



Atmospheric Retrievals in a Modern Python Framework

Mario Echeverri Bautista¹, Anton Verhoef¹, Ad Stoffelen¹, Maximilian Maahn²

(1) KNMI

(2) Leipzig University, Institute for Meteorology



Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Milieu



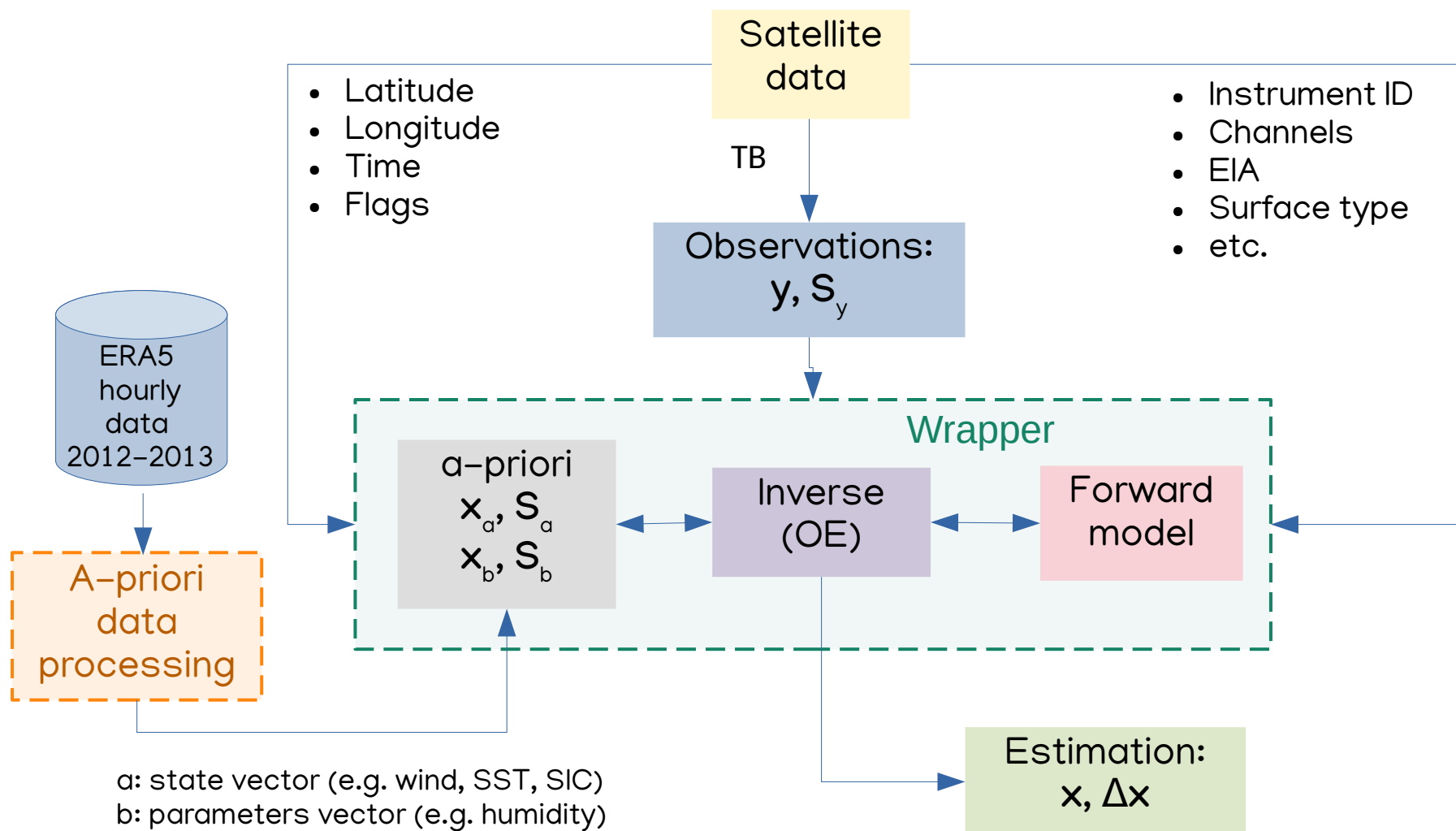
UNIVERSITÄT
LEIPZIG



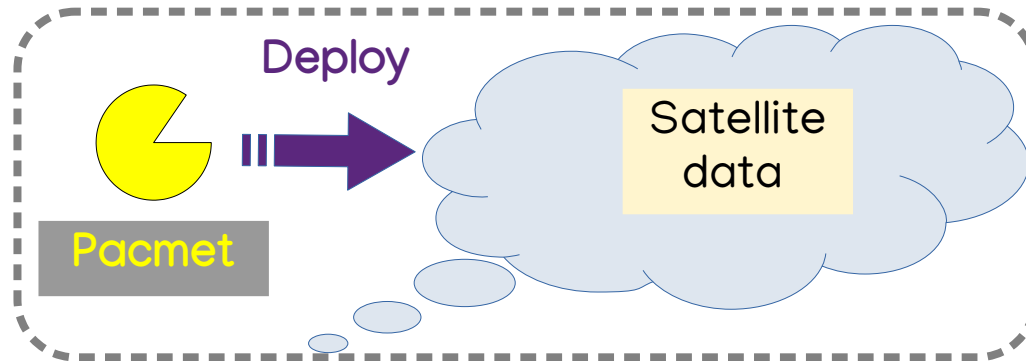
Today's story:

- The main idea
- From Observations to Atmosphere
- Going at scale using Pangeo
- Ongoing activities
- Wrap up

The OE pipeline:

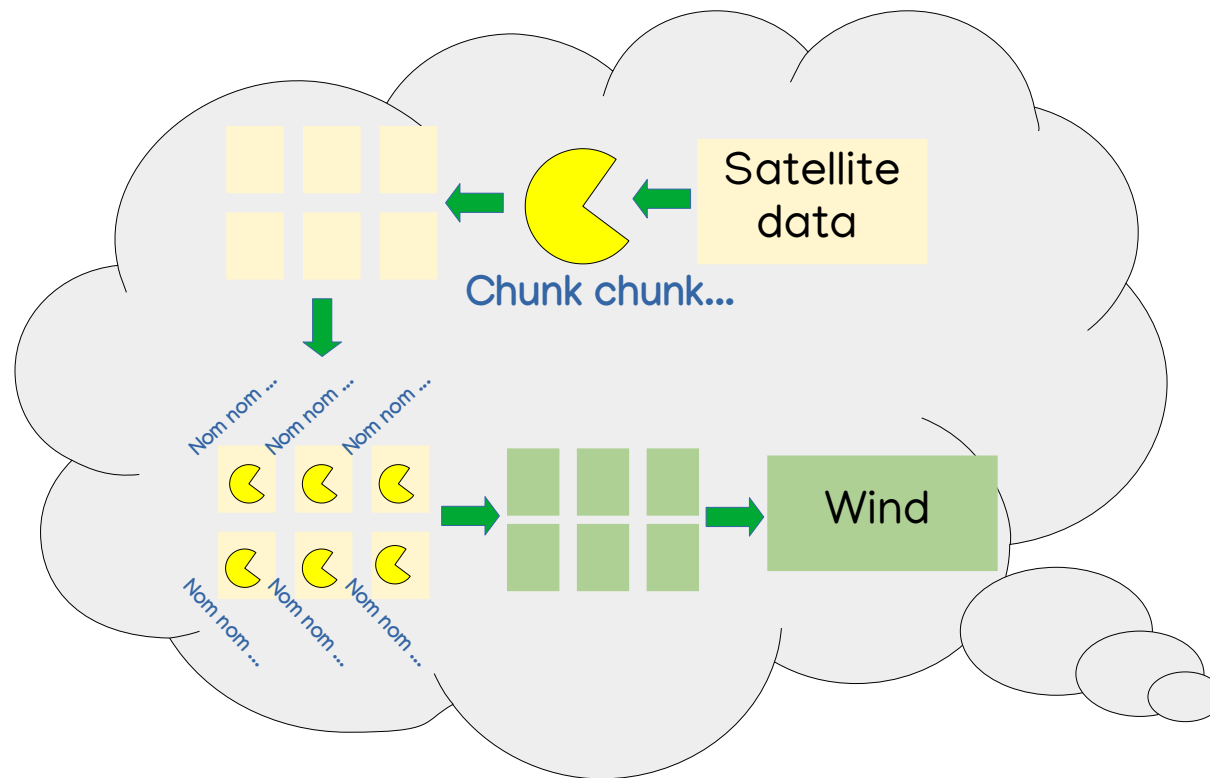


Data-centered framework:

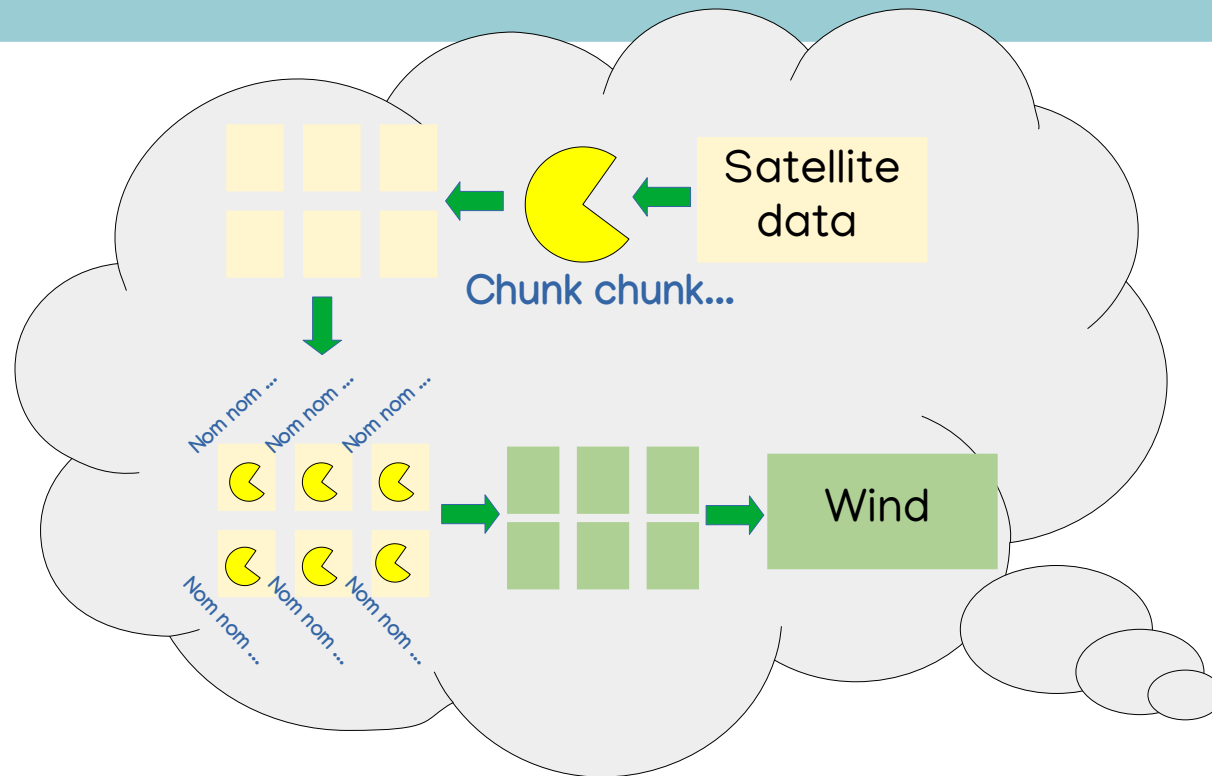


- We want to process a bunch of data on a “cloud”.
- Data downstream is increasing with time, solution needs to scale up easily.
- Software is modular; components are “easily” deployed.
- Desirable: Interface with modern Machine Learning libraries possible.

