

# DESTINATION EARTH

---

## AQUA

THE QUALITY ASSESSMENT TOOL FOR THE  
CLIMATE DIGITAL TWIN

*Silvia Caprioli and the AQUA team*



Politecnico  
di Torino

3rd Destination Earth User eXchange, Darmstadt, 15 October 2024

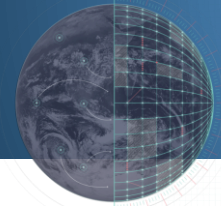


Funded by  
the European Union

**Destination Earth**

implemented by





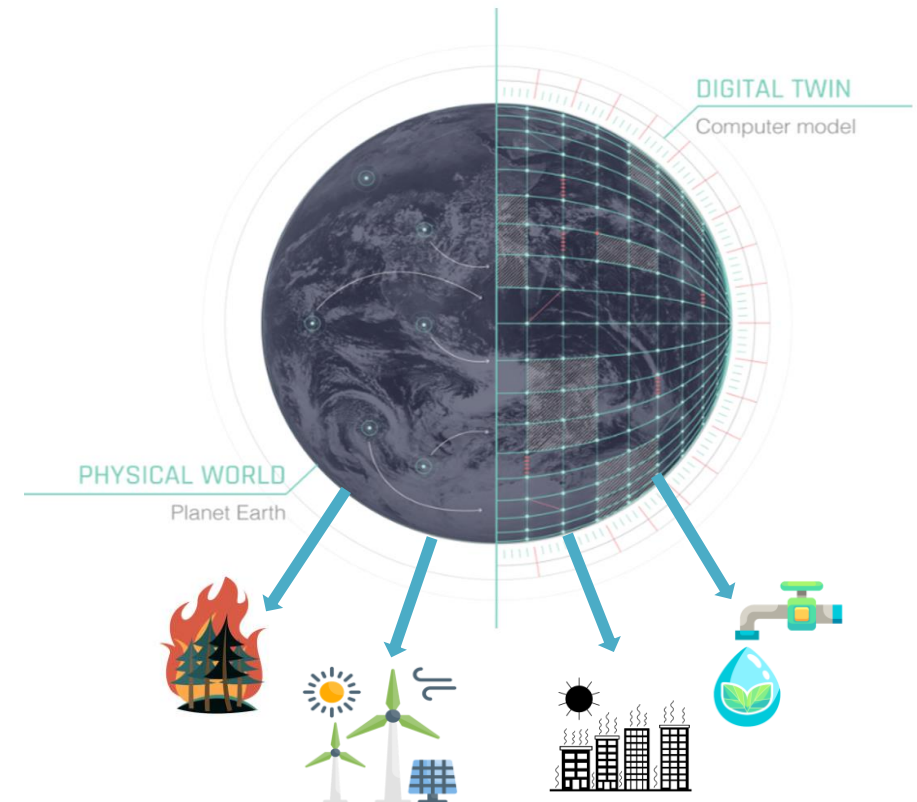
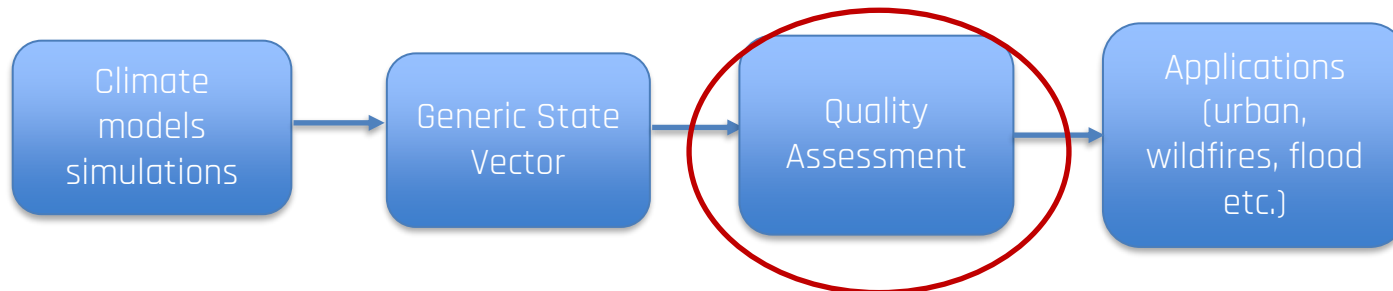
## Destination Earth: The Climate Digital Twin

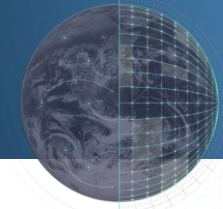
### The Climate Digital Twin

#### Key Features:

- ✓ Global climate simulations at **km-scale horizontal resolution**
- ✓ User-driven approach focused on **user interactivity**
- ✓ **Streaming framework** from climate model output to impact sectors
- ✓ **Quality assessment** and **uncertainty quantification** based on observations

#### AQUA Application for Quality Assessment

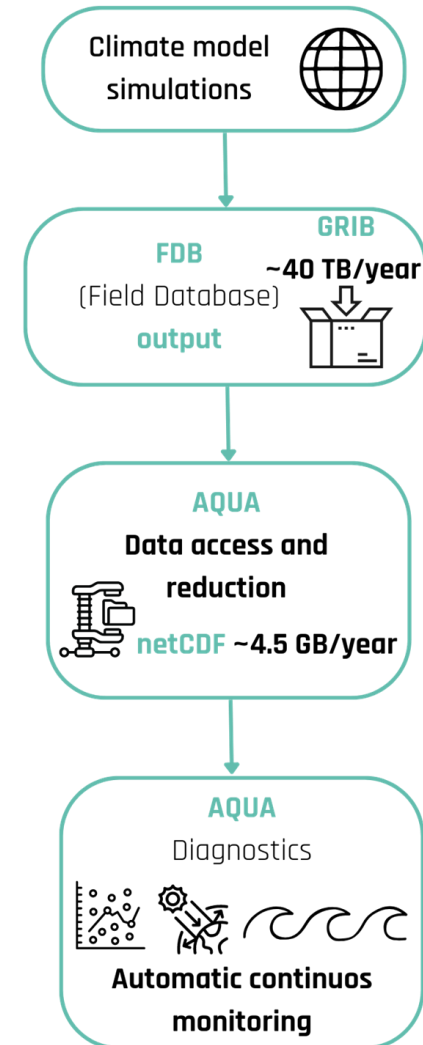


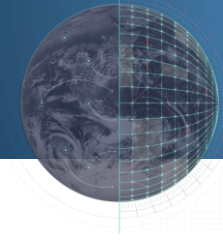


## Model evaluation in the Climate DT: The need for AQUA

Many tools are available for climate model analysis (ESMValTool, E3SM, CMEC...). However, Climate DT simulations presents specific challenges:

- Global climate simulations at **unprecedented horizontal** (5 km atm and ocean) and **temporal resolution** (1 hour)
- **Amount of data** for a single simulation **is critically high**: 40 TB per simulated year (at the moment: 5.7 PB of Climate DT data in total)
- **Data access** through FDB, with data both locally on HPC and remote on Data Bridge/Data Lake
- Flexible to support the introduction of a **streaming framework** for **continuous and automatic monitoring** of model output
- Deployment and portability on multiple European **pre-exascale supercomputers** (LUMI, Levante, MareNostrum5 etc.)

