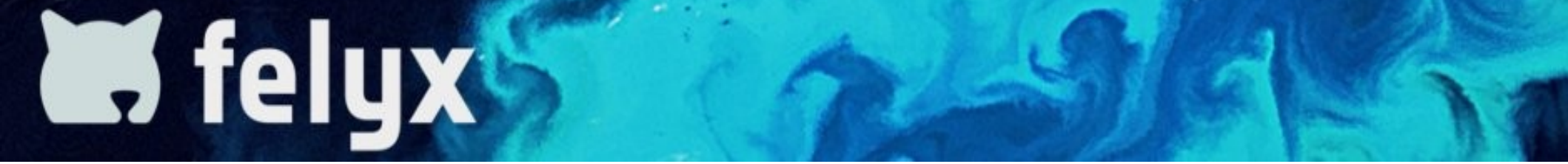


Felyx : application to CDAF

Jean-François Piollé – **Ifremer**

Sylvain Herlédan – **OceanDataLab**

Philippe Goryl, Craig Donlon, Veronica Guidetti – **ESA**

esa

Ifremer

pelamis scientific software

PML Plymouth Marine Laboratory

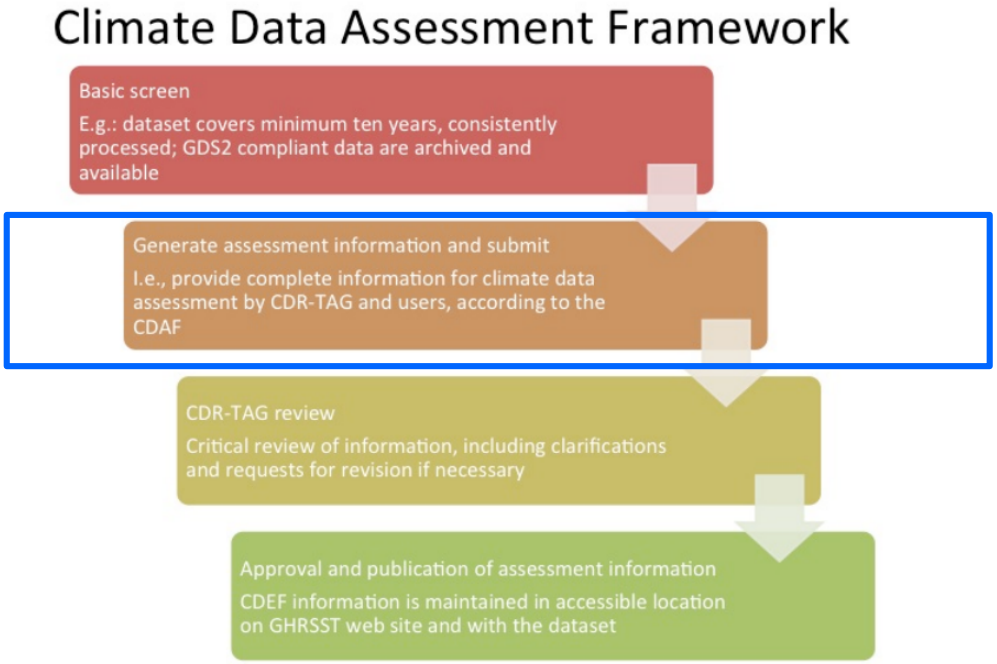
Use case 1 : essential climate variables

Context : GHRSST Climate Data Assessment Framework (CDAF) - « Creating a community multi-sensor match-up system is a GHRSST objective »

How to assess if a dataset is suitable for climate trend detection ?

Accuracy, sensitivity and consistency

Provide a tool to users to evaluate this on their own dataset using a common set of diagnostics and reference data