Data-centered framework: wind example



Data-centered framework: caveats



- Chunking is a logical operation, i.e. inexpensive*.
- “Nom nom...” refers to the processing our *chunks* of data.
- Number of *chunks* do not need to be same as number of workers: 🍷

* It generate tasks to be performed, this is not a ‘live’ operation, but the tasks are to be performed at some point.

The message, do not overdo with chunking.



Data Model and Data Formats

- We use **xarray**¹ as data interface.
- **xarray** follows the NetCDF data model (thus it follows CF conventions²):
 - **Self-Describing.** A netCDF file includes information about the data it contains.
 - **Portable.** A netCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.
 - **Scalable.** Small subsets of large datasets in various formats may be accessed efficiently through netCDF interfaces, even from remote servers.
 - **Appendable.** Data may be appended to a properly structured netCDF file without copying the dataset or redefining its structure.
 - **Sharable.** One writer and multiple readers may simultaneously access the same netCDF file.
 - **Archivable.** Access to all earlier forms of netCDF data will be supported by current and future versions of the software.
- This is highly beneficial in terms of embracing FAIR principles.
- **xarray**'s API handles among others **NetCDF**, **GRIB** and **zarr**.
- **xarray** can be seen as a multidimensional **Pandas** and labeled **Numpy**.

1) <https://docs.xarray.dev/en/stable/index.html>

2) <https://cfconventions.org/cf-conventions/cf-conventions.pdf>

Data Model and Data Formats: examples



GRIB (ERA5)

xarray.Dataset

Dimensions: (time: 96, step: 2, isobaricInhPa: 37, latitude: 724, longitude: 1440)

Coordinates:

number	()	int64	0	
time	(time)	datetime64[ns]	2014-01-01T0...	
step	(step)	timedelta64[ns]	06:00:00 18:0...	
isobaricInhPa	(isobaricInhPa)	float64	1e+03 975.0 9...	
latitude	(latitude)	float64	90.0 89.75 89...	
longitude	(longitude)	float64	-180.0 -179.8 ...	
valid_time	(time, step)	datetime64[ns]	dask.array<ch...	
surface	()	float64	0.0	

Data variables:

t	(time, step, isobaricInhPa, latitude, longitude)	float32	dask.array<ch...	
q	(time, step, isobaricInhPa, latitude, longitude)	float32	dask.array<ch...	
u10n	(time, step, latitude, longitude)	float32	dask.array<ch...	
v10n	(time, step, latitude, longitude)	float32	dask.array<ch...	
sp	(time, step, latitude, longitude)	float32	dask.array<ch...	
t2m	(time, step, latitude, longitude)	float32	dask.array<ch...	
lsm	(time, step, latitude, longitude)	float32	dask.array<ch...	
skt	(time, step, latitude, longitude)	float32	dask.array<ch...	

Attributes:

GRIB_edition : 1
GRIB_centre : ecmf
GRIB_centreDe... : European Centre for Medium-Range Weather Forecasts
GRIB_subCentre : 0
Conventions : CF-1.7
institution : European Centre for Medium-Range Weather Forecasts
history : 2022-03-11T10:35 GRIB to CDM+CF via cfgrib-0.9.9.0/ecCodes-2.22.1 with {"source": "/home/mario/Data/Covariance_means/MARS_api_data/ERA5_data/datasets/profiles_2014UC.grib", "filter_by_keys": {}, "encode_cf": [{"parameter", "time", "geography", "verti cal"]}]

NetCDF (*CMSAF-ish)

xarray.Dataset

Dimensions: (time: 45505, scene_across_track: 90, scene_channel: 5)

Coordinates:

scene_across_...	(scene_across_track)	int32	1 5 9 13 17 ... 34...	
scene_channel	(scene_channel)	int64	11 12 13 14 15	
time	(time)	datetime64[ns]	2014-09-09 ... 2...	

Data variables:

lat	(time, scene_across_track)	float64	dask.array<chun...	
lon	(time, scene_across_track)	float64	dask.array<chun...	
eia	(time, scene_across_track)	float32	dask.array<chun...	
sft	(time, scene_across_track)	float32	dask.array<chun...	
tb	(time, scene_across_track, scene_channel)	float32	dask.array<chun...	
global_channel...	(scene_channel)	int32	dask.array<chun...	
channel_uncert...	(scene_channel)	float32	dask.array<chun...	
wind	(time, scene_across_track)	float32	dask.array<chun...	
wind_err	(time, scene_across_track)	float32	dask.array<chun...	
chiSquareTest1	(time, scene_across_track)	float32	dask.array<chun...	
chiSquareTest2	(time, scene_across_track)	float32	dask.array<chun...	
chiSquareTest3	(time, scene_across_track)	float32	dask.array<chun...	
chiSquareTest4	(time, scene_across_track)	float32	dask.array<chun...	

Attributes:

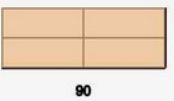
title : Environmental Scene 1
comment : feedhorn channels: h19, v19, v22
elevation_offset... : 0.4
azimuth_offset... : -0.3

Big data, chunks and processing: Nom nom...



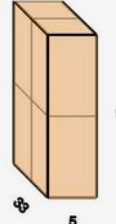
lat (time, scene_across_track) float64 dask.array<chun...

	Array	Chunk
Bytes	23.20 kiB	5.98 kiB
Shape	(33, 90)	(17, 45)
Count	5 Tasks	4 Chunks
Type	float64	numpy.ndarray



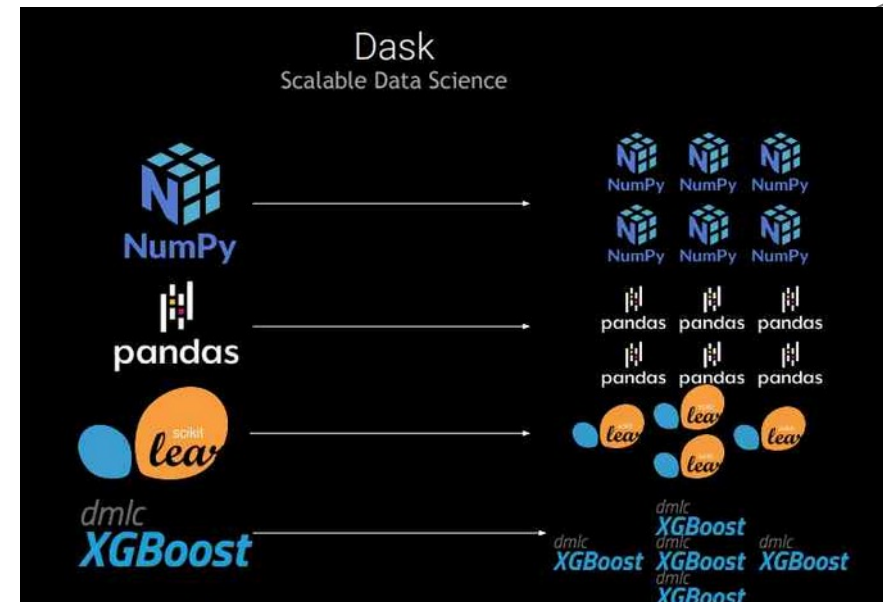
tb (time, scene_across_track, scene_channel) float32 dask.array<chun...

	Array	Chunk
Bytes	58.01 kiB	14.94 kiB
Shape	(33, 90, 5)	(17, 45, 5)
Count	5 Tasks	4 Chunks
Type	float32	numpy.ndarray



- xarray uses Dask¹ under the hood in order to handle big datasets
- Divide and conquer strategy: divide dataset in chunks and process

From 2022,



1) <https://docs.dask.org/en/stable/>
2) <https://www.nvidia.com/en-us/glossary/data-science/dask/>



Our application: wind retrieval

- Wind retrievals using optimal estimation and radiometer observations:
 - We use **pyOptimalEstimation** for performing the retrieval (core computation); open source project (ME contributor).
 - We use **pyResample** (pyTroll) for re-sampling (when needed).
 - We use **xarray + Dask** for multi-dimensional labeled datasets handling¹ and high level parallelization².
 - We use **RTTOV** from NWPSAF as radiative transfer model (i.e. forward model).

1) Handling includes Numpy and Pandas native support + Dask chunking

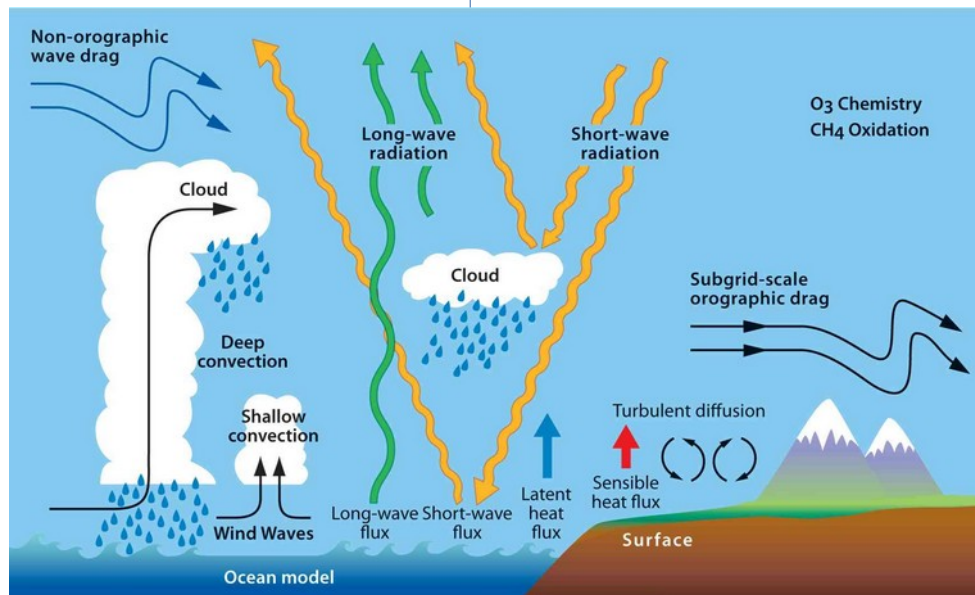
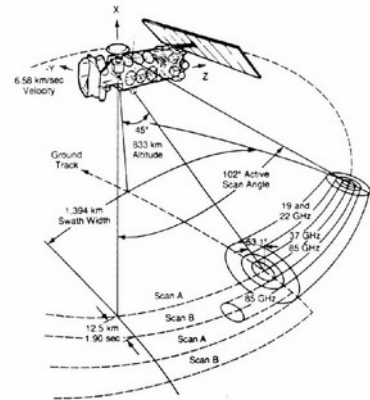
2) High level parallelization includes creation of cluster of workers and scheduling tasks to be executed.

Wind retrievals: Radiative transfer (forward model)



Top of the Atmosphere: TOA

Observations:
Brightness
temperature (TB)



profiles: Temp.,
Hum., Press.,
etc.