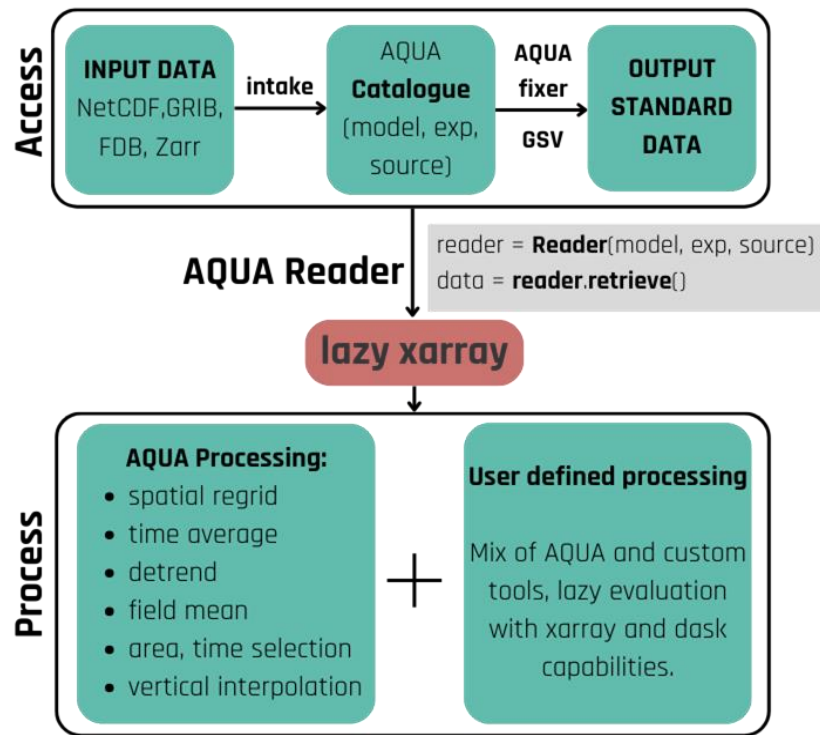




## The Philosophy of AQUA

AQUA provides a framework to **access**, **process** and **analyze** large volumes of climate data

- **making data access simple** (two lines of code to get a dask-enabled xarray)
- **dealing with all the technicalities** (grid weights, FDB requests, different variable names and units, ecCodes versions) **behind the scenes.**



**Python3** library package based on **xarray**, **dask**, **CDO** and **intake**

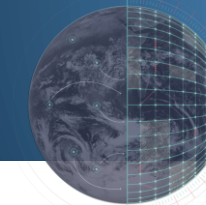
**Technical details are hidden** simple user experience

**Lazy access** to all data (parallel **dask** enabled) including **FDB**

**Modular structure** seamless integration between core functions

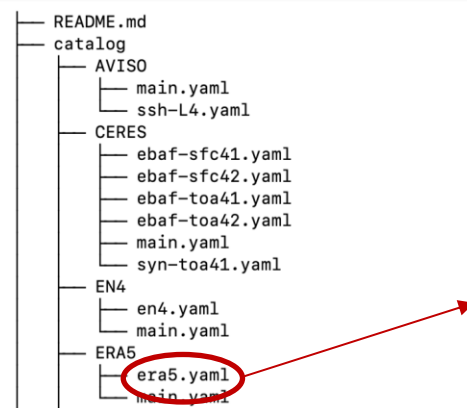
**Fast regrid capabilities** based on precomputed weights

**CI/CD** on every new code addition with around 75% code coverage for the framework



## AQUA core concepts: the Catalog

- Folder containing **paths** and **technical details** of all the sources that the user may want to access (models and observational datasets)
- Built on **intake** package
- **3 level hierarchy: model** (e.g. IFS, ERA5)  
**experiment** (e.g. historical, scenario)  
**source** (e.g. hourly-native, monthly-r100)
- **FDB GRIB** data as well as **NetCDF**, **Zarr** and other formats can be used with small effort
- Machine independent



ERA5 as netCDF

```

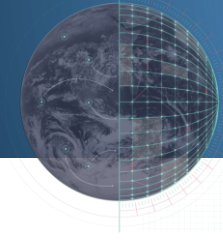
1  plugins:
2  |   source:
3  |   |   - module: intake_xarray
4
5  sources:
6  |   monthly:
7  |   |   description: ERA5 monthly data from 1940 to 2022
8  |   |   driver: netcdf
9  |   |   metadata:
10 |   |   |   source_grid_name: era5-r025
11 |   |   |   fixer_name: ERA5-destine-v1
12 |   |   args:
13 |   |   |   urlpath: '{{DATA_PATH}}/ERA5/mon/ERA5*_mon_full*.nc'
14 |   |   |   chunks:
15 |   |   |   |   time: 12
16 |   |   |   xarray_kwargs:
17 |   |   |   |   decode_times: True
  
```

IFS-NEMO on FDB

```

1  sources:
2  |   hourly-native-atm2d: &base-default
3  |   args: &args-default
4  |   request: &request-default
5  |   class: d1
6  |   dataset: climate-dt
7  |   activity: ScenarioMIP
8  |   experiment: SSP3-7.0
9  |   generation: 1
10 |   model: IFS-NEMO
11 |   realization: 1
12 |   resolution: high
13 |   expver: '0001'
14 |   type: fc
15 |   stream: clte
16 |   date: 20210101
17 |   time: '0000'
18 |   param: 167
19 |   levtype: sfc
20 |   step: 0
21 |   data_start_date: auto
22 |   data_end_date: auto
23 |   chunks: D # Default time chunk size
24 |   savefreq: h # at what frequency are data saved
25 |   timestep: h # base timestep for step timestep
26 |   timestepstyle: date # variable date or variable step
27 |   description: hourly data on native grid Tco2559 (about 5km).
28 |   driver: gsv
29 |   metadata: &metadata-default
30 |   fdb_home: /users/lrb_465000454_fdb/native
31 |   fdb_path: /users/lrb_465000454_fdb/native/etc/fdb/config.yaml
32 |   eccodes_path: /projappl/project_465000454/jvonhar/aqua/eccodes/eccodes-2.32.5/definitions
33 |   variables: [78, 79, 134, 137, 141, 148, 151, 159, 164, 165, 166, 167, 168, 186,
34 |   |   187, 188, 235, 260048, 8, 9, 144, 146, 147, 169, 175, 176, 177, 178, 179,
35 |   |   180, 181, 182, 212, 228]
36 |   source_grid_name: tco2559
37 |   fixer_name: ifs-destine-v1
  
```





## AQUA core concepts: the Reader

Once a catalog is added the user can retrieve the dataset with two lines of code!

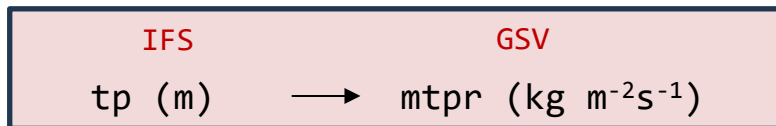


This is a lazy dask array!

```
from aqua import Reader
reader = Reader(model='IFS-NEMO', exp='historical-1990',
source='hourly-native-atm2d', regrid='r100')
data = reader.retrieve()
```

### Fixing

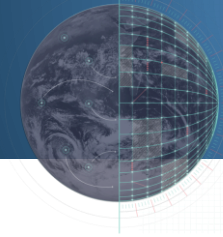
- Data are automatically retrieved in a **common metadata format** (GRIB standard)
- Fixes are defined when creating a catalog source and then **automatically enabled** when data are retrieved



