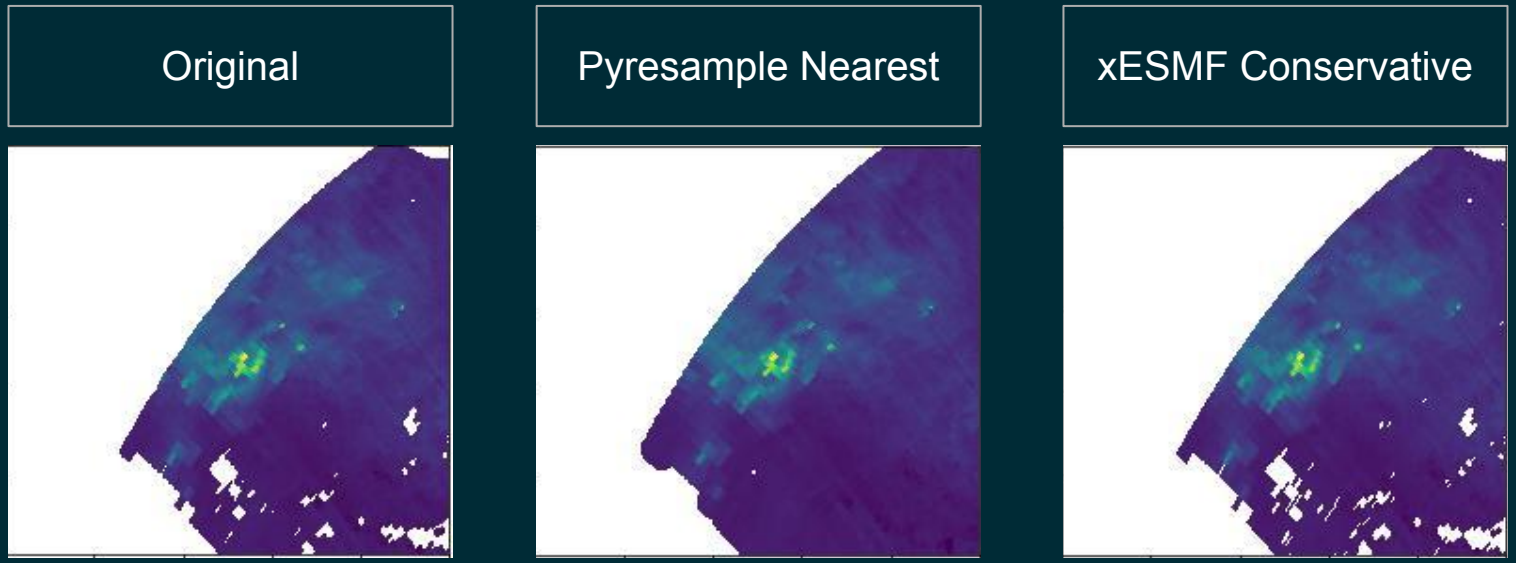
WGS84 0.1 deg x 0.1 deg



Lambert 1 km x 1 km



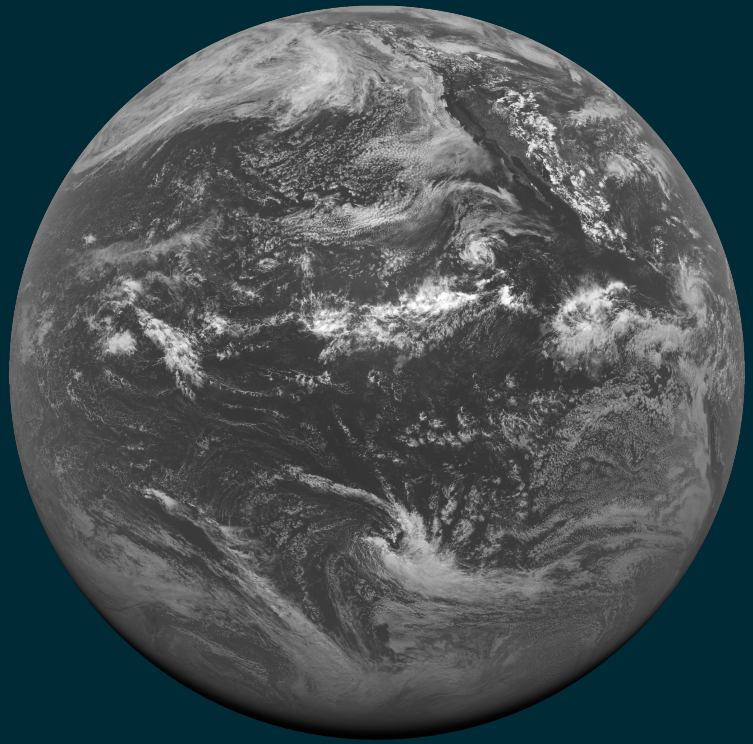
Resampling



TROPOMI L2 CO Total Column

Add conservative resampler to pyresample: https://github.com/pytroll/pyresample/issues/440