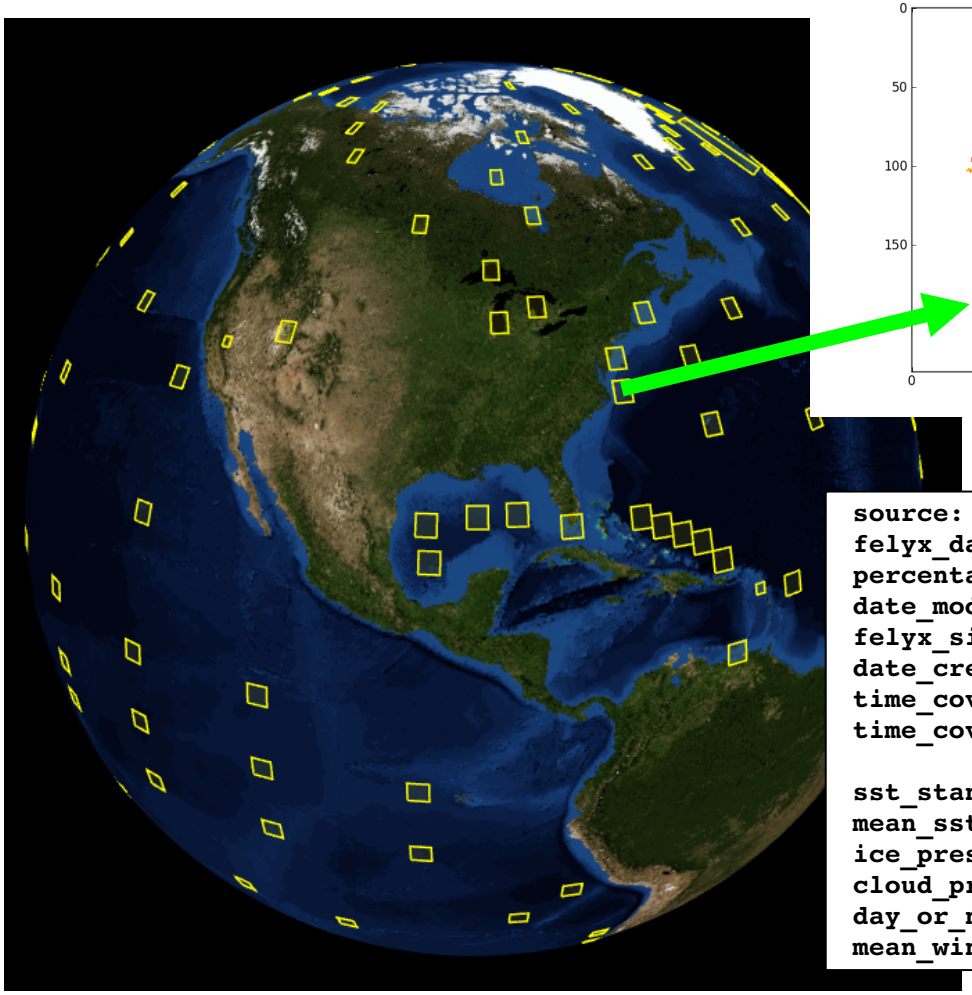


felyx

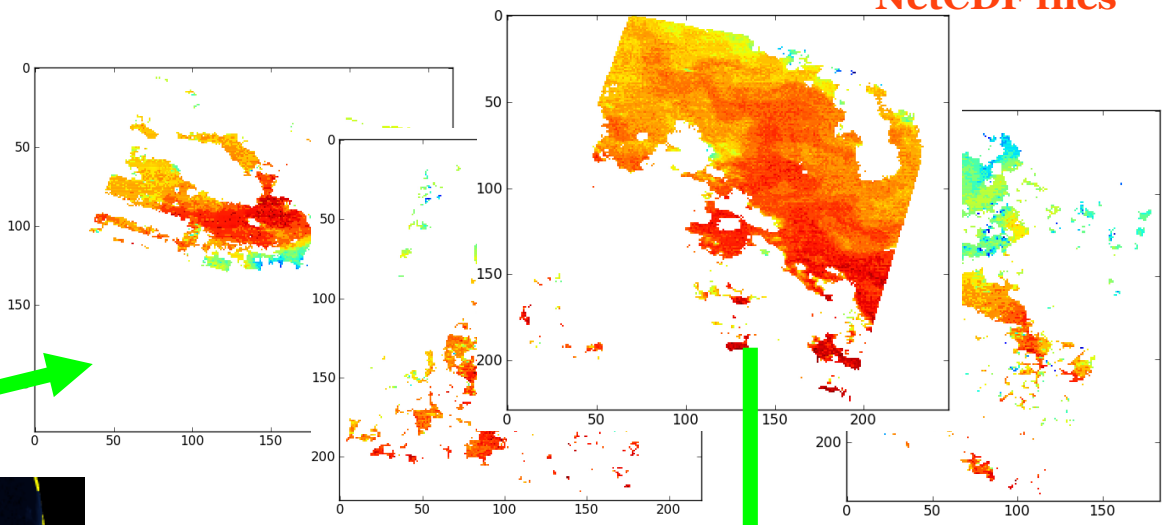
back-end concept

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

process quantitative, qualitative, stat metrics over miniprods



NetCDF files




```
source: 20130101-IFR-L4_GHRSST-SSTfnd-ODYSSEA-GL_B_010-v2.0-fv1.0.nc
felyx_dataset_name: ifr-l4-sstfnd-odyssea-glob_0_0_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
```

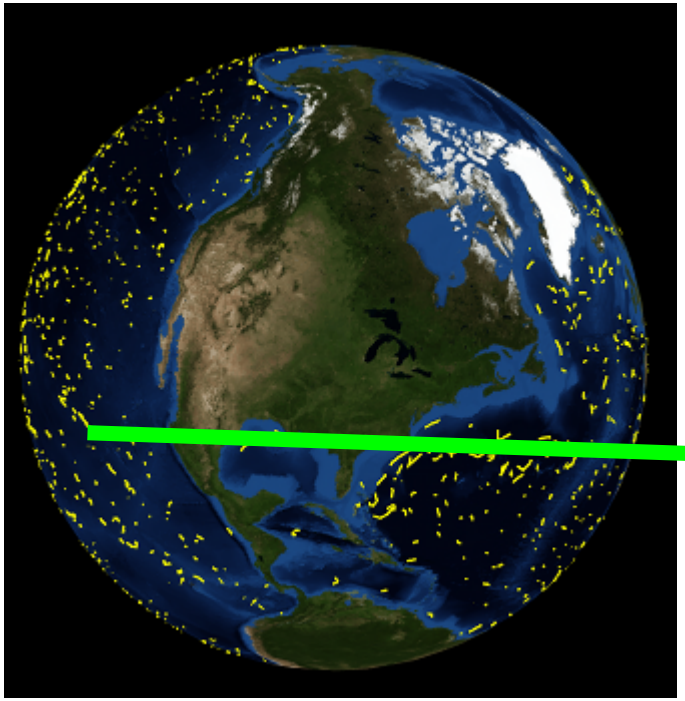

JSON files
indexed in a search
engine (ElasticSearch)

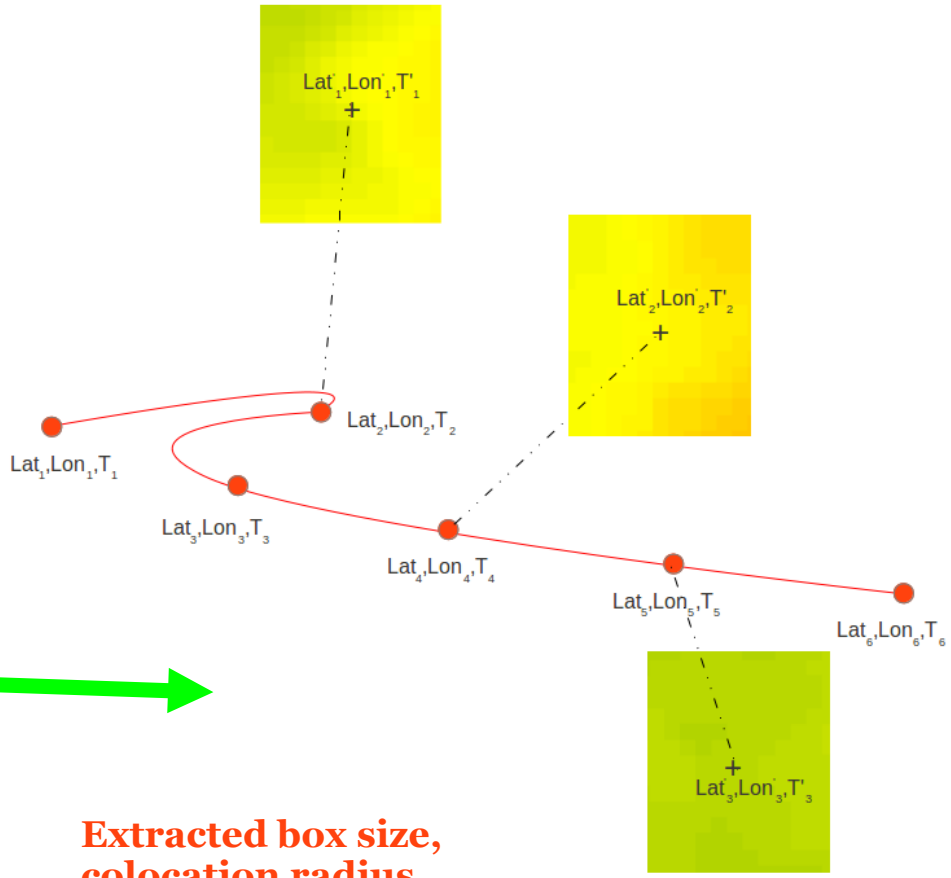
sites may be trajectories (buoys, cruise, hurricane)