- YAML file with **fixer\_name** that can be applied to multiple models/exps/sources

```
1 fixer_name:
2   ifs-destine-v1:
3     data_model: ifs
4     deltat: 3600
5     delete: [tprate]
6
7   vars:
8     # Derive precipitation from TP and drop tprate
9     mtr: # https://codes.ecmwf.int/grib/param-db/235055
10      nanfirst: true
11      source: tp
12      grib: true
13     # Evaporation and snowfall
14     mer: # https://codes.ecmwf.int/grib/param-db/?id=235043
15      source: e
16      grib: true
17      src_units: m
18      nanfirst: True # option to fix ifs bug in destine-v
19      #units: kg m**2 s**-1
20     msr: # https://codes.ecmwf.int/grib/param-db/?id=235031
21      source: sf
22      grib: true
23      src_units: m
24      nanfirst: True
25      #units: kg m**2 s**-1
26     # Latent and sensible heat
27     mslhf: # https://codes.ecmwf.int/grib/param-db/?id=2350
28      source: slhf
29      grib: true
30      nanfirst: True
31     msshf:
32      source: sshf
33      grib: true
34      nanfirst: True
```

```
ceres-ebaf-destine-v1:
  data_model: False
  vars:
    mtnlwrfl:
      derived: 0.-toa_lw_all_mon
      grib: true
      attributes:
        valid_min: -500
        valid_max: 0
        positive: down
    mtmswrf:
      derived: solar_mon-toa_sw_all_mon
      grib: true
      attributes:
        valid_min: 0
        valid_max: 1400
        positive: down
    msnlwrfl:
      derived: 0.-sfc_net_lw_all_mon
      grib: true
    msmswrf:
      derived: sfc_net_sw_all_mon
      grib: true
```



## AQUA core concepts: the Reader

Once a catalog is added the user can retrieve the dataset with two lines of code!



This is a lazy dask array!

```
from aqua import Reader
reader = Reader(model='IFS-NEMO', exp='historical-1990',
source='hourly-native-atm2d', regrid='r100')
data = reader.retrieve()
data_regridded = reader.regrid(data)
```

Prepare for regrid to 1° deg resolution

Actual regrid

### Regridding

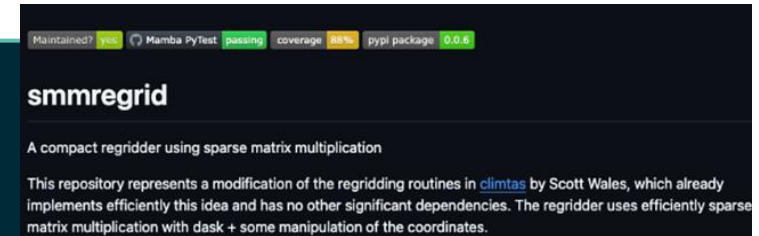
AQUA provides functions to interpolate and regrid data to match the spatial resolution of different datasets.

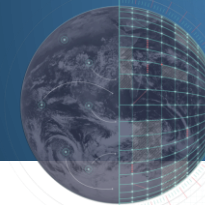
- Based on `smmregrid` package which operates **sparse matrix computation** based on **pre-computed weights**.
- Working on both native and **HealPix grids**
- Grids are defined in `yaml` file in AQUA and weights are evaluated with `CDO` only once.

```
grids:
# default
lon-lat:
  vert_coord: ["2d"]
  space_coord: [lon, lat]
lon-lat-depth:
  vert_coord: ["depth"]
  space_coord: [lon, lat]

# target regrid
r005: r7200x3600
r010: r3600x1800
r020: r1800x900
r025: r1440x720
r050: r720x360
r100: r360x180
r200: r180x90
r250: r144x72

# Generic Healpix
hp25-nested:
  space_coord: ["ncells"]
  vert_coord: ["2d"]
  path: '{{ grids }}/HealPix/hp25_nested_atm.nc'
hp27-nested:
  space_coord: ["ncells"]
  vert_coord: ["2d"]
  path: '{{ grids }}/HealPix/hp27_nested_atm.nc'
```





## AQUA processing

AQUA core classes and functions can be integrated in any analysis



- Use AQUA framework as a **library** in your analysis
- Use **available AQUA diagnostics**
- Build **your own AQUA diagnostic**

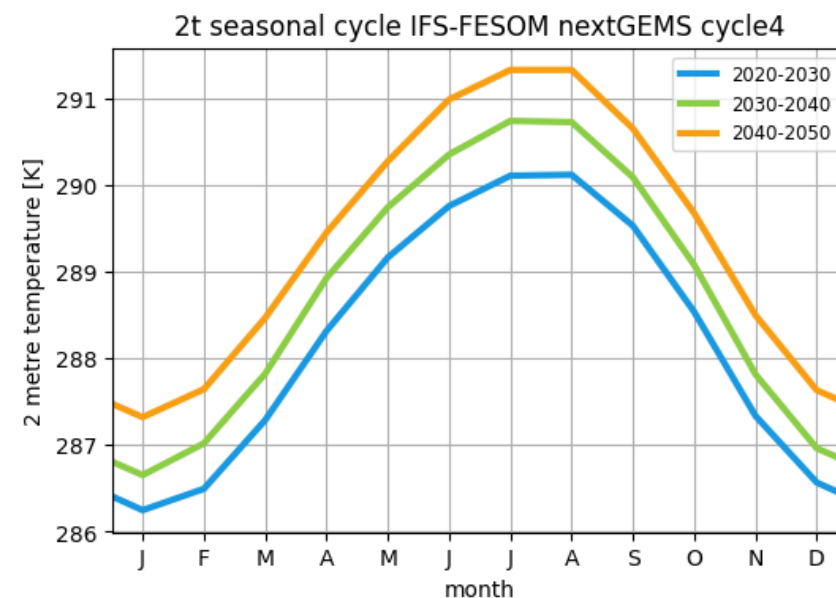
### Simple example of custom analysis

Custom processing for a seasonal cycle

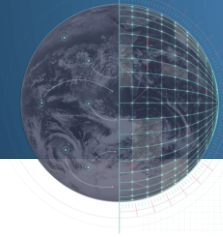
```
from aqua import Reader
reader = Reader(model='IFS-FESOM', exp='ssp370-ng4', source='lra-r100-monthly')
data_20_30 = reader.retrieve(var='2t', startdate='2020', enddate='2030')
cycle_20_30 = data_20_30['2t'].aqua.fldmean().groupby('time.month').mean('time')
```

Graphic utilities for a custom plot

```
from aqua.graphics import plot_seasonalcycle
plot_seasonalcycle(data=[cycle_20_30, cycle_30_40, cycle_40_50], data_labels=['2020-2030', '2030-2040', '2040-2050'], title='2t seasonal cycle IFS-FESOM nextGEMS cycle4')
```







## The diagnostics suite

### State-of-the-art

Provide general performance evaluation of the models and identification of biases and drifts.

Run on **low-resolution** data (daily/monthly, 1-deg)

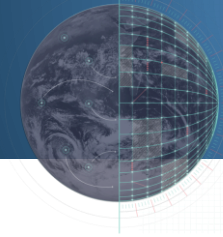
- **Performance indices:**  
evaluation of model biases compared with CMIP6
- **Timeseries and biases:**  
timeseries of temperature, precipitation etc.  
2d biases
- **Radiation budget:**  
model radiative budget imbalance
- **Sea ice extent:**  
seasonal cycles of sea ice extent in specific regions
- **Teleconnection indices:**  
NAO and ENSO indices and their pattern correlations
- **Ocean circulation evaluation:**  
evaluation of anomalies of oceanic variables.  
(temperature, salinity, density, mixed layer depth etc.)

