SENTINEL-3A CAL/VAL ENVIRONMENT AND APPLICATIONS

Jean-François Piollé, Ifremer
(Eumetsat/Copernicus visiting scientist)

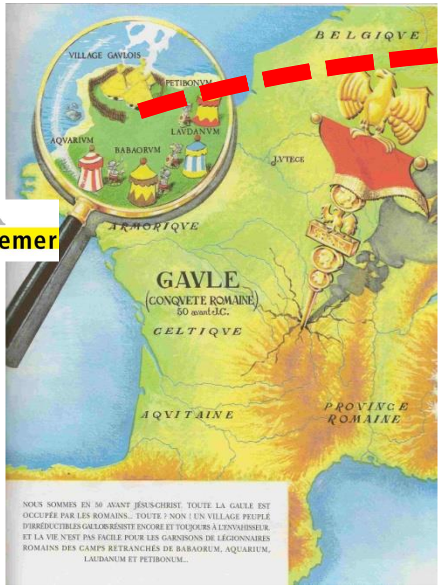
And EUMETSAT S3A SLSTR team : **Igor Tomazic, Anne O'Caroll, Prasanjit Dash**

Outline

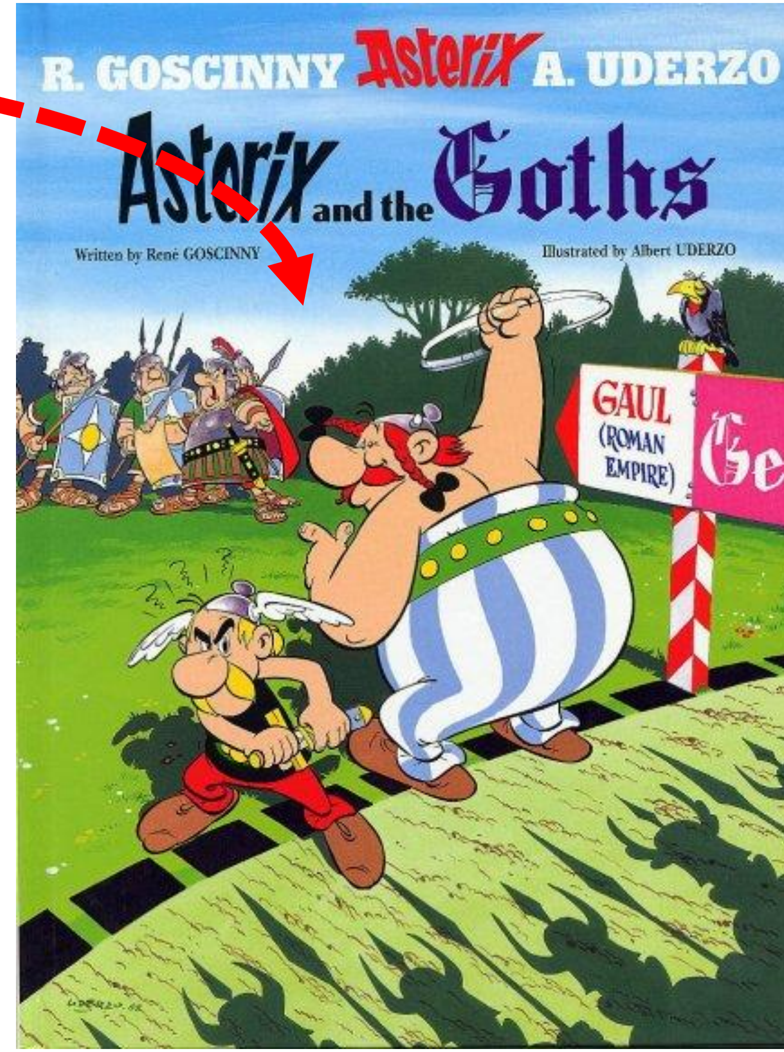
Trying to asssemble a consistent framework for Sentinel-3 cal/val and putting it in application

- General technical framework / environment: platform and management tools
- Used data
- Analysis and processing tools and applications: felyx, naiad, jupyter, and others
- Use cases
- Conclusions / way forward

A Britton in Germany…



- **Ifremer**, french marine institute, Brest, France
- **CERSAT**, satellite data center of Ifremer, part Laboratory for Ocean Physics and Satellite remote sensing (**LOPS**)
- Involvement in sea surface temperature community (GHRSST, OSI SAF) SST
- Validation and multi-sensor merging
- Long history at LOPS on cross-sensor synergy and intercomparison : colocation, indexing and search/extraction tools



http://cersat.ifremer.fr
http://www.umr-lops.fr

GENERAL S3 CAL/VAL FRAMEWORK

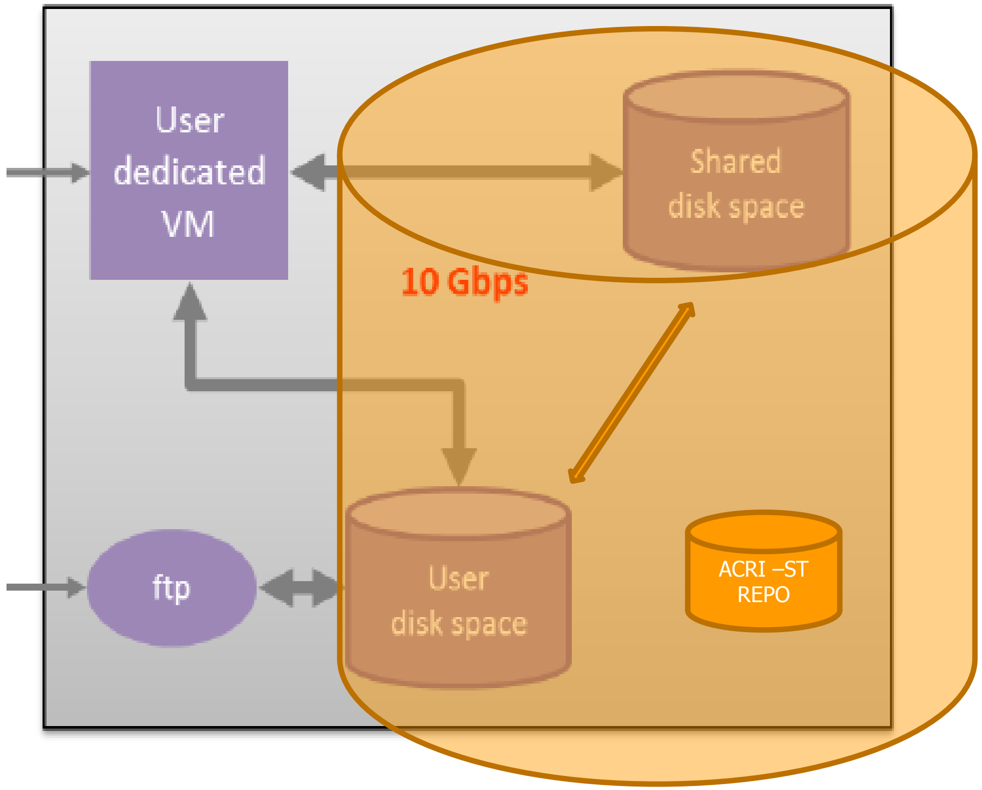
GSES

- GSES : «ground segment engineering service »

- Remote platform for Sentinel-3 cal/val
- Physically located in ACRI, Sophia-Antipolis, France

- Dedicated cloud for Sentinel-3 cal/val, shared by SST, ocean colour and altimetry subgroups

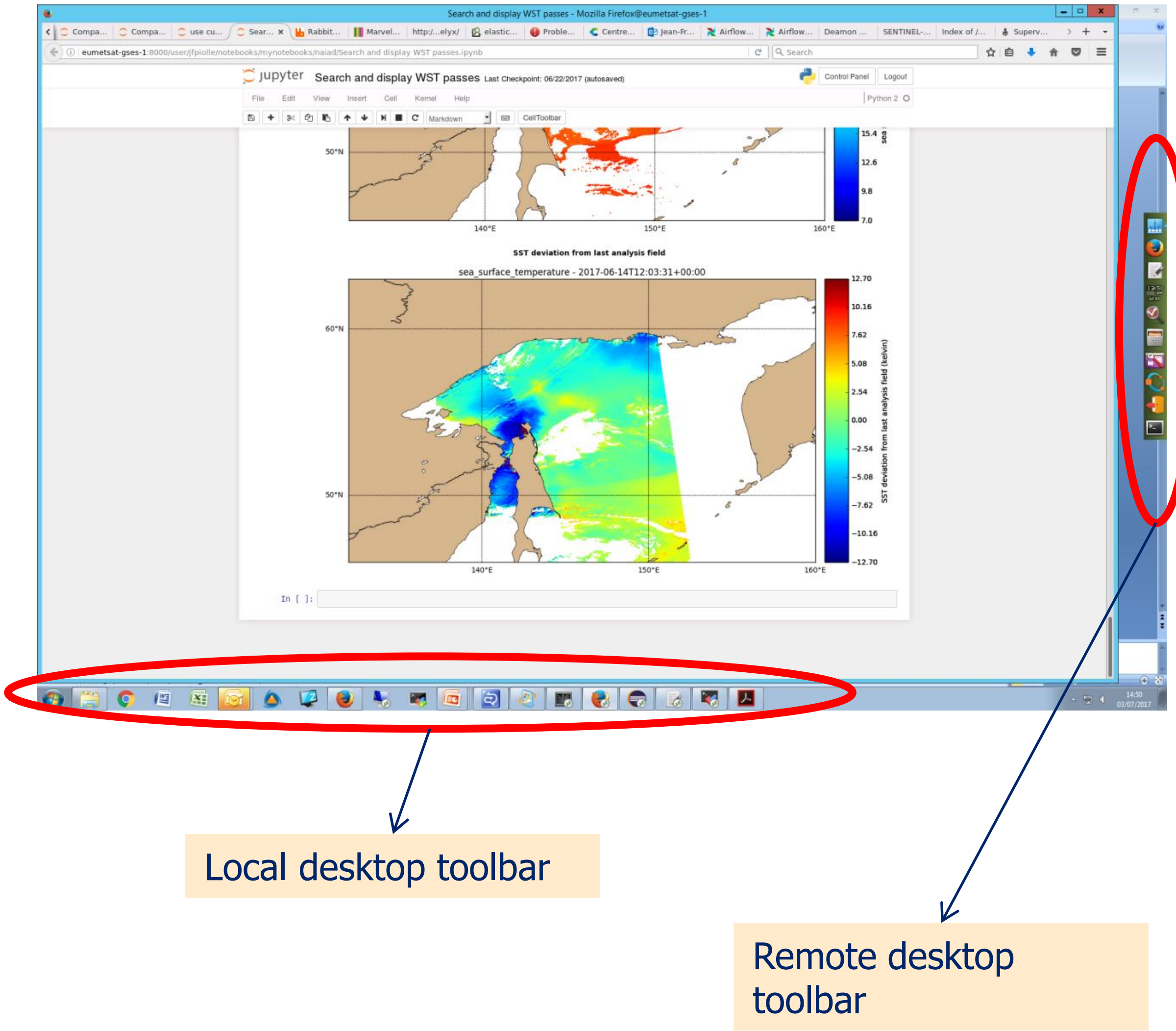
- Specs
    - 9 VMs (64 GB RAM, 8 cores)
    - storage : 1 PB
    - Ubuntu



EUMETSAT

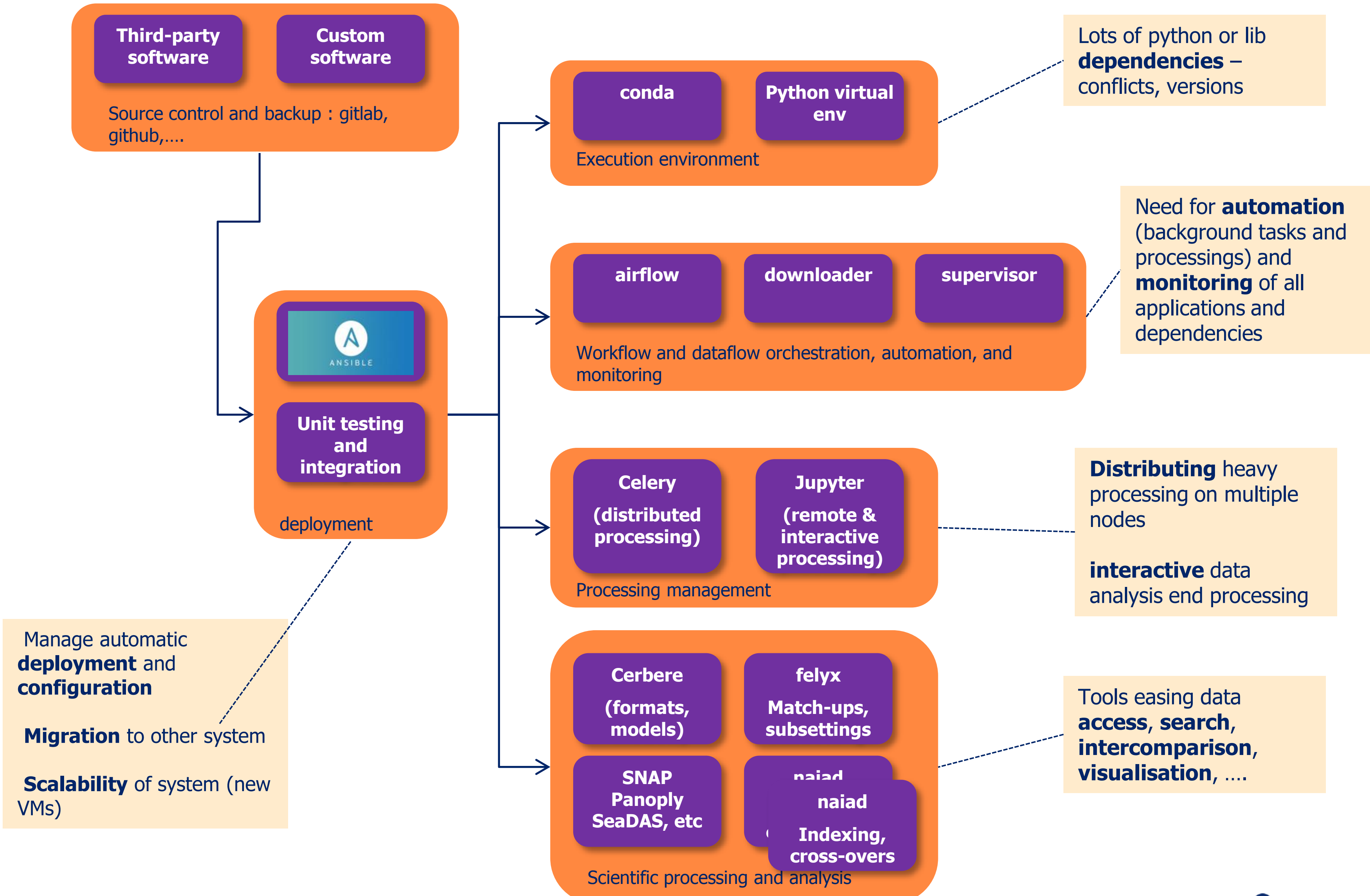
Access to GSES

- Remote desktop from windows or Linux : seamless access to a VM, merged with your own local environment – works pretty well
- Flexible deployment of software and applications
- FTP (and web) access from Eumetsat
- Same infrastructure used by ACRI for reprocessing : direct access to reprocessed data
- But : data duplication



EUMETSAT

Building the GSES cal/val environment



Environment / deployment management

Source control / Git

Eumetsat : https://gitlab.eumetsat.int/sen3/

Ifremer : https://git.cersat.fr/

All non Sentinel-3 or non GSES specific projects hosted at Ifremer

All software and configuration hosted on a Git server (serves also as backup)

Unit testing and continuous integration

Tested with gitlab / docker

Some unit testing implemented (and re-used for s3checker)

Premature wrt current software or GSES stability

Deployment with **ansible** (https://www.ansible.com)

Ansible is an IT automation tool

Deployment procedure (suite of instructions) described in a ***playbook***

Description of cal/val framework in a configuration file (hosts, storage, roles of each host, etc…)

Automatic execution of installation **and configuration**

Community contributions for open source COTS

Ansible

playbook

```yaml
# install supervisor
---

- hosts: supervisor
  # set to 'true'/'yes' to activate privilege escalation
  become: true
  become_user: '{{ oper_account }}'

  vars_files:
    - vars/configuration_vars.yml

  vars:

    # installed in operation space
    user: oper


  roles:
    - { role: supervisor,
        become: true,
        become_user: root,
      }

  post_tasks:

    - name: restart supervisor
      become_user: root
      service:
        name: supervisor
        state: restarted
      tags: supervisor

      # @TODO sudo update-rc.d supervisor defaults
```

High level keywords for common installation/configuration tasks (shorter than a script)

Nested call to sub-playbooks (« roles ») allow quick description of third-party dependencies installations (databases, etc…)

Hosts configuration

```ini
# hosts for staging environment
# -----------------------------

[all]
eumetsat-gses-1
eumetsat-gses-2
eumetsat-gses-3
eumetsat-gses-4
eumetsat-gses-5
eumetsat-gses-6
eumetsat-gses-7
eumetsat-gses-8
eumetsat-gses-9

[all:vars]
deployment_environment=staging

# user in which workspace to perform the installation
user=oper


# host for installation procedures (only one!)
# -------------------------------------------

[common_install]
eumetsat-gses-1

# hosts where to deploy a postgresql server
# ----------------------------------------

[postgres_servers]
eumetsat-gses-3


# supervisor hosts
# ---------------
# supervisor is used to start/stop/restart and monitor daemon process
# deploy supervisor on every host where on of the following services
# is running:
#     - airflow
#     - elasticsearch
#     - jupyterhub
#     - felyx-worker
#     - felyx-frontend

[supervisor]
eumetsat-gses-1
eumetsat-gses-2
eumetsat-gses-3
eumetsat-gses-4
eumetsat-gses-5
eumetsat-gses-6
eumetsat-gses-7
eumetsat-gses-8
eumetsat-gses-9


# jupyterhub
# ----------

[jupyterhub]
eumetsat-gses-1
```

List all the hosts of the framework and what software to deploy on

Installation can be replayed when adding new hosts

Execution environment for python

Conda (http://conda.pydata.org)

Because of the number of packages required for scientific computing and dependencies/version conflict issues, python is now rarely used through system level packages => python virtual environment

Conda provides pre-installed environments, with most used scientific packages or just packages with their dependencies

Extends concept of python virtual environment => manages C/C++ libs dependencies

Is also used in our case to manage our own in-house packages

Conda is also language agnostic

The regular cal/val user environment is *calvaluser* and can be loaded with the following command:

```
source activate calvaluser
```

If you want to leave this environment:

```
source deactivate calvaluser
```

You can list the other available conda environments (mostly for some background operation tasks) with the following command (though you should not need them if you are not a tool developer or a tool yourself!):

```
conda info --envs
```

Edit

Installed scientific packages

To list the available packages, just run:

```
conda list
```

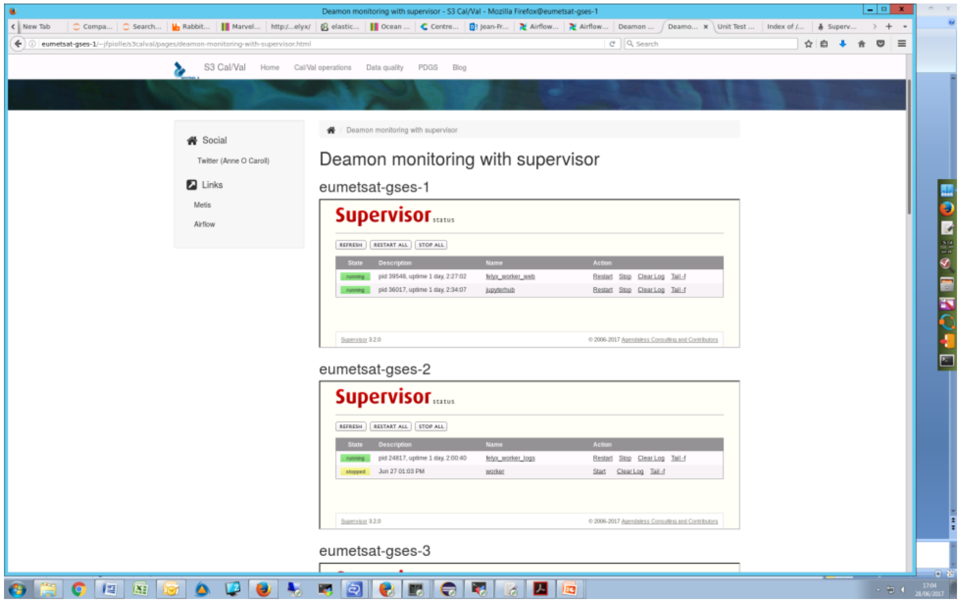
Edit

Predefined environments

calvaluser : for users, jupyterhub, etc...
calvaloper : for python softwares and routine processings
testenv : for local user testing

EUMETSAT

supervisor




```
command=bash launch-worker.sh
directory=/srv/supervisor/airflow
stopsignal=QUIT
stopasgroup=true
killasgroup=true
autorestart=true
user=s3ocean
stdout_logfile=/mount/common-storage/ogs/airflow-worker-
eumetsat-gses-4-stdout.log
stderr_logfile=/mount/common-storage/logs/airflow-worker-
eumetsat-gses-4-stderr.log
environment=AIRFLOW_HOME="/mount/common-
storage/workdata/staging/oper/stage_airflow",PATH="/mount/home/
s3ocean/staging/oper/miniconda3/bin/:/usr/local/sbin:/usr/local
/bin:/usr/sbin:/usr/bin:/sbin:/bin",AIRFLOW_SPOOL="/mount/commo
n-storage/workdata/staging/oper/stage_airflow/spool"
```

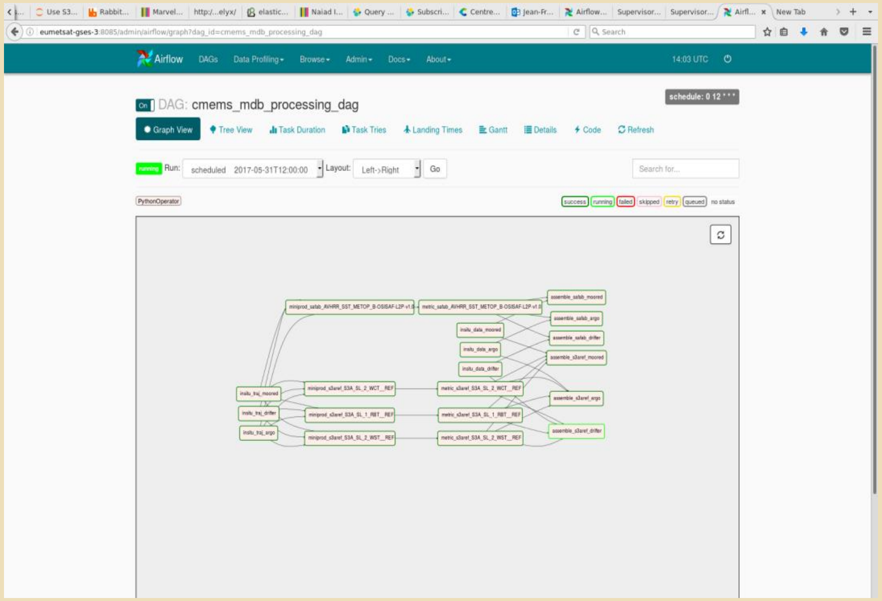
- Supervision of all background tasks (daemon)
  - Airflow
  - Felyx
  - Jupyterhub
  - …
- Ensures they stay alive
- Automatically restarts processed down
- Start/stop on demand process
- Status, access to logs
- Deployed on each host
- Centralized monitoring possible
- Automatically configured when deploying with ansible playbooks (when included in the playbook!)