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

felyx

Match-ups

Extraction of miniprods colocated with in situ data measurements (works the same for drifters, Argo floats or moorings)

```
find /home/cerdata/provider/neodc/l4/esacci_sst/ -name
'*.nc' | xargs -ifoo felyx-miniprod foo ostia-esacci-
l4-v01.0 --no-empty-miniprods=analysed_sst --dynamic
-o /home3/br156-200/project/felyx/manifest/
```

In CDAF context, just a preconfigured script :

```
$ run_miniprods
```

```
.
├── cci13001
│   └── ostia-esacci-l4-v01.0
│       ├── 2006
│       │   ├── 159
│       │   │   └── 20060608120000_cci13001_ostia-esacci-l4-v01.0_20060608120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 160
│       │   │   └── 20060609120000_cci13001_ostia-esacci-l4-v01.0_20060609120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 161
│       │   │   └── 20060610120000_cci13001_ostia-esacci-l4-v01.0_20060610120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 162
│       │   │   └── 20060611120000_cci13001_ostia-esacci-l4-v01.0_20060611120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 163
│       │   │   └── 20060612120000_cci13001_ostia-esacci-l4-v01.0_20060612120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 164
│       │   │   └── 20060613120000_cci13001_ostia-esacci-l4-v01.0_20060613120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 165
│       │   │   └── 20060614120000_cci13001_ostia-esacci-l4-v01.0_20060614120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 166
│       │   │   └── 20060615120000_cci13001_ostia-esacci-l4-v01.0_20060615120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 167
│       │   │   └── 20060616120000_cci13001_ostia-esacci-l4-v01.0_20060616120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 168
│       │   │   └── 20060617120000_cci13001_ostia-esacci-l4-v01.0_20060617120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 169
│       │   │   └── 20060618120000_cci13001_ostia-esacci-l4-v01.0_20060618120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 170
│       │   │   └── 20060619120000_cci13001_ostia-esacci-l4-v01.0_20060619120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 171
│       │   │   └── 20060620120000_cci13001_ostia-esacci-l4-v01.0_20060620120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 172
│       │   │   └── 20060621120000_cci13001_ostia-esacci-l4-v01.0_20060621120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
│       │   ├── 173
│       │   │   └── 20060622120000_cci13001_ostia-esacci-l4-v01.0_20060622120000-ESACCI-L4_GHRSST-SSTdepth-OSTIA-GLOB_LT-v02.nc
```

esa Ifremer pelamis scientific software PML Plymouth Marine Laboratory

Metrics production and registration (optional)

```
$ find /home3/br156-200/project/felyx/manifest_dynamic/ostia-esacci-l4-v01.0 -name '*.manifest'
| xargs -ifoo felyx-metric -o /home3/br156-200/project/felyx/metrics_dynamic/ostia-esacci-l4-
v01.0 --no-clean –storage file_and_elastic_search foo
```

In CDAF context, just a preconfigured script :

```
$ run_metrics
```

Metrics

Metrics are operators applied to the content of the extracted miniprods

Metrics processing can be limited to the site mask content (by default).

Conditions can be applied to data before calculation (field vs threshold : « quality equal good »)

Metric operators can be :

- Statistical : mean, standard deviation, median, skewness, kurtosis, min, max, …
- Data manipulation : center value
- Conditions : ice presence, cloud coverage, night time/day time
- Could qualitative too => any operation returning a sngle float or string value

Examples of metrics

For a GHRSST L2P product

mean sea surface temperature (quality >= acceptable)	`mean( {"field": "sea_surface_temperature", "must_have": [{"operator": "greater_equal", "field": "quality_level", "value": 4}]} )`
mean sses bias (quality >= acceptable)	`mean( {"field": "sses_bias", "must_have": [{"operator": "greater_equal", "field": "quality_level", "value": 4}]} )`
mean sses standard deviation (quality >= acceptable)	`mean( {"field": "sses_standard_deviation", "must_have": [{"operator": "greater_equal", "field": "quality_level", "value": 4}]} )`
mean wind speed	`mean( {"field": "wind_speed"} )`
day or night status	`day_or_night( {} )`
sea surface temperature standard deviation (quality >= acceptable)	`standard_deviation( {"field": "sea_surface_temperature", "must_have": [{"operator": "greater_equal", "field": "quality_level", "value": 4}]} )`
ice presence	`ice_presence( {} )`

Other metric examples

Sst difference between sat and in situ SST

Time difference between sat and in situ SST

Mean SST in NxN box

Stddev in NxN box

Quality level

Clear sky percentage in NxN box

Ice percentage in NxN box

Sun zenital angle

Nightime/daytime value

….

Everything that is required as condition or input later to filter, compare, select, produce diagnostics, etc...