### Frontier

Provide insights into physical/dynamical processes not yet explored in classical climate simulations

Run on **high-resolution** data (hourly, km-scale)

- **Sea Surface Height variability:**  
to evaluate surface ocean dynamics
- **Tropical Rainfall:**  
analysis of tropical precipitation extremes
- **Tropical cyclones detection, tracking and zoom-in:**  
detection and tracking of tropical cyclones.  
Set of variables (precipitation, wind etc...) around the Tc's center at high resolution



## Running diagnostics in AQUA

Diagnostics in AQUA are organized in a **modular structure** as separate Python classes/functions

- Run them independently and in parallel
- Easy to add custom diagnostics

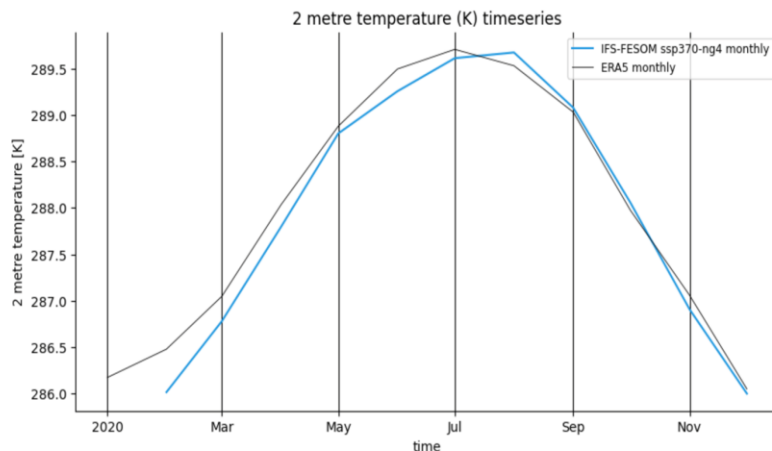
### Use directly a diagnostics' function in a Jupyter notebook

```
from global_time_series import Timeseries
```

✓ 4.3s

```
ts = Timeseries(var='2t', models='IFS-FESOM', exps='ssp370-ng4', sources='lra-r100-monthly',
               startdate='2020-01-01', enddate='2020-12-31',
               std_startdate='2020-01-01', std_enddate='2020-12-31',
               loglevel='INFO')
```

✓ 0.0s

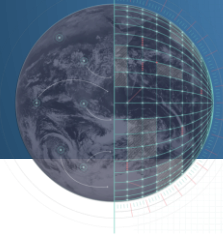


### Run a single diagnostic with a cli

```
series/cli> python cli_global_time_series.py -c config_time_series_atm.yaml
```

```
diagnostics > global_time_series > cli > ! config_time_series_atm.yaml
1 # The first model in this block can be overridden
2 # in the command line by using:
3 # --model, --exp, --source, --outputdir
4 models:
5   - model: 'IFS-NEMO'
6     exp: 'ssp370'
7     source: 'lra-r100-monthly'
8   outputdir: './'
9
10 # This is the list of variables that will be plotted as timeseries
11 timeseries: ['2t', 'mtrp', 'mer', 'mstl',
12             'tcc', 'lcc', 'mcc', 'hcc',
13             'mslhf', #Mean surface latent heat flux
14             'msshf', #Mean surface sensible heat flux
15             'msdswrf', #Mean surface downward short-wave radiation flux
16             'msdlwrf', #Mean surface downward long-wave radiation flux
17             'msnswrf', #Mean surface net short-wave radiation flux
18             'msnlwrf', #Mean surface net long-wave radiation flux
19             'mtnlwrf', #Mean top net long-wave radiation flux
20             'mtnswrf', #Mean top net short-wave radiation flux
21             ]
22 timeseries_formulae: ['mtnlwrf+mtnswrf']
23
24 gregory:
25   ts: '2t'
26   toa: ['mtnlwrf', 'mtnswrf']
27   monthly: True
```

**config file** where the user can specify model, exp, source, variables, output dir etc.



## Running diagnostics in AQUA

Diagnostics in AQUA are organized in a **modular structure** as separate Python classes/functions



- Run them independently and in parallel
- Easy to add custom diagnostics

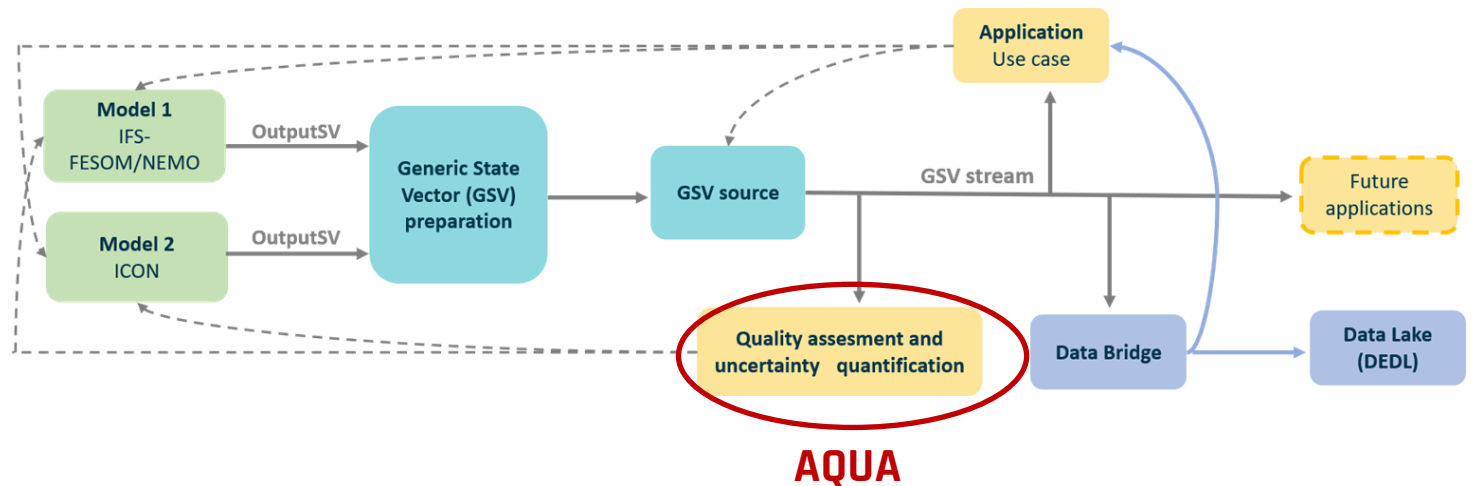
### Run all the diagnostics for a complete analysis

```

2024-04-04 14:34:11: Setting loglevel to WARNING
2024-04-04 14:34:11: Atmospheric model: IFS-NEMO
2024-04-04 14:34:11: Oceanic model: IFS-NEMO
2024-04-04 14:34:11: Experiment: historical-1990
2024-04-04 14:34:11: Source: lra-r100-monthly
2024-04-04 14:34:11: Machine: lumi
2024-04-04 14:34:11: Output directory: ./output
2024-04-04 14:34:11: Machine set to lumi in the config file
2024-04-04 14:34:11: Creating output directory ./output
2024-04-04 14:34:11: Running setup checker
2024-04-04 14:35:25: Finished setup checker
2024-04-04 14:35:25: Running tropical_rainfall
2024-04-04 14:35:25: Running global_time_series
2024-04-04 14:35:25: Running seasonal_cycles
2024-04-04 14:35:25: Running radiation
2024-04-04 14:35:25: Running teleconnections
2024-04-04 14:35:25: Running atmglobalmean
2024-04-04 14:35:25: Running global_time_series
2024-04-04 14:35:25: Running teleconnections
2024-04-04 14:35:26: Running ocean3d_drift
2024-04-04 14:35:26: Running ocean3d_circulation
2024-04-04 14:35:26: Running seaice_extent
2024-04-04 14:35:26: Running seaice_conc
2024-04-04 14:35:26: Running ecmean
    
```

### Real-time monitoring of ongoing simulations

Integration into the general Climate-DT **workflow** is in progress





## Real time monitoring: the AQUA explorer

Browser address bar: aqua-web-climatedt.2.rahtiapp.fi

Page title: AQUA Diagnostics Explorer

Navigation menu: AQUA Diagnostics Explorer, Home, About, Experiments

Header: Welcome to the AQUA Diagnostics Explorer

Text: This page summarizes results from AQUA diagnostics applied to ongoing DE340 simulations.