http://supervisord.org

EUMETSAT

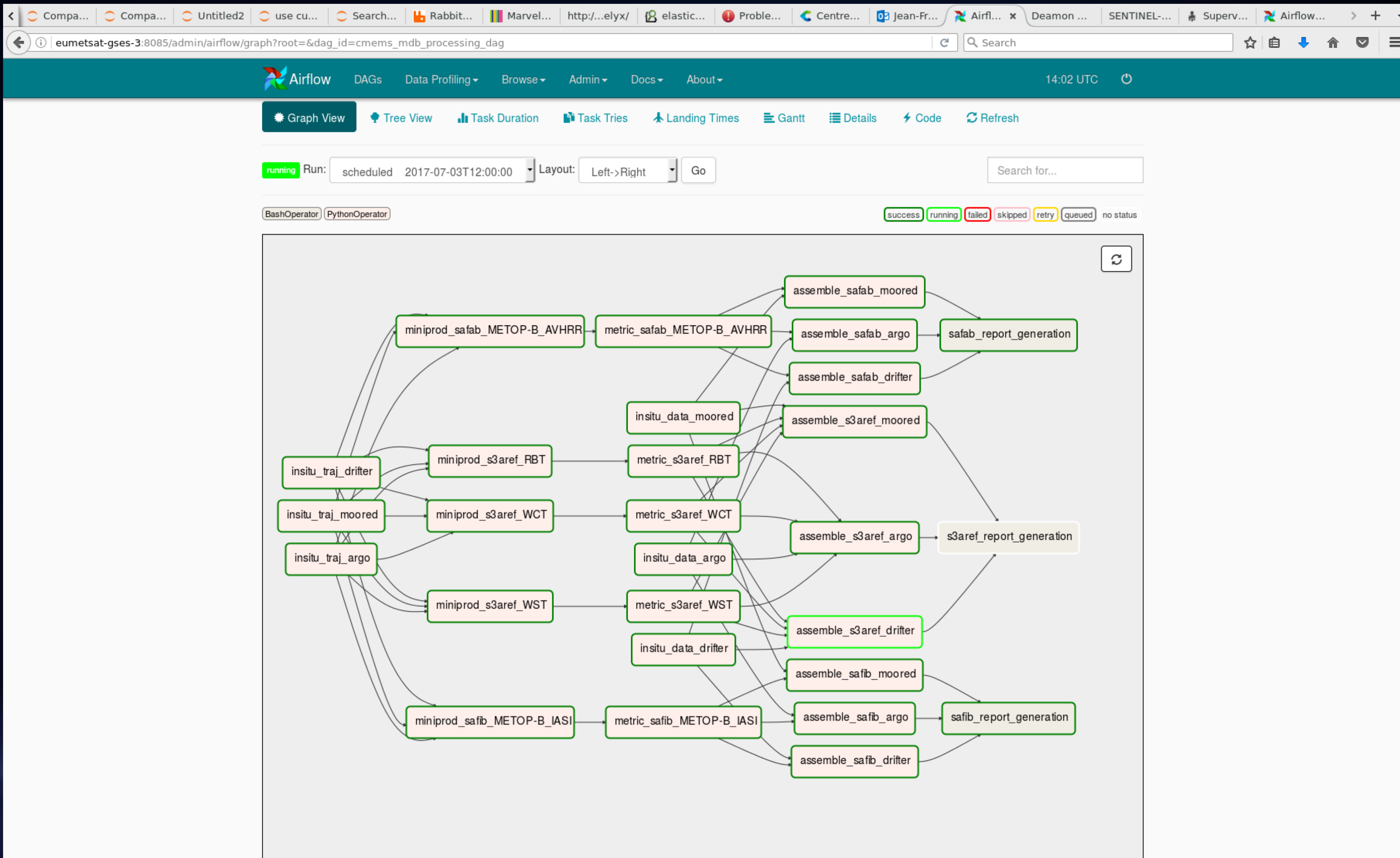
Airflow

- Task scheduler
- Schedules and runs processing workflows
    - Handle dependencies, conditions
    - Workflows implemented in python
    - Tasks distributed on different VMs (celery framework)
    - Alert through email and web interface
    - Quite complex to handle at first but very powerful framework for running and monitoring background tasks – time saver on the long run!!
- Deployed and installed on GSES with Ansible, controlled by supervisor
- High availability, very effective
- Absolutely essential for routine tasks



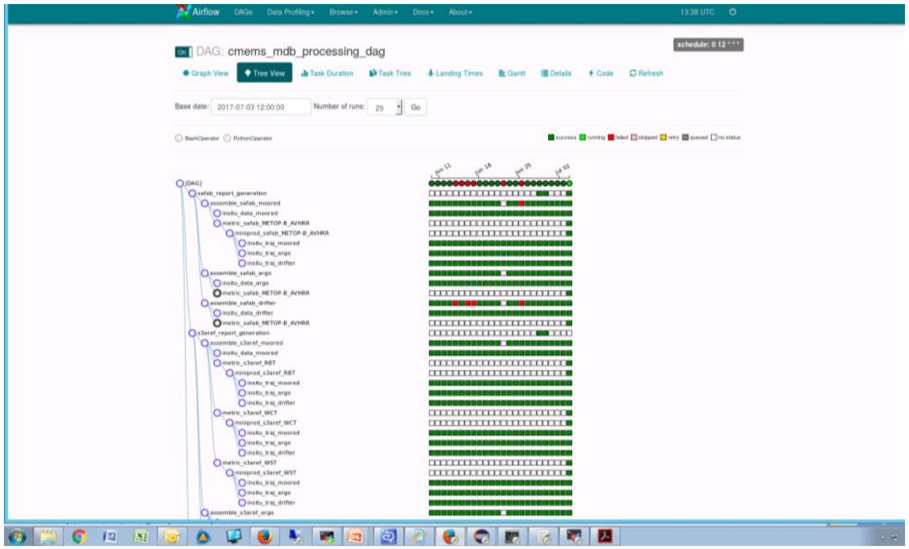
https://airflow.incubator.apache.org

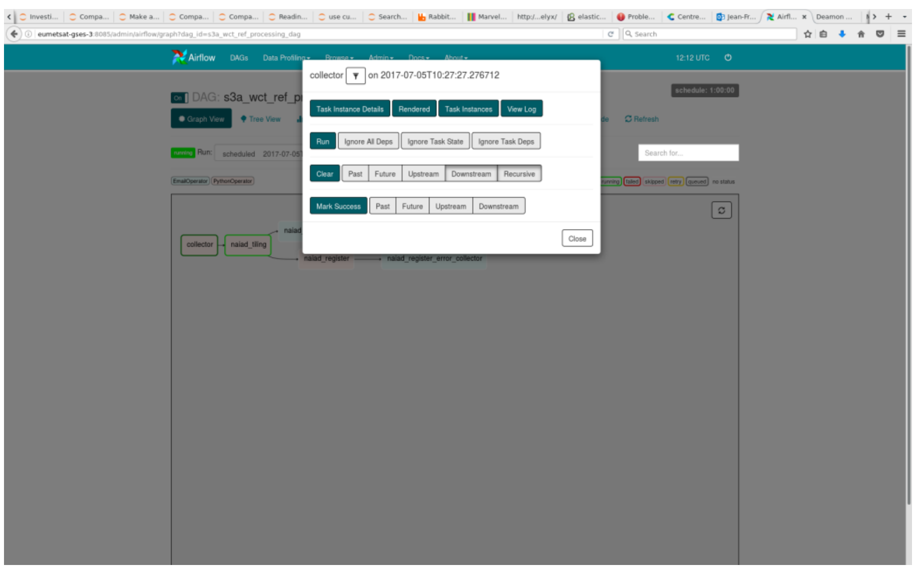
Airflow



EUMETSAT

Airflow






```python
# default arguments for DAG operators
args = {
    'owner': 's3ocean',
    'depends_on_past': False,
    # defines the rule by which dependencies are applied for the task to get
    # triggered
    'trigger_rule': TriggerRule.ALL_DONE,
    # max time allowed for the execution of this task instance, if it goes
    # beyond it will raise and fail.
    'execution_timeout': timedelta(hours=2),
    'email': ['igor.tomazic@eumetsat.int', 'jean-francois.piolle@eumetsat.int'],
    }

# check environment variables
# --------------------------

# DAG name
dag_name = "clean_logs_dag"

# processes
# ---------
clean_es_logs_task = "clean_es_logs"


# spools
# ------


# output directories
# ------------------



# DAG
# ---
# the DAG will be triggered daily (everyday at 10:00 UTC)

dag = airflow.DAG(
    dag_name,
    # the description for the DAG to e.g. be shown on the webserver
    description="Clean past logs",
    # Defines how often that DAG runs, this timedelta object gets added to your
    # latest task instance's execution_date to figure out the next schedule
    schedule_interval="0 1 * * *",
    # The timestamp from which the scheduler will attempt to backfill
    start_date=datetime.utcnow() - timedelta(days=2),
    # maximum number of active DAG runs, beyond this number of DAG runs in a
    # running state, the scheduler won't create new active DAG runs
    max_active_runs=7,
    # specify how long a DagRun should be up before timing out / failing, so
    # that new DagRuns can be created
    dagrun_timeout=timedelta(hours=24),
    # Perform scheduler catchup (or only run latest)? Defaults to True
    catchup=True,
    default_args=args
    )



def get_es_clean_command(date):
    """Return the command to clean es logs"""
    return (
```


EUMETSAT

Dataflows

Sentinel-3A data	versions
SL_1_RBT___	REF : reference – internal OPE : operational version REP : reprocessings (two) NRT : near real time NTC : non time critical Multiple IPF releases
SL_2_WCT___ (internal)	
SL_2_WST___	
OL_1_ERR___	
OL_1_EFR___	
OL_2_WRR___	
OL_2_WFR___	
SR_2_WAT___	

The data to assess

Comparison data	Source
OSI SAF METOP-B AVHRR L2P	OSI SAF (Ifremer)
OSI SAF SEVIRI L2P	OSI SAF (Ifremer)
OSI SAF METOP-B IASI L2P	OSI SAF (Ifremer)
OSI SAF METOP-A IASI L2P	OSI SAF (Ifremer)
NOAA VIIRS L2P	NOAA
METOP-B IASI L1	Eumetsat
METOP-B IASI L2	Eumetsat

Similar products / sensors to compare with

Reference data	Source
OSTIA	UK Met office
CMC	PO.DAAC
OSI SAF MDB NRT	Ifremer
CMEMS in situ data	Ifremer
Radiometer data	NOC
MOBY	NOAA
Aeronet	NASA
Rephy	Ifremer

The « ground truth »

Multiple dataflows to maintain updated

Multiple sources for download to monitor

Scripts + more advanced tools for easier management

« downloader »

Maintaining complete **up-to-date** collections of reference and S3A datasets

Ensure **completeness**, continuous update / replacement

Store data in a **common organization** (possibly different from provider)

Purge data history (rolling archive) without impact on download

Ensure **data integrity** (use of checksum when provided)

Monitor data source (access, availability of the data) and raise issues

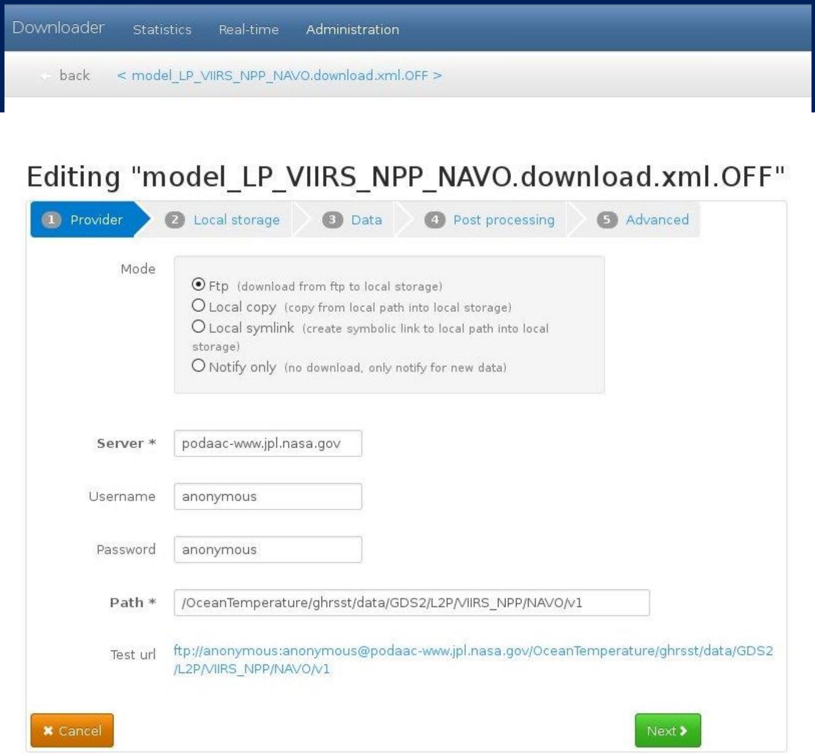
No on-the-shelf tool existing with enough intelligence to address all this issues

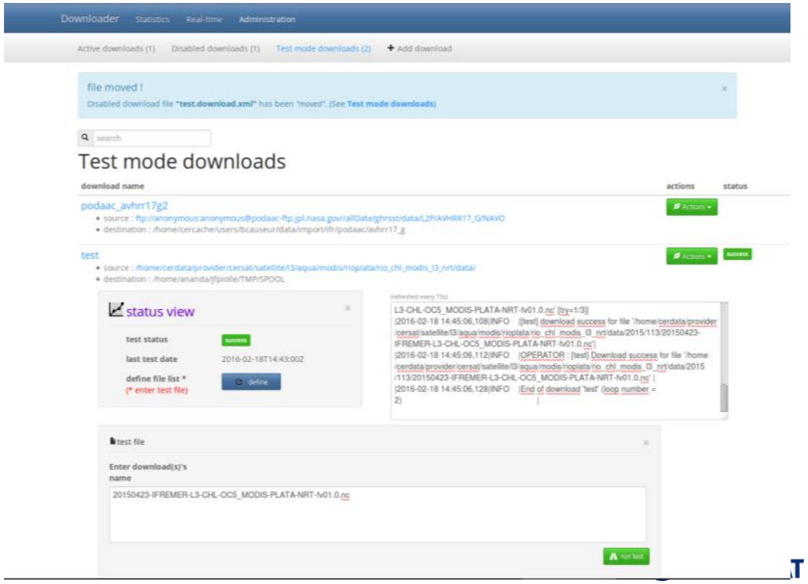
CERSAT tool, in python - Based on years of experimenting various issues with data mirroring (GHRSST GDAC, scatterometer,…)

Not a FTP client : built on robust existing client (lftp) for transfer issues (interruption, resume, integrity)

add « intelligence » layer to decide which data should be downloaded or not (not a pure mirror)

backend daemon and web front-end for configuration and monitoring





« downloader »

(Re)download configurable wrt any file change : timestamp, size

Filters on filenames to select relevant files only

Extract sensing time from filename for intelligent download (file selection or update based on sensing time)

Block or limit (re)download (time window)

Organize downloaded data (product/YYYY/DDD) whatever the organization at provider

Uncompress files

Check integrity if checksum provided

Sent notification of newly downloaded file (ex: to RabbitMQ) => data driven producer/consumer processing

Can be used to scan local network repo, symbolic link instead of copy => trigger data driven processing from a spool

Comes with configuration and monitoring web GUI

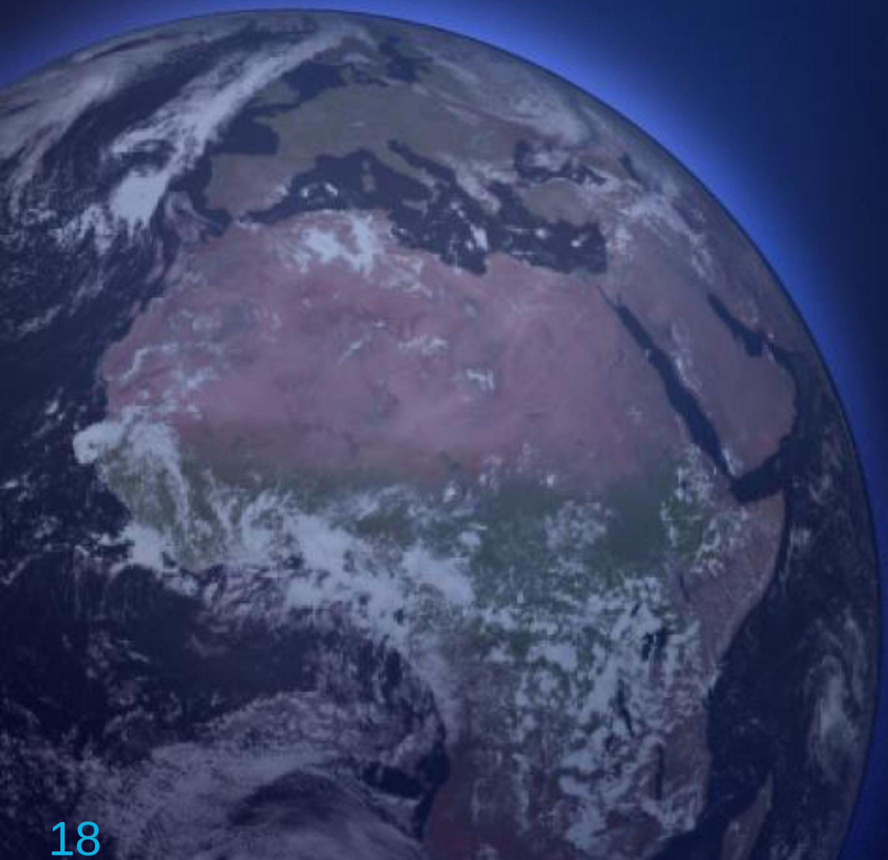
Operates as a background daemon : automatic and continuous update

Ongoing upgrade for SAFE and Datahub selection

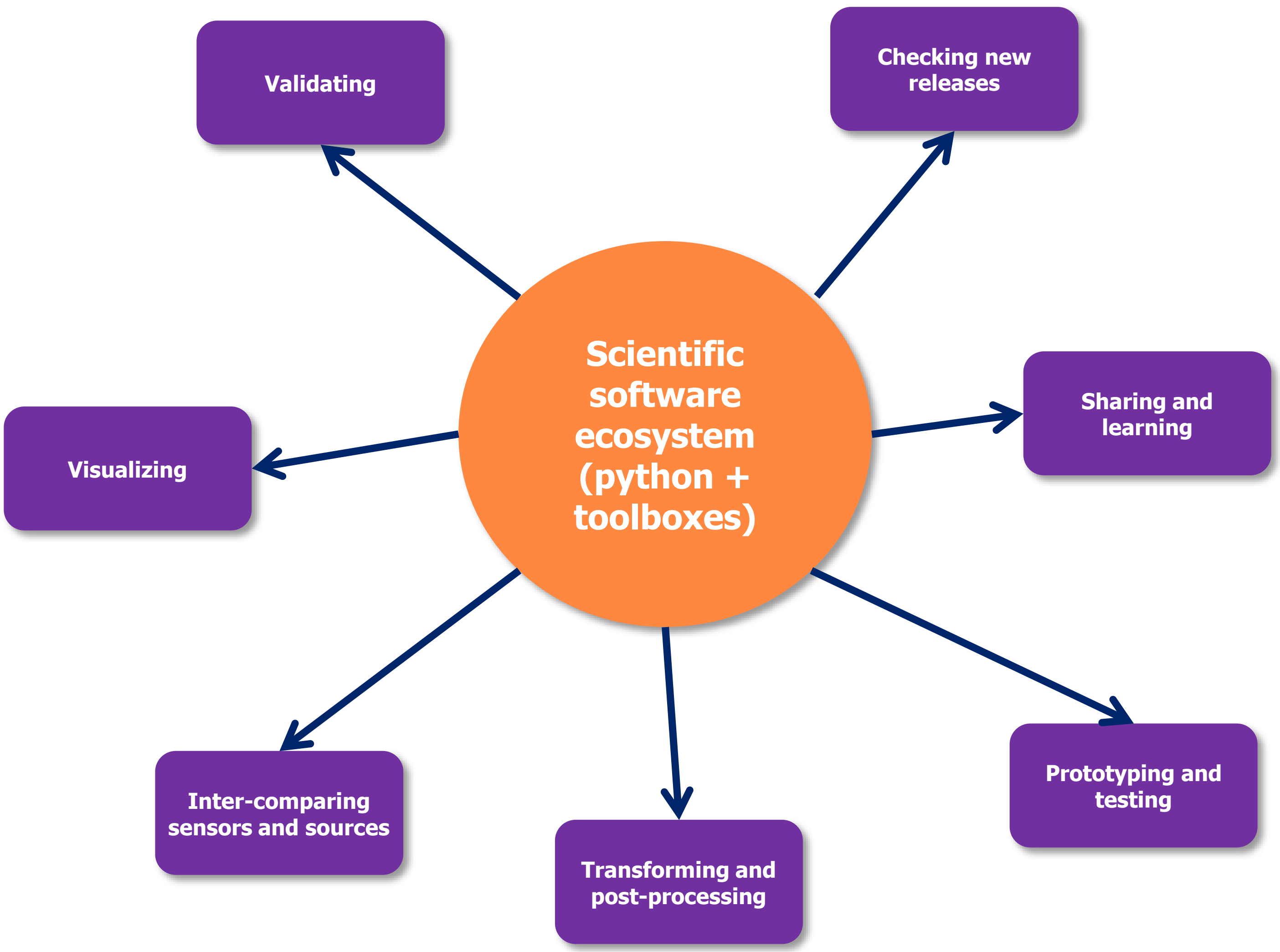
File selection options

Operation options

CALV/VAL TOOLS AND APPLICATION



scientific software framework



EUMETSAT

Python scientific packages

- Data readers
  - **cerbere** : generic data model classes and readers for various formats
  - **cerberecontrib-s3** : all OLCI and SLSTR products
  - **cerberecontrib-eps** : IASI L1 product for IASI, extendable to other products

- Data analysis
  - **cerplot** : display cerbere objects (swath, grids, etc…)
  - **cerinterp** : resampling of Cerbere objets (swath, grids, etc…)
  - **s3mdbreader** : read MDB data – abstract layers over files, helper functions for cloud filtering, closest valid pixel selection. **To be replaced**.
  - **s3analysis** : sandbox for S3 data analysis (SLSTR and OLCI). Converters for felyx in situ data, helper functions for cloud screening/quality level calculation, MDB filtering functions (@Gary), *MDB production statistics (@Gary)*
  - **sst_binner** : librairies and commands to build L3 from any SST products (L1 and L2)
  - **cloudleaks** : gross cloud leakage detection and automatic cross over detection with other sources, case browser (jquery) based on generated quicklooks
  - **naiad** : python bindings for swath data spatial/temporal and multi-criteria search or cross-over detection between sensors
  - **felyx_mdb** : commands for triggering a end to end match-up production workflow (used to process match-ups over a specific day for instance) – used also in Airflow

- Product verification
  - **s3checker** : systematic checks to test new product releases, bug fixes or corrections – should be updated continuously

Cerbere : data abstraction layer in python

- Generic **python API** to access and describe file content (different data formats) and observation patterns
- Abstract layer to build generic tools and applications upon it
- Implemented at Ifremer, used by a few other people, also access layer for softwares like felyx, naiad, syntool, cal/val tools and routines
- Generic data file model (similar to netCDF) – **mapper** :
  - Standard geolocation dimensions : row/cell, x/y, lat/lon, time,…
  - Other dimensions
  - Standard geolocation fields
  - Instrumental / geophysical fields : multi-dimensional arrays (incl.)
    - Variables attributes : no explicit scale factor, transformation performed in memory
  - Metadata (global attributes)

- Generic observation patterns - **datamodel**
  - Swath, Grid, Trajectory (along-track) , Image, TimeSeries, GridTimeSeries,….
  - Generic functions
    - save : format to similar format (dimensions, global attributes, etc…) any data following the same observation pattern
    - extraction of subsets, etc…
- Complemented by some companion packages
  - mappers for other formats (Sentinel-3/SAFE, IASI/EPS)
  - **Also alleviates complexity of SLSTR products**
  - generic packages based on the cerbere datamodel concept : ancillary fields, display, resampling/interpolation, ocean parameter calculation,

Doc/tutorial : http://cerbere.readthedocs.io/en/latest/

EUMETSAT

« big data » technologies

Elasticsearch : nosql type of database (alternative solution to SOLR, Cassandra, Hbase,…), with geospatial extensions : used for geospatial information indexing and search