A forward model connects
the unknowns with the
observations:

$$TB = F(f, \theta, b, \text{profiles}, \text{NSWS})$$

Surface of the ocean

Surface unknowns:
Wind speed at 10m
height

Optimal Estimation (OE): inversion

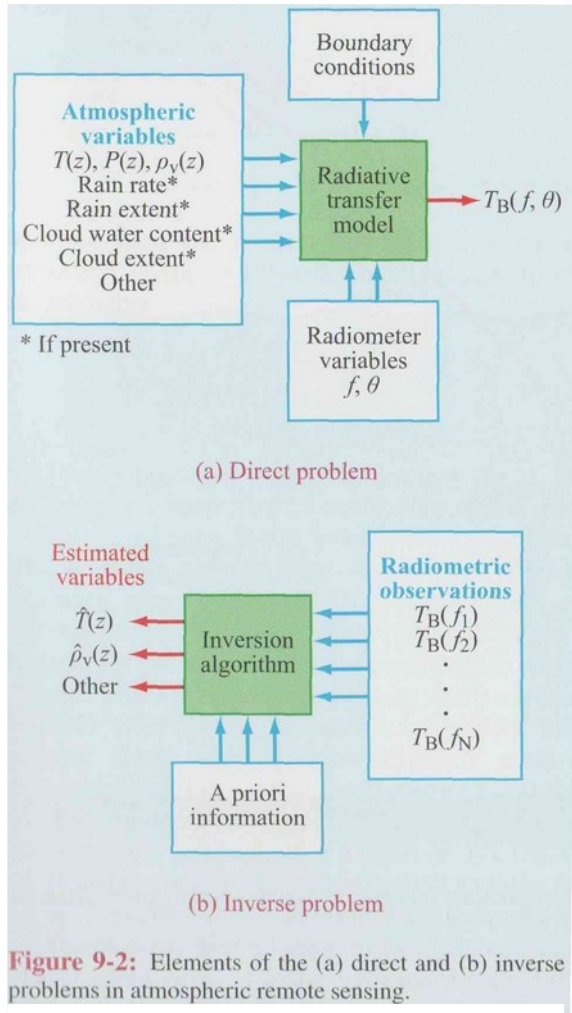
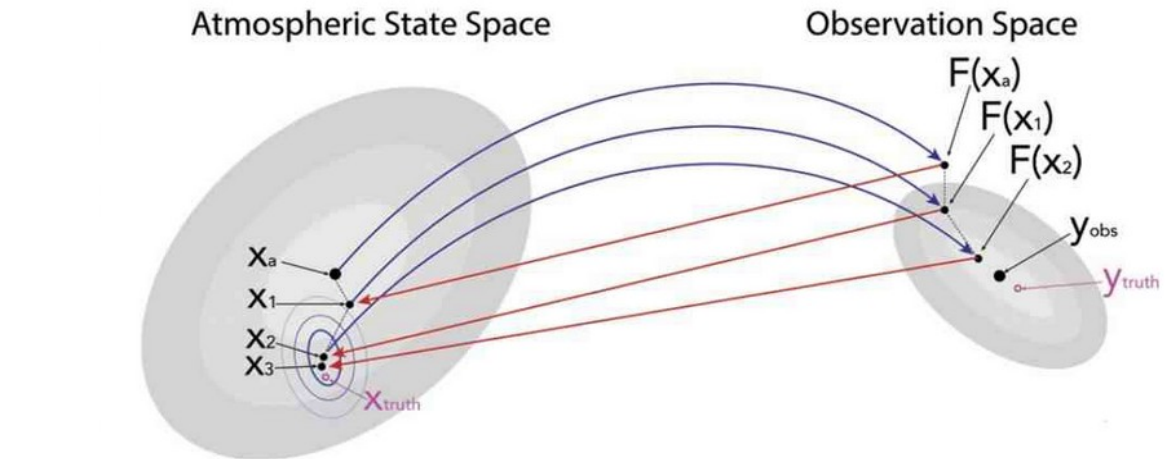


Figure 9-2: Elements of the (a) direct and (b) inverse problems in atmospheric remote sensing.

From *



Estimated Atmospheric state:

$$\mathbf{x}_{i+1} = \mathbf{x}_a + (\mathbf{S}_a^{-1} + \mathbf{K}_i^T \mathbf{S}_e^{-1} \mathbf{K}_i)^{-1} \mathbf{K}_i^T \mathbf{S}_e^{-1} [\mathbf{y} - \underbrace{F(\mathbf{x}_i, \mathbf{b})}_{\text{Forward model}} + \mathbf{K}_i(\mathbf{x}_i - \mathbf{x}_a)],$$

Convergence criteria:

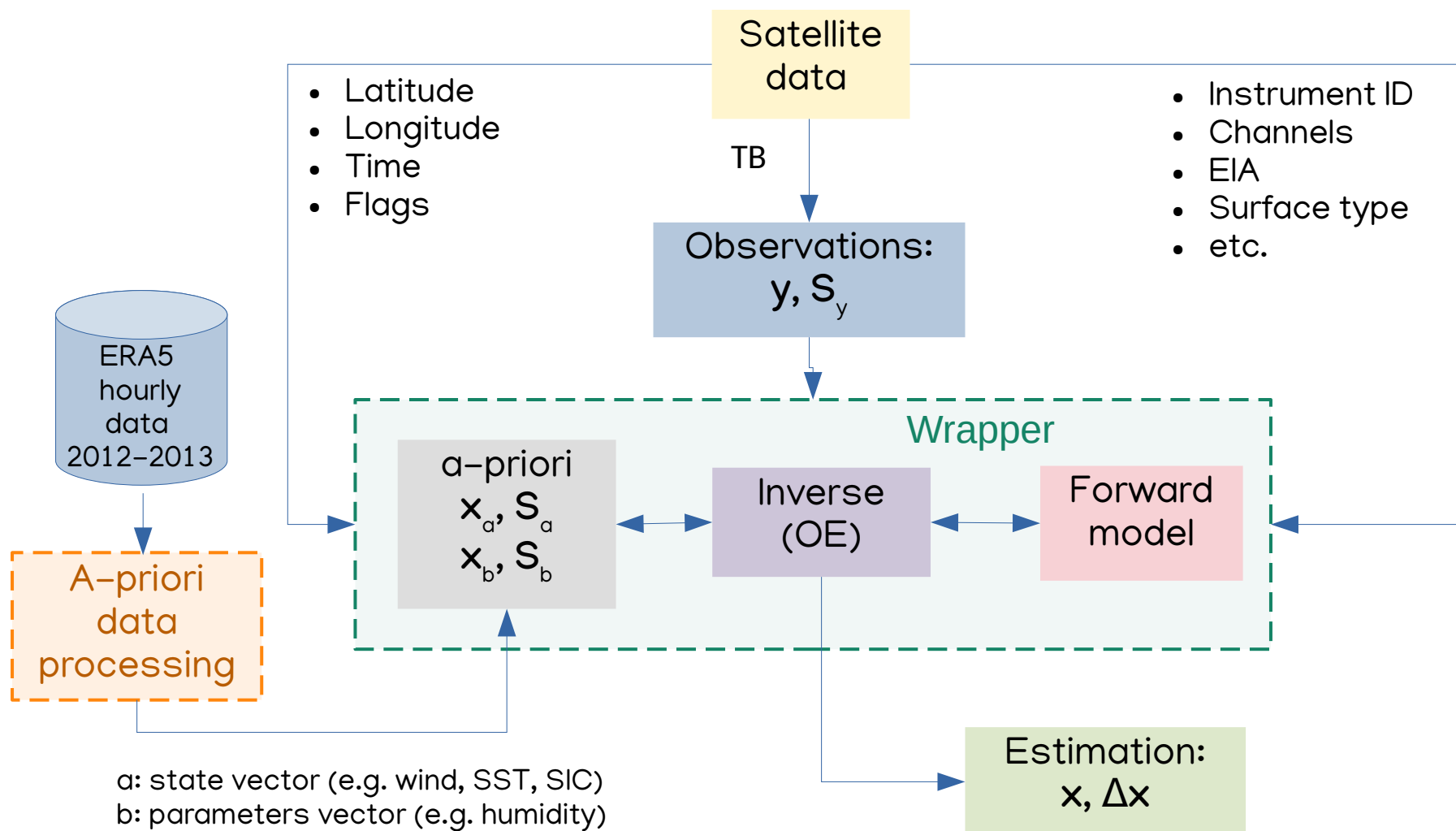
$$(\mathbf{x}_i - \mathbf{x}_{i+1})^T \mathbf{S}_i^{-1} (\mathbf{x}_i - \mathbf{x}_{i+1}) \ll \text{length}(\mathbf{x})$$

Uncertainty of the estimation:

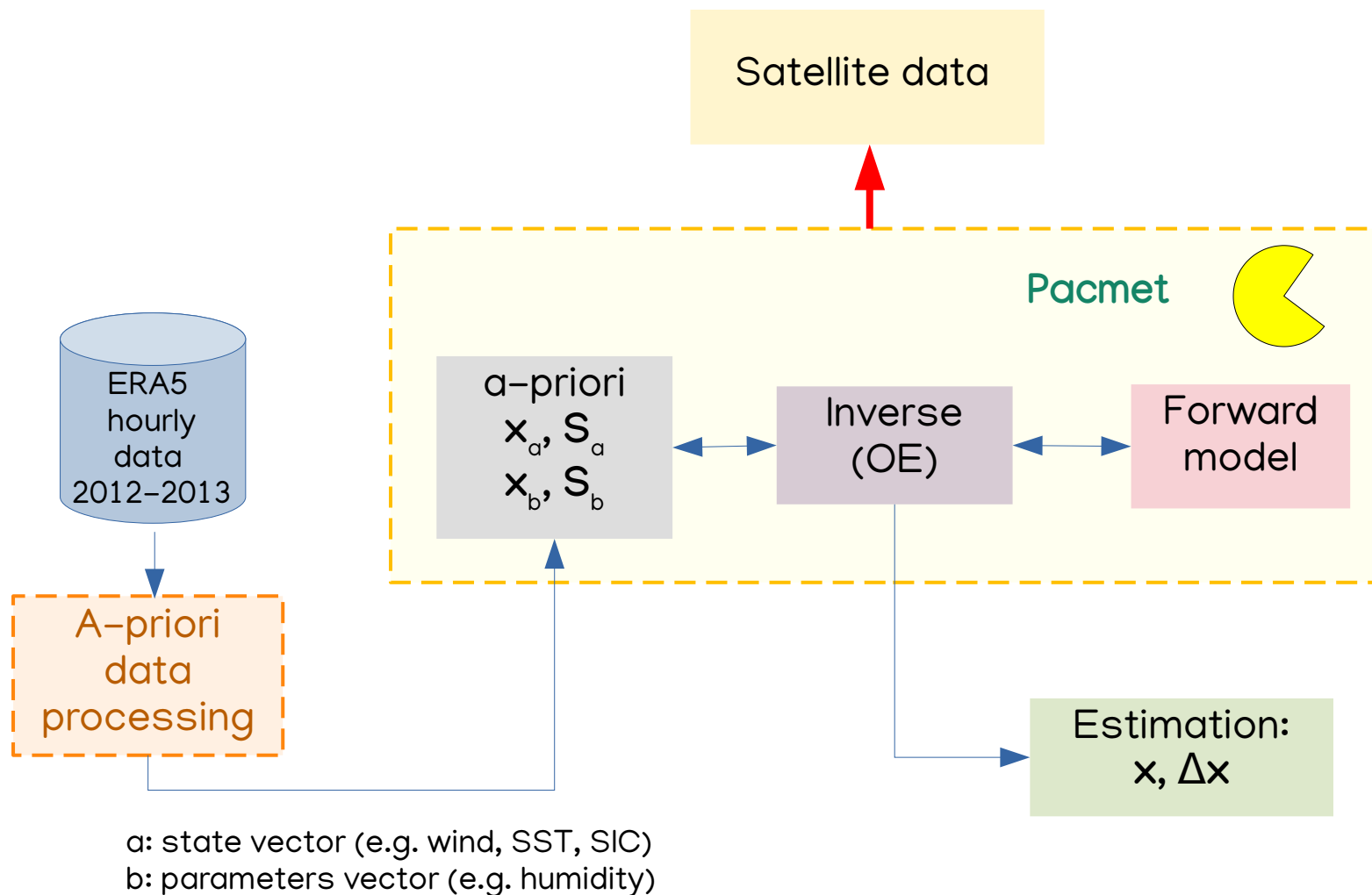
$$\mathbf{S}_i = (\mathbf{S}_i^{-1} + \mathbf{K}_i^T \mathbf{S}_e^{-1} \mathbf{K}_i)^{-1},$$

Scheme and equations from [1]

The OE pipeline:



The OE pipeline:





How: Load in lazy format

```
Dataset = xr.open_dataset('today.nc').chunk({"time": chunk_size_time,  
                                             "scene_across_track": chunk_size_s_a_t})
```

xarray.Dataset

► Dimensions: (time: 45505, scene_across_track: 90, scene_channel: 5)

▼ Coordinates:

scene_across_...	(scene_across_track)	int32	1 5 9 13 17 ... 34...	 
scene_channel	(scene_channel)	int64	11 12 13 14 15	 
time	(time)	datetime64[ns]	2014-09-09 ... 2...	 