Text: Please choose an experiment from the menu on the left.

- HR = high resolution, about 5km
- SR = standard resolution, about 10 km
- MR = medium resolution, about 25 km
- LR = low resolution, about 144 km

Text: For full documentation visit [The AQUA documentation](#).

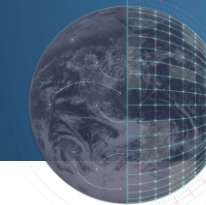
**Real-time monitoring** is already implemented for the running DestinE simulations through the [AQUA explorer](#)

Diagnostics are run everyday on Climate DT runs and all the results are published on a dedicated website.

Developed by AQUA using infrastructure by CSC

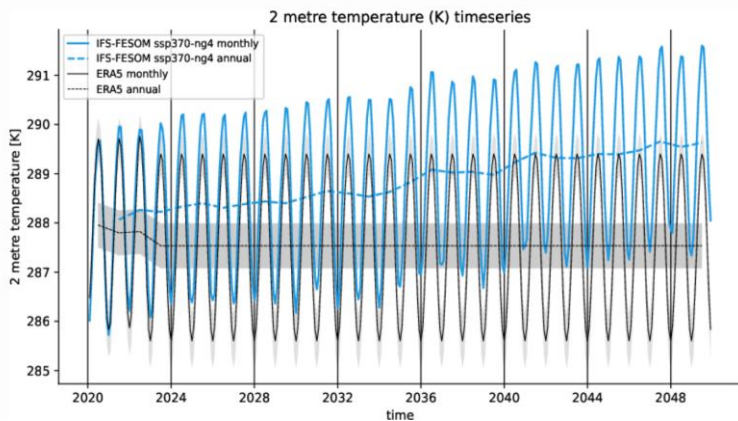
- Automatic experiment page generation
- Integrated and dynamical figure caption generation
- Single tab for each experiment
- It will be replaced by a dashboard