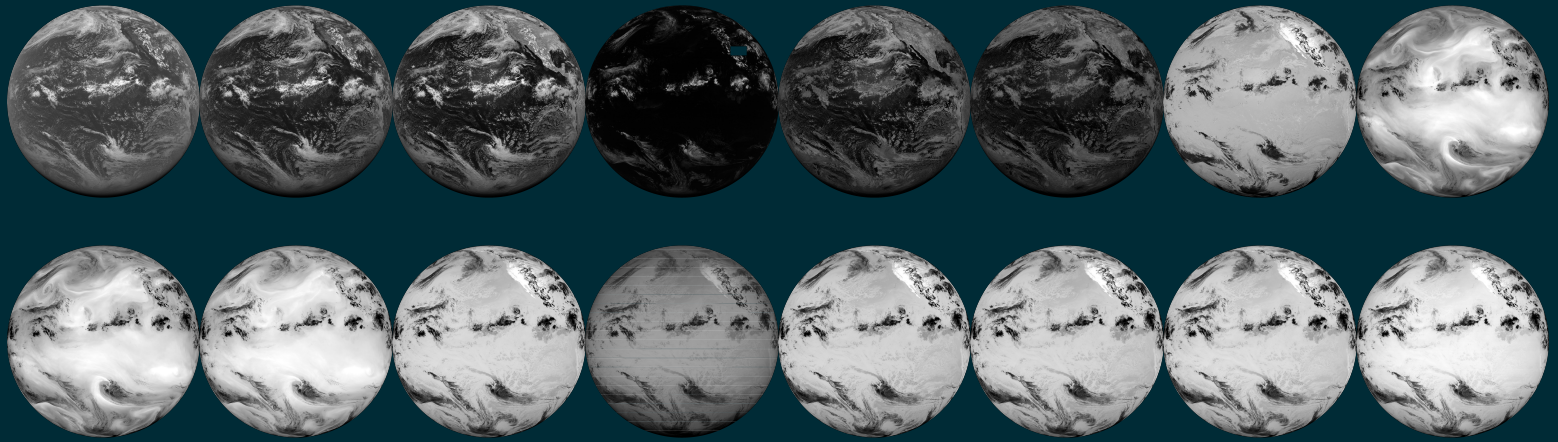
Qucikview



```
>>> scn.show('C01')
```

Data source: GOES-17 ABI

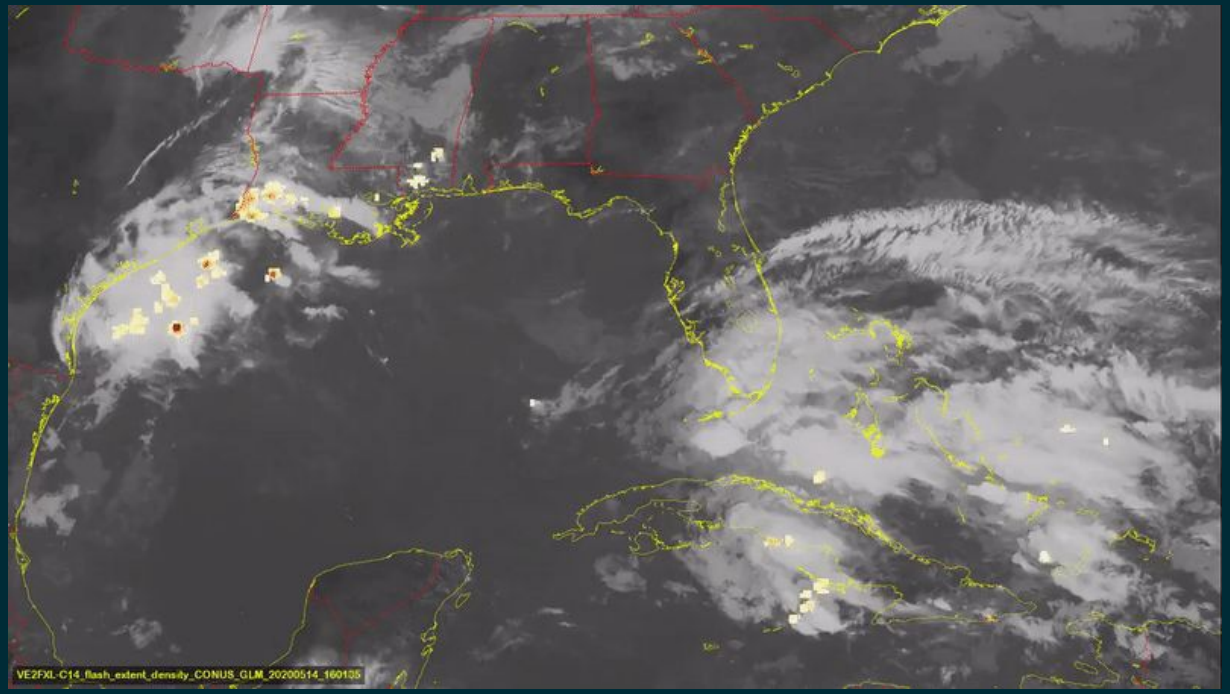
Qucikview



Quickview of all channels
(C01 ~ C16)