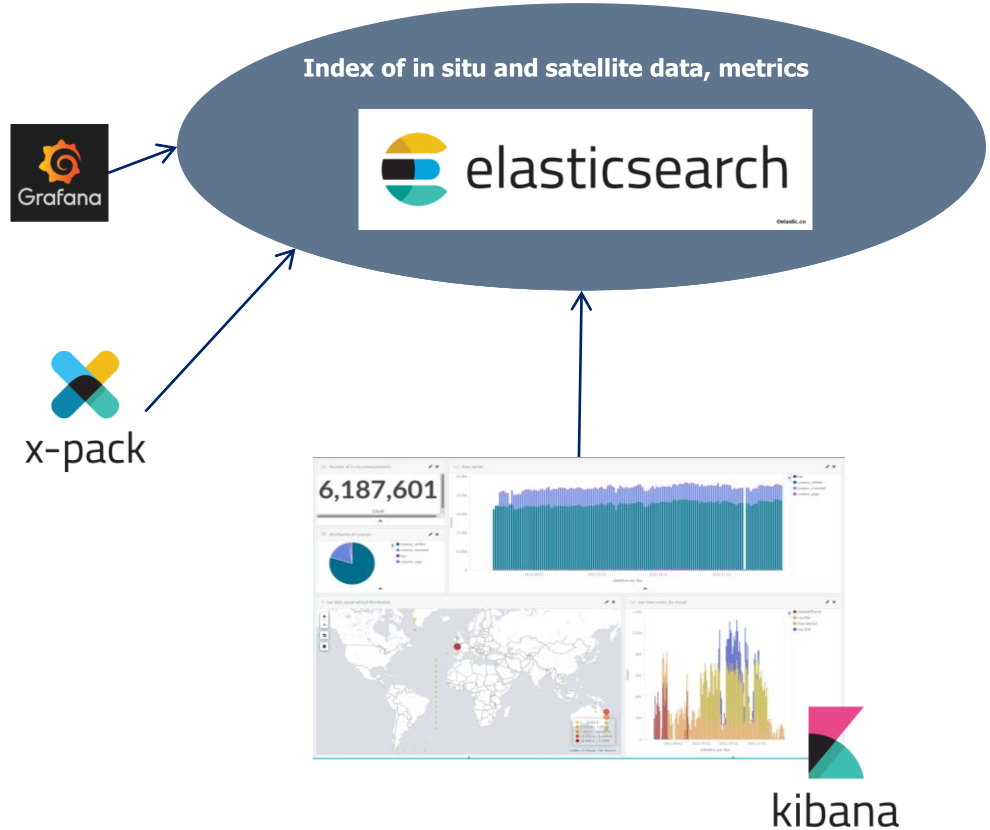
Take advantage of distributed environment (here GSES VMs) – scale easily

Several third-party tools and analytics tools to leverageits full power)

Analyzing, alerting, finding patterns

Barely scratching the surface now : machine learning, …

Two main tools based on : **Felyx** and **Naiad**



MATCH-UP DATABASES WITH FELYX

Background

Intercomparison of different sources of data is a key asset when working with earth observations

- Validation (cal/val) against in situ or other sensors
- Algorithm development and improvement
- Combination of different parameters from different sources (synergy, ancillary data,…)
- Monitoring and detection of issues

Today's sensor reach data **volume** and available **bandwidth** limitations of most users, plus **complexity** of managing multiple datastreams

Tools are required to extract the **relevant amount of information** only to perform the above tasks

felyx

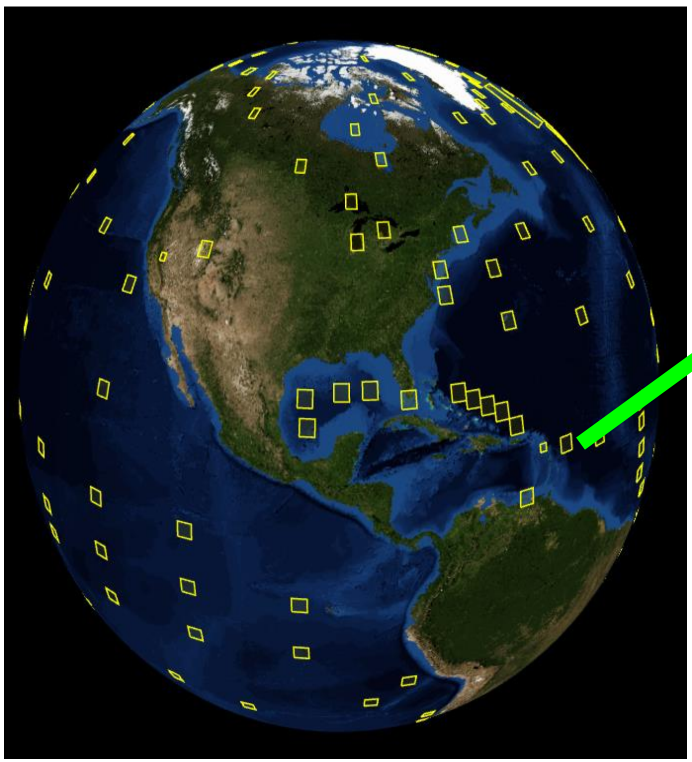
- Intended for **satellite to in situ match-up extraction** and systematic data **extraction over user defined area or locations :**
  - Command line based – query through RESTful and python APIs.
- **Main functions**
  - Extraction of file subsets over static or moving locations
  - Extraction and indexing of metrics over the subsets for analytics
  - Assembling with in situ data
- **Main outputs**
  - Miniprods and metrics
  - Assembled multi-sensor match-up files
  - Display of metrics, alert detection through analytics tools
- **Implementation**
  - Open source software in python
  - Relies on existing open source frameworks for big data and distributed processing : **ElasticSearch**, **RabbitMQ**, **Celery**, ….

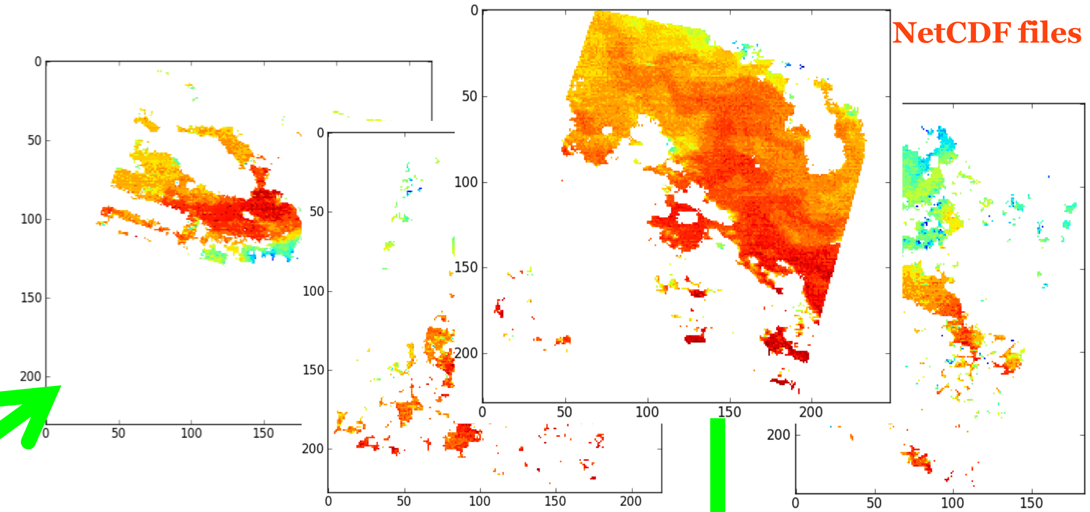
elasticsearch RabbitMQ

Felyx for MDB production

extract **miniprods** (subsets) over static and dynamic sites

process quantitative, qualitative, stat metrics over miniprods





NetCDF files

source: 20130101-IFR-L4_GHRSST-SSTfnd-ODYSSEA-GLOB_010-v2.0-fv1.0.nc
felyx_dataset_name: ifr-l4-sstfnd-odyssea-glob_010_v2.1
percentage_coverage_of_site_by_miniprod: 100.0
date_modified: 2014-04-18T10:30:21
felyx_site_identifier: ukm005
date_created: 2014-04-18T10:30:21
time_coverage_start: 2013-01-01T00:00:00
time_coverage_stop: 2013-0101T00:00:00

sst_standard_deviation : 1.34
mean_sst : 286.289
ice_presence: 0
cloud_presence": 46.80
day_or_night: "night"
mean_wind_speed: 4.8388

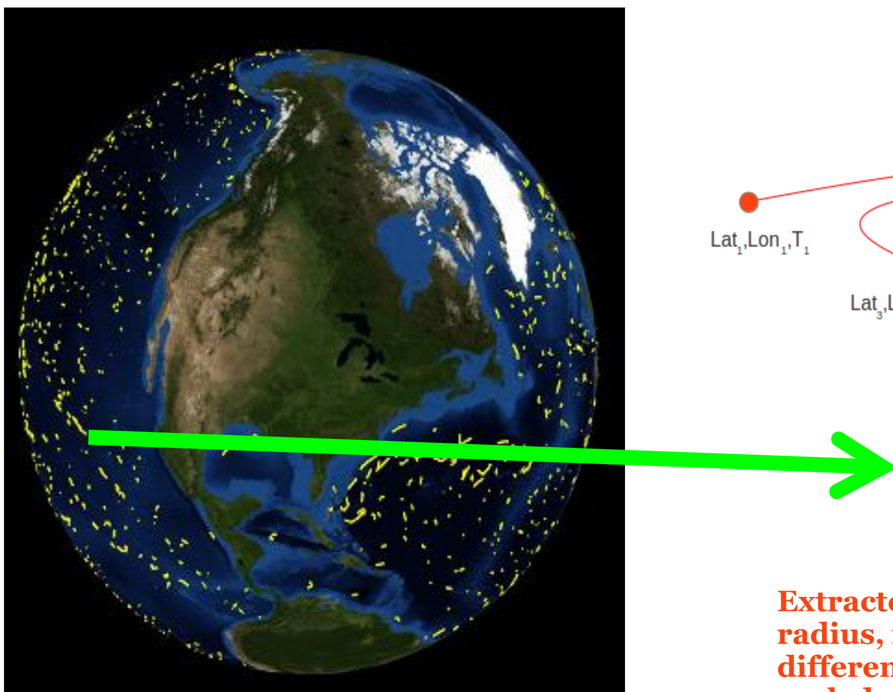
JSON files indexed in a search engine (ElasticSearch)

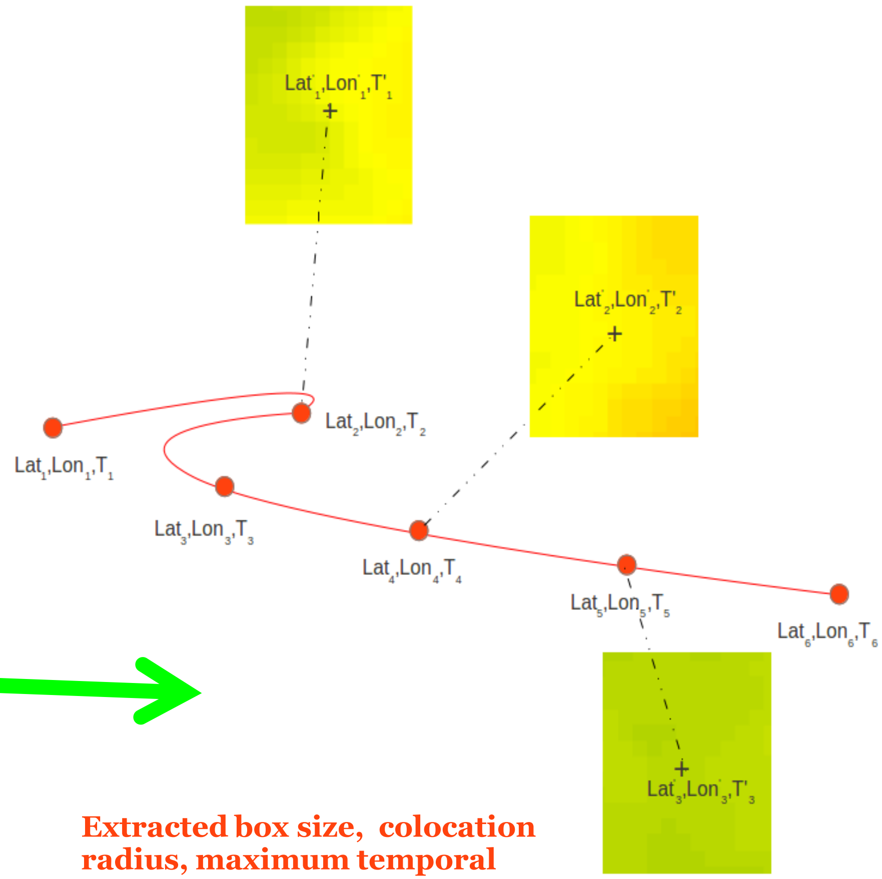
Felyx for MDB production

sites may be trajectories (buoys, cruise, hurricane)

MINIPROD's centred on trajectory locations closest in time locations closest in time

trajectory files ingested through import web service (CSV file)





Extracted box size, colocation radius, maximum temporal difference can be adjusted for each dataset

In situ sources

- Benefit on general frameworks:
  - CMEMS
    - Integration with Copernicus/CMEMS service for the provision of moored and drifting buoys and Argo data : collection and availability of all data in the same format and quality control
    - Canadian & european GDACs for surface drifters being created
    - Expected improvements in quality control and metadata

  - in situ radiometer

    - http://www.shipborne-radiometer.org/
    - High quality data
    - Common format and content has been agreed
    - Shared repository will be soon available assembling all these data
    - Currently used in felyx : cruises from ABoM, NOC, RSMAS and DMI

- All these data formatted in felyx format and available on ftp for ingestion into other MDB (request jfpiolle@ifremer.fr)
- Felyx + in situ data : framework for consistent MDB production for each GHRSST product

Felyx match-up database workflow

Colocation window : 2h (12h for Argo), 5km

21 x 21 pixel boxes

+/- 6h of in situ data history

In situ data :

- Copernicus/CMEMS (Coriolis)
- ISAR radiometer on opportunity ships (delayed-mode)

Sentinel-3 data :

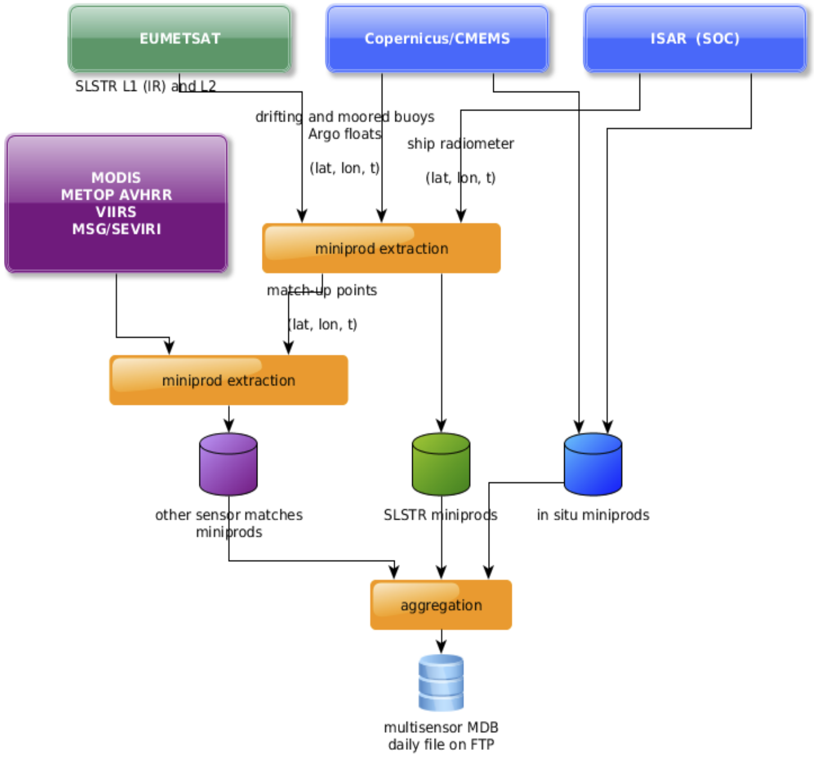
- L1 infra-red channels
- L2 (SST) – all fields, incl. meteo and ancillary fields

Other sensor data

- Metop-B/AVHRR, MSG/SEVIRI, OLCI, (MODIS, VIIRS)

Resampling of all data to SLSTR grid

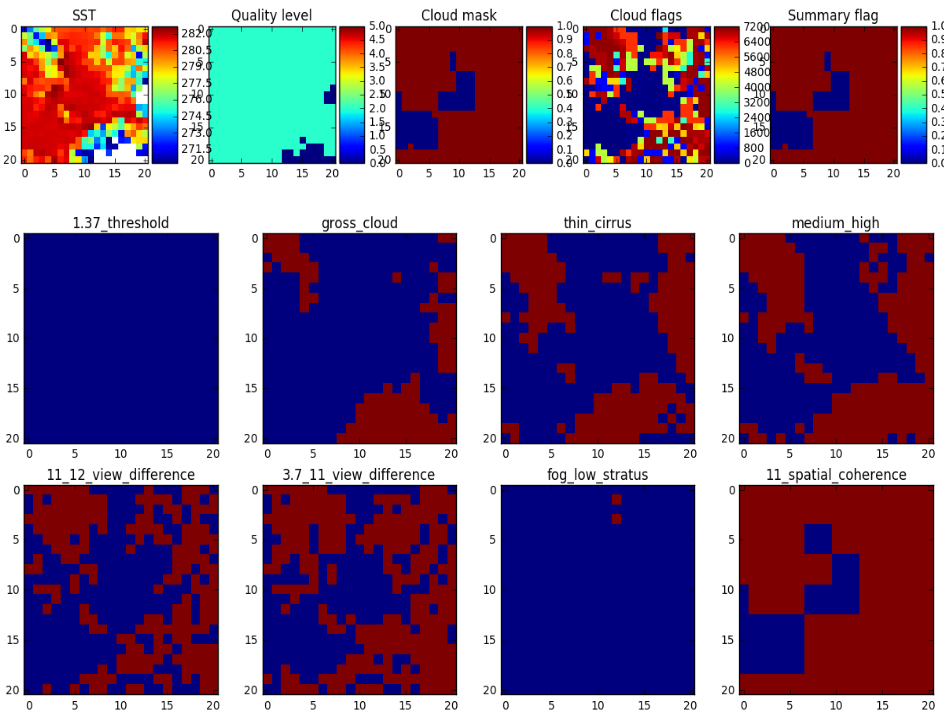
Daily aggregated match-up files on FTP : stack all matchups into a single file.



EUMETSAT

Content of match-up

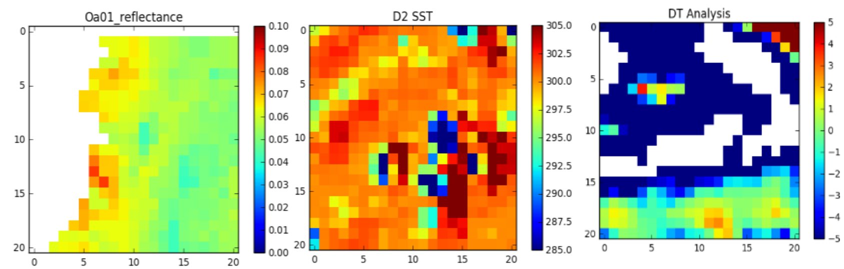
All fields from RBT (L1), WCT and WST (L2)



More than 600 variables from L1 to L2…..

21x21 boxes extracted with all fields for each match-up can be used to test and assess new algorithms or post-processing on a larger scale and time period in a fast way, with in situ information to directly estimate theimprovement.

All fields from cross-overs and complementary files



Cross-over fields from OLCI, METOP, VIIRS

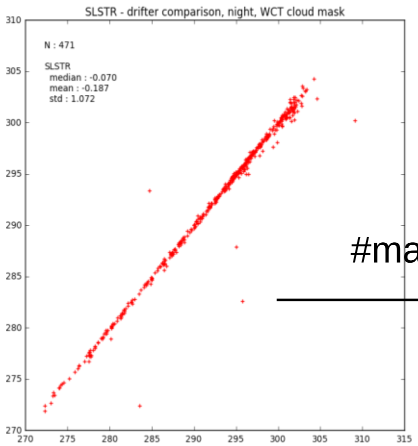
Complementary files from post processing of match-ups (prototype SST, quality level, etc…)

Ancillary fields (OSTIA dSST)

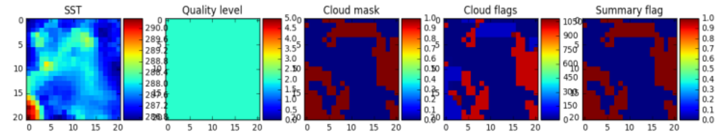
In situ buoy history centered on match-up

EUMETSAT

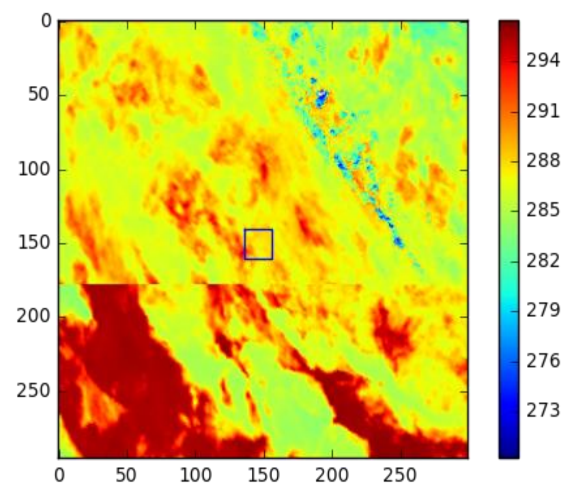
Traceability to source information



#match-up to full box



#Full box to #source granule (and #offsets within granule)



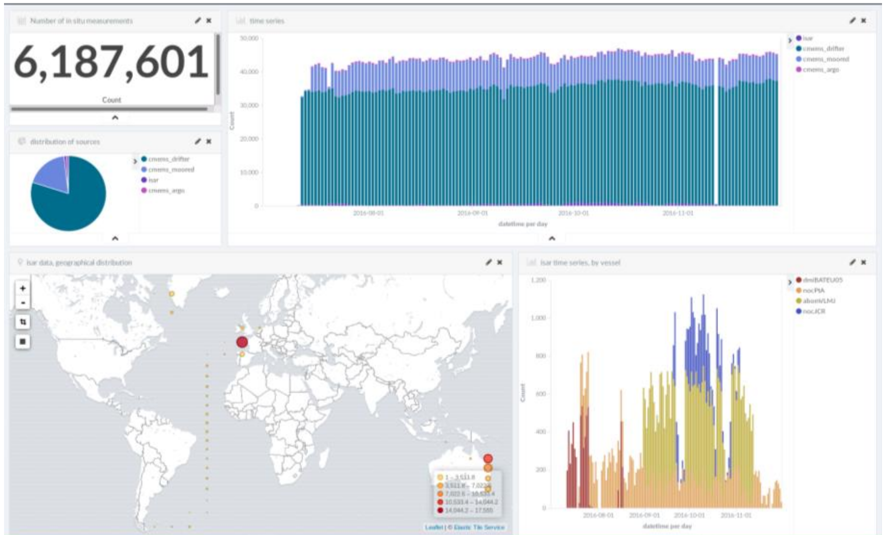
Full traceability of information in match-up

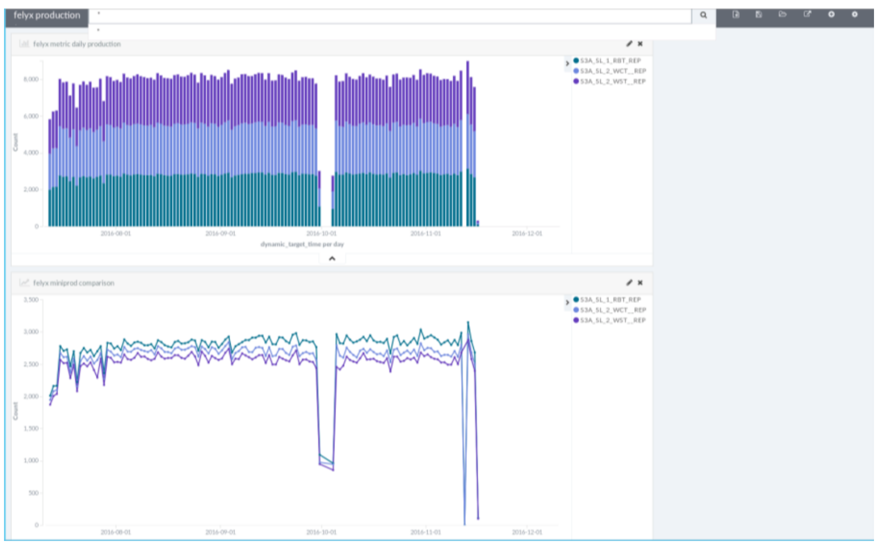
Traceability analysis example with jupyter notebooks

Existing match-up databases

Match-up database	Primary products	Complementary products	Availability
OSI SAF **SLSTR** MDB	SLSTR NRT products (**OPE**)	METOP (about 50%) SEVIRI (about 30%) SST prototype OSTIA	July 2016 - present
Eumetsat **SLSTR** MDB	SLSTR NRT products (**REF**)		May 2017 - present
Eumetsat reprocessed **SLSTR** MDB	SLSTR REP v4	OLCI SST prototype OSTIA	July – Nov 2016
Eumetsat reprocessed **SLSTR** MDB	SLSTR REP v5	SST prototype OSTIA	Nov 2016 - April 2017
Eumetsat **IASI** MDB	Eumetsat & OSI SAF L2P METOP-A METOP-B IASI		June 2017 - onward
Eumetsat METOP-B **AVHRR** MDB	OSI SAF L2P METOP-B AVHRR		June 2017 – onward
Eumetsat **OLCI** MDB	OLCI L2 WFR		July 2017 - onward