Complete match-up assembling (ex : monthly file)

```
felyx-build-matchups --datasets-name ostia-esacci-l4-v01.0 --site-collection cci_gtmba --start
2000-01-01 --stop 2000-01-02 -v --add-dynamic-data --history 360 --in-memory
```

Match-up files format

/home/jfpiolle/GHRSST17/format.txt

Add in situ data to the server

```
curl -XPOST -k -F type=dynamic -F
file=@gtmba.csv
'https://felyx.cersat.fr/services/csv/'
```

Will be pre-ingested in CDAF context. Operation is to be performed on any other in situ dataset not shipped with the tool.

In situ trajectory format

target;cm6101531;drifting buoy 6101531;drifting buoy with identifier 6101531;{cmems_drifter};{};felyx;
position;
6.66200;40.74300;20160519T010042;
6.65400;40.74200;20160519T020008;
6.63700;40.74400;20160519T040004;
6.63000;40.74700;20160519T050044;
6.62000;40.75100;20160519T070003;
6.61800;40.75300;20160519T080013;
6.61600;40.75400;20160519T090043;
6.61500;40.75500;20160519T100012;
6.61200;40.75500;20160519T110013;
6.61000;40.75400;20160519T120043;
6.60700;40.75300;20160519T130010;
6.60500;40.75200;20160519T140012;
6.60200;40.75000;20160519T150008;
6.59800;40.74900;20160519T160035;
6.59300;40.74700;20160519T170042;
6.58800;40.74600;20160519T180010;
6.58100;40.74500;20160519T190012;
6.57500;40.74500;20160519T200042;
6.56900;40.74400;20160519T210013;
6.56200;40.74400;20160519T220206;
6.55600;40.74500;20160519T230113;
target;cm44857;drifting buoy 44857;drifting buoy with identifier 44857;{cmems_drifter};{};felyx;
position;
-35.66300;42.75000;20160519T000000;
-35.65700;42.75100;20160519T010000;
-35.65300;42.74900;20160519T020000;
-35.65300;42.74500;20160519T030000;
-35.66000;42.74100;20160519T040000;
-35.67000;42.73600;20160519T050000;
-35.68500;42.73200;20160519T060000;

Both trajectories and data should be merged in future version – this step won't be necessary anymore

In situ data format

target ; 6101531

fields ; Latitude ; Longitude ; Time ; solar_zenith_angle ; climatology_water_temperature ; water_temperature_0 ; water_temperature_1 ; water_temperature_2 ; water_temperature_3 ; water_temperature_4 ; meas_depth_0 ; meas_depth_1 ; meas_depth_2 ; meas_depth_3 ; meas_depth_4 ; quality_level_0 ; quality_level_1 ; quality_level_2 ; quality_level_3 ; quality_level_4 ; rejection_flag_0 ; rejection_flag_1 ; rejection_flag_2 ; rejection_flag_3 ; rejection_flag_4 ;

position ;

40.7430000305 ; 6.66200017929 ; 2016-05-19T01:00:42 ; 115.61 ; 17.56 ; 16.7600002289 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7420005798 ; 6.65399980545 ; 2016-05-19T02:00:08 ; 109.79 ; 17.56 ; 16.7600002289 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7439994812 ; 6.63700008392 ; 2016-05-19T04:00:04 ; 92.6 ; 17.56 ; 16.7600002289 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7470016479 ; 6.63000011444 ; 2016-05-19T05:00:44 ; 82.29 ; 17.56 ; 16.6800003052 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 38 ;

40.7509994507 ; 6.61999988556 ; 2016-05-19T07:00:03 ; 60.06 ; 17.56 ; 16.6000003815 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7529983521 ; 6.61800003052 ; 2016-05-19T08:00:13 ; 48.73 ; 17.56 ; 16.8400001526 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7540016174 ; 6.61600017548 ; 2016-05-19T09:00:43 ; 37.78 ; 17.56 ; 16.6800003052 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7550010681 ; 6.61499977112 ; 2016-05-19T10:00:12 ; 28.06 ; 17.56 ; 16.6800003052 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7550010681 ; 6.61199998856 ; 2016-05-19T11:00:13 ; 21.6 ; 17.56 ; 16.6000003815 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7540016174 ; 6.61000013351 ; 2016-05-19T12:00:43 ; 21.72 ; 17.56 ; 16.5200004578 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7529983521 ; 6.60699987411 ; 2016-05-19T13:00:10 ; 28.34 ; 17.56 ; 16.4400005341 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

40.7519989014 ; 6.60500001907 ; 2016-05-19T14:00:12 ; 38.13 ; 17.56 ; 16.4400005341 ; null ; null ; null ; null ; 0.5 ; 50.0 ; null ; null ; null ; 4 ; 0 ; 0 ; 0 ; 0 ; 2 ; 38 ; 38 ; 38 ; 38 ;

Both trajectories and data should be merged in future version

Improvements for CDAF (but not only)

Interface to run each of previous operation

- Select in situ source
- Run miniprod production and registration
- Run match-up extraction
- Etc...

Possible deployment options

Run remotely as a service :

- Felyx is deployed on a remote host (cloud)
- A user pushes its dataset
- Match-ups and comparisons are run through a web interface
- (+) easy to run for users
- (+) comparison with other datasets possible, central repository for metrics / diagnostic of all datasets
- (-) requires a host for the service
- (-) requires push of complete dataset

Run locally on a cluster :

- Install from source code and run felyx on your local cluster
- (+) best performances (processing time)
- (-) installation requires still some work of our team to make it easier

Run locally on a single machine (virtual machine or docker)

- Install a fully pre-configured felyx system shipped pre-installed on a virtual machine
- (+) easy to run
- (-) slower performances
- Tutorial for running a felyx VM : http://felyx.readthedocs.io/en/latest/virtualbox_installation.html

felyx

Access and visualization

In [4]:

```
from datetime import datetime

from pyfelyx.query.selection import MetricSelection
from pyfelyx.query.instance import Instance

inst = Instance(url='http://localhost:1080/felyx')

query = MetricSelection(
                            sites=['cci53057'],
                            datasets=['ostia-esacci-l4-v01.0'],
                            metrics=['matchup_sst', 'insitu_sst'],
                            start=datetime(2001,1,1),
                            end=datetime(2001,12,31)
                            )

result = query.run(inst)
```