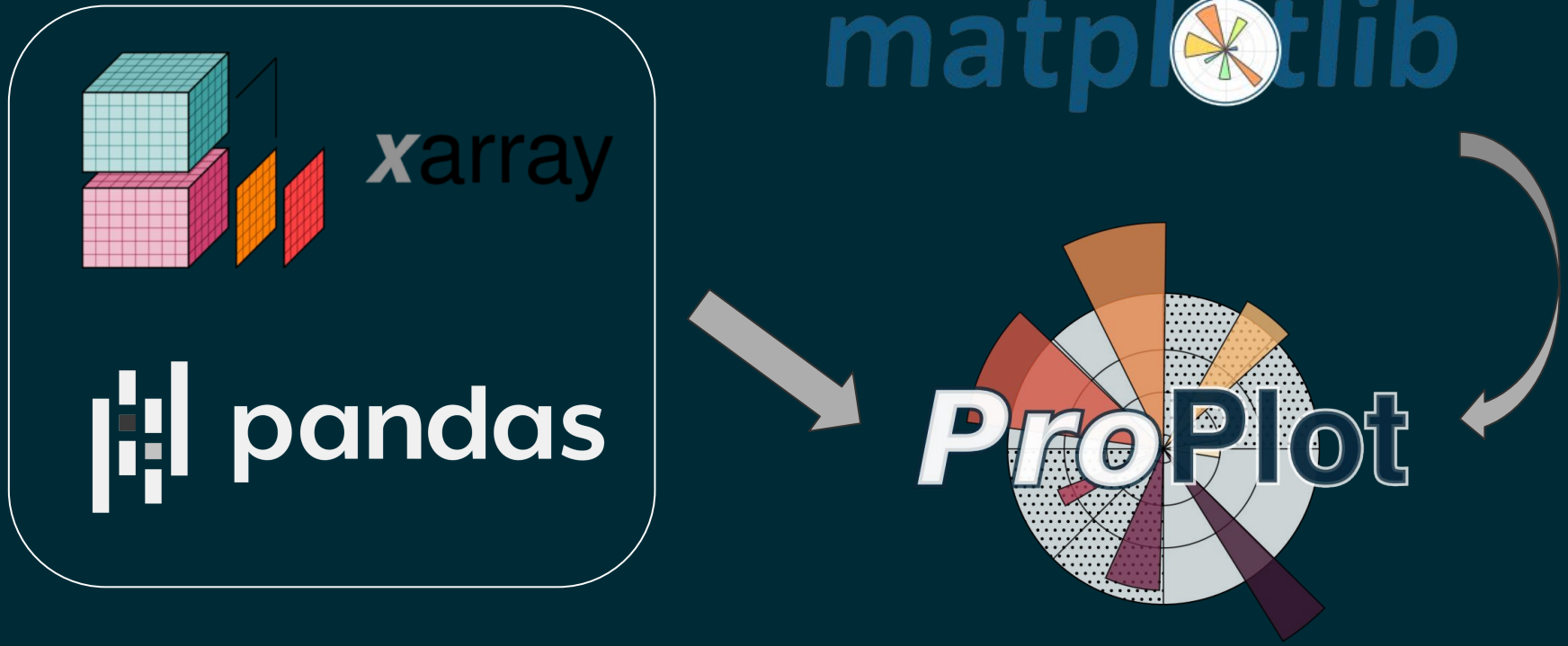
Data source: GOES-17 ABI

Qucikview



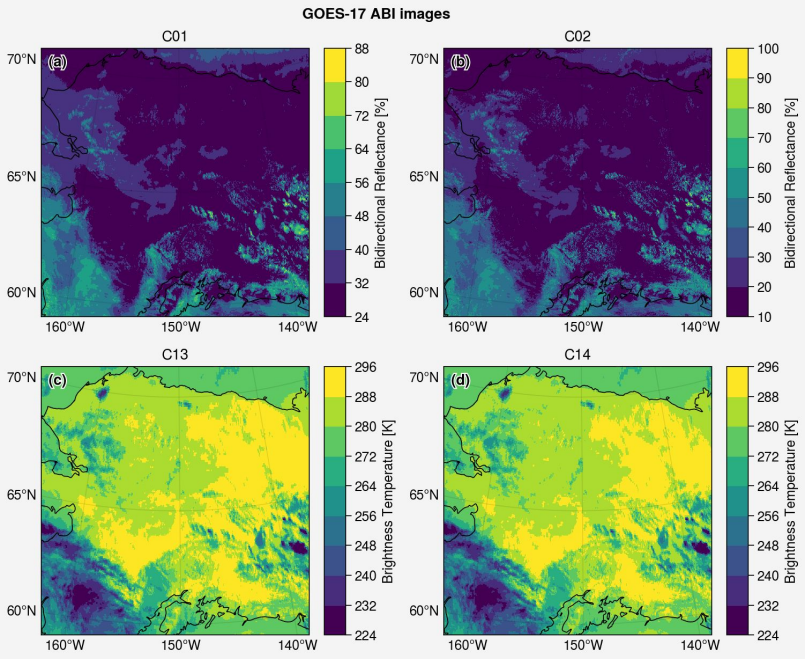
Data source: GOES-16 ABI + GLM

Scientific Figure



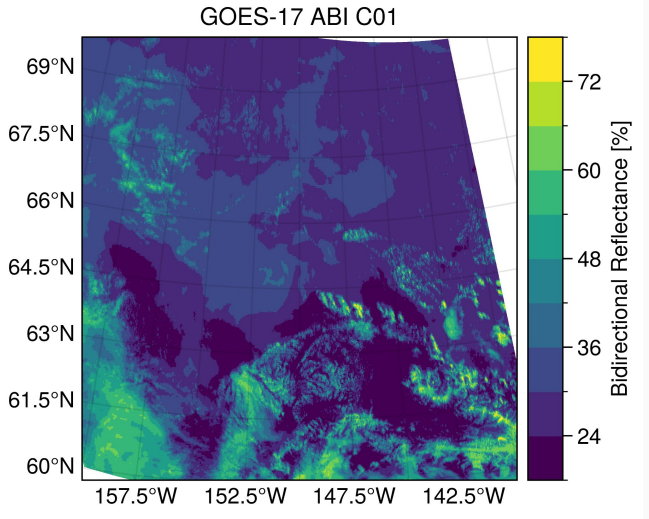
Scientific Figure

```
import proplot as pplt

crs = scn_resample['C01'].attrs['area'].to_cartopy_crs()

fig, axs = pplt.subplots(proj=crs, nrows=2, ncols=2)

......

>>> axs.format(reso='med', coast=True,
               latlines=5, lonlines=10, labels=True,
               title=['C01', 'C02', 'C13', 'C14'],
               suptitle='GOES-17 ABI images',