Match-up content statistics



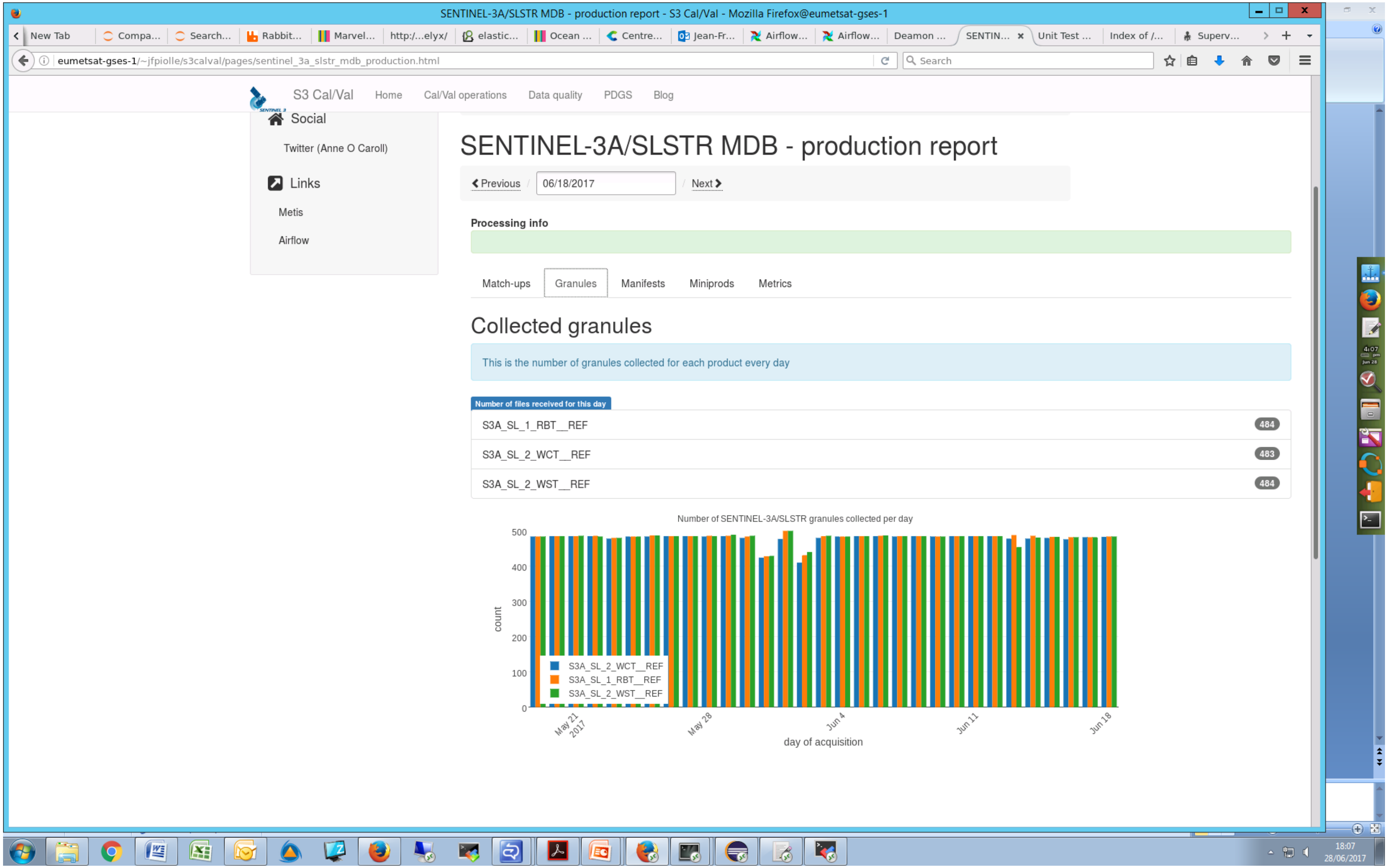


Typical match-up distribution for SLSTR, all weather conditions :

- more than 40.000 in situ measurements per day

- ~2000 match-ups / day for buoys
- ~350 match-ups / day for moored buoys
- ~600 match-ups / day for argo floats

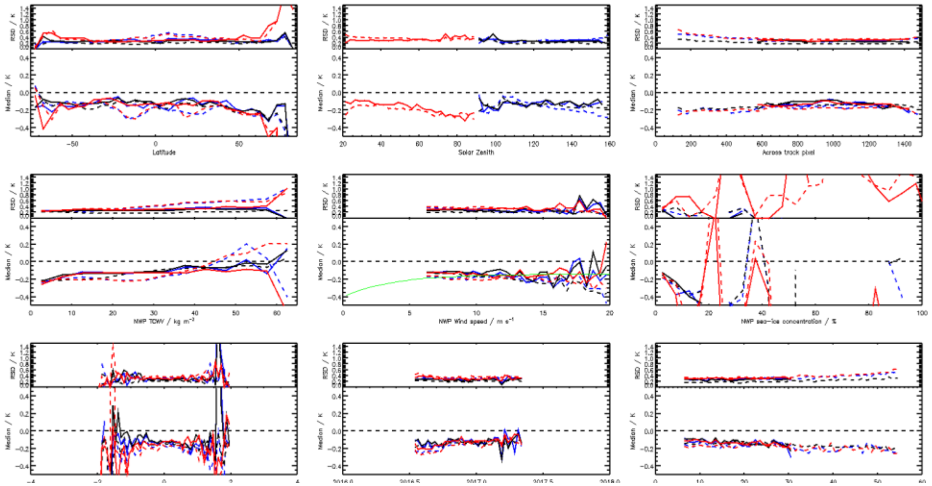
Production monitoring

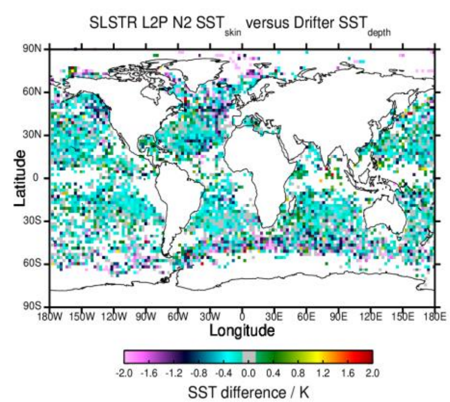


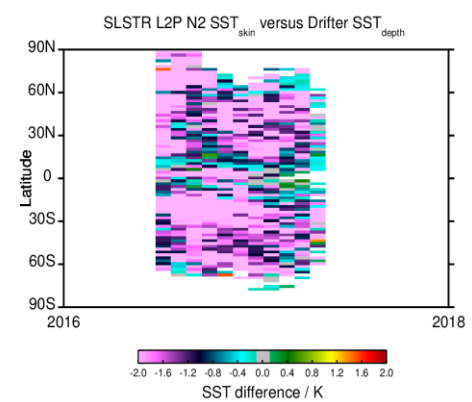
Application of SLSTR MDB(s)

- Used by different groups at Eumetsat, within S3VT and MPC Sentinel-3

- Major asset in:
  - L1 cloud screening validation (RAL)
  - L2 SST coefficient estimation (Univ. Of Reading)
  - L2 Quality level stratification and uncertainties estimation (Univ. Of Leicester)
  - SST validation : OSI SAF (Meto-France / DMI / MetNo), NOAA, Eumetsat
  - Metis intercomparison framework







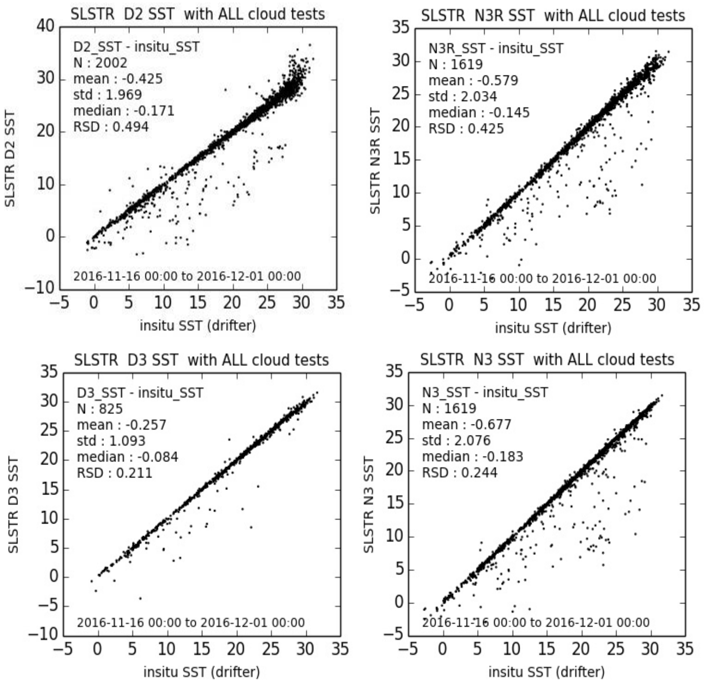
Quality monitoring statistics to be updated periodically for control and monitoring

Courtesy: G.Corlett, Univ. Of Leicester

EUMETSAT

OSI SAF SLSTR federated activity

- Funded by Eumetsat
- SST experts from Ocean & Sea ice SAF (Meteo-France, DMI and MetNo)
- Global assessment and specific on high latitudes with in situ data collection from ISAR in situ radiometer onboard arctic sea cruise and drifters + sea ice temperature
- Based on felyx generated match-up databases



(WCT_SST – insitu_SST) with cloud clearing REC

algo	N	mean	std	median	RSD
N2	5293	-1.520	3.449	-0.330	0.560
N3R	1911	-0.617	2.098	-0.146	0.430
N3	1911	-0.727	2.165	-0.192	0.245
D2	2653	-0.735	2.654	-0.188	0.561
D3	966	-0.294	1.179	-0.096	0.209

26586 cases, 5299 clear cases **19.9 %**

(WCT_SST – insitu_SST) with cloud clearing ALL

algo	N	mean	std	median	RSD
N2	4130	-0.973	2.608	-0.296	0.479
N3R	1619	-0.579	2.034	-0.145	0.425
N3	1619	-0.677	2.076	-0.183	0.244
D2	2002	-0.425	1.969	-0.171	0.494
D3	825	-0.257	1.093	-0.084	0.211

26586 cases, 4133 clear cases **15.5 %**

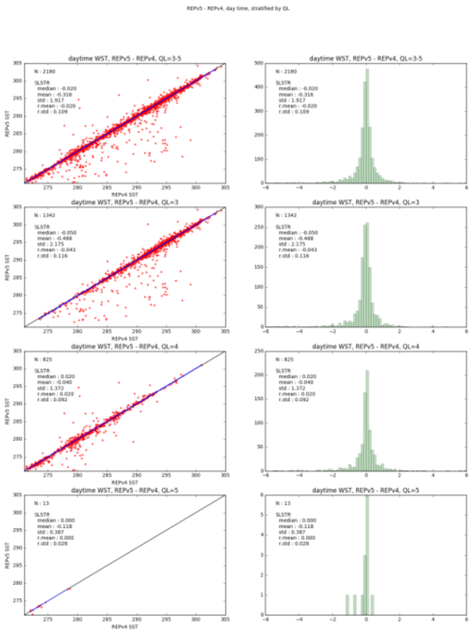
No correction : skin WCT SST vs bulk insitu SST

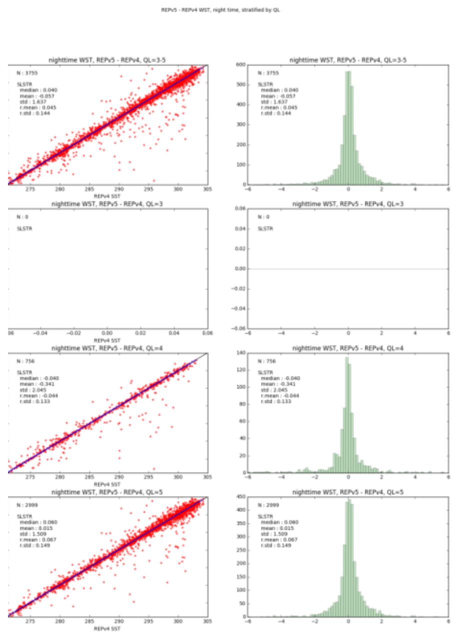
Courtesy: Anne Marsouin, Meteo-France

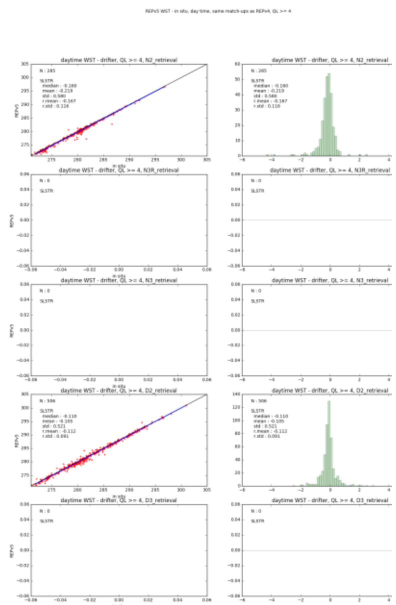
Intercomparison of MDBs

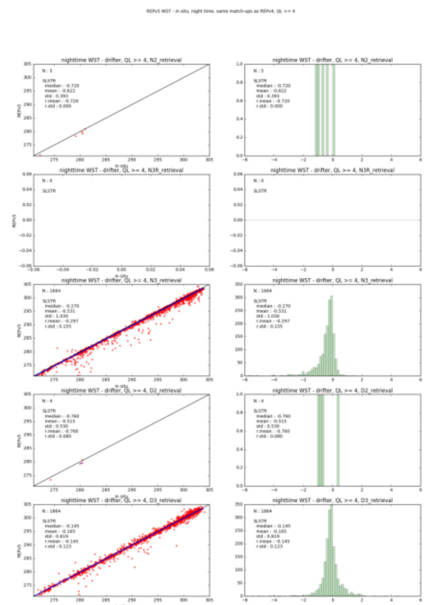
Assessment of algorithm improvements

All match-ups are uniquely identified through buoy id and time and location : this makes easy to intercompare different versions of product with each other, through « match-ups of match-ups » (left) or respective comparison of each version to the same in situ values (right)



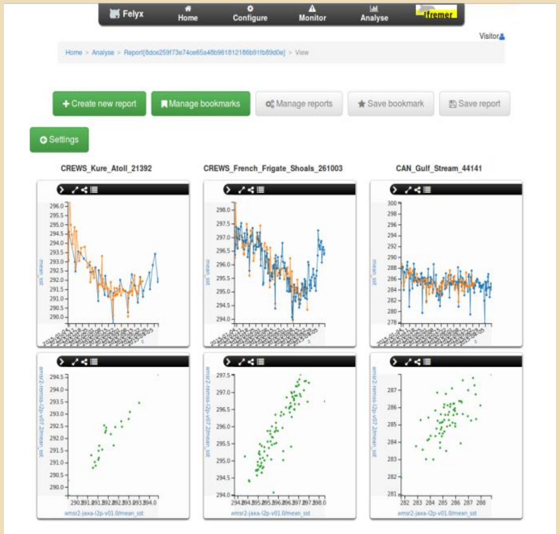






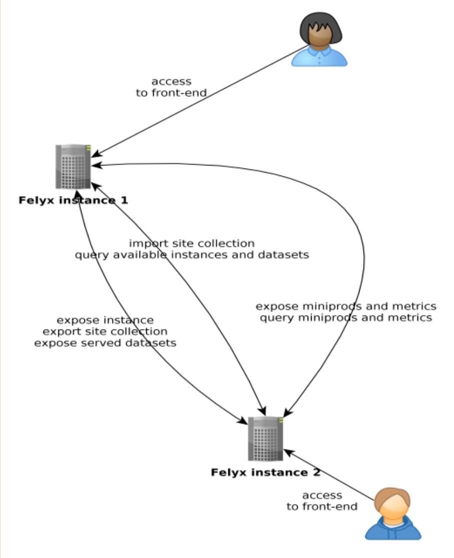
Comparison of reprocessing v5 vs v4 for SLSTR SST product

Other felyx features

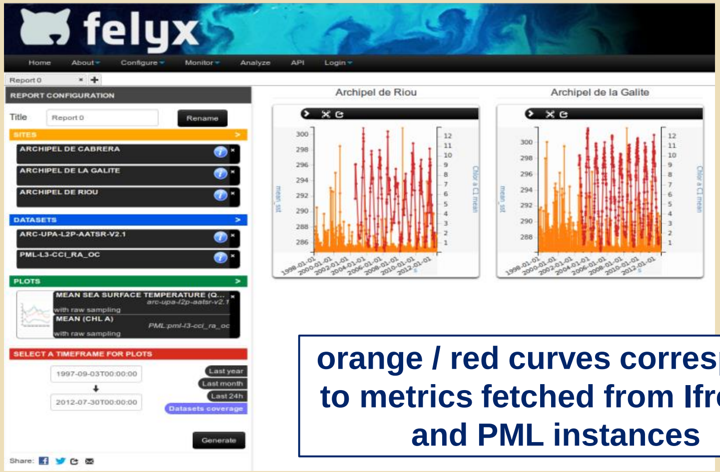


Felyx natively embeds a web front-end with plotting capabilities for the match-ups and miniprods

Ability to design reports, automate share them through a repository



Federated queries



orange / red curves correspond to metrics fetched from Ifremer and PML instances

NAIAD : DATA INDEXING AND SEARCH

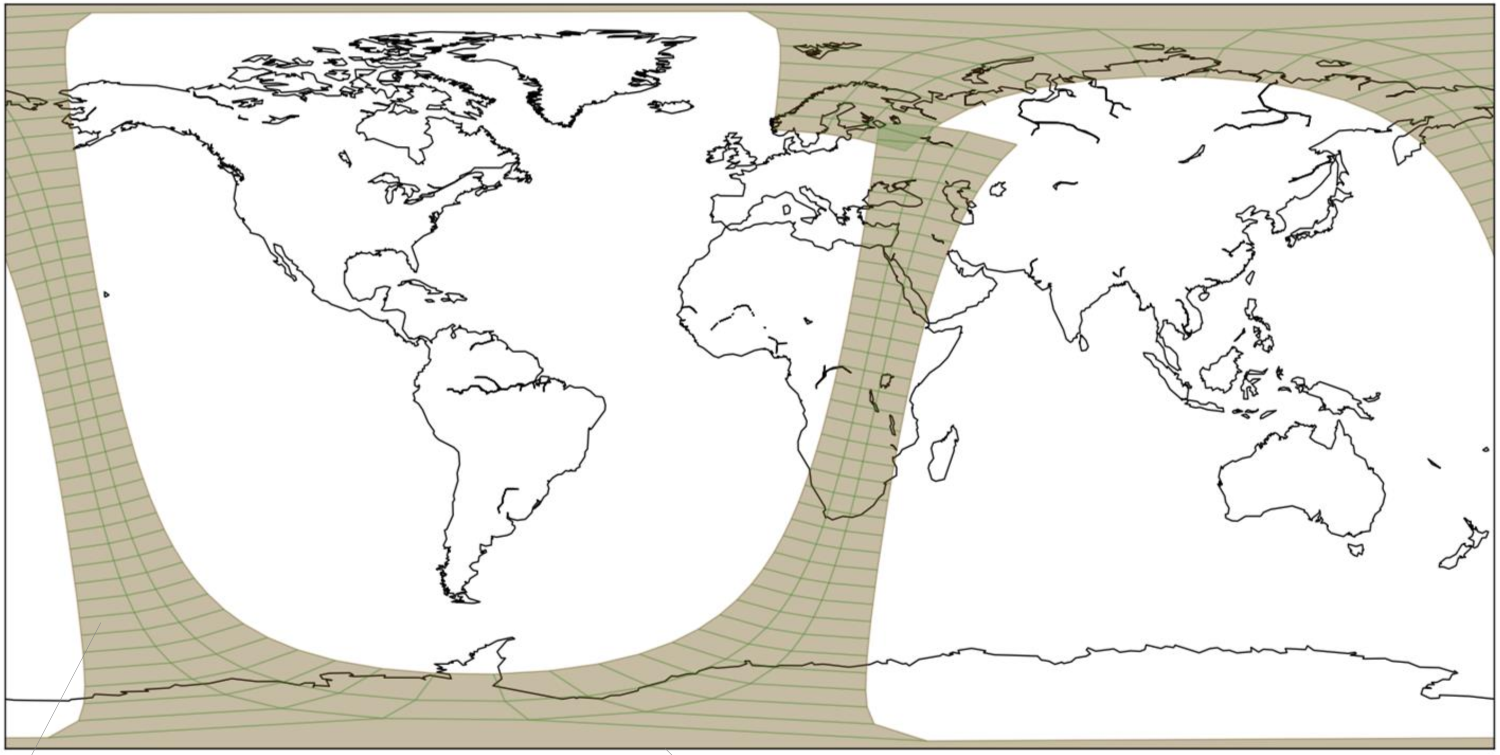
Naiad

- Intended for **satellite to satellite cross-over** detection
- Indexing of observation data as temporally bounded geographical shapes
- Command line or API based
- **Main functions**
  - Search file or file subset wrt multiple criteria : spatial, temporal, properties and metadata
  - Cross-search in different datasets, with time window constraint (cross-overs)
- **Main outputs**
  - List of file subsets (file name, indices)

http://naiad.readthedocs.io/en/develop/

Naiad ENHANCED SATELLITE ARCHIVE DATAMINER

Naiad – data tiling



Product / file

Tile geographical shape (polygon)

temporal coverage

any numerical (quantitative) or text (tag, qualititative, metadata) properties

Product / file

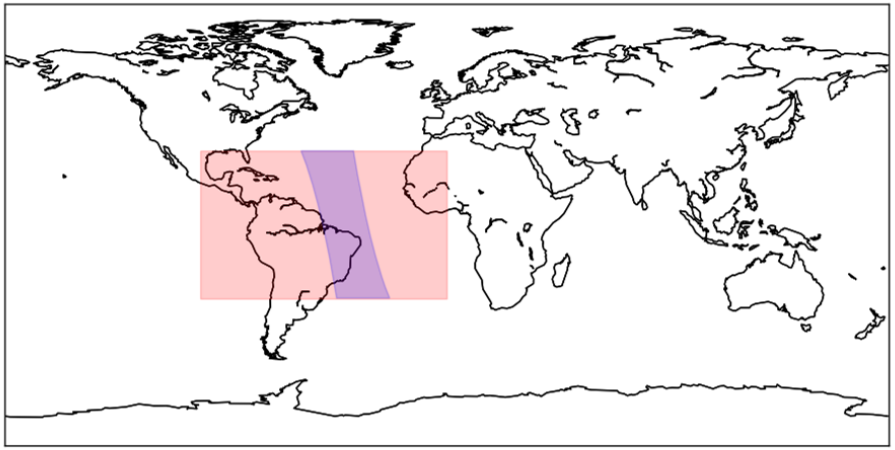
Sensor footprint geographical shape

temporal coverage

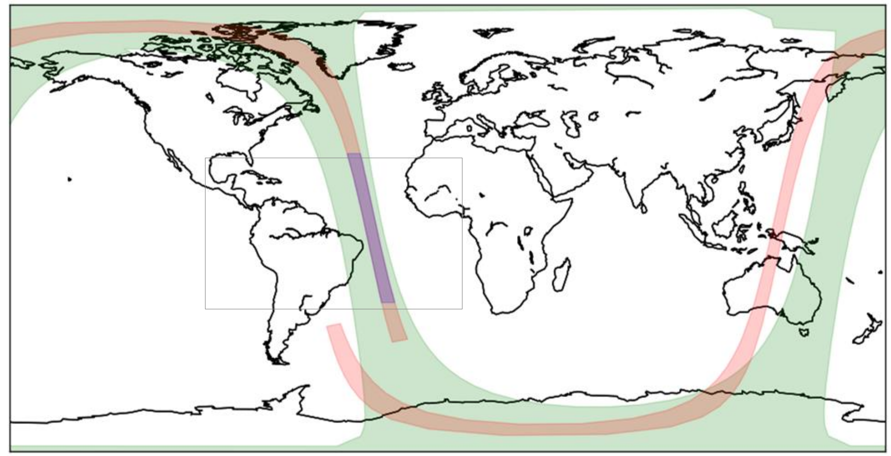
any numerical (quantitative) or text (tag, qualititative, metadata) properties