In [12]:

```
from datetime import datetime

from pyfelyx.query.selection import MetricSelect
from pyfelyx.query.instance import Instance

inst = Instance(url='http://localhost:1080/felyx

query = MetricSelection(
                            sites=['cci53057'],
                            datasets=['ostia-esa
                            metrics=['matchup_ss
                            start=datetime(1991,
                            end=datetime(2010,12
                            )

result = query.run(inst)
```

In [5]:

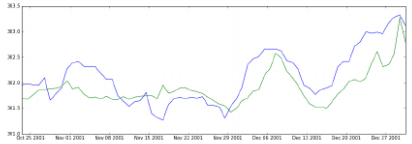
```
matchup_sst = result['results']['ostia-esacci-l4-v01.0']['cci53057']['metr
ics']['matchup_sst']
insitu_sst = result['results']['ostia-esacci-l4-v01.0']['cci53057']['metri
cs']['insitu_sst']
times = result['results']['ostia-esacci-l4-v01.0']['cci53057']['times']
```

In [13]:

```
matchup_sst = result['results']['ostia-esacci-l4
ics']['matchup_sst']
times = result['results']['ostia-esacci-l4-v01.0
```

In [6]:

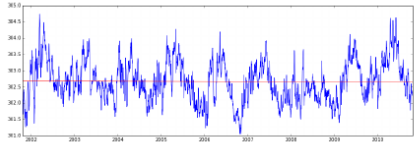
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15,5))

plt.plot(times, matchup_sst)
plt.plot(times, insitu_sst)
plt.show()
```



In [14]:

```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15,5))

plt.plot(times, matchup_sst)

#Linear regressison -polyfit
from scipy import polyval, polyfit, sqrt

days = [(_- datetime(2007,1,1)).days for _ in times]
(ar,br)=polyfit(days,matchup_sst,1)
xr=polyval([ar,br], days)

print('Linear regression using polyfit')
print('regression: a=%f b=%f' % (ar,br))
plt.plot(times,xr,'r-')

plt.show()
```

```
Linear regression using polyfit
regression: a=-0.000012 b=302.664413
```



API

Allow to query, process and display programmatically metrics and miniprods

Build custom diagnostics and plots

Cross-queries

Example : usage with iPython notebooks

FTP/OpenDAP access to miniprods can be offered too.

esa

Ifremer

pelamis scientific software

PML Plymouth Marine Laboratory

felyx

localhost:8888/notebooks/git/felyx-notebooks/CCI time series and trend.ipynb

jupyter **CCI time series and trend** Last Checkpoint: 05/05/2016 (autosaved)

File Edit View Insert Cell Kernel Help Python 2

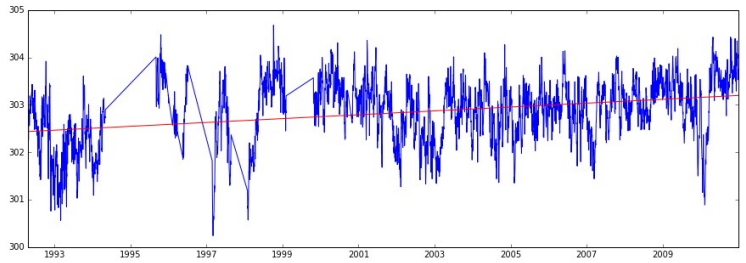
Code CellToolbar

In [1]:
```
from datetime import datetime

from pyfelyx.query.selection import MetricSelection
from pyfelyx.query.instance import Instance

inst = Instance(url='http://localhost:1080/felyx')

site = 'cci52307'

query = MetricSelection(
                            sites=[site],
                            datasets=['ostia-esacci-l4-v01.0'],
                            metrics=['matchup_sst'],
                            start=datetime(1991,9,1),
                            end=datetime(2011,12,31)
                            )

result = query.run(inst)
```

In [2]:
```
matchup_sst = result['results']['ostia-esacci-l4-v01.0'][site]['metrics']['matchup_sst']
times = result['results']['ostia-esacci-l4-v01.0'][site]['times']
```

In [3]:
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15,5))

plt.plot(times, matchup_sst)

#Linear regressison -polyfit
from scipy import polyval, polyfit, sqrt

days = [(_ - datetime(2007,1,1)).days for _ in times]
(ar,br)=polyfit(days,matchup_sst,1)
xr=polyval([ar,br], days)

print('Linear regression using polyfit')
print('regression: a=%f b=%f' % (ar,br))
plt.plot(times,xr,'r-')

plt.show()
```

```
Linear regression using polyfit
regression: a=0.000111 b=303.035726
```



In [ ]:

esa Ifremer pelamis scientific software PML Plymouth Marine Laboratory

felyx

localhost:8888/notebooks/git/felyx-notebooks/buoy comparison with ESA CCI data.ipynb

jupyter **buoy comparison with ESA CCI data** Last Checkpoint: 27 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help Python 2

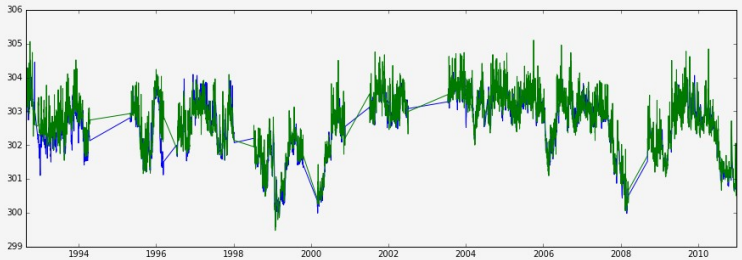
Code CellToolbar

In [12]:
```
from datetime import datetime
import numpy

from pyfelyx.query.selection import MetricSelection
from pyfelyx.query.instance import Instance

inst = Instance(url='http://localhost:1080/felyx')

site = 'cci52307'
site = 'cci52321'

query = MetricSelection(
                            sites=[site],
                            datasets=['ostia-esacci-l4-v01.0'],
                            metrics=['matchup_sst', 'insitu_sst'],
                            start=datetime(1991,1,1),
                            end=datetime(2011,12,31)
                            )

result = query.run(inst)
```