               abc='(a)', abcloc='ul',)
```



Scientific Figure

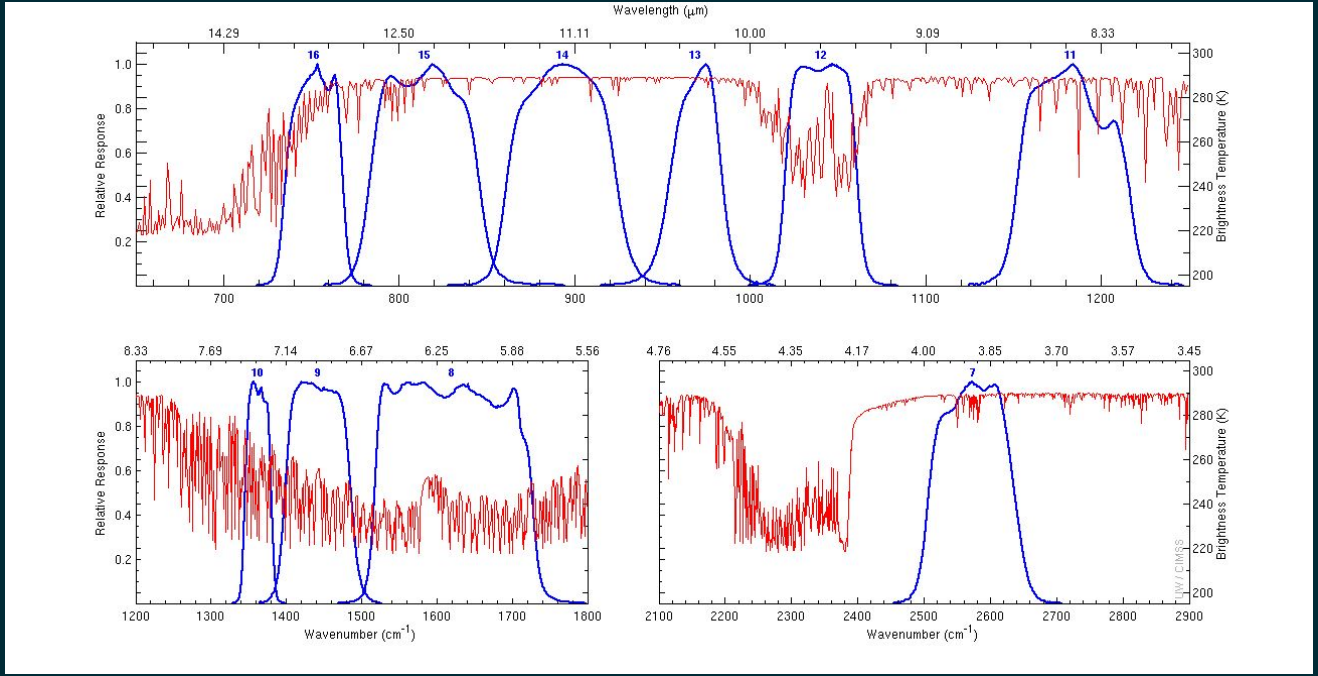
```
import cartopy.crs as ccrs

# crop data to lon/lat region
>>> scn_crop = scn.crop(ll_bbox=(-160, 60, -140, 70))
crs_crop = scn_crop['C01'].attrs['area'].to_cartopy_crs()
```




```
# set plotting projection
crs_plot = ccrs.LambertAzimuthalEqualArea(central_latitude=65, central_longitude=-150)
fig, axs = pplt.subplots(proj=crs_plot)
scn_crop['C01'].plot(ax=axs, transform=crs_crop)

axs.format(lonlim=(-160, -140), latlim=(60, 70), labels=True, title='GOES-17 ABI C01')
```