Naiad - queries

Simple search



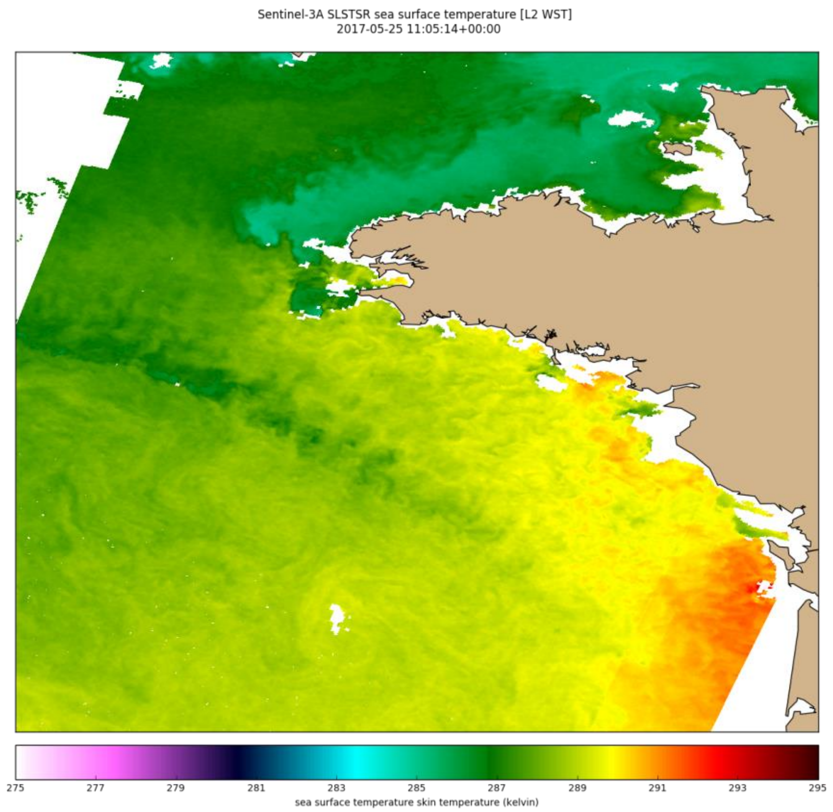
Cross-over search

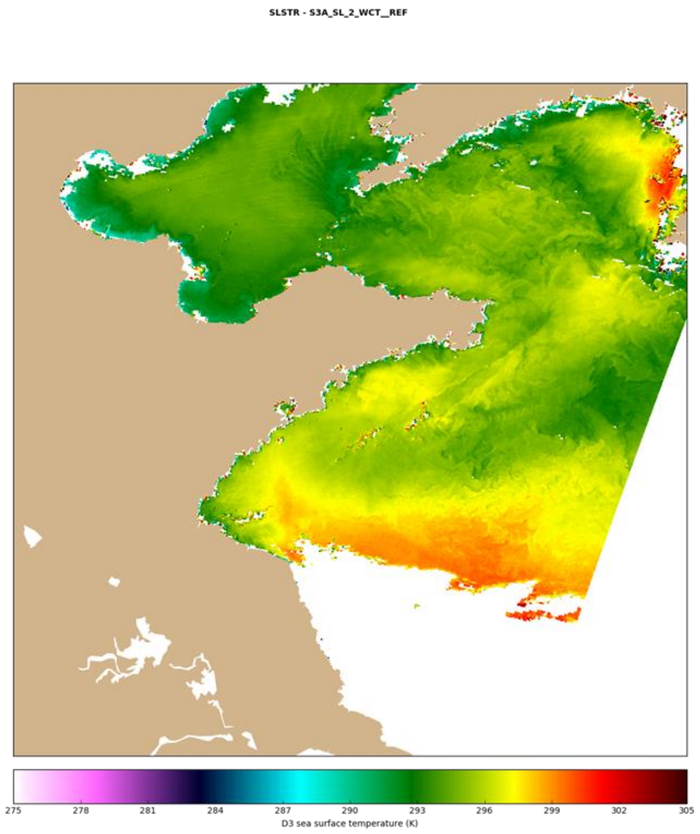


```
SHOW 0 8
W_XX-EUMETSAT-Darmstadt,HYPERSPECT+SOUNDING,MetOpA+IASI_C_EUMP_20100701004153_19184_eps_o_l1.nc

Reference
Name      : W_XX-EUMETSAT-Darmstadt,HYPERSPECT+SOUNDING,MetOpA+IASI_C_EUMP_20100701004153_19184_eps_o_l1.nc
Time range: 2010-07-01 01:45:32 to 2010-07-01 02:04:44
Slice     : {'cell': slice(40, 119, None), 'row': slice(477, 621, None)}
Geometry   :POLYGON ((-74.3743285021517 30, -68.90709065955365 30, -68.65699768066406 29.06100082397461, -60.12200164794922 -9.196999549865723, -56.58900070190
43 -23.4950008392334, -54.69640015258144 -30, -60.13921621269841 -30, -62.56700134277344 -19.75699996948242, -70.01499938964844 13.71700000762939, -72.4209976
1962891 23.25600051879883, -74.3743285021517 30))
Crossover :
Name      : 20100701-ATS_NR_2P-UPA-L2P-ATS_NR__2PNPDE20100701_020310_000046842090_00432_43572_3066-v01.nc
Time range: 2010-07-01 02:08:32 to 2010-07-01 02:27:22
Slice     : {'cell': slice(0, 511, None), 'row': slice(2152, 9684, None)}
Geometry   :POLYGON ((-74.3743285021517 30, -68.90709065955365 30, -68.65699768066406 29.06100082397461, -60.12200164794922 -9.196999549865723, -56.58900070190
43 -23.4950008392334, -54.69640015258144 -30, -60.13921621269841 -30, -62.56700134277344 -19.75699996948242, -70.01499938964844 13.71700000762939, -72.4209976
1962891 23.25600051879883, -74.3743285021517 30))
```

Results as images, text or json document

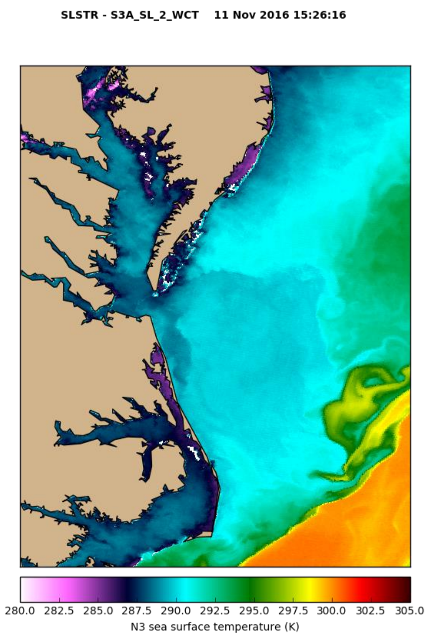
Use case : clear sky image search

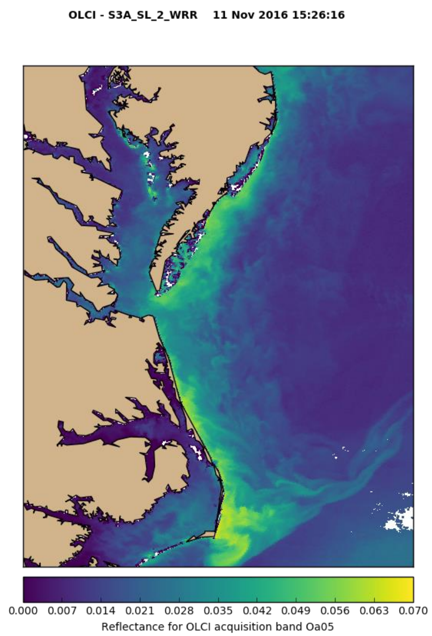


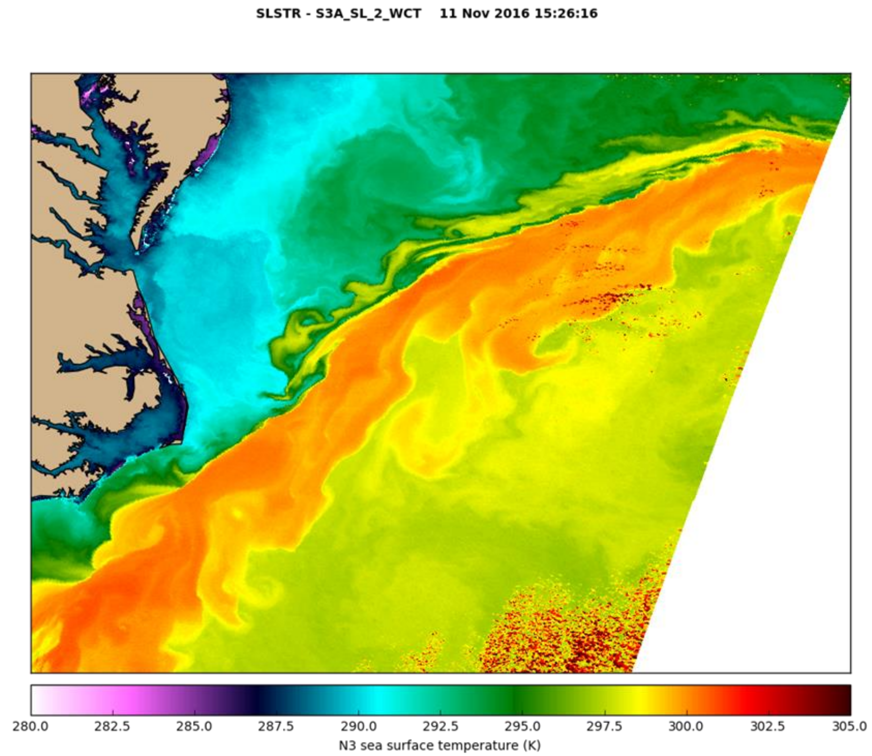


Search based on metadata registered in Elasticsearch

Use case : scene selection, comparison







- clear sky image search
- cross-overs between SLSTR / OLCI or other pairs of sensors

EUMETSAT

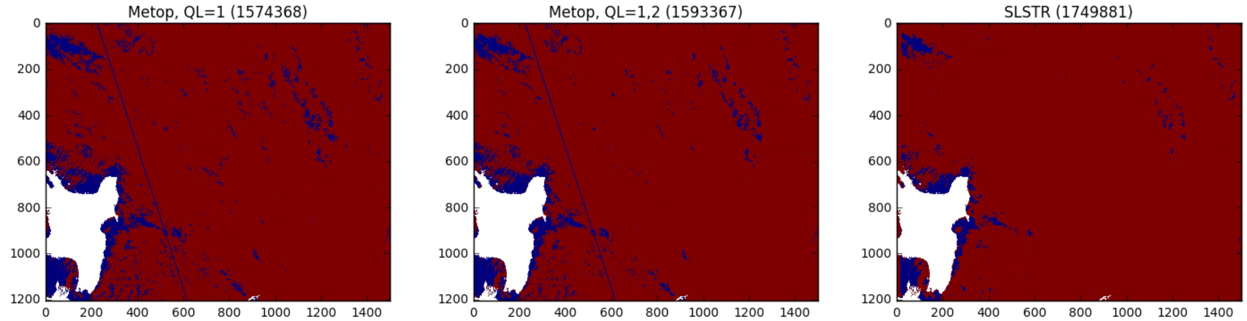
Use case : external cloud mask

Assess quality of SLSTR L1/L2 cloud mask by direct comparison with cloud mask provided by other instruments, resampled on SLSTR image grid

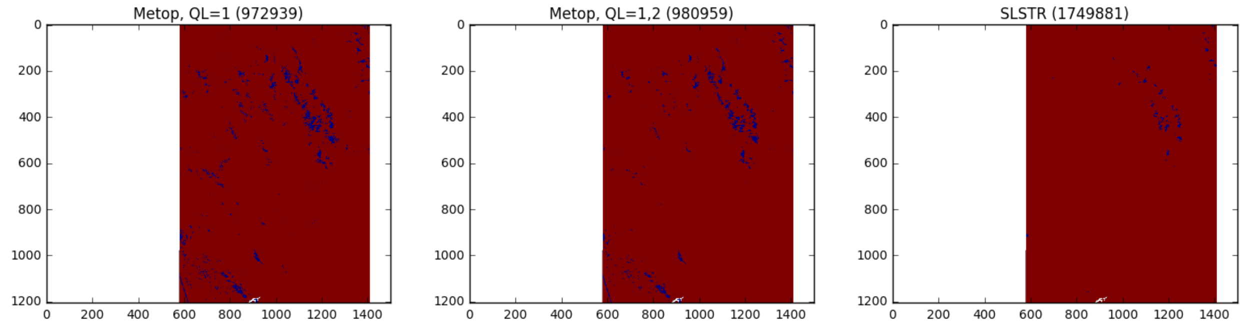
We construct a new cloud mask for SLSTR by remapping the cloud mask from Metop-B cross-overs onto SLSTR granules.

Two masks are generated : native cloud mask from Metop-B (QL=1) and GHRSST mask (QL=1,2)

Intendend for statistical comparison of respective mask extent (NOT at pixel level as clouds may be shifting)