▼ Data variables:

lat	(time, scene_across_track)	float64	dask.array<chun...	 
lon	(time, scene_across_track)	float64	dask.array<chun...	 
eia	(time, scene_across_track)	float32	dask.array<chun...	 
sft	(time, scene_across_track)	float32	dask.array<chun...	 
tb	(time, scene_across_track, scene_channel)	float32	dask.array<chun...	 
global_channel...	(scene_channel)	int32	dask.array<chun...	 
channel_uncert...	(scene_channel)	float32	dask.array<chun...	 
wind	(time, scene_across_track)	float32	dask.array<chun...	 
wind_err	(time, scene_across_track)	float32	dask.array<chun...	 

How: what is this?

```
Dataset = xr.open_dataset('today.nc').chunk({"time": chunk_size_time,
                                             "scene_across_track": chunk_size_s_a_t})
```

xarray.Dataset

► Dimensions: (time: 45505, scene_across_track

▼ Coordinates:

scene_across_... (scene_across_track)

scene_channel (scene_channel)

time (time)

▼ Data variables:

lat (time, scene_across_track)

lon (time, scene_across_track)

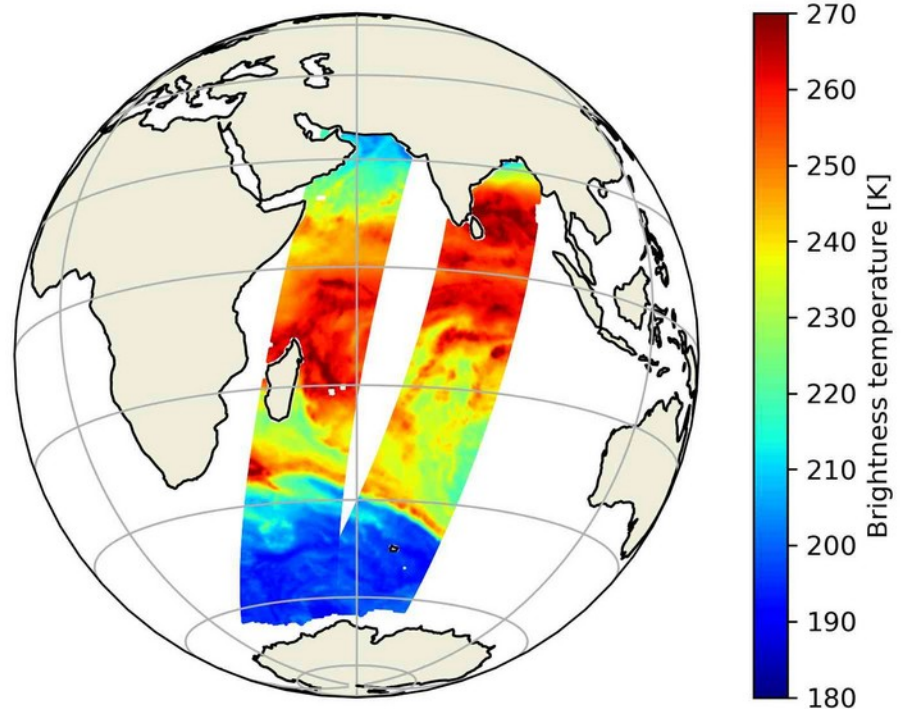
eia (time, scene_across_track)

sft (time, scene_across_track)

tb (time, scene_across_track, scene_c

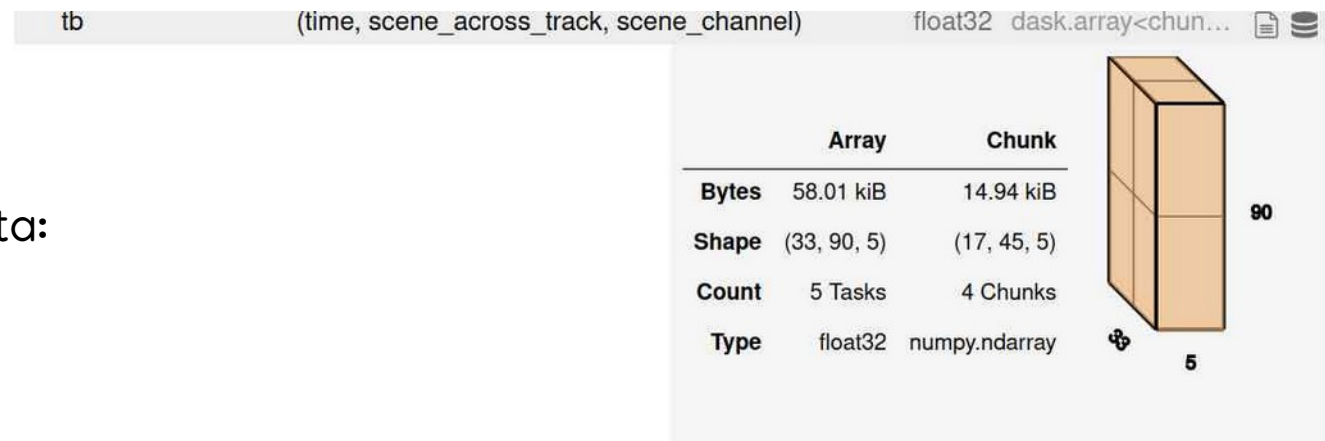
global_channel... (scene_channel)

channel_uncert... (scene_channel)

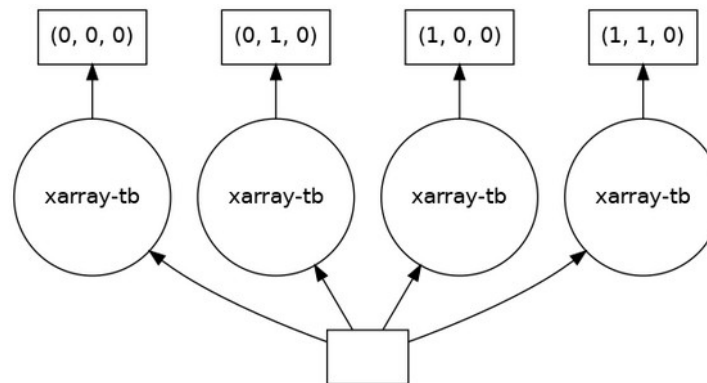


How: chunks and tasks, simple

Chunks of data:



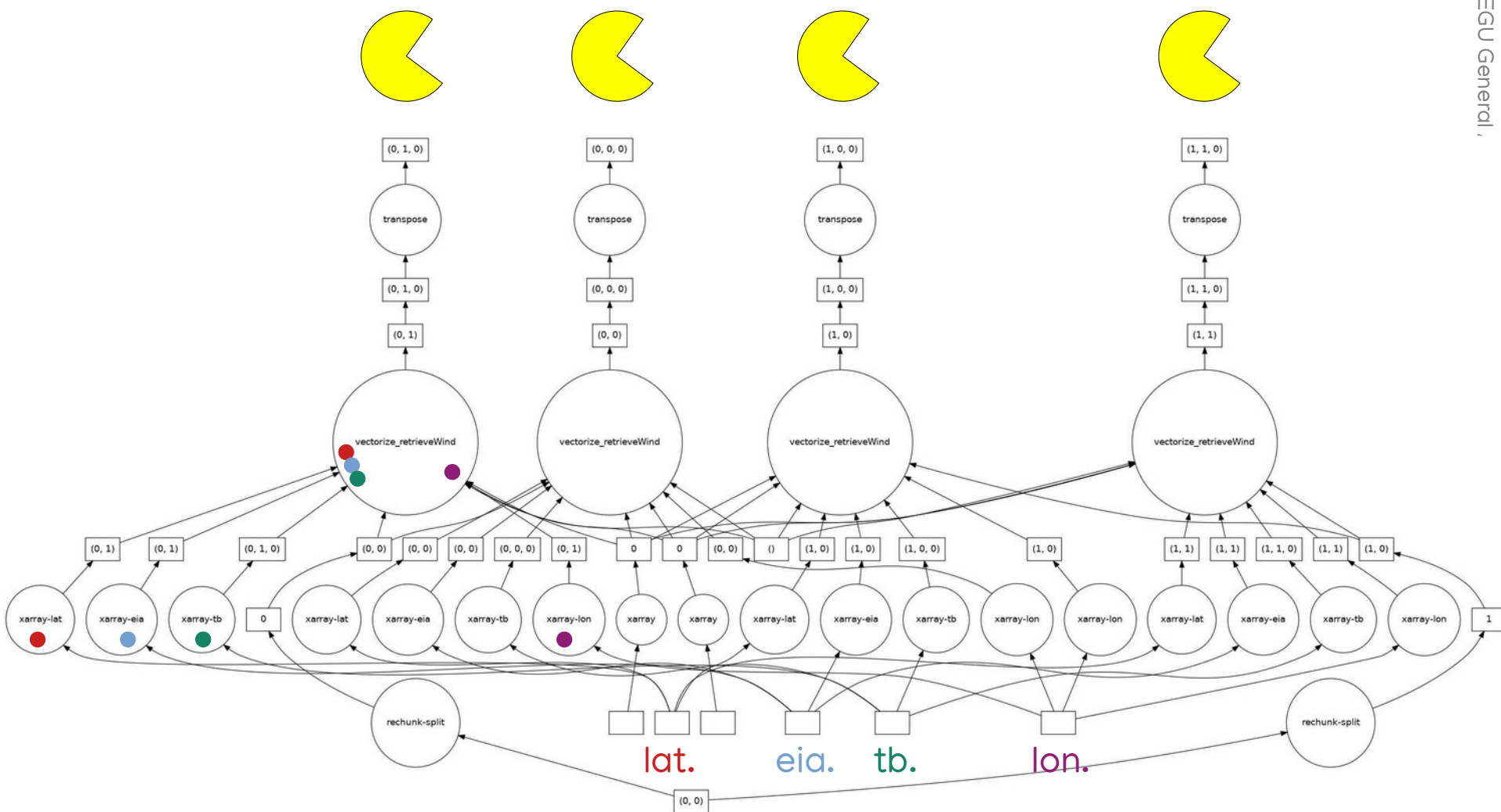
Graph of tasks:







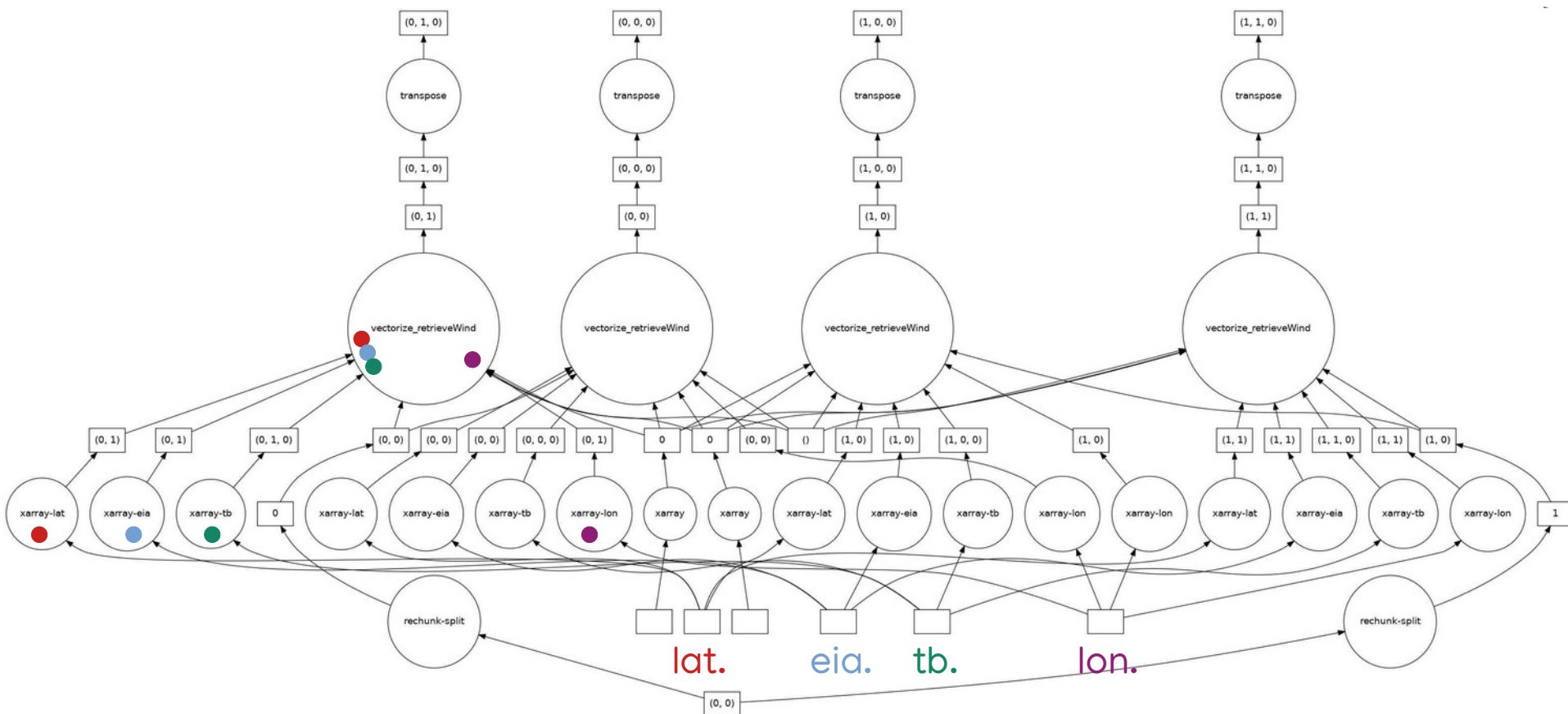
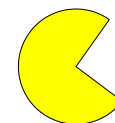
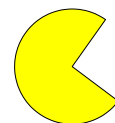
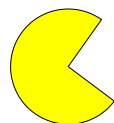
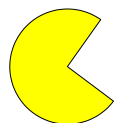
How: tasks scheduled among workers





How: tasks scheduled among workers

Cluster →





How: Computation clusters

- **Dask** offers a broad spectrum of cluster managers in its API:
 - LocalCluster (i.e. your computer with all its workers)
 - SSHCluster (i.e. a cluster of computers connected through ssh)
 - CloudProvider (i.e. leveraging cloud native API's)
 - FargateCluster (AWS)
 - Elastic Compute Cloud, EC2 (AWS)
 - Elastic Container Service (AWS)
 - DropletCluster (DigitalOcean)
 - GCPCluster (Google Cloud)
 - AzureVMCluster (Microsoft Azure)



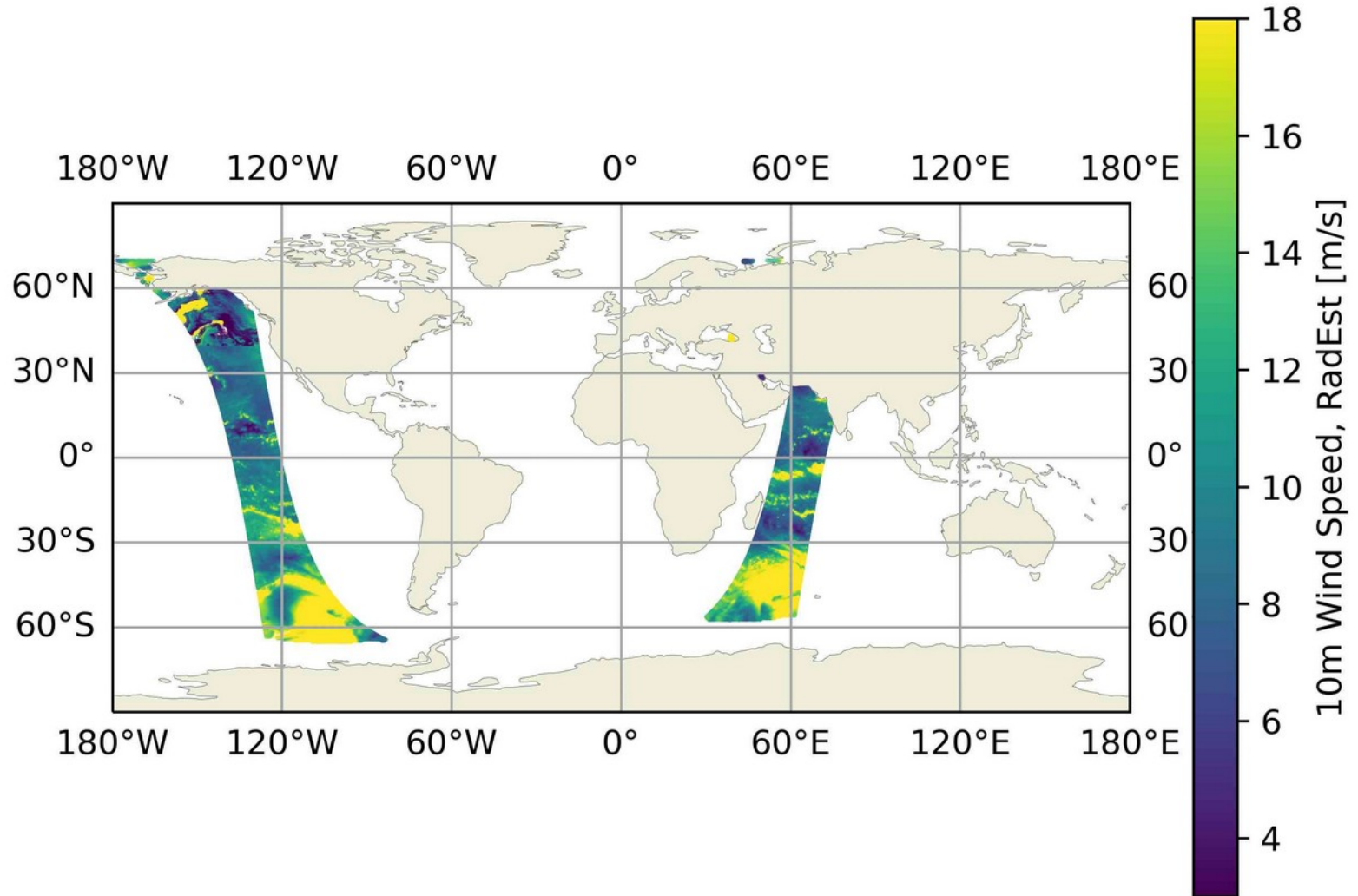
Some results:

- Wind retrievals using CMSAF SSMIS (F16) Temperature Brightness (2 hours in September 2014).
- For testing purposes we use a-priori data computed using ERA5:
 - Mean ($\mathbf{x}_a$, $\mathbf{x}_b$): Climatological mean (2012–2013)
 - Covariance ($\mathbf{S}_a$, $\mathbf{S}_b$): natural variability in the dataset*.

* This is not representative of the background error covariance, which turns out to be a whole area of research on its own, but it gives a good starting point to test.