Composites

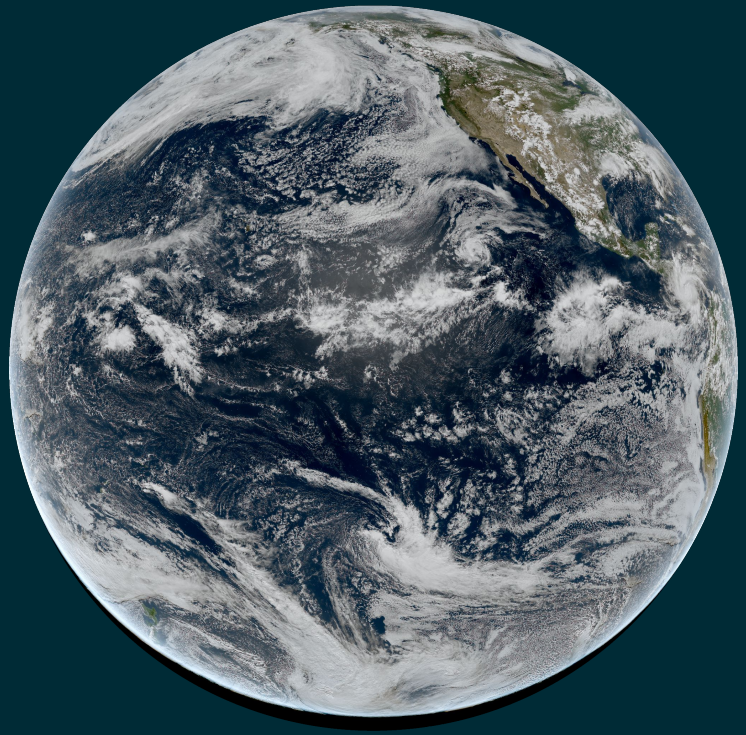
Spectral Response Functions
(SRF) of GOES-17 ABI



Composites

Pyspectral

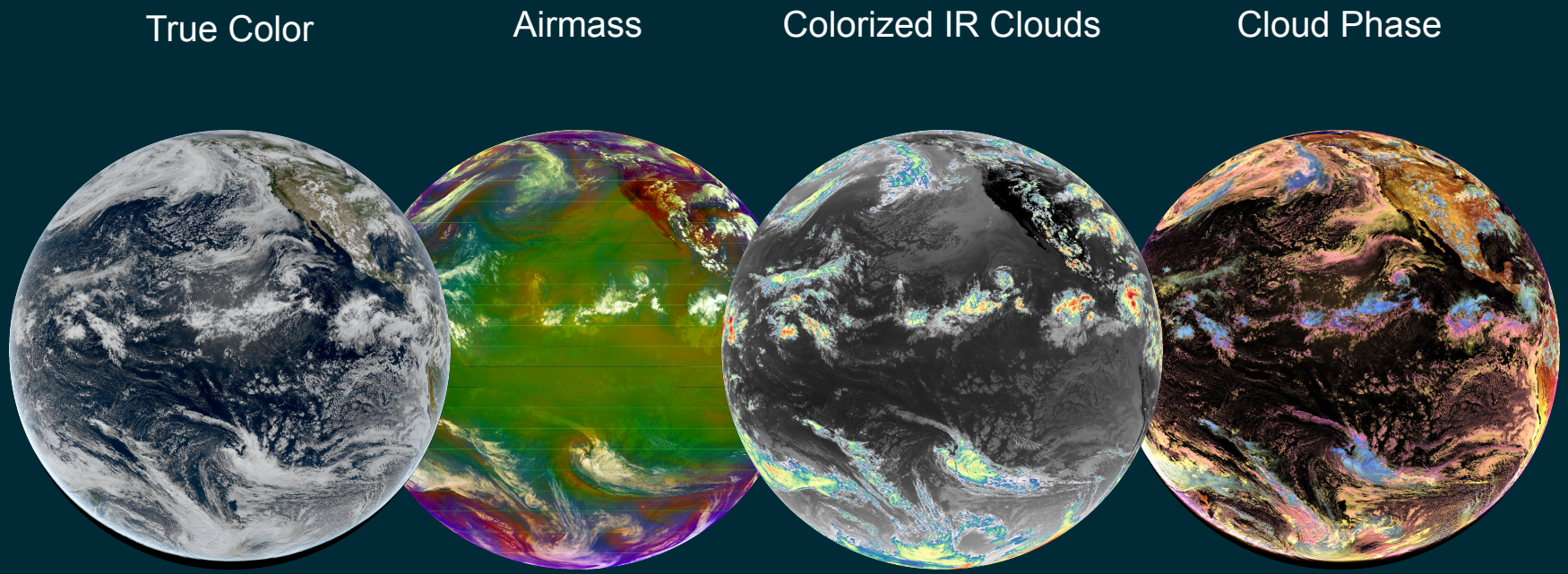
Corrections
to
Channels



```
scn.show('true_color_nocorr')
```

```
scn.show('true_color')
```