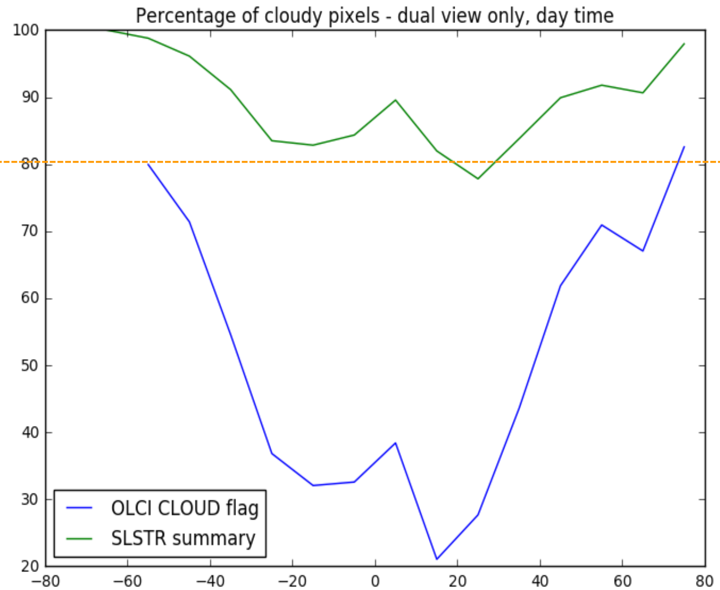
Full swath mask

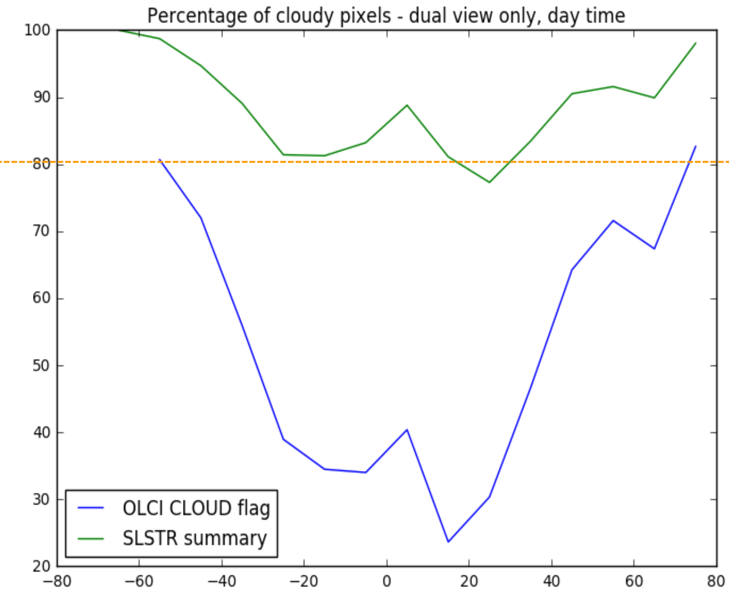


Nadir view only

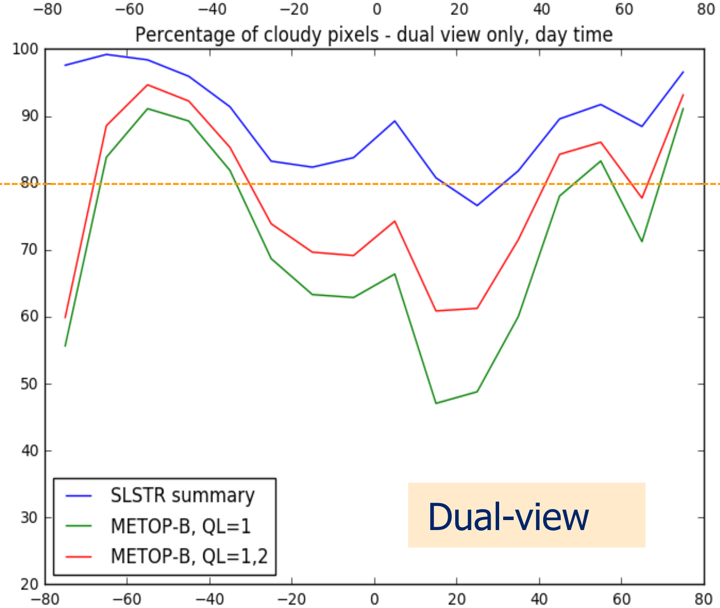
EUMETSAT

Comparisons with OLCI and SLSTR cloud masks, latitude dependency, day time

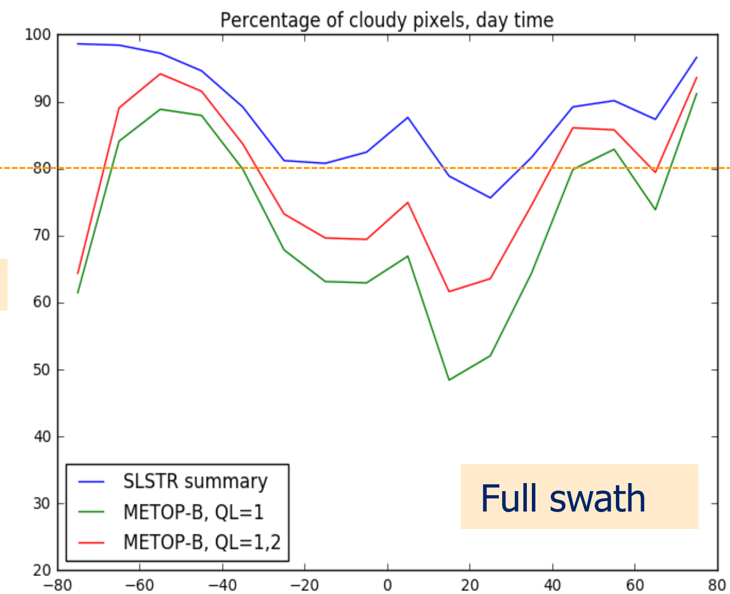




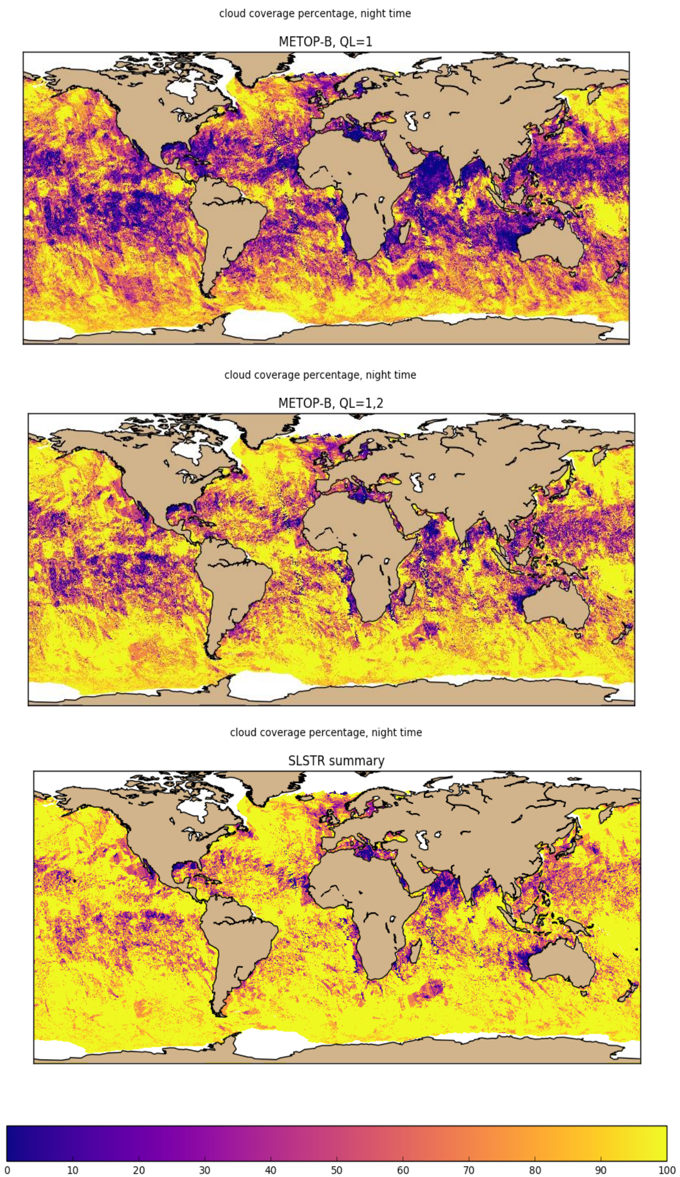
OLCI



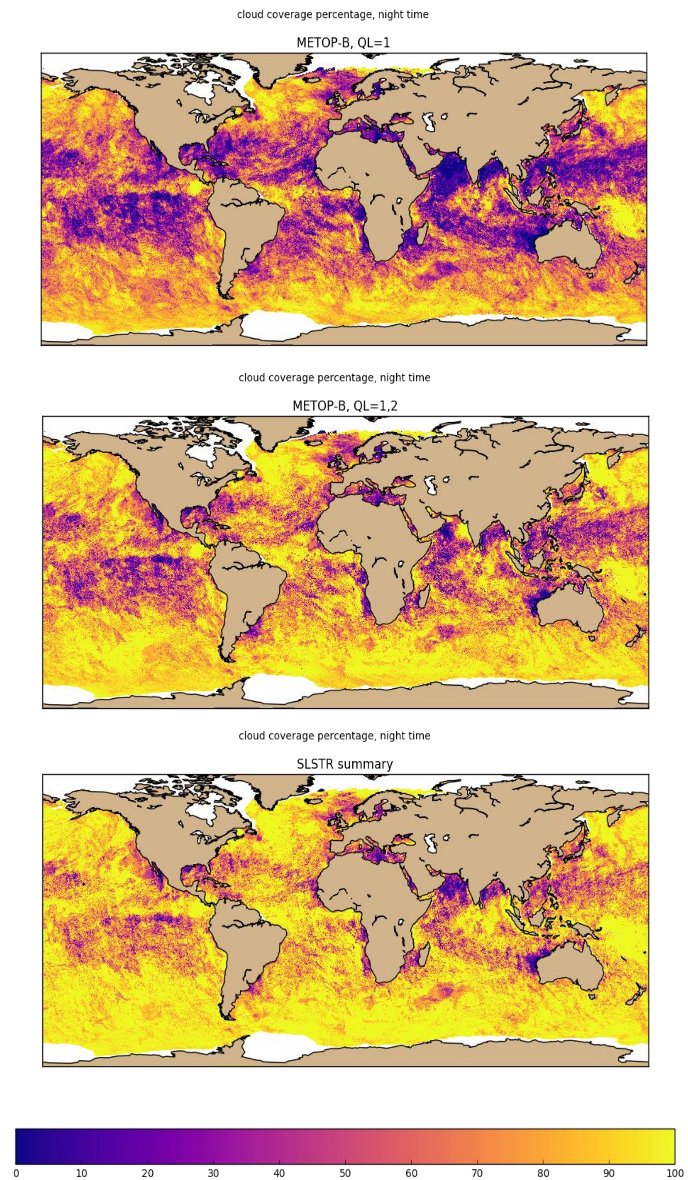
METOP



SLSTR / Metop-B, night time

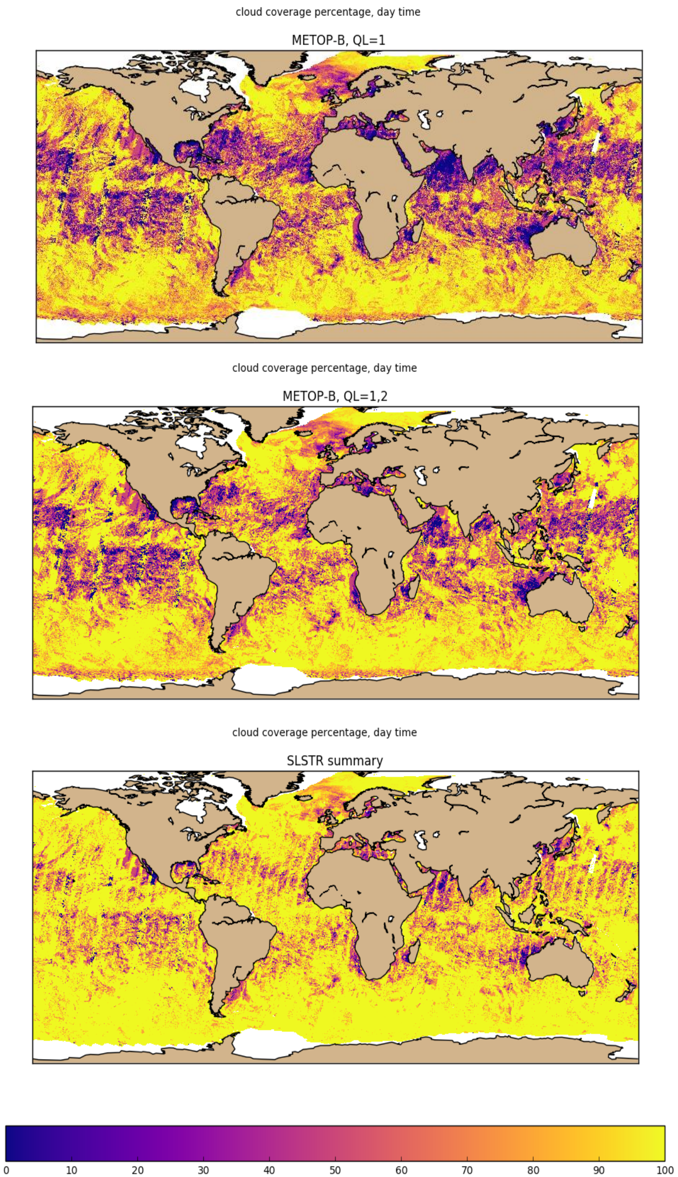


Dual-view SLSTR

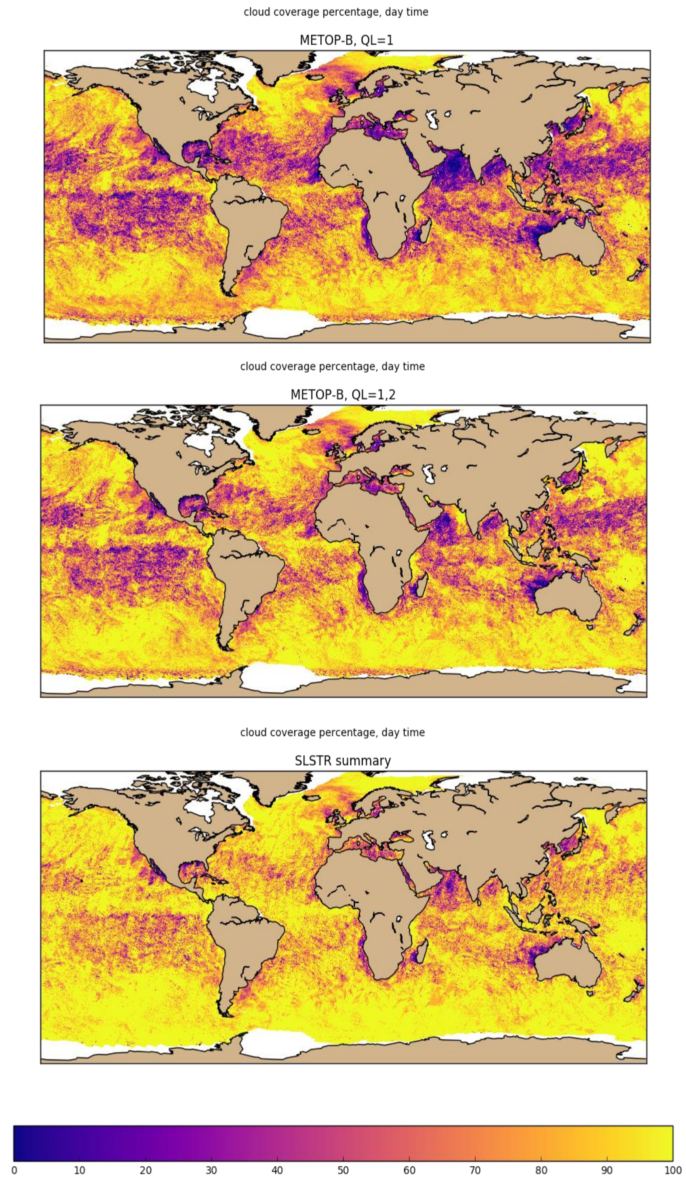


Full swath SLSTR

SLSTR / Metop-B, day time

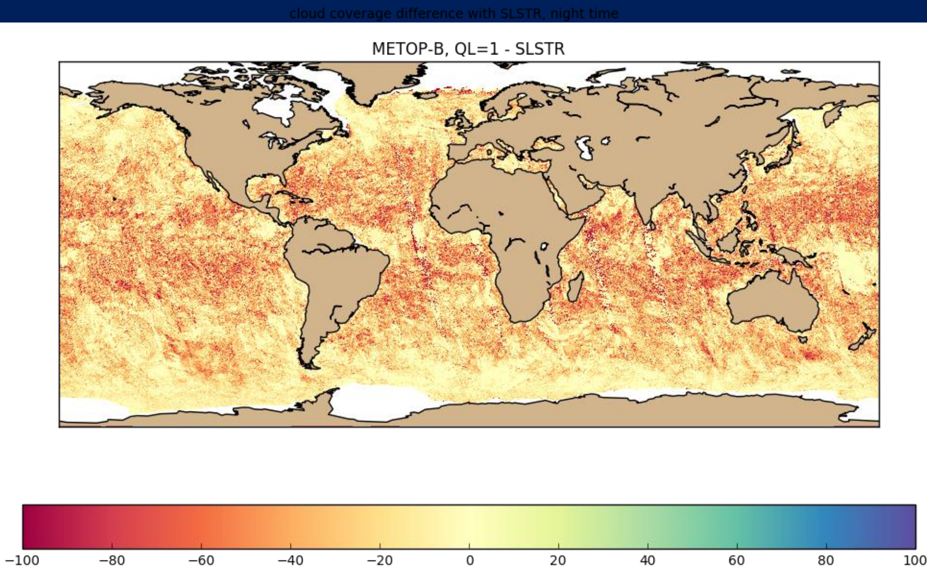


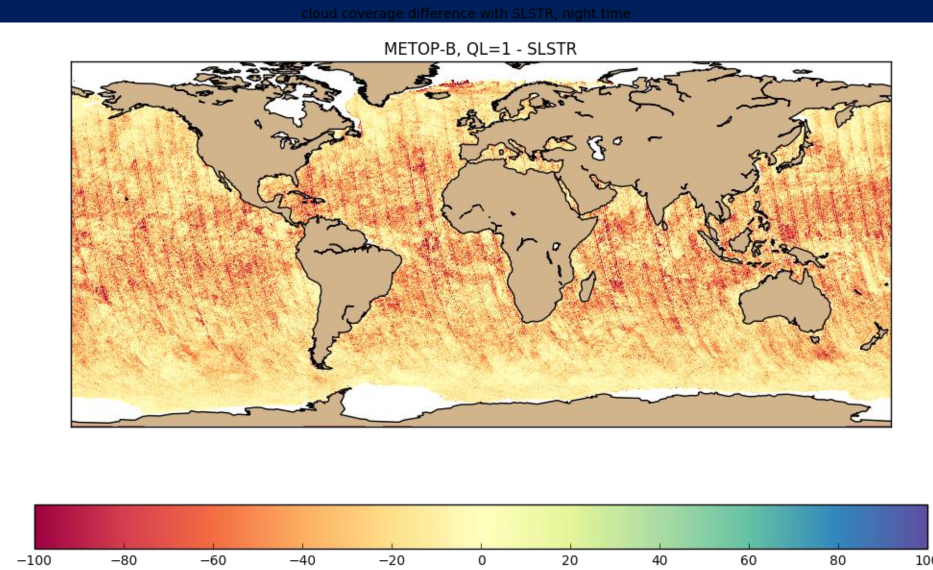
Dual-view SLSTR

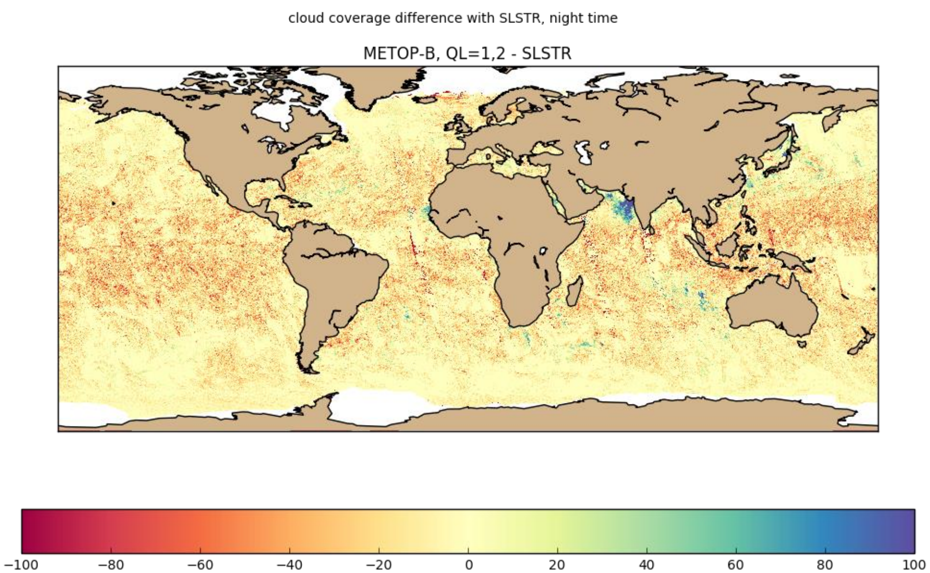


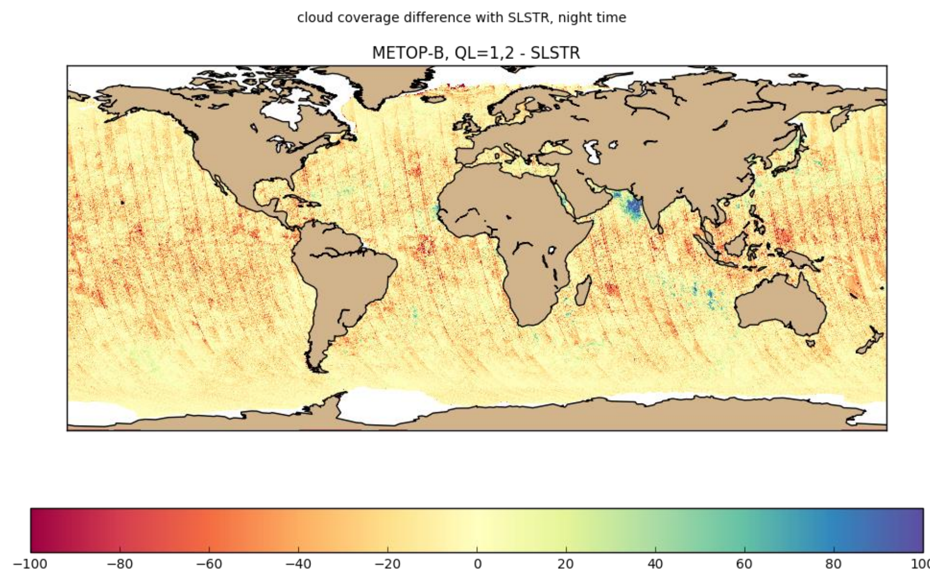
Full swath SLSTR

SLSTR / Metop-B, night time







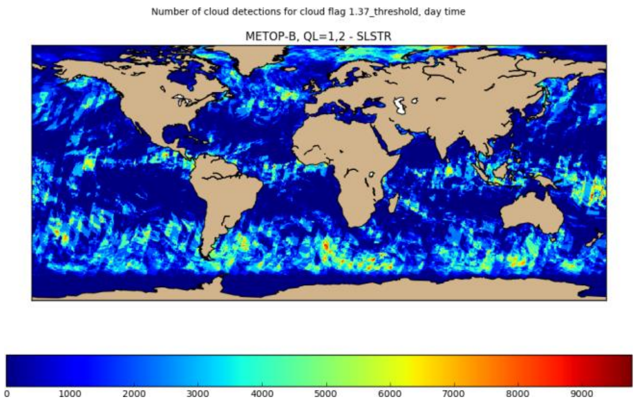


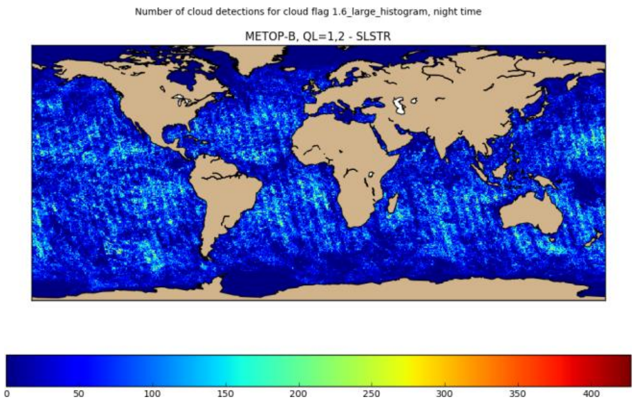
Dual-view SLSTR

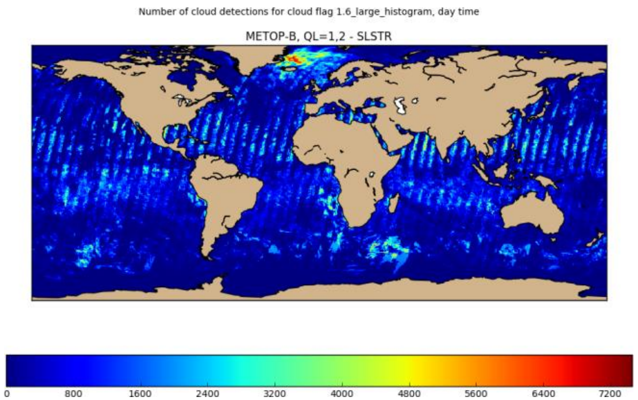
Full swath SLSTR

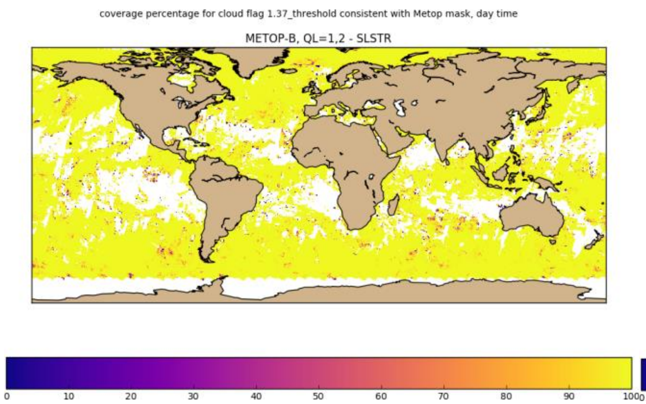
EUMETSAT

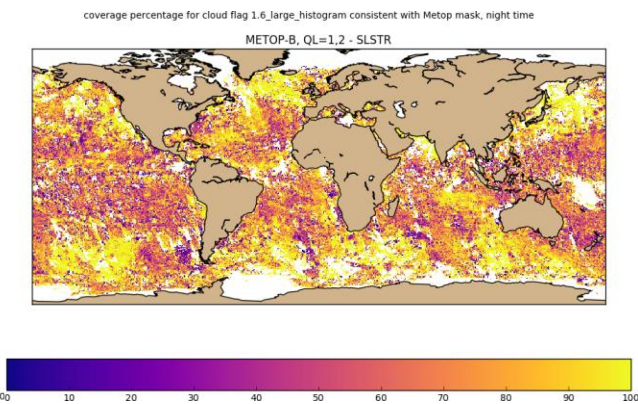
SLSTR / METOP-B, stats per cloud mask test