In [13]:
```
matchup_sst = result['results']['ostia-esacci-l4-v01.0'][site]['metrics']['matchup_sst']
insitu_sst = result['results']['ostia-esacci-l4-v01.0'][site]['metrics']['insitu_sst']
times = result['results']['ostia-esacci-l4-v01.0'][site]['times']


print type(insitu_sst)
```

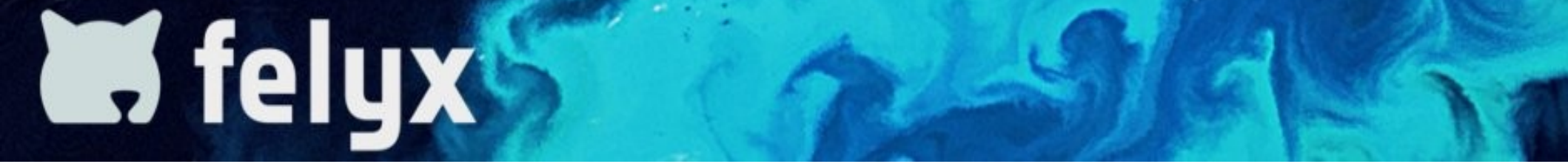
```
<class 'numpy.ma.core.MaskedArray'>
```

In [14]:
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15,5))

plt.plot(times, matchup_sst)
plt.plot(times, insitu_sst)
plt.show()
```



In [15]:
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15,15))

plt.plot(insitu_sst, matchup_sst, "o")
plt.show()
```

esa Ifremer pelamis scientific software PML Plymouth Marine Laboratory



localhost:8888/notebooks/git/felyx-notebooks/buoy comparison with ESA CCI data.ipynb

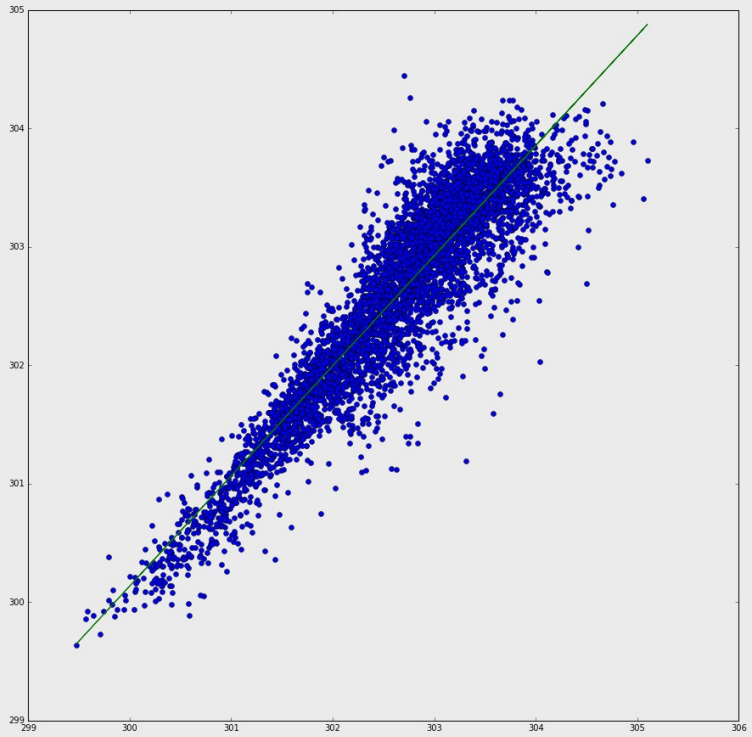
jupyter buoy comparison with ESA CCI data Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

299 1994 1996 1998 2000 2002 2004 2006 2008 2010

In [16]:
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(15,15))