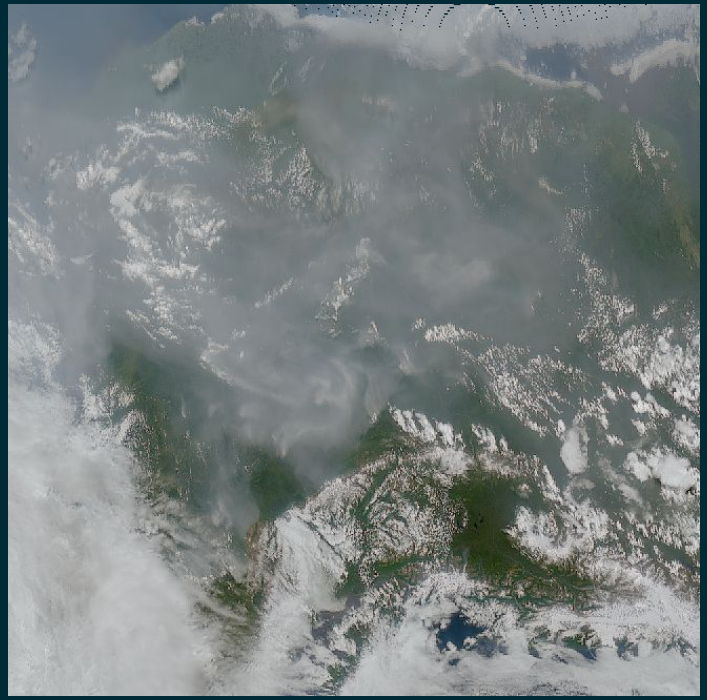
Composites



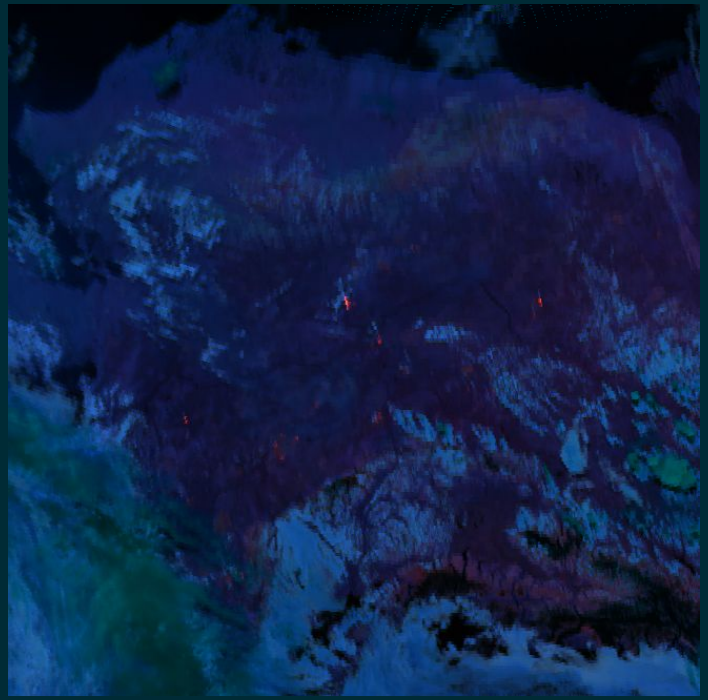
Data source: GOES-17 ABI

Composites

True Color



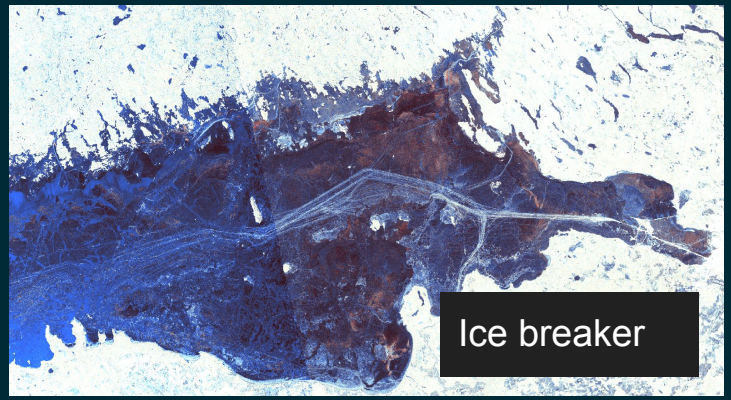
Fire Temperature

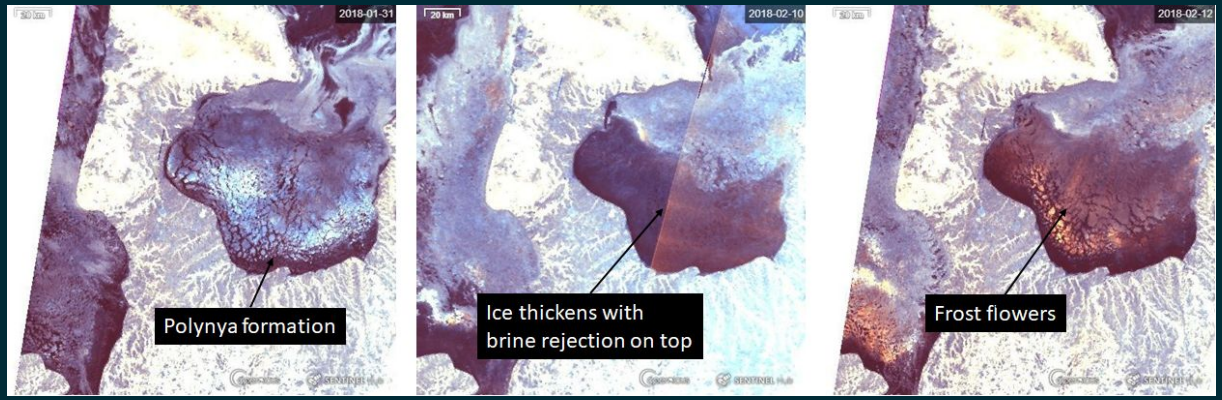


Data source: GOES-17 ABI

Composites

SAR-Ice

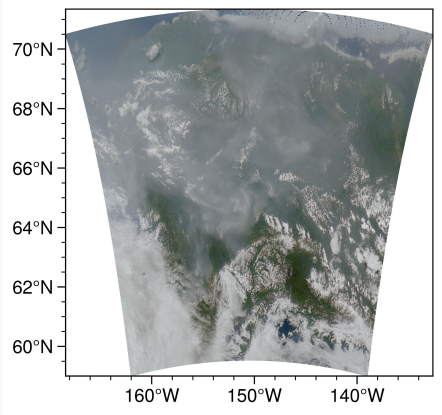




Data source: Sentinel 1 SAR

Scientific Figure

```
from satpy.writers import get_enhanced_image

# get lon/lat
lon_alaska, lat_alaska = scn_alaska['true_color'].area.get_lonlats()

# get enhanced image
>>> img_array = get_enhanced_image(scn_alaska['true_color']).data.clip(0, 1).load()

# Plot data using pcolormesh
m = axs.pcolormesh(lon_alaska, lat_alaska, img_array[0, :, :],
facecolor = img_array.transpose('y', 'x','bands').values.reshape(-1, 3))

# This is necessary to let the `color` argument determine the color
m.set_array(None)

axs.format(xformatter='deglon', yformatter='deglat', grid=False)
```