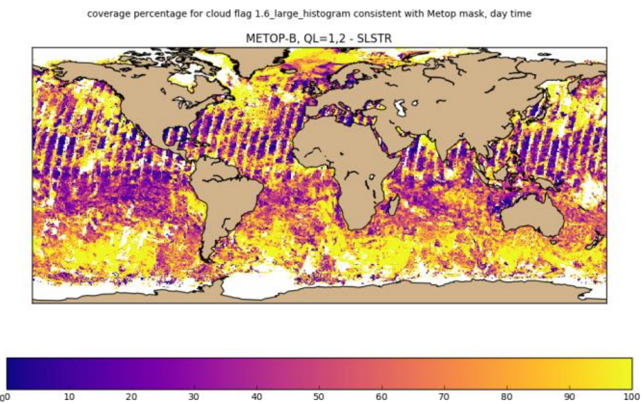












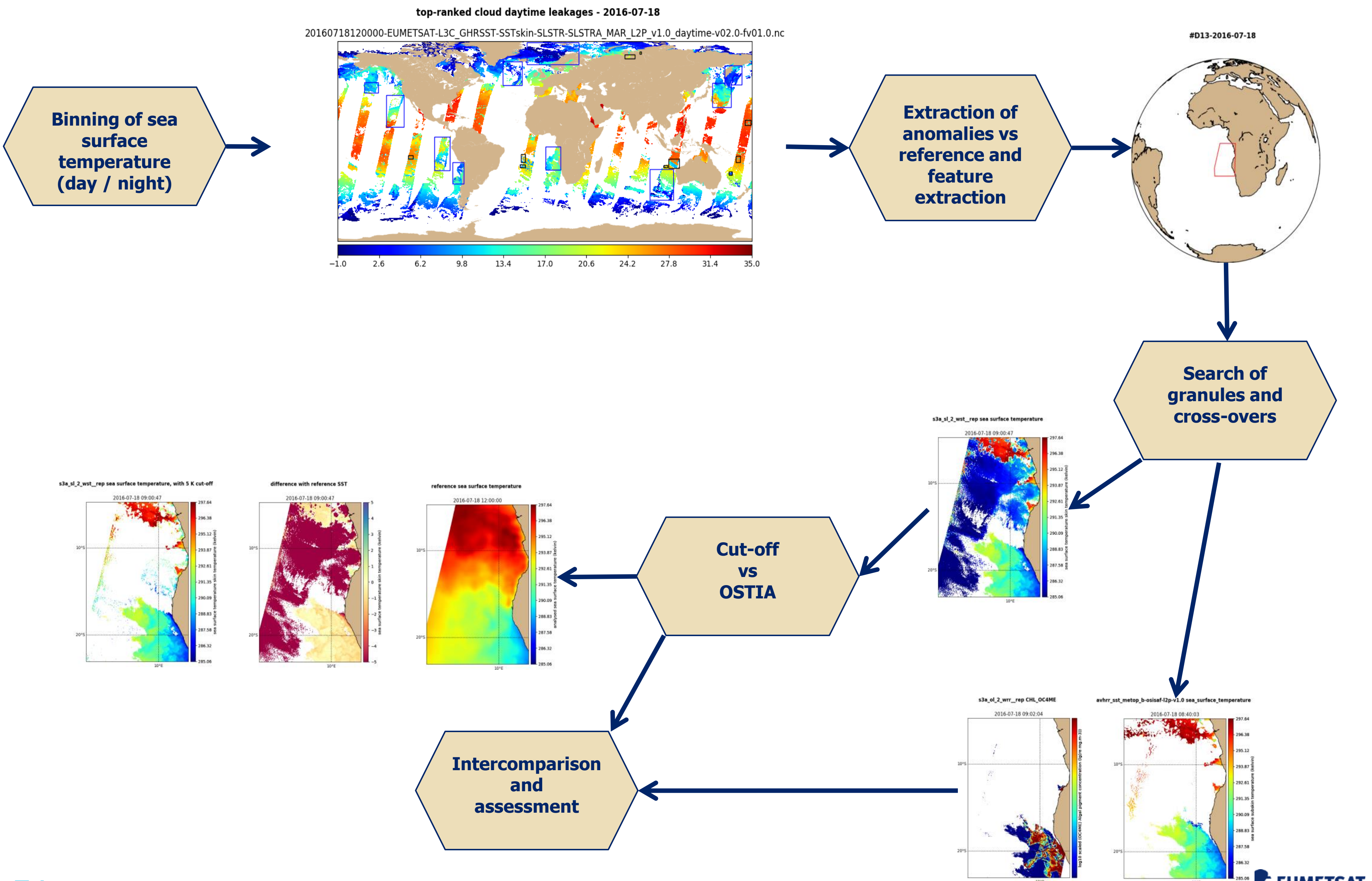
1.37 threshold

1.6 large histogram, night time

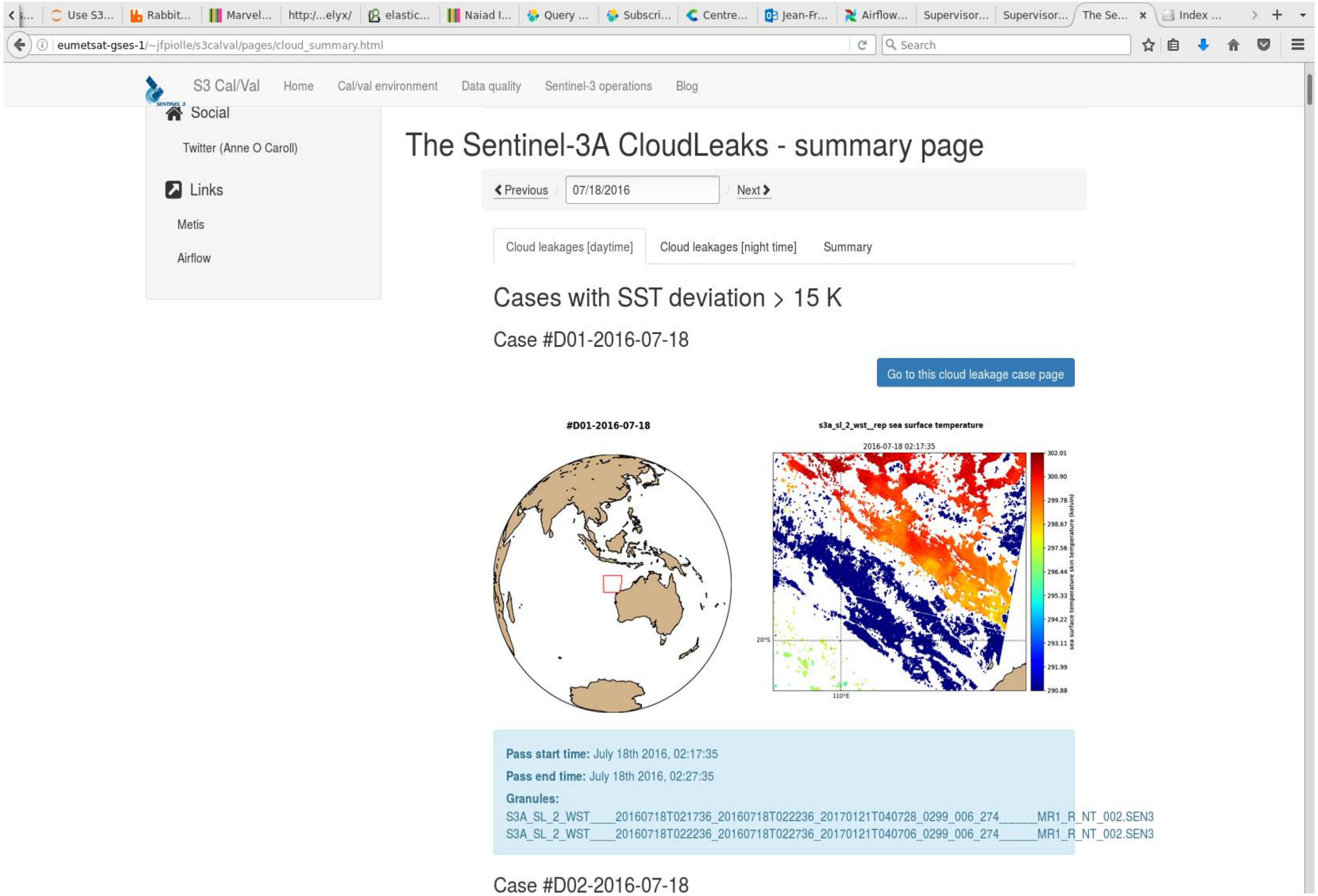
1.6 large histogram, day time

EUMETSAT

Use case : cloud leakage detection workflow

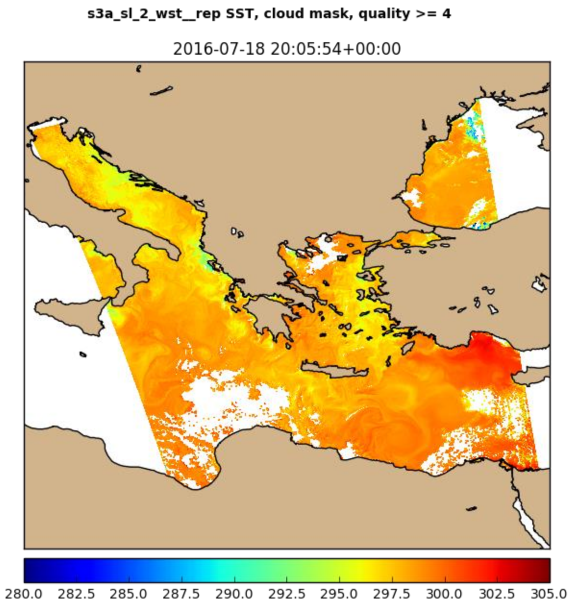


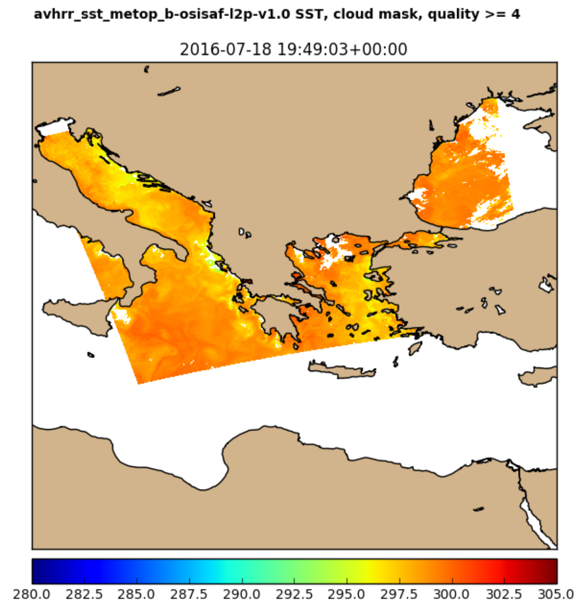
Catalogue of cloud cases

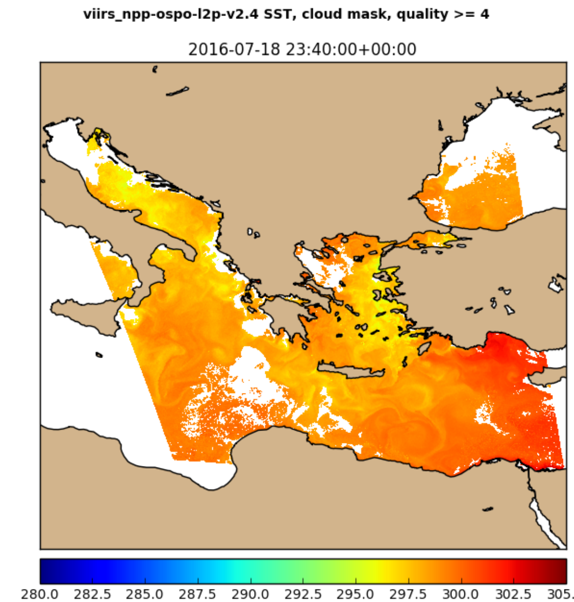


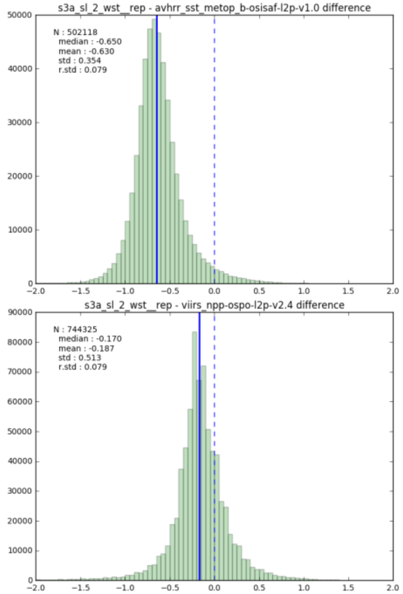
EUMETSAT

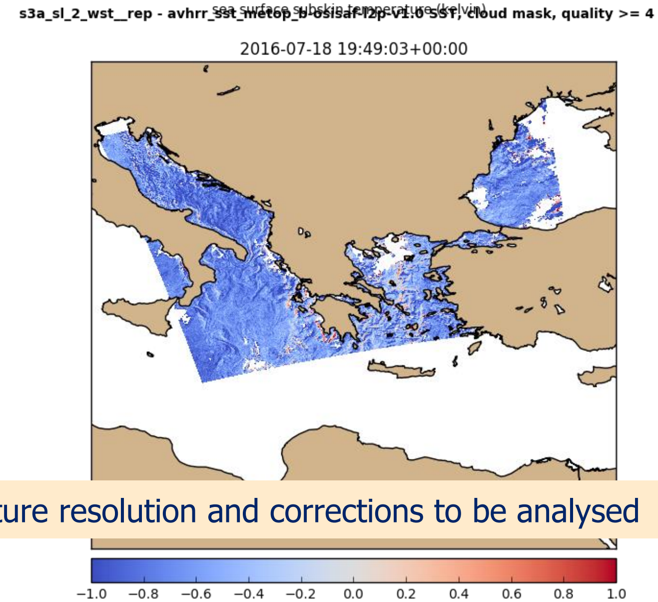
Use case : cross-over comparisons

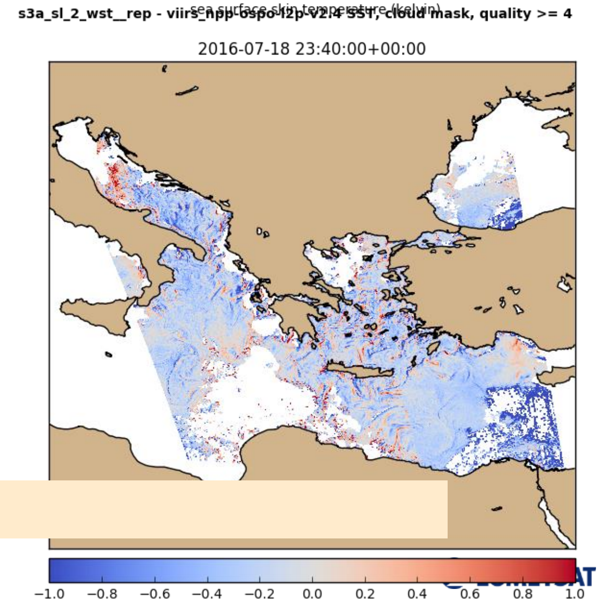








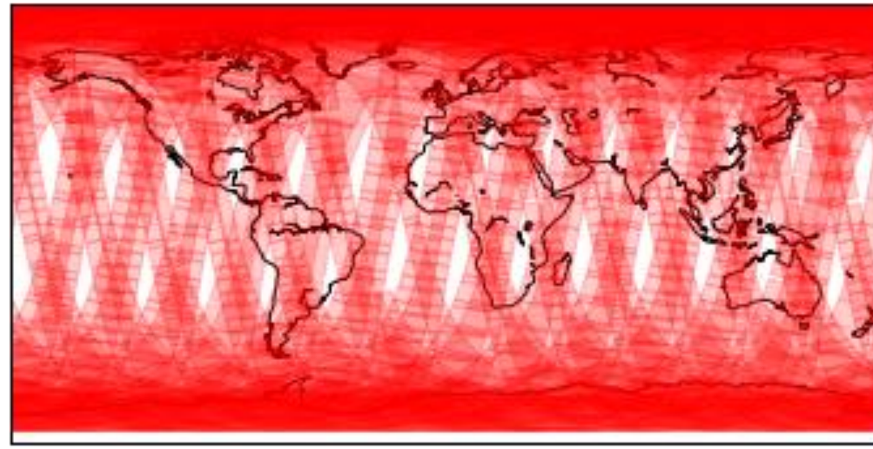




Feature resolution and corrections to be analysed

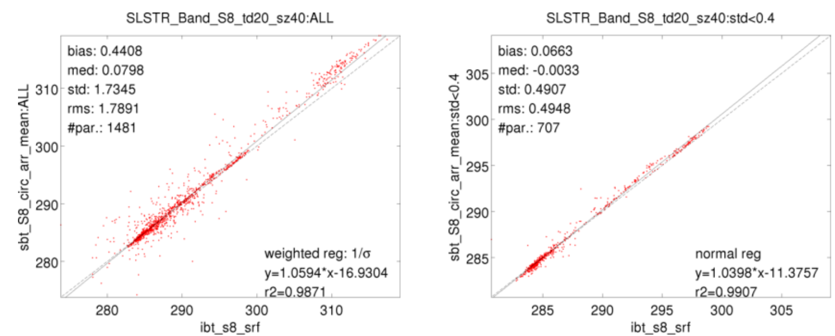
IASI L1C / SLSTR L1

```python
# search colocated files on "good" use case
# -----------------------------------------

start = datetime(2016, 11, 4)
end = datetime(2016, 11, 6)
reference = 'test_PDGS_SL_1_RBT____NR'.lower()
crossed = ['test_IASI_xxx_1C_M02'.lower()]
intersection_percentage = 10.
time_window = timedelta(minutes=5.)

lonmin, latmin, lonmax, latmax = -180, -90., 180., 90
area = shapely.geometry.asPolygon([
         (lonmin, latmax),
         (lonmax, latmax),
         (lonmax, latmin),
         (lonmin, latmin),
         (lonmin, latmax),
         ])
#constraints = [('slstr_class_summary_1km_cloudy_region_pix', 'gt', 30.),
#                ('slstr_class_summary_1km_cloudy_region_pix', 'lt', 70.),
#                ('solar_zenith_angle_min', 'lt', 70)
#               ]


search = ColocationSearch(reference, crossed, area, start, end, time_window,
                          precise=False, method='granule',
                         # granule_constraints=granule_constraints,
                         percent=intersection_percentage
                         )
```



5 min window

SLSTR vs. IASI (S8)



Courtesy: Igor Tomazic, Eumetsat

EUMETSAT

JUPYTER

jupyter

- http://jupyter.org/
- Python (but not only) in your web browser
- Embeds and mixcode, visualisation, explainations, equations in « notebooks »
- Growingly popular for interactive science
- Can run different languages (over 40)
- Can mix in some shell instructions
- Can be exported as html pages, pdf documents, .rst documents, LateX, python script
- Widgets for more interactivity, small task interfaces
- Complemented by **jupyterhub** which is single-user => allow multi-user access : a jupyter notebook server is spawned for each user

In S3 cal/val framework

- Quick visual development
- Sharing results with methodologie
- Learning tools and libraries
- Repeating analysis scenarii (new product release, longer time series,…)
- Advanced data analysis interfaces
- (Dashboards, report production)

Interactive integration of our different pieces of software jupyter Display an ensemble mean Last Checkpoint: 06/22/2017 (autosaved)

Display an ensemble mean of L3

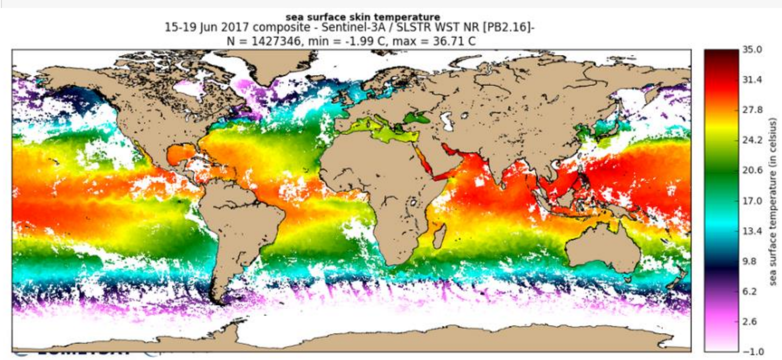
The ensemble mean can be built in a terminal with command line tools such as **cdo**.

For instance:

cdo ensmean /mount/common-storage/workdata/staging/oper/sst_binner/glob/0.1/v1.0/nighttime_nrt_ref/l3/slstra-mar-l2p-v1.0/2017/16[4-9]/*.nc mean_sst.nc

In [5]:

```python
import datetime
import os

from cerbere.datamodel.grid import Grid
from cerbere.mapper.ncfile import NCFile
from cerplot.mapping import CerMap

ensemble_mean = "/mount/home/jfpiolle/tmp/mean_sst.nc"

# load the data from ensemble mean with cerbere
grid = Grid()
ncf = NCFile(ensemble_mean)
grid.load(ncf)

data = grid.get_values("sea_surface_temperature") - 273.15


# display with cerplot package
subtitle = (
    "15-19 Jun 2017 composite - Sentinel-3A / SLSTR WST NR [PB2.16]"
    "- \nN = {0}, min = {1:.2f} C, max = {2:.2f} C").format(data.count(), data.min(), data.max())
img = CerMap(
        grid,
        fieldname="sea_surface_temperature",
        data=data,
        palette="medspiration",
        coastline=True,
        rivers=False,
        contouring='mesh',
        title=None,
        subtitle=subtitle,
        rangevalues=[-1.,35.],
        legendlabel="sea surface temperature (in celsius)",
        logo="eumetsat-copernicus",
        figsize=(14., 6),
        #background='bluemarble',
        )
#img.save('ensemble_mean_sst.png', crop=True)
img.show()
```



In [ ]:

jupyter

EUMETSAT

Interactive match-up outlier investigation with Jupyter

In [2]:

```
from s3analysis.slstr.mdb.analysis import get_basic_mask
from s3analysis.slstr.cloud import cloud_mask, DEFAULT_CLOUDMASK

# basic validity mask (sat angl < 55., wind speed > 6 m/s)
basic_mask = get_basic_mask(data_sat)

print "Number of match-ups : ", len(basic_mask)

print "Number of valid match-ups : ", (numpy.count_nonzero(basic_mask))
print "Number of invalid match-ups : ", (basic_mask).size - (numpy.count_nonzero(basic_mask))


# select only the match-ups where WST fields are defined
valid_sst = (
        basic_mask &
        ~numpy.ma.getmaskarray(data_sat['WST']["sst_theoretical_uncertainty"]) &
        (data_sat['WST']["quality_level"] > 2) &
        (numpy.ma.fabs(data_sat['WST']["dt_analysis"]) <= 5.) &
        ~numpy.ma.getmaskarray(data_insitu["water_temperature"])
        )


print "Final number of valid clear sky match-ups : ", numpy.count_nonzero(valid_sst)


slstr_sst = data_sat['WST']["sea_surface_temperature"] - 273.15
insitu_sst = data_insitu["water_temperature"]


cloudybox = cloud_mask(data_sat_box['WCT']["cloud_in"])
confidence = (data_sat_box['WCT']['confidence_in'][:] & 16384) > 0
```

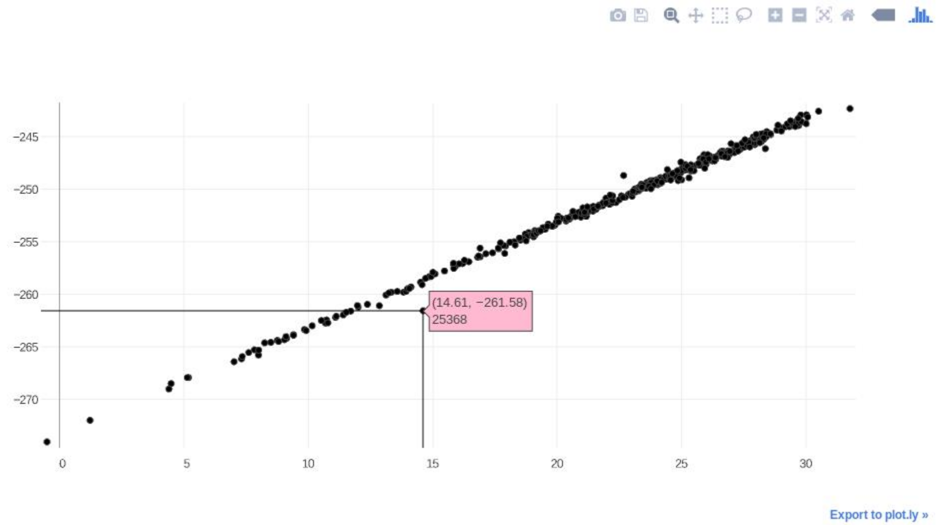
In [3]:

```
# additional filter to keep only nighttime data
night = (data_insitu['solar_zenith_angle'] > 90.)
```

In [4]:

```
# achtung! plotly neeeds to be installed in your environment (pip install plotly

import plotly.graph_objs as go
import numpy as np
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

# allow inline plot with plotly
init_notebook_mode(connected=True)

# Create a interactive scatterplot SST vs in situ with plotly
trace = go.Scattergl(
    x = insitu_sst[night & valid_sst],
    y = slstr_sst[night & valid_sst] - 273.15,
    text = numpy.arange(len(slstr_sst))[night & valid_sst],
    mode = 'markers',
    marker = dict(
        color = 'FFBAD2',
        line = dict(width = 1)
    )
)
data = [trace]
iplot(data)

print len(insitu_sst[night & valid_sst]), ' match-ups'
```

```
print len(insitu_sst[night & valid_sst]), ' match-ups'
```

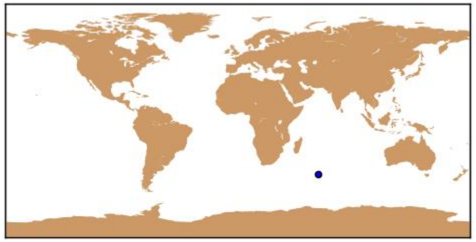


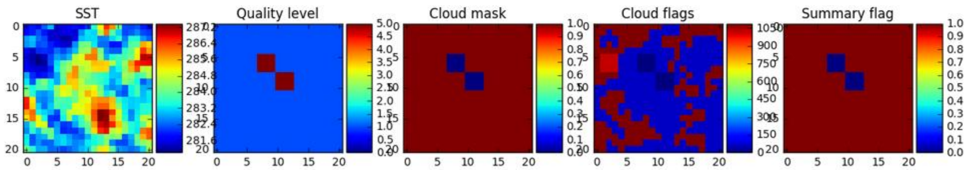

```
459  match-ups
```


EUMETSAT

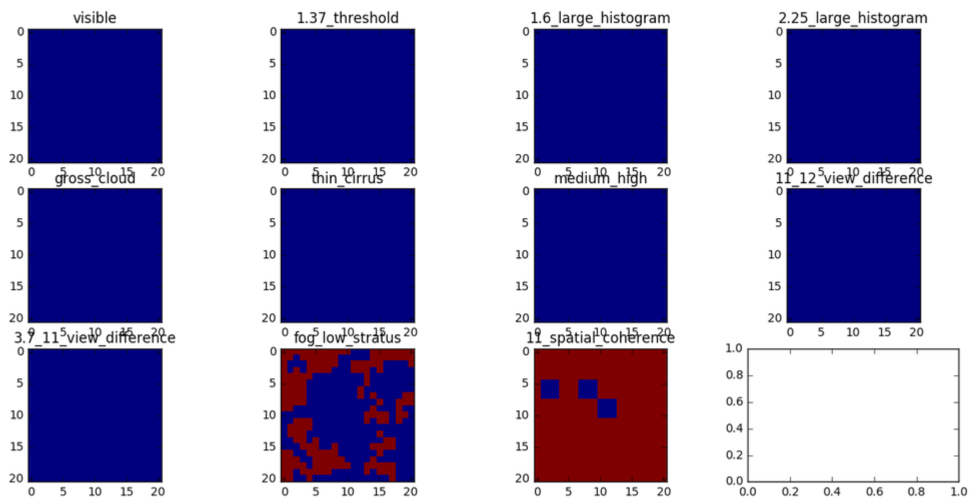
Interactive match-up outlier investigation with Jupyter

```
In situ value : 14.610000 K
SST - in situ difference : -3.040000 K
Traceability:
....WST file :  S3A SL 2 WST    20170628T185854 20170628T190154 20170628T202840 0179 019 198 5220 MAR F NR 002.SEN3
```






```
Used mask flags :  ['visible', '1.37_threshold', '1.6_large_histogram', '2.25_large_histogram', 'gross_cloud', 'thin_cirr
us', 'medium_high', '11_12_view_difference', '3.7_11_view_difference', 'fog_low_stratus', '11_spatial_coherence']
```




```python
# display match-up info
print "SST value : %f K" % slstr_sst[choice]
print "In situ value : %f K" % insitu_sst[choice]
print "SST - in situ difference : %f K" % (slstr_sst[choice] - insitu_sst[choice])

print "Traceability:"
print "....WST file : ", data_sat['WST']['origin'][choice]

# locate match-up on map
from mpl_toolkits.basemap import Basemap
m = Basemap()
m.drawmapboundary()
m.fillcontinents(color='#cc9966')
x, y = m(data_insitu['lon'][choice], data_insitu['lat'][choice])
m.scatter(x, y)


# plot cloud and SST
plot_mask(choice)
```


EUMETSAT

Interactive match-up outlier investigation with Jupyter

trace back to original file

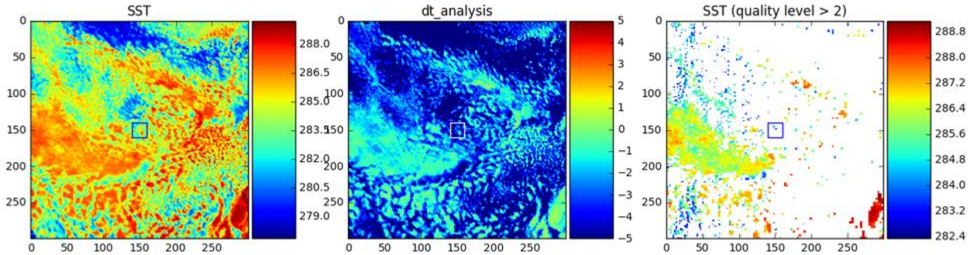
Here we access the content of the original file from which the match-up was extracted, and display a larger area around the match-up location.

This require to have access to the original SLSTR files!