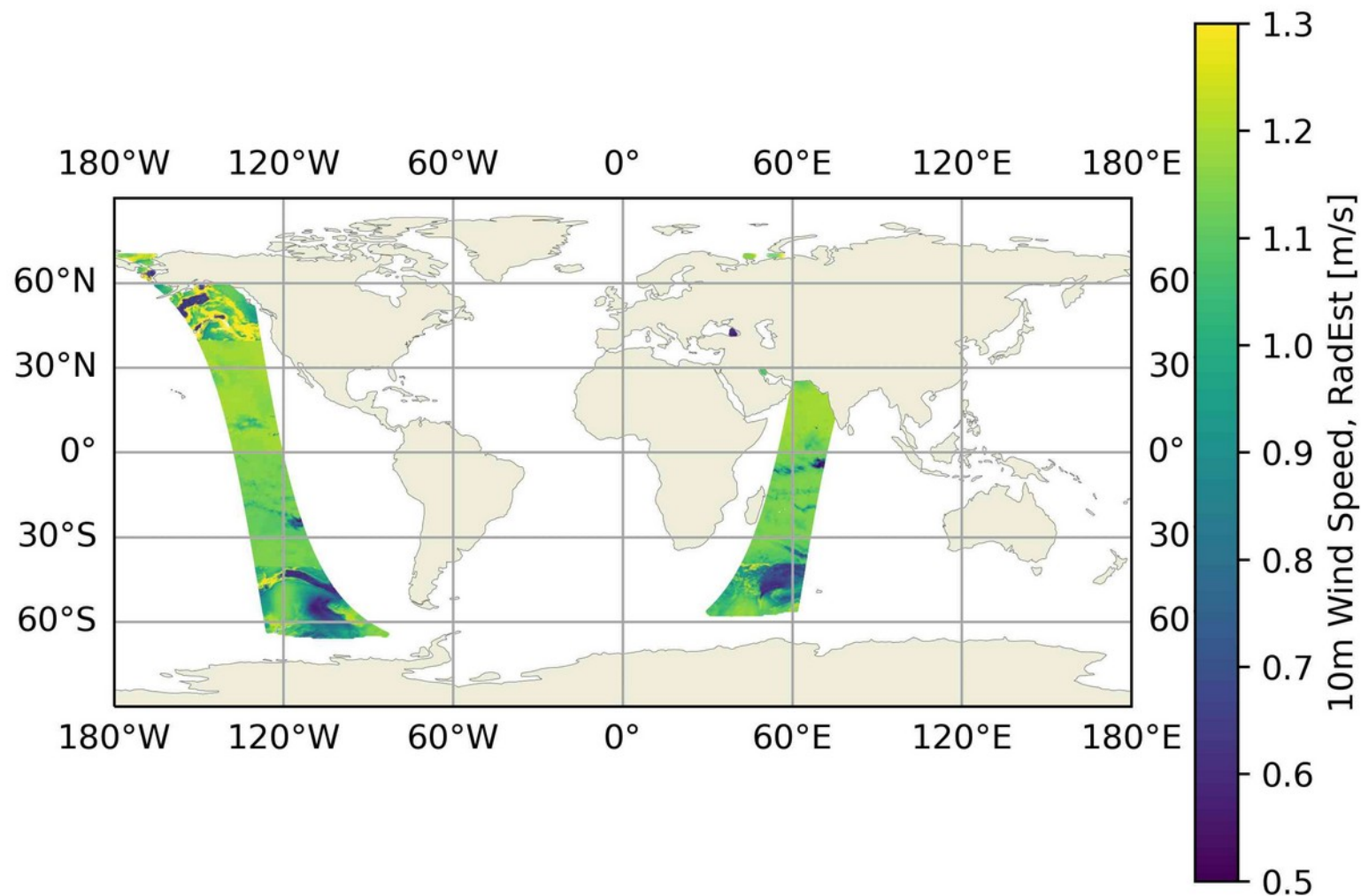
Some results: retrieved wind speed

X_{op}



Some results: uncertainty of retrieval

X_{err}

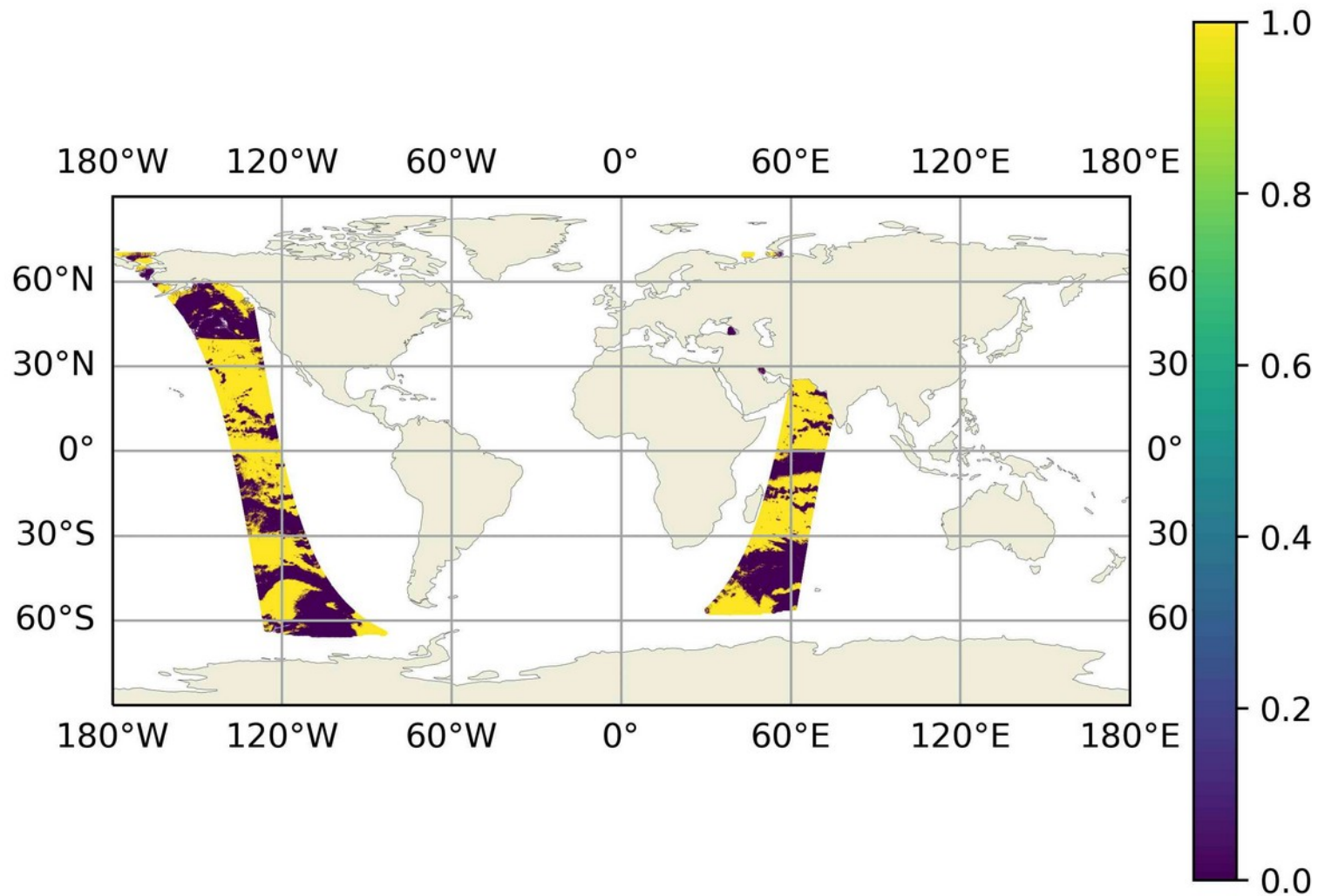




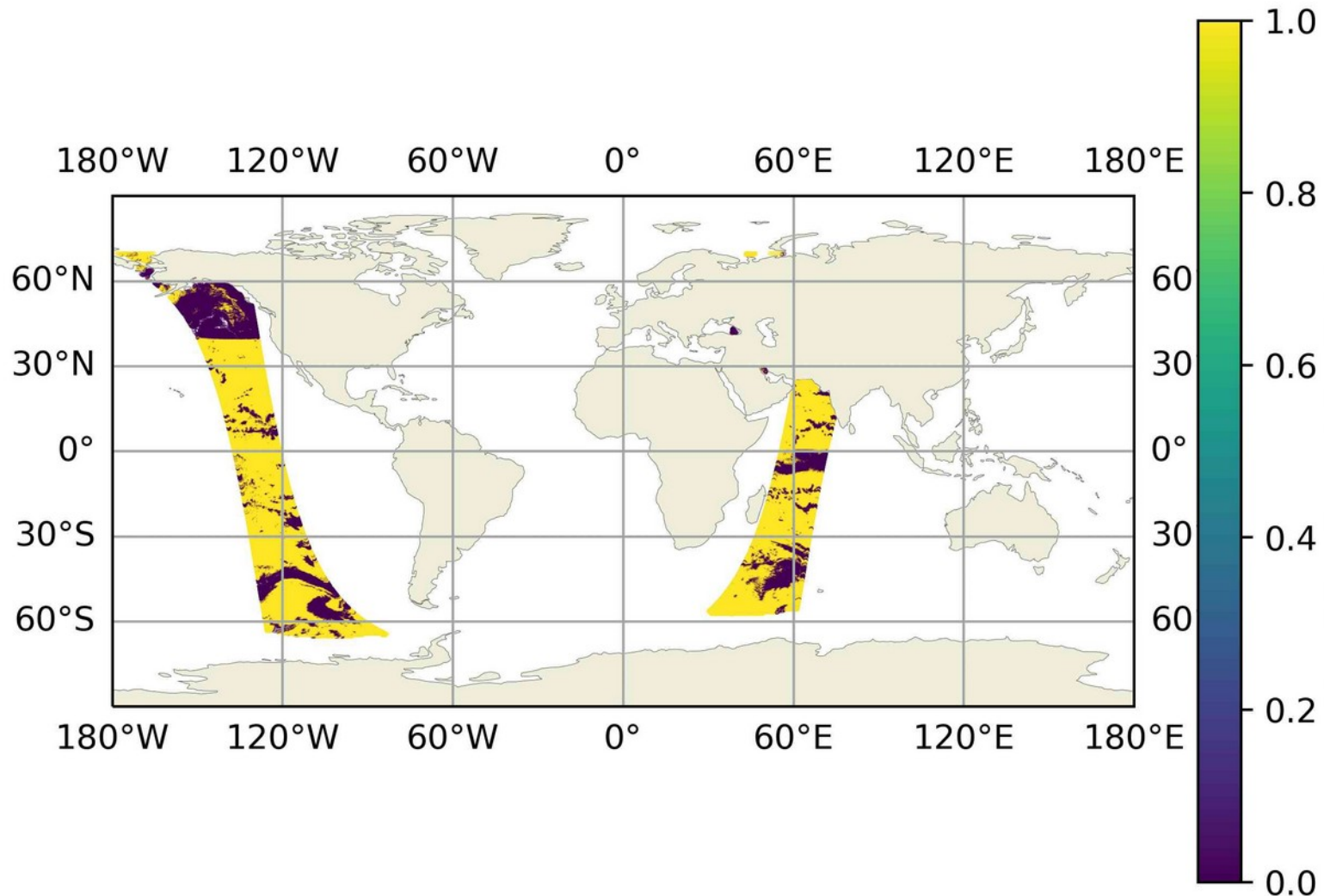
Statistical tests suite

- It is desirable to have at hand tools to evaluate the quality of the retrievals.
- Rodgers proposed a few diagnostics based on χ^2 testing (e.g. test whether or not a particular vector belongs to a given Gaussian distribution):
 - T0, y_optimal vs Obs.: $(y_{op} - y_{obs})$
 - T1, Obs. vs y_prior: $(y_{obs} - y_a)$
 - T2, y_optimal vs y_prior: $(y_{op} - y_a)$
 - T3, x_optimal vs x_prior: $(x_{op} - x_a)$