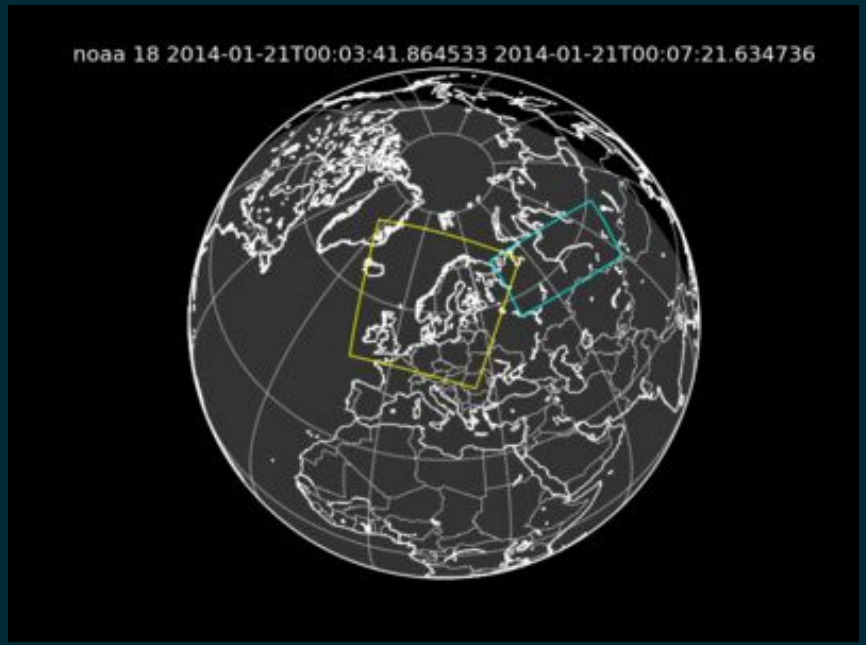
Writers

Satpy Writers

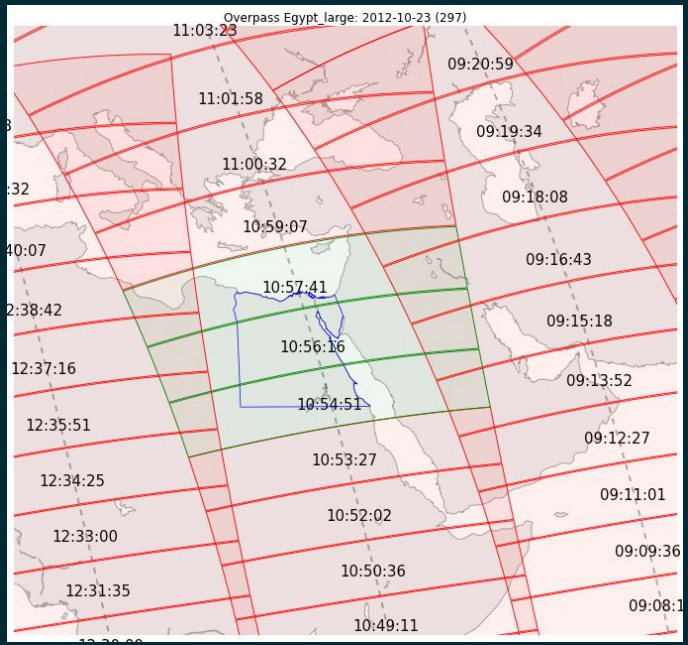
Description	Writer name	Status	Examples
GeoTIFF	`geotiff`	Nominal	
Simple Image (PNG, JPEG, etc)	`simple_image`	Nominal	
NinJo TIFF (using `pyninjotiff` package)	`ninjotiff`	To be deprecated (use ninjogeotiff)	
NetCDF (Standard CF)	`cf`	Pre-alpha	`Usage example`
AWIPS II Tiled NetCDF4	`awips_tiled`	Beta	
GeoTIFF with NinJo tags	`ninjogeotiff`	Beta	

Others

Pytroll-Schedule



Pyorbital



Processing EO satellite data is **EASY** ;)

Pytroll@Slack

Pytroll@GitHub

pytroll@googlegroups.com

Jupyter Notebooks: https://github.com/zxdawn/satpy_luchtalk

Composites

```
from pyresample import load_area

area_alaska = load_area('../scripts/test_areas.yaml', 'alaska_2km')

# resample to interested area
scn_alaska = scn.resample(area_alaska)

# export to image file
scn_alaska.save_datasets(filename='{sensor}_{name}_alaska_{start_time:%Y%m%d-%H%M%S}.png',
                         base_dir='../figures/',
                         datasets=['cira_fire_temperature', 'true_color'],
                         writer='simple_image')
```

Resampling

Resampling to Rectilinear rids

```
>>> from pyresample import create_area_def

# define the Rectilinear grid
>>> alaska_rect = create_area_def('alaska_Rectilinear',
>>>                               {'proj': 'longlat', 'datum': 'WGS84'},
>>>                               area_extent=[-165, 60, -140, 70],
>>>                               resolution=0.01,
>>>                               units='degrees',
>>>                               description='Alaska 0.01x0.01 degree lat-lon grid')

>>> scn_abi_alaska_rect = scn_abi.resample(alaska_rect)
>>> scn_abi_alaska_rect.show('C01')
```

Resampling

Resampling to Curvilinear Grids

```
# you can use `satpy/utils/coord2area_def.py`` to create the area
# python coord2area_def.py alaska_1km laea 60 70 -165 -140 1
>>> alaska_curv = load_area('../scripts/test_areas.yaml', 'alaska_1km')

>>> scn_abi_alaska_curv = scn_abi.resample(alaska_curv, resampler='nearest')
```