In [6]:
```
print data_sat['WST']['dynamic_target_center_index'][choice]
```

```
[681 200]
```

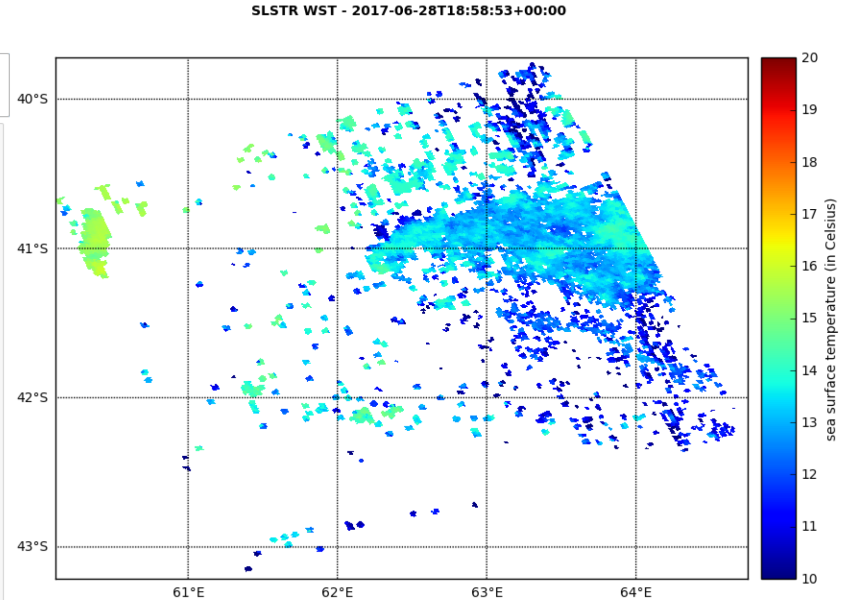
In [7]:
```
# get full path name
from naiad.utils.filelocator import FileLocator
locator = FileLocator()
fname = locator.get_full_path(data_sat['WST']['origin'][choice], 's3a_sl_2_wst__ref')

# define large subset
row, cell = data_sat['WST']['dynamic_target_center_index'][choice]
boxwidth = 300
boxheight = 300
larger_box = {'row': slice(max(0, row - boxheight / 2), row + boxheight / 2),
              'cell': slice(max(0, cell - boxwidth / 2), cell + boxheight / 2)
             }

# load data into a cerbere swa=th object
from cerbere.mapper.safeslfile import SAFESLWSTFile
from cerbere.datamodel.swath import Swath
wstfile = SAFESLWSTFile(fname)
swath = Swath()
swath.load(wstfile)
```

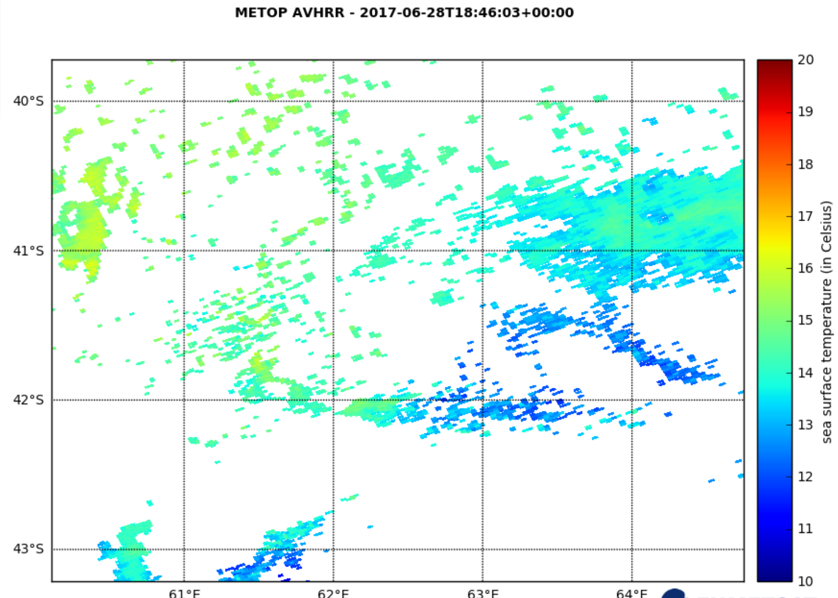


EUMETSAT

fetch Metop image

In [19]:
```
# import necessary packages
import shapely

from naiad.utils.filelocator import FileLocator
from naiad.queries.server import Server
from naiad.queries.search import SpatioTemporalSearch

from cerbere.mapper.ghrsstncfile import GHRSSTNCFile
from cerbere.datamodel.swath import Swath
from cerplot.mapping import CerMap

%matplotlib inline

# provides Naiad server URL
es = Server("http://eumetsat-gses-5:9200/")

# =========== DEFINE HERE YOUR SEARCH CRITERIA ===========

# define the geographical search box
lats = swath.get_lat(slices=larger_box)
lons = swath.get_lon(slices=larger_box)
area = shapely.geometry.box(lons.min(), lats.min(), lons.max(), lats.max())

# define start and end of search interval
start = swath.get_start_time() - datetime.timedelta(hours=1)
end = swath.get_end_time() + datetime.timedelta(hours=1)

# define the naiad indice (product name) to crawl
product = 'avhrr_sst_metop_b-osisaf-l2p-v1.0'

# compose the query
search = SpatioTemporalSearch([product], area, start, end)

# execute the query
res = search.run(es)
```

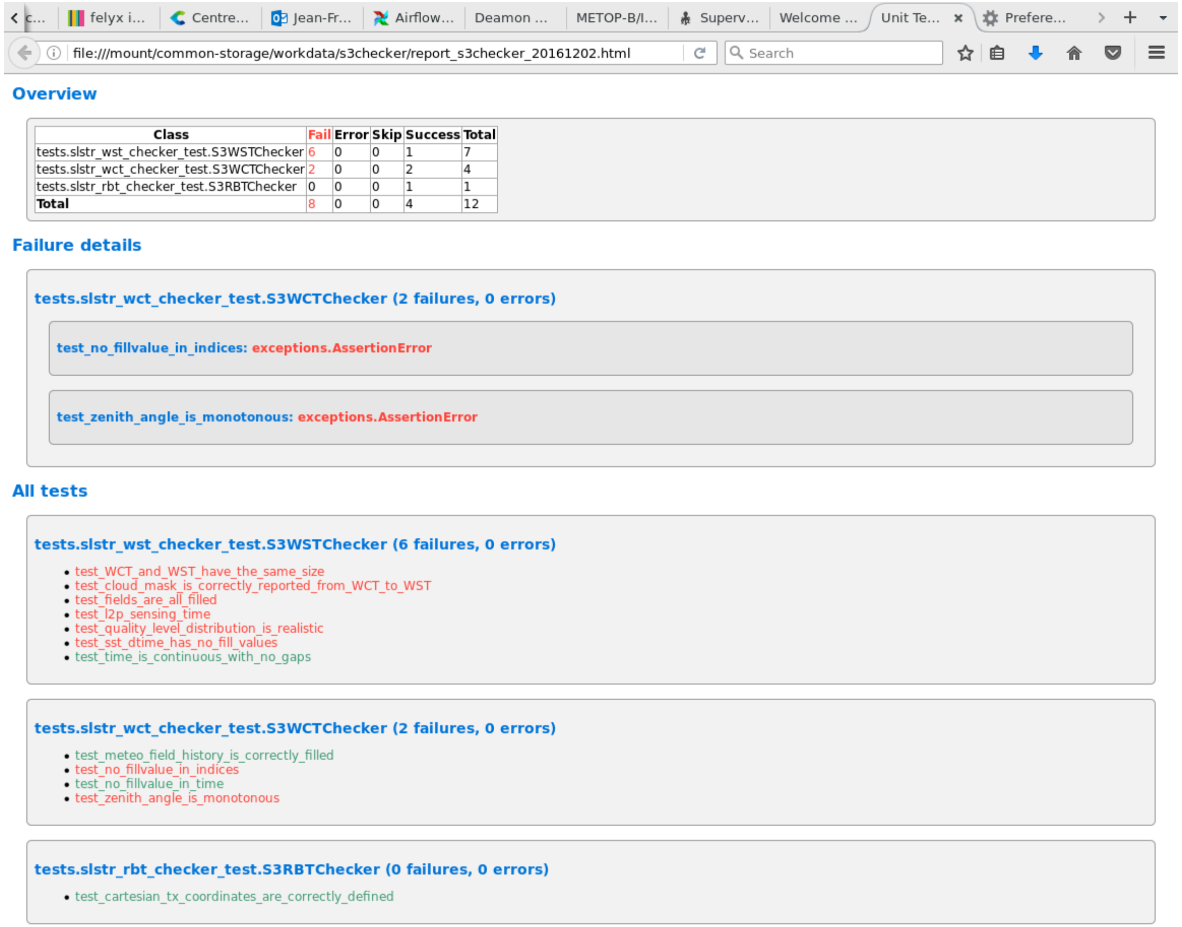




EUMETSAT

MISC

Fix / regression verification : « s3checker »



file:///mount/common-storage/workdata/s3checker/report_s3checker_20161202.html

Overview

Class	Fail	Error	Skip	Success	Total
tests.slstr_wst_checker_test.S3WSTChecker	6	0	0	1	7
tests.slstr_wct_checker_test.S3WCTChecker	2	0	0	2	4
tests.slstr_rbt_checker_test.S3RBTChecker	0	0	0	1	1
Total	8	0	0	4	12

Failure details

tests.slstr_wct_checker_test.S3WCTChecker (2 failures, 0 errors)

test_no_fillvalue_in_indices: exceptions.AssertionError

test_zenith_angle_is_monotonous: exceptions.AssertionError

All tests

tests.slstr_wst_checker_test.S3WSTChecker (6 failures, 0 errors)

- test_WCT_and_WST_have_the_same_size
- test_cloud_mask_is_correctly_reported_from_WCT_to_WST
- test_fields_are_all_filled
- test_l2p_sensing_time
- test_quality_level_distribution_is_realistic
- test_sst_dtime_has_no_fill_values
- test_time_is_continuous_with_no_gaps

tests.slstr_wct_checker_test.S3WCTChecker (2 failures, 0 errors)

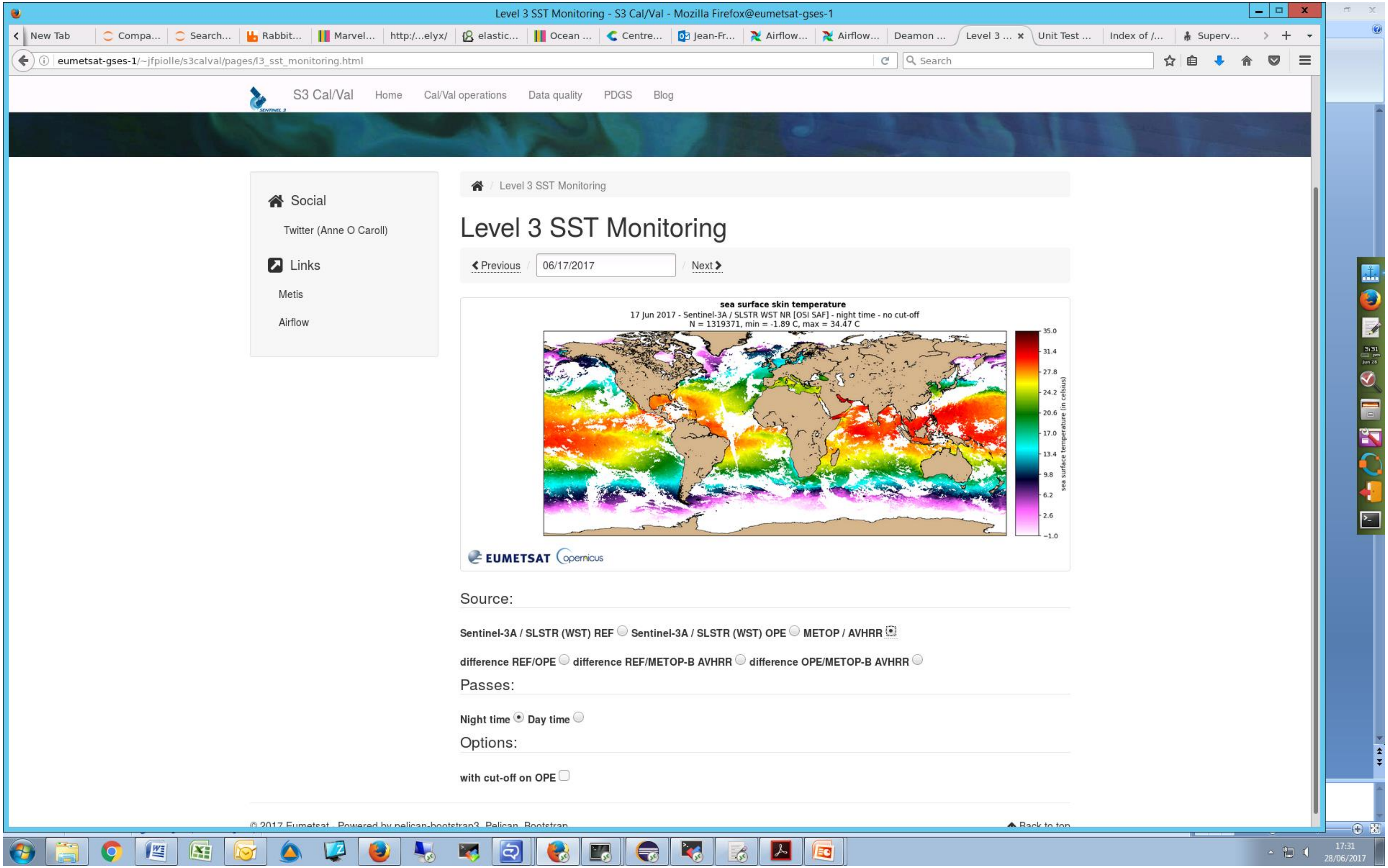
- test_meteo_field_history_is_correctly_filled
- test_no_fillvalue_in_indices
- test_no_fillvalue_in_time
- test_zenith_angle_is_monotonous

tests.slstr_rbt_checker_test.S3RBTChecker (0 failures, 0 errors)

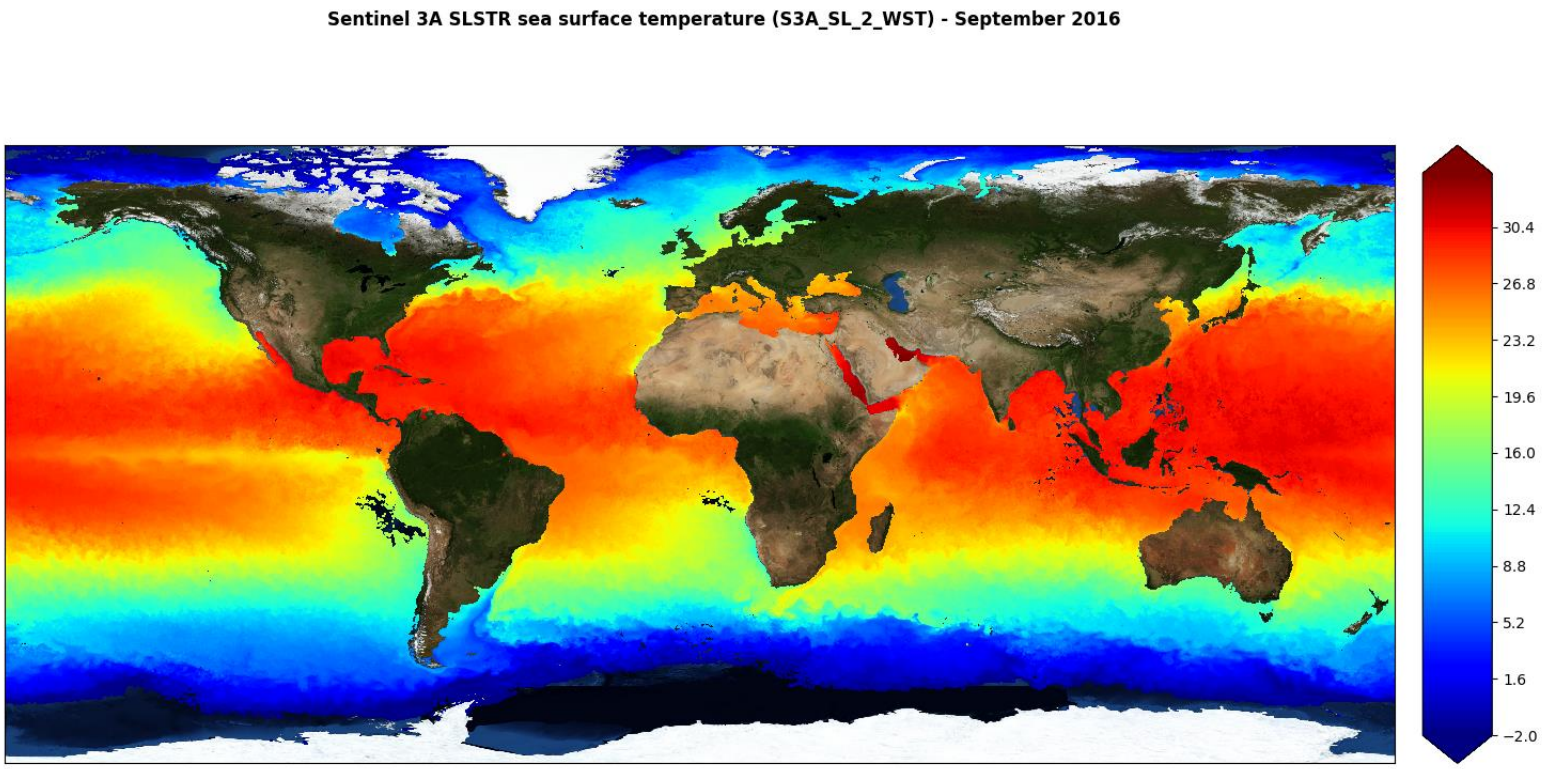
- test_cartesian_tx_coordinates_are_correctly_defined

Based on python unitary test framework – uses also scientific packages implemented for S3 data analysis

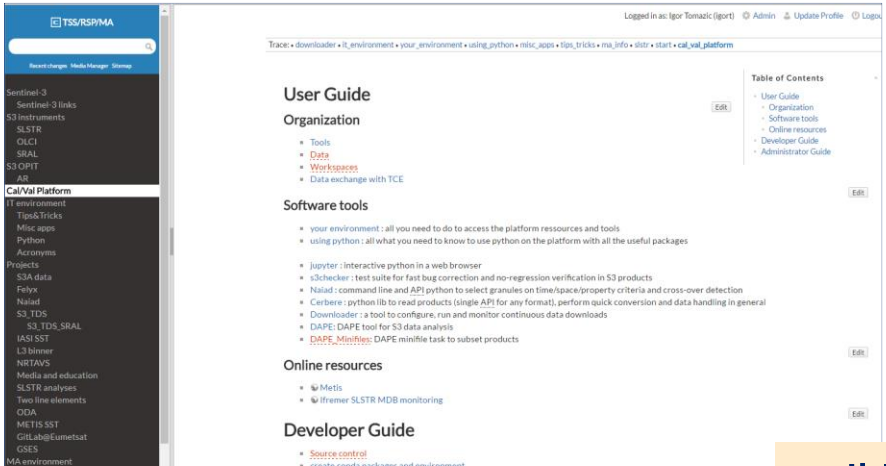
Configurable regridder/binner



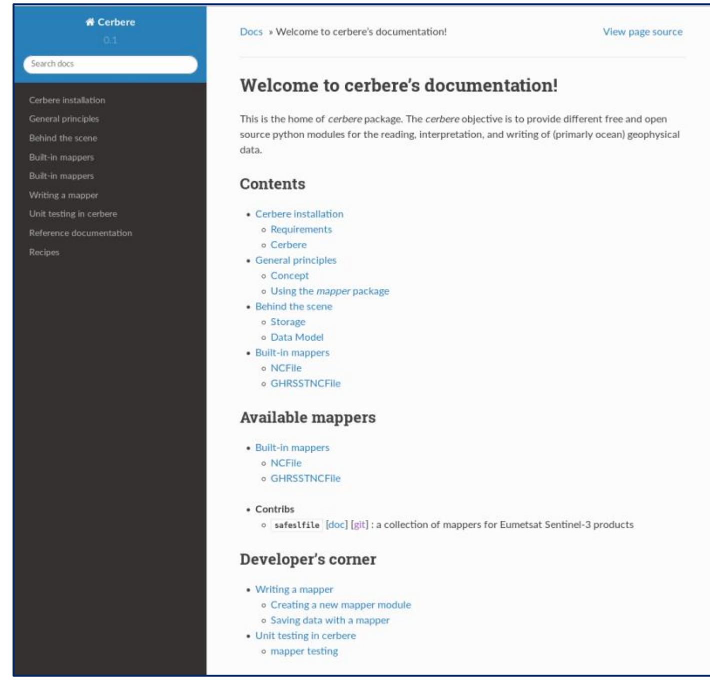
Configurable regridder/binner



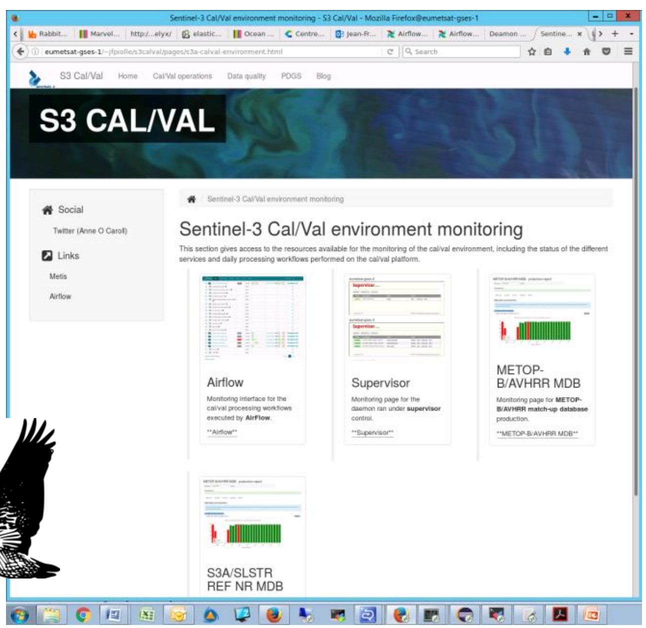
Complementary interaction tools



wiki



Software and python package documentation
[**sphinx** / readthedocs.org, **rst** format]

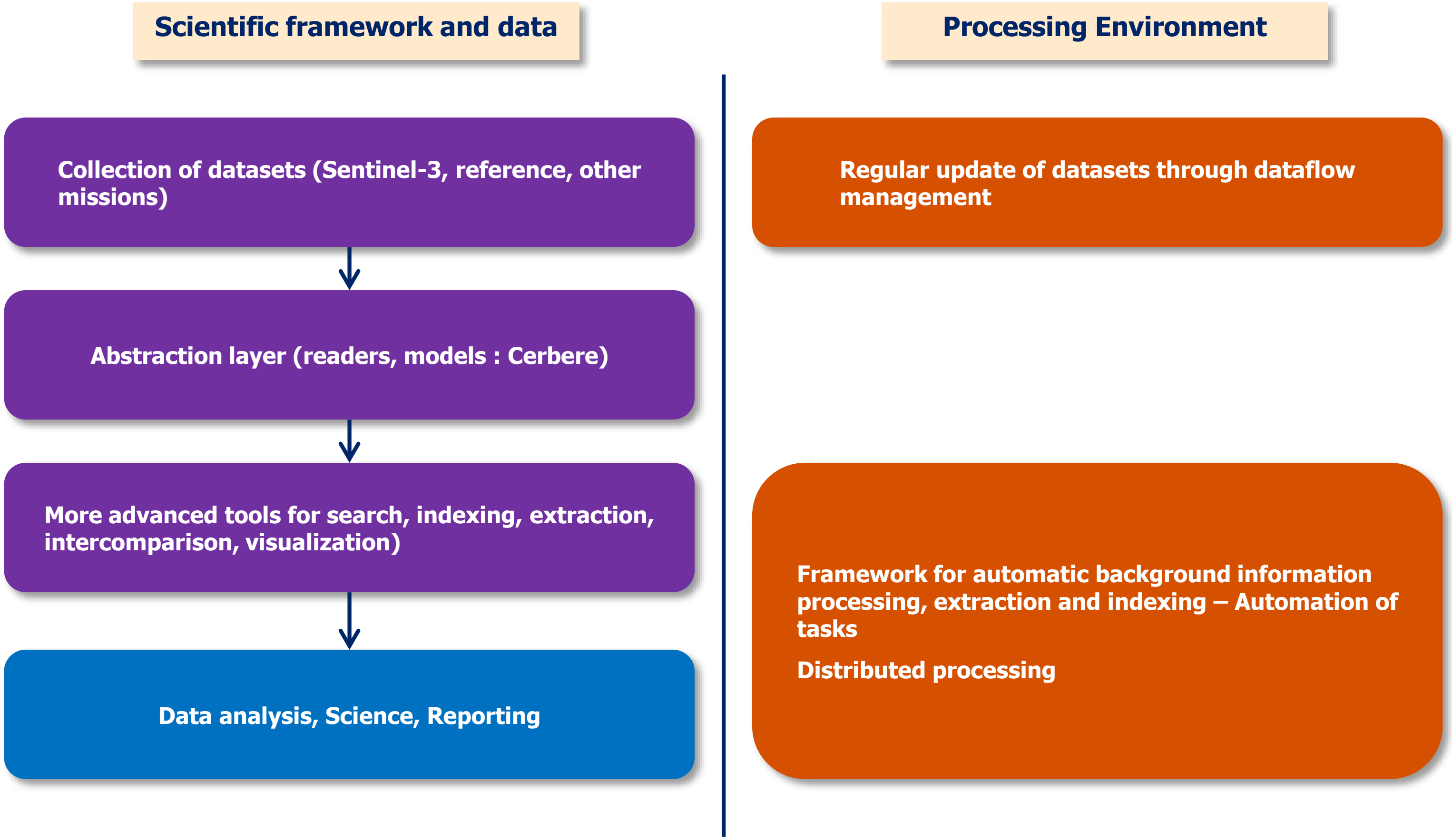


Pelican – static web site generator
[python, **rst** format]

Placeholder for various interfaces, monitoring tools, infos,…

CONCLUSION!!

« exploitation platform concept »



Prototype for a consistent cloud based « thematic exploitation platform », cal/val oriented

conclusion

- Common tasks for anybody working on cal/val but trying to improve integration and bridges between tools
- Mostly demonstrated on SST but moving to OLCI
- Still in demonstration mode and further consolidation is required
- But was an asset in SLSTR cal/val
  - Large usage of MDBs in data assessment
  - Detection of anomalies
  - Comparison with other instruments
- Some functionalities of deployed tools still to be further exploited

Improvements / new developments

- Prototyping (for – at least – L2) : testing improvements, algorithm changes, comparing with operational processor

- Alerting : automatic warning on data quality degradation (exemple: black body case)
  - A bit tedious to check all interfaces every day
  - Issues not always obviously raised by the existing indicators
  - Possible implementation through **felyx**, analytics tools from ES ecosystem, L3 products, cross-overs

- Data quality analysis : new indicators required
  - Aggregation of data : spatial and seasonal patterns (not only SST but other fields : distribution of QL, bias, cloud flags, etc…)
  - Spectral analysis

- Eumetsat products intercomparison : SLSTR, AVHRR, SEVIRI, IASI
  - Systematic differences in cross-overs, L3 differences, MDB
  - Identification and analysis of major differences
  - Leverage on available platform and tools
  - More cross analysis and involvement with OSI SAF team

THANKS TO RSP AND SLSTR TEAM FOR SUPPORTING THIS ACTIVITY!