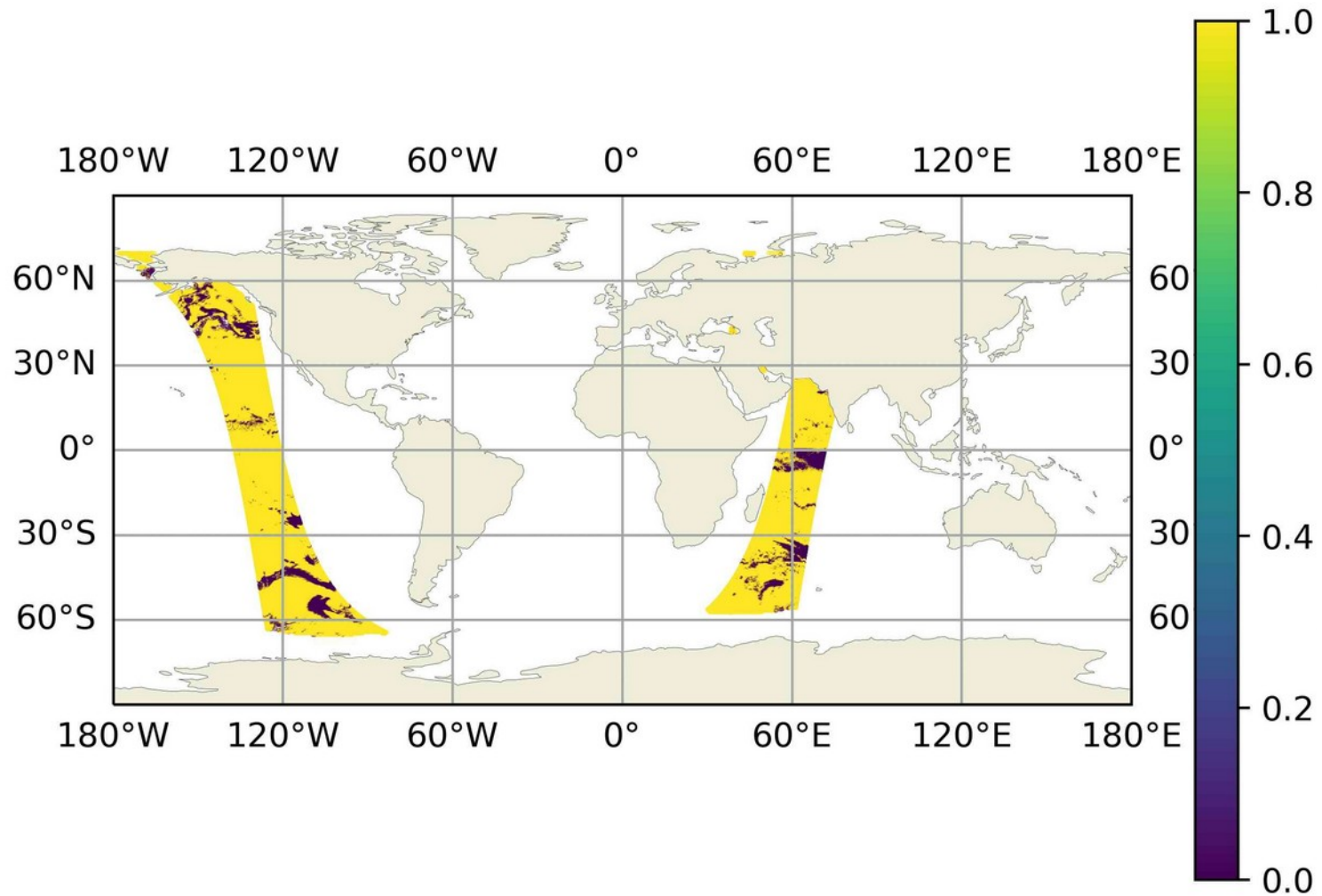
Some results: test ($y_{op} - y_{obs}$)



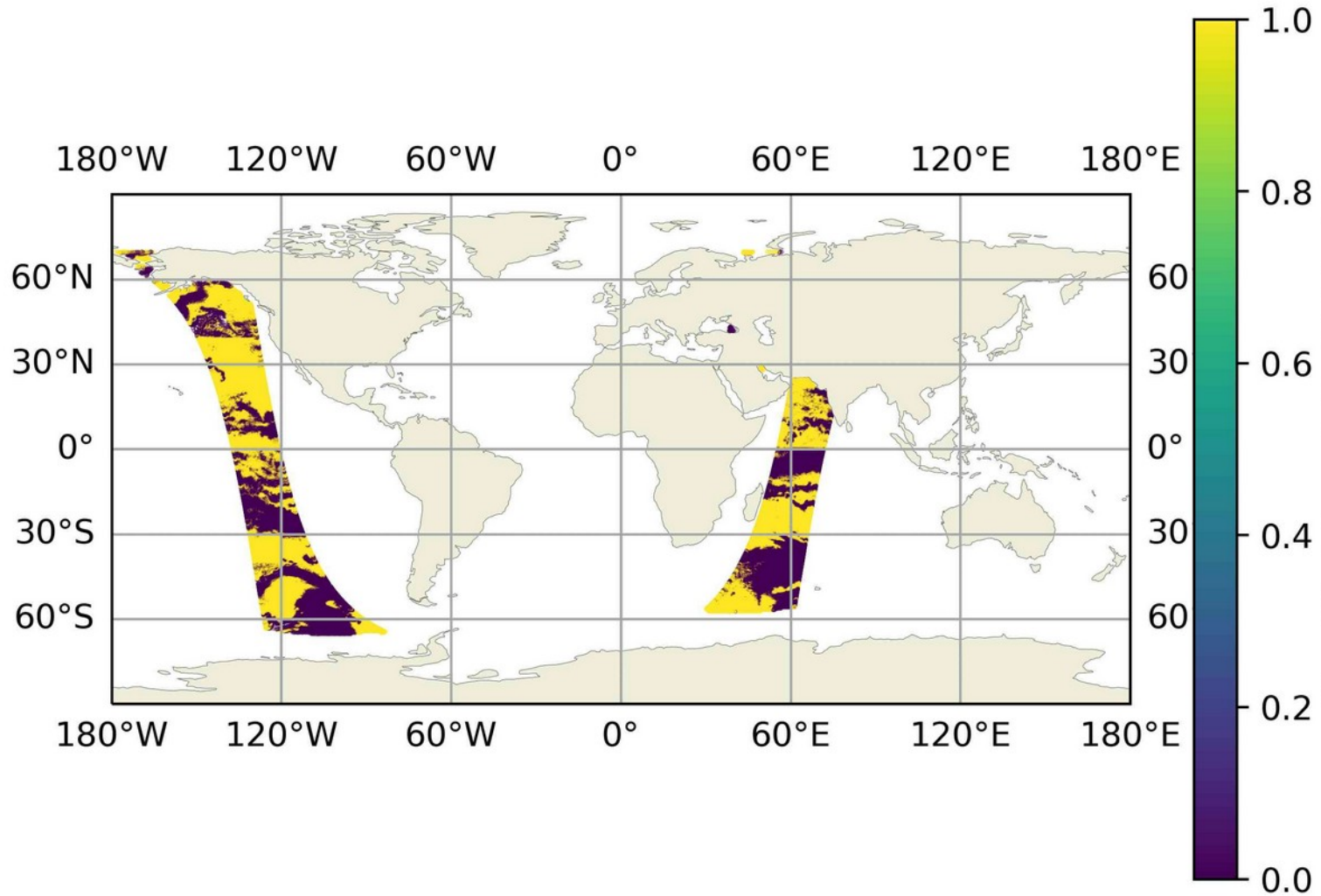
Some results: test ($y_{\text{obs}} - y_{\alpha}$)



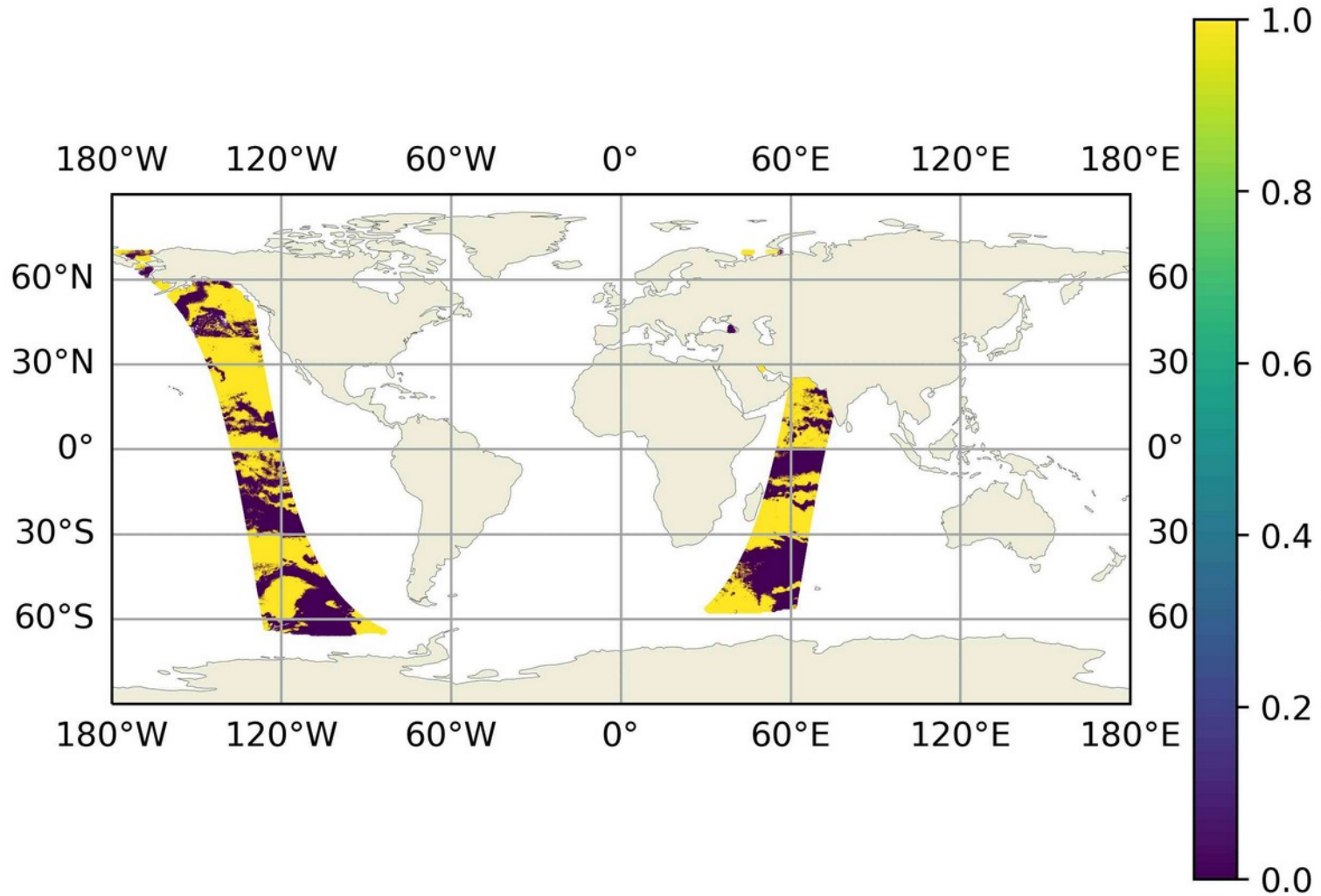
Some results: test ($y_{op} - y_a$)