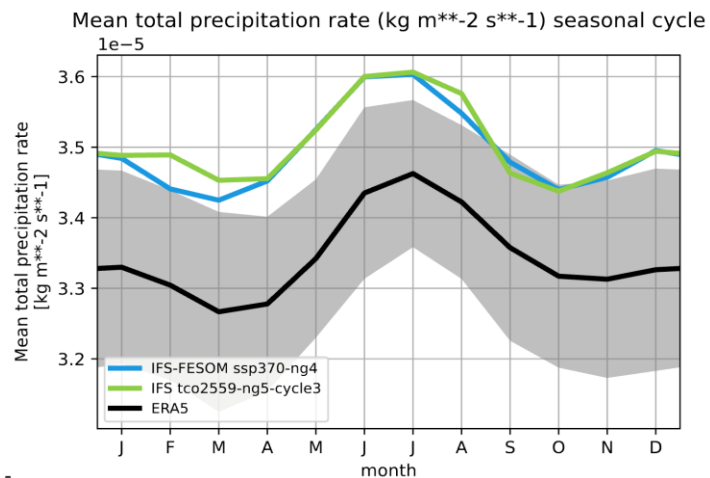


# Example plots

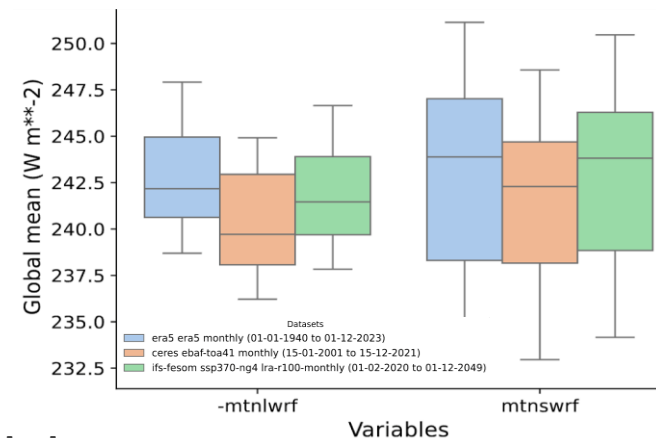
## Global timeseries



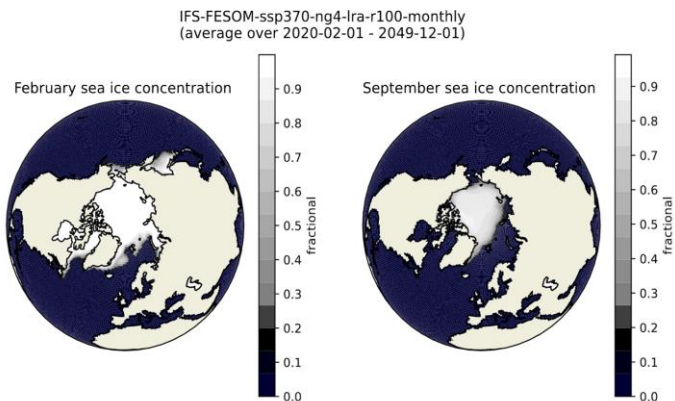
## Seasonal Cycles



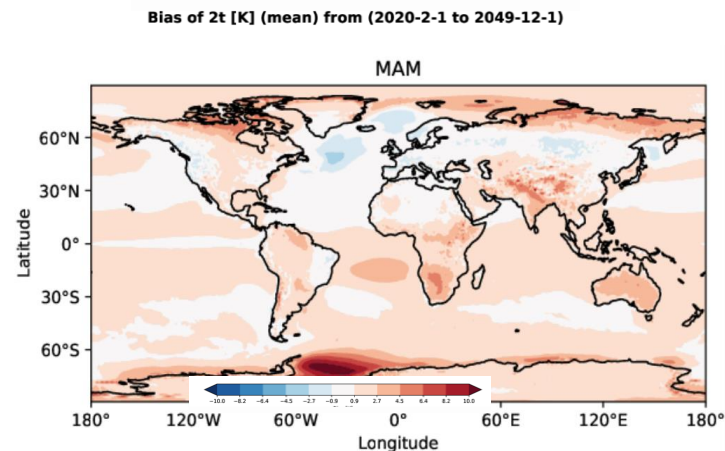
## Radiation budget



## Sea ice extent



## Seasonal Biases

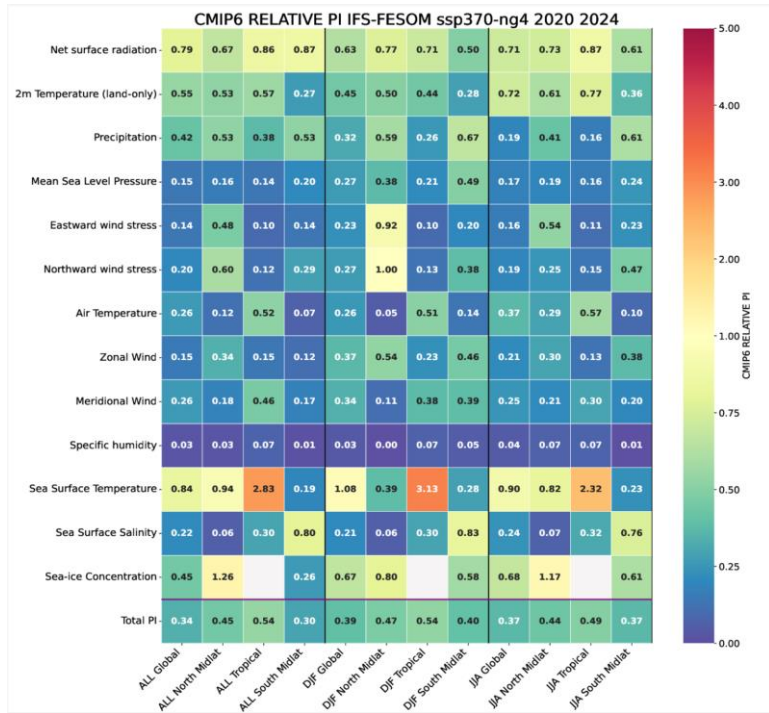




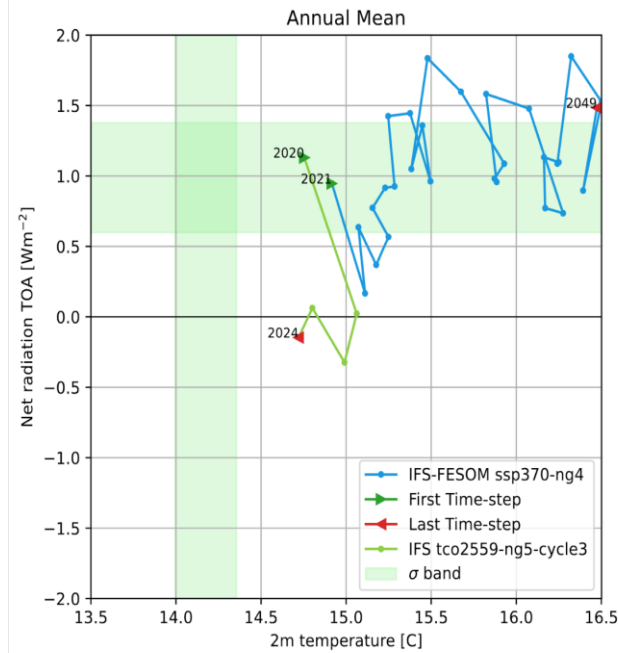


# Example plots

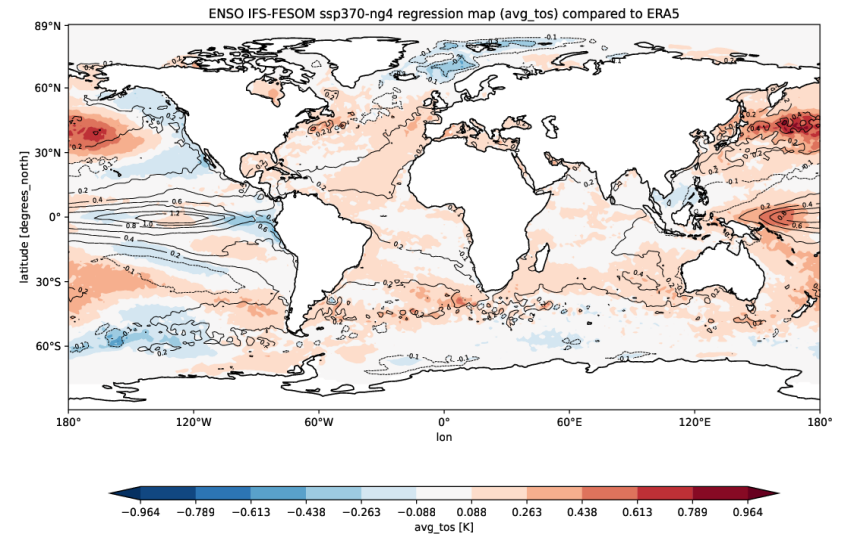
## Performance indices

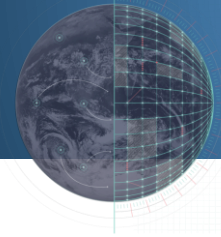


## Gregory plots



## Teleconnections maps





## Development plans

---

- Keep **AQUA** updated with respect with **changes in data portfolio** and eccodes versions, **HPC machines** and **Data Bridge** specifications for Climate DT purposes
- Transition toward **open source package** by refactoring the installation procedure (done) the diagnostics structure and the aqua-analysis tool (in progress); final aim is to deliver pypi and conda packages (planned), port the documentation to readthedocs (planned).
- **Expand and rationalize diagnostics**, both from the scientific (e.g. uncertainty quantification), the technical (efficiency and code quality) and the visualization point of view. Looking forward to integrate diagnostics from other developers too!



## THANK YOU FOR YOUR ATTENTION!

The code is on GitHub and will be **Open Source** soon!

<https://github.com/DestinE-Climate-DT/AQUA>

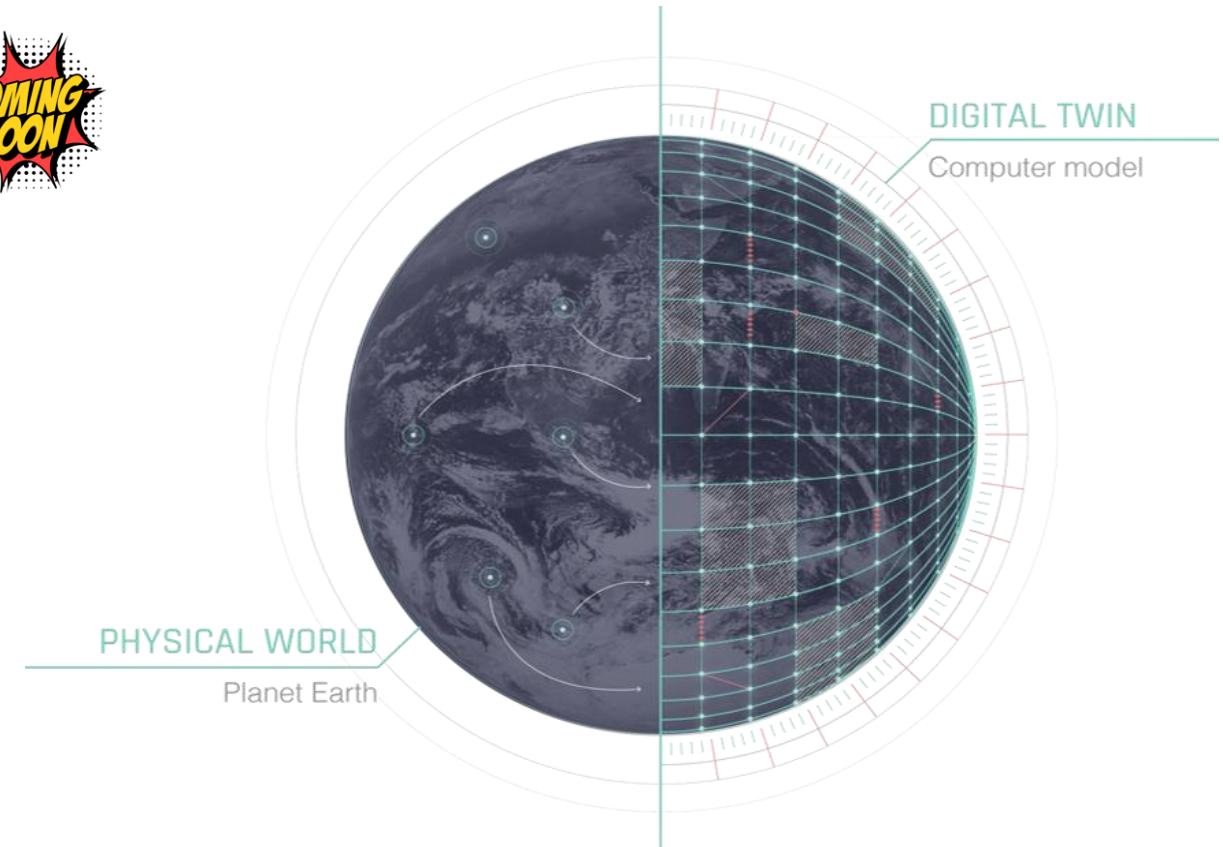
<https://github.com/DestinE-Climate-DT/Climate-DT-catalog>



Fill out this form with your email  
and GitHub username to be added  
to the repo!