plt.plot(insitu_sst, matchup_sst, "o")
plt.plot(insitu_sst, numpy.poly1d(numpy.polyfit(insitu_sst, matchup_sst, 1))(insitu_sst))
plt.show()
```



In [ ]:

esa

Ifremer

pelamis scientific software

PML Plymouth Marine Laboratory

felyx

config your report

Visitor

Home > Analyse > Report[874c3a03d95ecef5437f1681ae66ebcfde8f42cf]

PLOTS VIEW | REPORTS | BOOKMARKS

Show

Sites

can002
Gulf Stream (Buoy Station 44141)

ghr118
Coral Reef Early Warning System Station French Frigate Shoals (Iridium)

ghr115
Coral Reef Early Warning System Station Kure Atoll

Datasets

No preview

OSTIA L4 product from the ESA SST CCI project, produced using OSTIA reanalysis system v2.0
ostia-esacci-l4-v01.0

No preview

Plots

HISTOGRAM
Mean Sea Surface Temperature
ostia-ukmo-l4-glob-v2.0
Bin size: 1

TIME SERIES
Mean Sea Surface Temperature
ostia-ukmo-l4-glob-v2.0
Resampling: none

Time range

START DATE/TIME
1991-09-01 12:00:00

END DATE/TIME
2014-11-02 12:00:00

Funded by the European Space Agency
Designed and built by Ifremer with the support of Pelamis and PML.
Code licensed under GPLv3, documentation under Sphinx.

esa Ifremer PML Plymouth Marine Laboratory pelamis scientific software

Contact administrator

esa Ifremer pelamis scientific software PML Plymouth Marine Laboratory

felyx

Felyx | Home | Configure | Monitor | Analyse | Ifremer

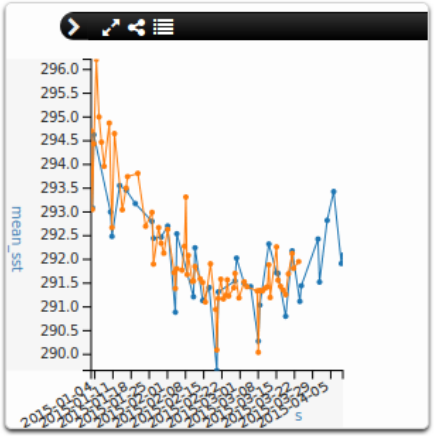
Visitor

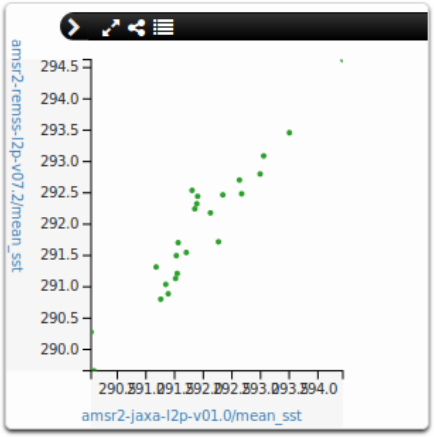
Home > Analyse > Report[8dce259f73e74ce65a48b961812186b91fb89d0e] > View

+ Create new report | Manage bookmarks | Manage reports | Save bookmark | Save report

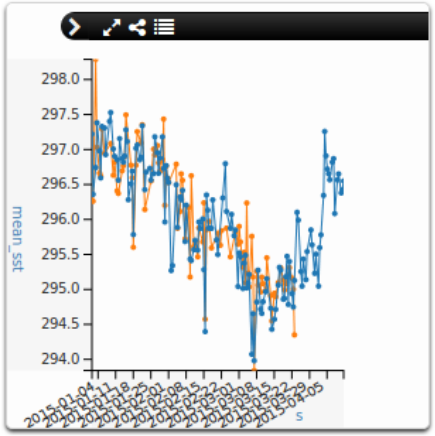
Settings

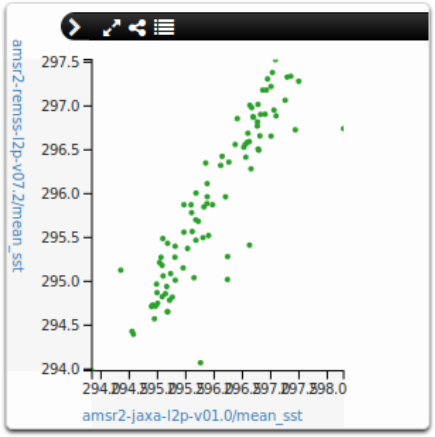
CREWS_Kure_Atoll_21392



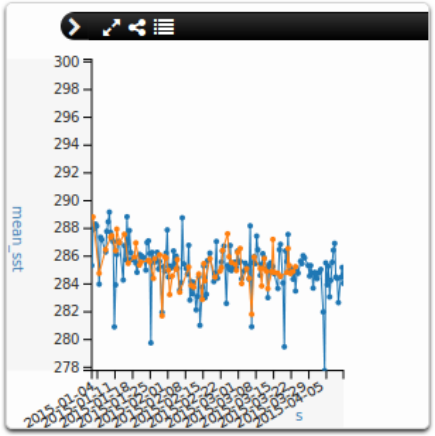


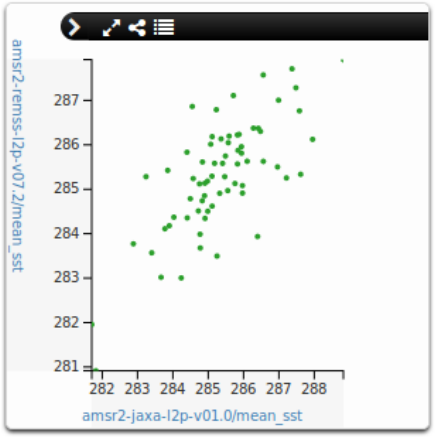
CREWS_French_Frigate_Shoals_261003





CAN_Gulf_Stream_44141





Plymouth Marine Laboratory

CDAF / complementarity with SQUAM

Baseline : felyx produced the match-ups vs reference in situ : miniprods and metrics

Option 1 : plots and diagnostics are produced in python from the registered metrics (like in previous examples)

(+) full python solution

(+) very dynamic and flexible : anybody can change the area or buoy of interest, selection and plot criteria on-the-fly or offline, add new ones using these examples, use as resources for teaching, etc.

(-) all diagnostics and plots for CDAF have to be coded (python/matplotlib equivalent to SQUAM plots)

Option 2 : plots and diagnostics are produced by SQUAM routines from the registered metrics = SQUAM is a new client of a felyx server

(-) dependency on IDL : can it be run end-to-end without any licence ?

(-) no flexibility to modify code (without licence) ?

(+) re-use existing pieces of code of SQUAM

(+) interactive web display without the need of SQUAM backend ?

(+) no need to generate match-up files (just the metrics)

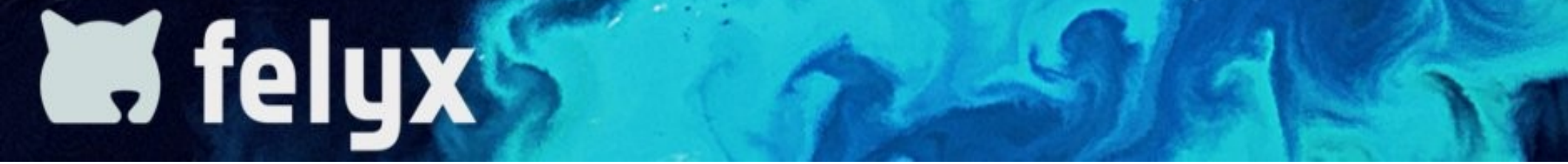