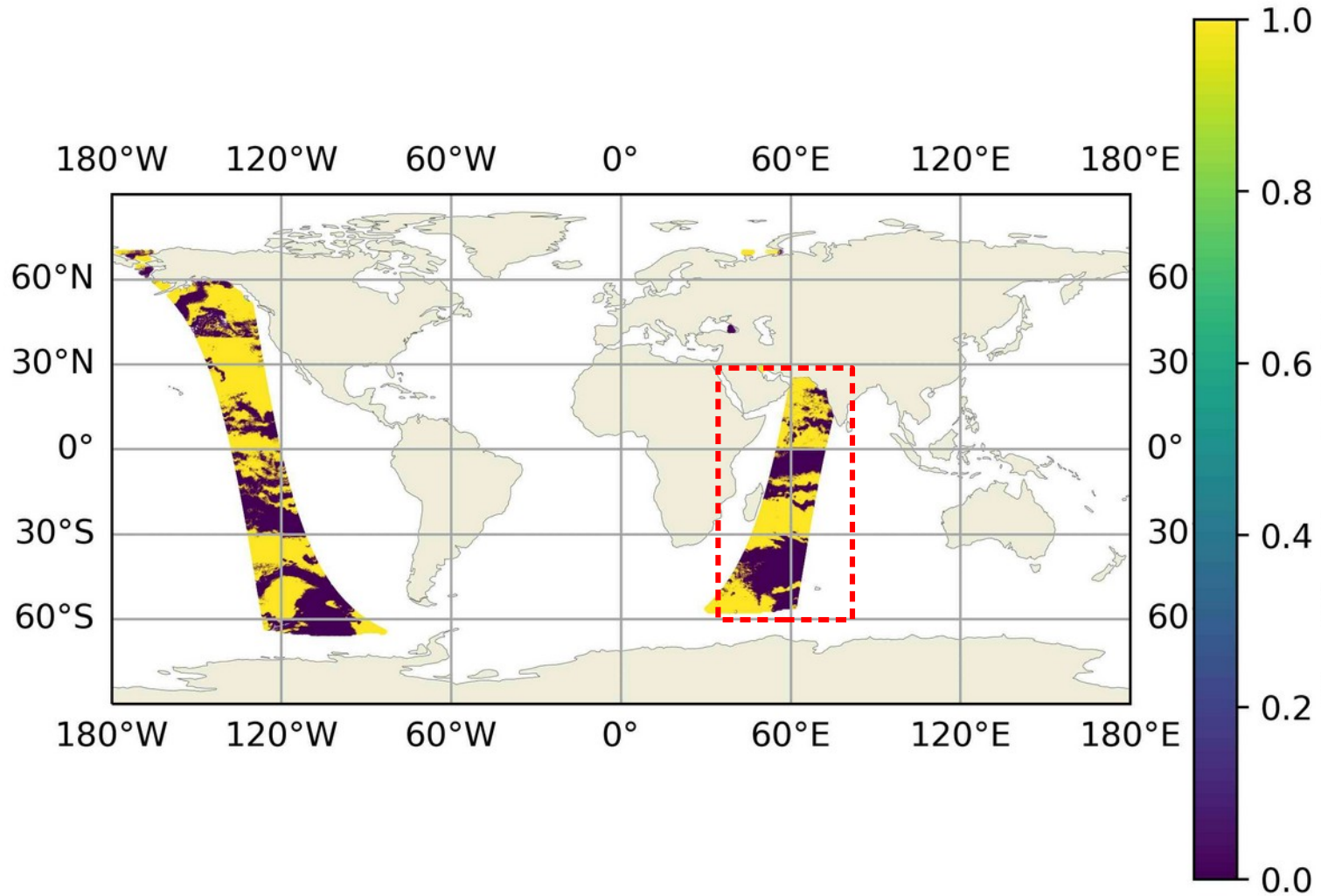
Some results: test ($x_{op} - x_a$)



Some results: test ($x_{op} - x_a$)



Some results: test ($x_{op} - x_a$)



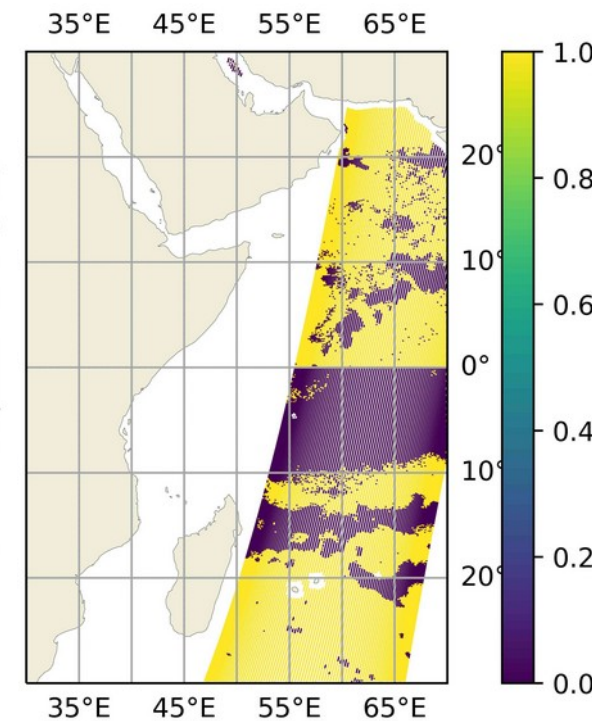
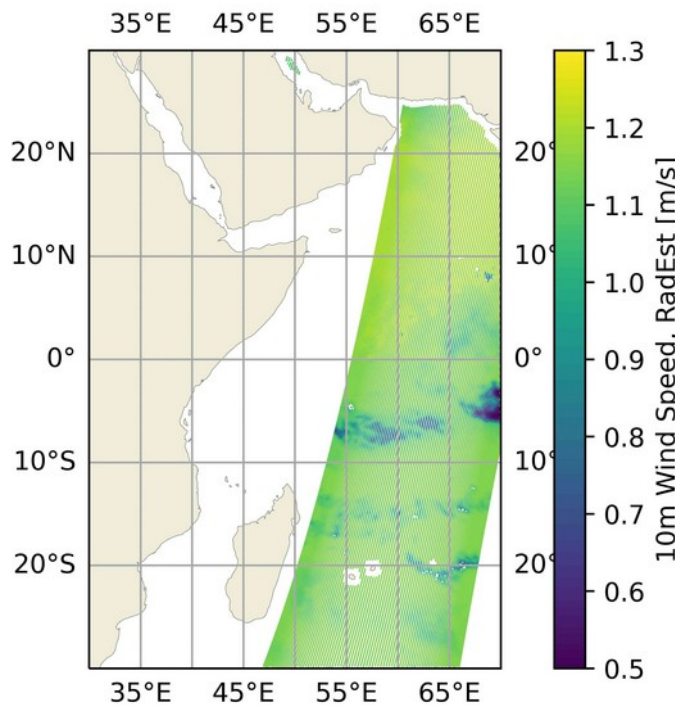
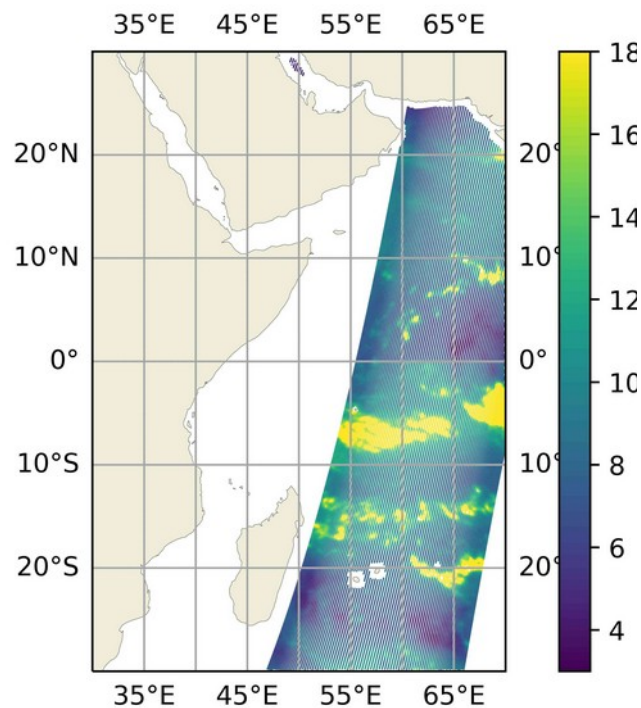


Some results: All pass

Wind speed

Wind
uncertainty

T0 & T1 & T2 & T3

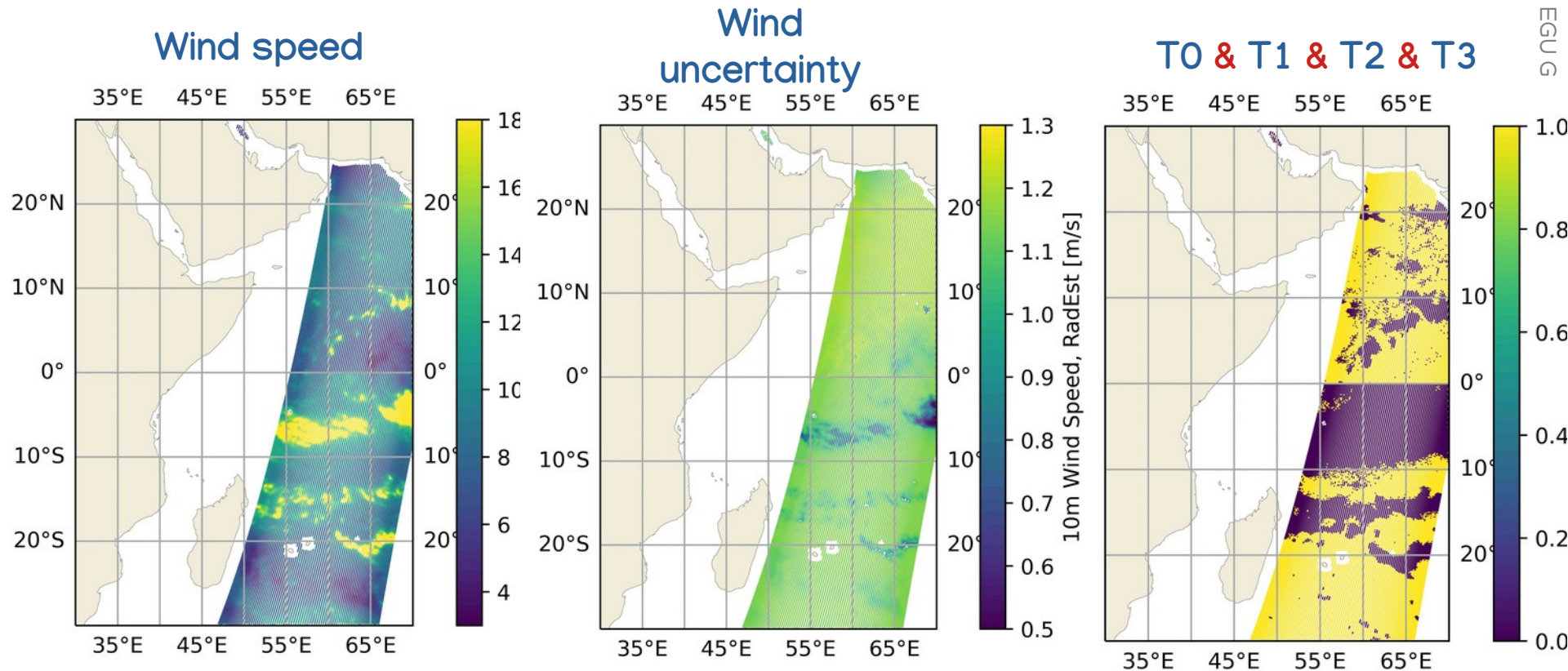


EGU General

13556



Some results: All pass



T0, y_{optimal} vs Obs.: $(y_{\text{op}} - y_{\text{obs}})$

T1, Obs. vs y_{prior} : $(y_{\text{obs}} - y_{\text{a}})$

T2, y_{optimal} vs y_{prior} : $(y_{\text{op}} - y_{\text{a}})$

T3, x_{optimal} vs x_{prior} : $(x_{\text{op}} - x_{\text{a}})$



Work in progress:

- Containerization:
 - Pack the whole “app” in a deploy-friendly container > Docker containers.
- Cloud resources:
 - What and how to use > AWS.
- A-priori error covariances:
 - Properly estimate the error variability within the ERA5 dataset (e.g. forecast model error).
- Documentation:
 - Reports
 - Markdown (for nicely documented code).

To conclude:

- A classical physical retrievals scheme was re-cast using modern Python.
- The Pangeo stack is being actively used in order to facilitate the connection between low level algorithms and high level interfaces.
- The Pangeo stack provides tools that can be integrated in a satellite observations processing pipeline and opens doors to high performance deployment in a straightforward manner.



References

- [1] M. Maahn et al, “Optimal Estimation Retrievals and Their Uncertainties What Every Atmospheric Scientist Should Know”, BAMS, Vol. 101, Issue 9, Sept. 2020
- [2] C. D. Rodgers, “Inverse Methods for Atmospheric Sounding, Theory and Practice”, World Scientific Pub., 2000.



Thanks!



Questions?