Aqua core team Contacts:

- [m.nurisso@isac.cnr.it](mailto:m.nurisso@isac.cnr.it)
- [p.davini@isac.cnr.it](mailto:p.davini@isac.cnr.it)
- [silvia.caprioli@polito.it](mailto:silvia.caprioli@polito.it)
- [natalia.nazarova@polito.it](mailto:natalia.nazarova@polito.it)
- [jost.hardenberg@polito.it](mailto:jost.hardenberg@polito.it)
- [supriyo.ghosh@bsc.es](mailto:supriyo.ghosh@bsc.es)



## BACKUP SLIDES

# AQUA core concepts: the Reader

Once a catalog is added the user can retrieve any dataset with two lines of code!



This is a lazy dask array!

```
from aqua import Reader
reader = Reader(model='IFS-NEMO', exp='historical-1990',
source='hourly-native-atm2d', regrid='r100')
data = reader.retrieve()
```

> ~

[4] ✓ 0.2s

... xarray.Dataset

► Dimensions: (ncells: 6599680, time: 106608)

▼ Coordinates:

lon	(ncells)	float64	0.0 0.3142 ... -0.6283 -0.3142	📄	☰
lat	(ncells)	float64	1.57 1.57 1.57 ... -1.57 -1.57	📄	☰
time	(time)	datetime64[ns]	1990-01-01 ... 2002-02-28T23:00:00	📄	☰

▼ Data variables:

tc1w	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
tc1w	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
sp	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
tcwv	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
sd	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
chnk	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
msl	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
blh	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
tcc	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
10u	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
10v	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
2t	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
2d	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
lcc	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
mcc	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰
hcc	(time, ncells)	float64	dask.array<chunksize=(24, 6599680), meta=np.ndarr...	📄	☰



## Other available Reader methods

### Time aggregation:

```
reader = Reader(model="IFS", exp="tco2559-ng5", source="ICMGG_atm2d")
data = reader.retrieve()
daily = reader.timmean(data, freq='daily')
```

### Spatial averaging:

```
reader = Reader(model="IFS", exp="tco2559-ng5", source="ICMGG_atm2d")
data = reader.retrieve()
regional_mean = reader.fldmean(data, lon_limits=[-50, 50], lat_limits=[-10,20])
```

### Detrend:

```
reader = Reader(model="IFS", exp="tco2559-ng5", source="ICMGG_atm2d")
data = reader.retrieve()
daily = reader.detrend(data['2t'], dim='time')
```

### Vertical interpolation

```
reader = Reader(model="IFS", exp="tco2559-ng5", source="ICMU_atm3d",
regrid='r100')
data = reader.retrieve()
field = data['u'].isel(time=slice(0,5)).aqua.regrid()
interp = field.aqua.vertinterp(levels=[830, 835], units='hPa', method='linear')
```

### Accessors:

Please notice that all methods have accessors as **data.aqua.fldmean()** so that they can be called inline!

**LRA: AQUA** had to address the extreme large volume of full resolution data in the Climate-DT Phase1. The solution was to create a tool which exploiting the parallelization capabilities of AQUA - based on dask - could **postprocess high resolution data in few minutes moving from high spatial and temporal resolution to monthly 1deg x 1deg data**

This is done combining AQUA fixer, regridding and time averaging functionalities and it is the foundation of AQUA monitoring.

On the long term we plan to deprecate the LRA if low resolution data will be produced on the model side.

In next release, LRA will be integrated in the command line so that it can be called with ``aqua lra -c config.yaml``

```
1 #This include the option for the regridding
2 target:
3   resolution: r100
4   frequency: monthly
5   outdir: /pfs/lustrep3/projappl/project_465000454/data/AQUA/LRA
6   tmpdir: /pfs/lustrep3/projappl/project_465000454/padavini/tmp
7
8 loglevel: INFO
9
10 # Set to True if lra-r100-monthly-zarr should be created and checked
11 zarr: False
12 verify_zarr: False
13
14 slurm:
15   partition: small
16   username: padavini
17   account: project_465000454
18   time: 10:00:00
19   mem: 256GB
20
21 data:
22   ICON:
23     historical-1990:
24       hourly-hpz10-atm2d:
25         vars: ['mtpr', '2t', 'skt', 'msl', 'tcc', 'sd', 'tcw', 'tclw', 'msr', 'mslh', 'mssh',
26             'msnswrf', 'msnlwrf', 'msdwlwrf', 'msdswrf', 'mtnswrf', 'mtnlwrf', 'mtdswrf']
27         workers: 12
28       hourly-hpz10-atm3d:
29         vars: ['q', 't', 'u', 'v']
30         workers: 6
31       daily-hpz10-oce2d:
32         workers: 16
33         vars: ['avg_sithick', 'avg_siconc']
34       daily-hpz10-oce3d:
35         workers: 10
36         vars: ['avg_thetao', 'avg_so']
37
```