Option 3 : plots and diagnostics are produced by SQUAM routines from the assembled match-up files

(+/-) same as above

Data access from files rather than web services

Option 4 : combination of option 1 and option 2/3

Hard-code display of SQUAM + custom plot/diagnostics in python



Thanks

esa

Ifremer

pelamis scientific software

PML Plymouth Marine Laboratory

Sentinel-3 MDB : workflow

Colocation window : 2h, 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 :

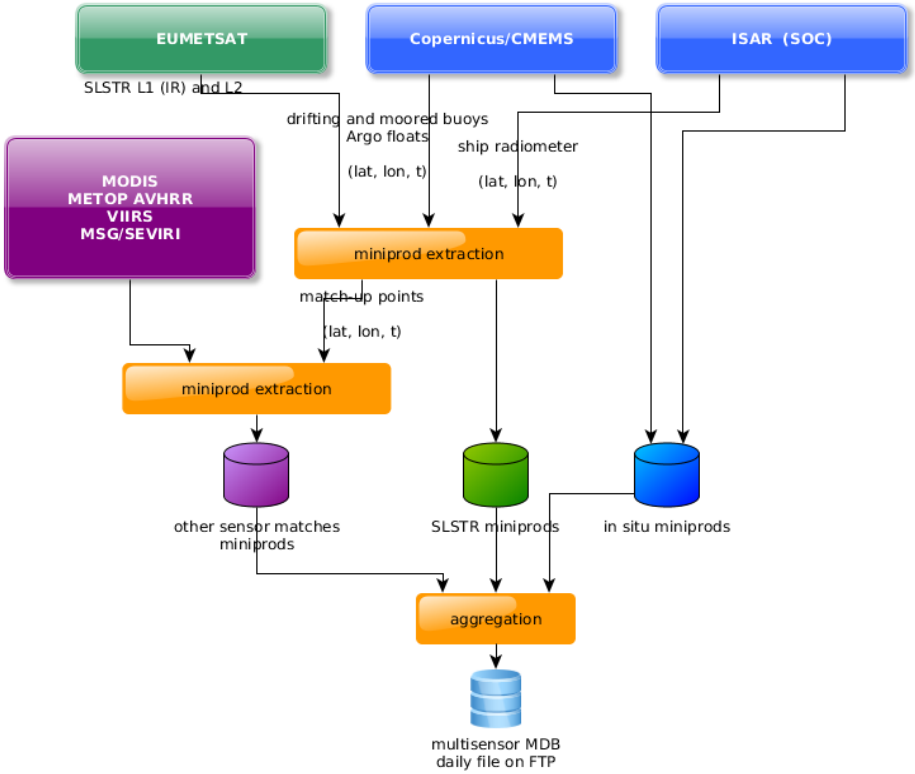
- L1 infra-red channels
- L2 (SST) – all fields, incl. meteo and ancillary fields

Other sensor data

- Metop-B/AVHRR, MODIS, VIIRS, MSG/SEVIRI

Resampling of all data to SLSTR grid

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



Distributed and scalable architecture

Backend : process miniprods and metrics

Front-end : system configuration, web interface and API to query miniprods and metrics

Celery :perform distributed processing

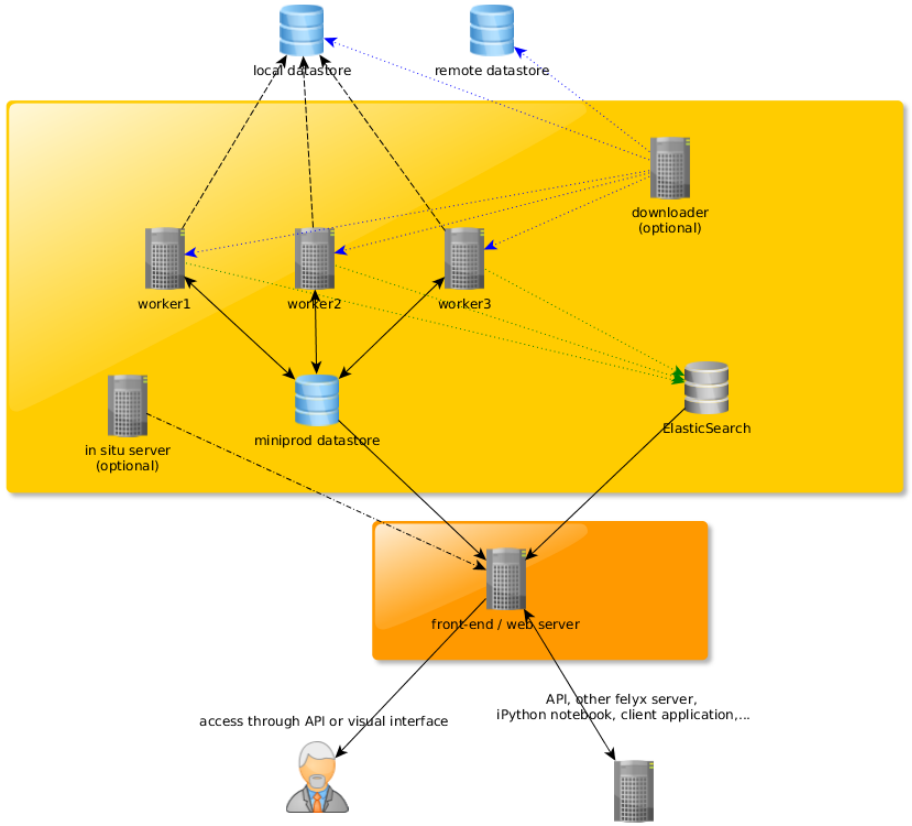
Elastic-search : (no sql) database

In situ server : implements similar back-end/front-end functions and store in situ data

Downloader : plug felyx to local or remote data stores (automatic processing)

Access through RESTful API and web interface

Cross-felyx server queries



Installation and deployment

- source code will be freely available (GIT server), GPLv3 license : users can do pretty much what they want, modification (sharing is encouraged), commercial usage allowed
- Fully documented : installation, administration, user and developer guide, reference documentation
- Software can be installed from source code or deployed using the provided linux container (http://en.wikipedia.org/wiki/LXC)
- Building downstream applications on top of felyx servers:

  Instances share a common, base RESTfull API.

  Users are encouraged to build their own applications, and to host their own instance with their own data, configured to their needs.!
- System tailoring : parameter, datasets, metrics, front-end, ….

  Felyx is fully written in python (back-end) and javascript (front-end), using third party components compatible with above licensing

Felyx is agnostic and extensible

Can handle any type of acquisition pattern :

Swath, grid, trajectories or along-track, …

Can handle any kind of geophysical parameter

Can handle any kind of domain : ocean, land, atmosphere, cryosphere, …

Can handle new data format through an extensible plugin mechanism that can be tailored by any administrator of felyx system

Metrics processing can be extended too through plugin mechanism

Documentation : felyx.readthedocs.org

Source code and packages : python implementation, git server, GPLv3 open source licence. Access link will be provided on http://felyx.org

Useful dependencies :

Cerbere package : generic API to various formats and observation patterns.

Documentation : http://cerbere.readthedocs.fr

Git : https://git.cersat.fr/cerbere