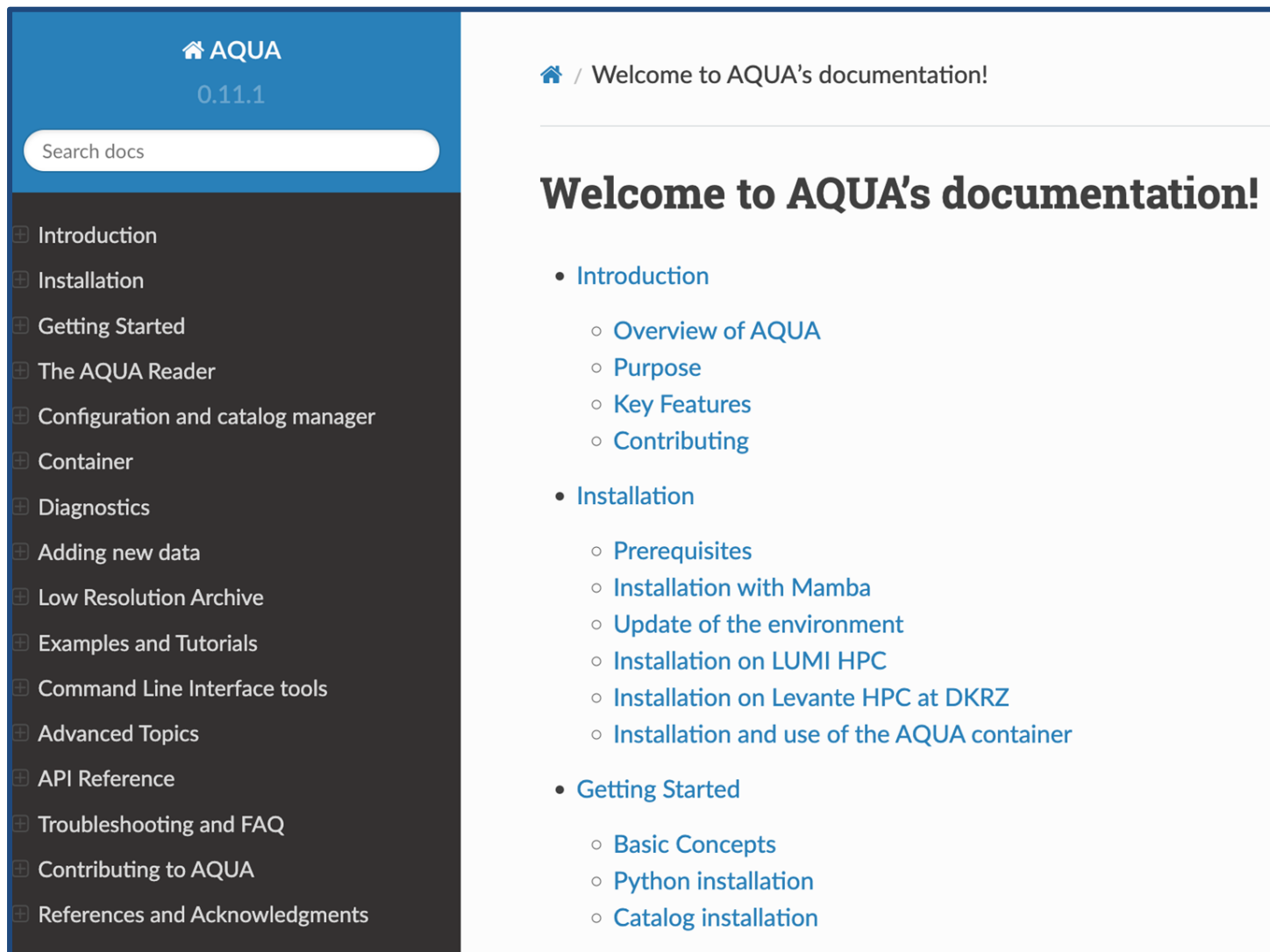
# AQUA code and documentation

So far these are **two private repositories** on the Climate-DT **GitHub** (**AQUA**: <https://github.com/DestinE-Climate-DT/AQUA> and **Climate-DT-catalog** <https://github.com/DestinE-Climate-DT/Climate-DT-catalog>)

**AQUA documentation** with **Sphinx** is available at <https://aqua-web-climatedt.2.rahtiapp.fi/documentation/index.html>

- General description
- How to install, how to run
- Illustrative examples
- API references
- FAQ for HPC deployment
- Container instructions

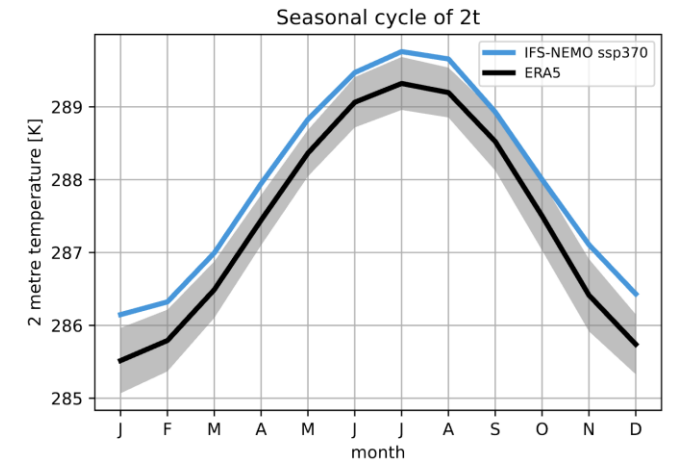
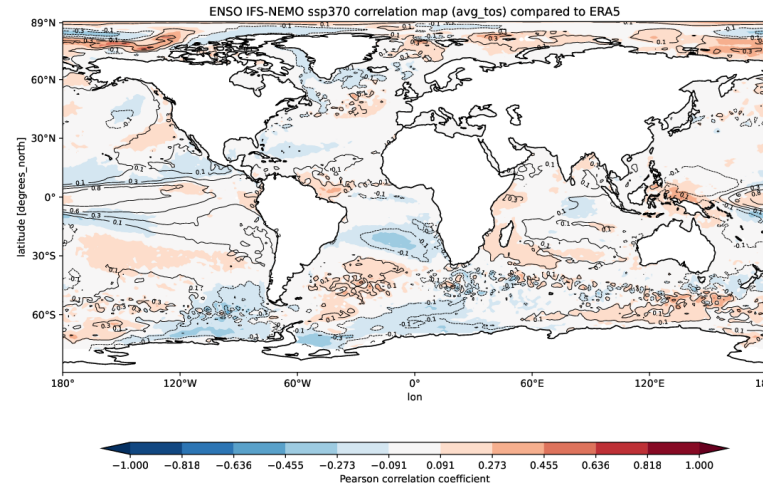
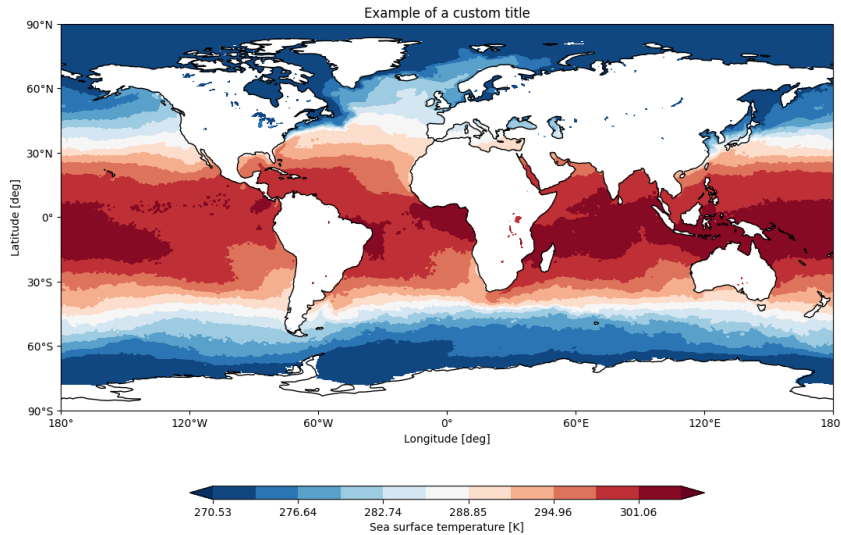
Frequent releases - current is v0.11.2



The screenshot displays the AQUA documentation website. The top navigation bar is blue with the AQUA logo and version number 0.11.1. Below the navigation bar is a search bar labeled 'Search docs'. A dark sidebar on the left contains a list of navigation items, each with a plus icon: Introduction, Installation, Getting Started, The AQUA Reader, Configuration and catalog manager, Container, Diagnostics, Adding new data, Low Resolution Archive, Examples and Tutorials, Command Line Interface tools, Advanced Topics, API Reference, Troubleshooting and FAQ, Contributing to AQUA, and References and Acknowledgments. The main content area is white and features a welcome message: 'Welcome to AQUA's documentation!'. Below the welcome message is a list of topics: Introduction (with sub-items: Overview of AQUA, Purpose, Key Features, Contributing), Installation (with sub-items: Prerequisites, Installation with Mamba, Update of the environment, Installation on LUMI HPC, Installation on Levante HPC at DKRZ, Installation and use of the AQUA container), and Getting Started (with sub-items: Basic Concepts, Python installation, Catalog installation).

# Graphical tools

Despite lagging a bit behind with respect to other features, some **basic functions for plotting** are already available within AQUA, as `plot_single_map()`, `plot_single_map_diff()` and `seasonal_cycle()`



These functions are used by the AQUA diagnostics but can be used on whatever xarray