Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Providentia v2.5.0
Training Session

18/02/2025

Paula Serrano | Alba Vilanova | Dene Bowdalo

Introduction

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Providentia is designed to generate **on-the-fly analysis and reports** of experiments, with respect to processed ground-based observational data

	Mean	StdDev	p5	Median	p95
observations	27.68	14.60	4.31	27.03	54.13
EMEP	33.44	9.52	17.35	33.25	49.48
MONARCH	30.27	11.96	11.55	29.21	51.06
SILAM	26.29	12.31	7.66	25.27	49.34

Functionalities

Structure

Download

Allows you to download the observations and experiments.

Interpolation

Matches experiments to observation networks.

Dashboard

To analyse in real time BSC experiment results, using observational data.

Report

Creates detailed pdf reports for in-depth analysis.

Library

Uses backend functions within Jupyter notebooks.

Structure

Download

Allows you to download the observations and experiments.

Interpolation

Matches experiments to observation networks.

Dashboard

To analyse in real time BSC experiment results, using observational data.

Report

Creates detailed pdf reports for in-depth analysis.

Library

Uses backend functions within Jupyter notebooks.

Structure

Download

Allows you to download the observations and experiments.

Interpolation

Matches experiments to observation networks.

Dashboard

To analyse in real time BSC experiment results, using observational data.

Report

Creates detailed pdf reports for in-depth analysis.

Library

Uses backend functions within Jupyter notebooks.

Dashboard

	Mean	StdDev	p5	Median	p95
observations	27.68	14.60	4.31	27.03	54.13
EMEP	33.44	9.52	17.35	33.25	49.48
MONARCH	30.27	11.96	11.55	29.21	51.06
SILAM	26.29	12.31	7.66	25.27	49.34

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Structure

Download

Allows you to download the observations and experiments.

Interpolation

Matches experiments to observation networks.

Dashboard

To analyse in real time BSC experiment results, using observational data.

Report

Creates detailed pdf reports for in-depth analysis.

Library

Uses backend functions within Jupyter notebooks.

Report

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Structure

Download

Allows you to download the observations and experiments.

Interpolation

Matches experiments to observation networks.

Dashboard

To analyse in real time BSC experiment results, using observational data.

Report

Creates detailed pdf reports for in-depth analysis.

Library

Uses backend functions within Jupyter notebooks.

Library

```python
import providentia as prv
```

```python
provi = prv.load("interactive_template.conf")
```

```python
# make a map plot (p95 statistic)
provi.make_plot('map-p95', labela='EMEP', map_extent=[-180, 180, -90, 90], plot_options=['domain'])
```

```
Warning: Width and/or height have not been passed. The default values will be set.
Warning: More than 1 network or species defined, can only plot for 1 pair. Taking EBAS|sconco3.
```

Set up

Set up

HPC | **Local**

Connect to HPC machines

Clone Providentia

Create a configuration file

Download data

Interpolate your experiments

Launch the tool

Set up

HPC | **Local**

Connect to HPC machines

Clone Providentia

Create a configuration file

Download data

Interpolate your experiments

Launch the tool

Connect to HPC machines

In your local machine, open and edit the SSH configuration file with:

```
vi .ssh/config
```

```
Host mn5
    HostName glogin4.bsc.es
    User bsc032XXX
    IdentityFile ~/.ssh/id_rsa
    ForwardX11 yes
    ForwardX11Trusted yes
    Compression yes
    Ciphers aes128-gcm@openssh.com
    ForwardX11Timeout 7d
```

```
Host nord3v2
    HostName nord4.bsc.es
    User bsc032XXX
    IdentityFile ~/.ssh/id_rsa
    ForwardX11Trusted yes
    ForwardX11 yes
    ForwardAgent yes
    Compression yes
```

Set up

HPC | **Local**

Connect to HPC machines

Clone Providentia

Create a configuration file

Download data

Interpolate your experiments

Launch the tool

Clone Providentia

If you are working on an HPC machine, we recommend cloning the directory in gpfs (e.g. /gpfs/scratch/bsc32/bsc032XXX) from MN5 (glogin4) to have access from MN5 and Nord3v2.

Enter the project's GitLab page and clone Providentia in your working directory:

```
git clone https://earth.bsc.es/gitlab/ac/Providentia
```

The latest releases are saved in the production branch, but we suggest users to stay in the master branch to be able to stay up-to-date.

Directories

Providentia relies on the next directories that store essential files for its operation.

```
├── configurations/   Contains configuration files required to run all Providentia modes.
├── logs/             Stores output files generated by Interpolation.
├── notebooks/        Holds template notebooks.
├── plots/            Saved plots if Providentia is used as a module.
├── reports/          Saved reports generated in Report.
├── saved_data/       Stores configuration, NetCDF and NumPy files if Providentia is used as a module.
└── settings/         Contains files that configure various aspects of Providentia.
```

settings directory

basic_stats.yaml	Defines the stats properties
color_palettes.yaml	Defines the possible color palettes
data_paths.yaml	Defines dataset paths categorized by machine.
experiment_bias_stats.yaml	Defines the experiment stats properties
init_prov.yaml	Stores initialization settings, including non-ghost available networks and resolutions.
interp_experiments.yaml	Specifies locations for non-interpolated experiments.
remove_extreme_stations.yaml	Defines criteria for filtering stations if you want to automatically remove them.
exceedances.yaml	Stores threshold values for exceedance statistics
fairmode.yaml	Stores configurations for the Fairmode plots.
plot_characteristics.yaml	Configures plot appearance settings.
report_plots.yaml	Defines plot types per report type.

Set up

HPC
Local
Connect to HPC machines
Clone Providentia
Create a configuration file
Download data
Interpolate your experiments
Launch the tool

Create a configuration file

Configuration files are used to know **what data to interpolate** (interpolation mode), **download** (download mode), **read and plot** (dashboard, report and library modes).

By default, they are read from the folder configurations and are composed of sections (mandatory) and subsections (optional). When reading and filtering the data, subsection take all values defined in their parent sections.

The arguments that you can set are defined in: https://earth.bsc.es/gitlab/ac/Providentia/-/wikis/Configuration-files.

Example of a configuration file

configurations/configuration_name.conf

Section ([name])

Subsection ([[name]])

```
[PRV_sconco3_a365]
network = EBAS
species = sconco3
resolution = hourly
start_date = 20180101
end_date = 20180601
experiments = cams61_chimere_ph2-eu-000 (CHIMERE)
temporal_colocation = True
spatial_colocation = True
report_type = standard
report_summary = True
report_stations = False
report_filename = PROVIDENTIA_Report
report_title = Report

  [[Barcelona]]
  latitude = 39.8, 41.8
  longitude = 1.5, 2.5
  area_classification = keep: rural || remove: urban-suburban
```

Sections and subsections

When launching the tool multiple sections, the modes behave differently:

- **Interpolation**: The data for the first section will be interpolated unless you specify it otherwise. Subsections are not taken into account.
- **Dashboard**: You will be prompted to choose one specific section-subsection pair.
- **Report**: A report will be generated per section and inside each report there will be plots per each subsection.
- **Library**: The first section will be chosen unless you specify it otherwise.
- **Download**: The data for all sections will be downloaded. If only one section needs to be downloaded, it can be specified with the --section key.

Defining experiments

Experiments can be configured in various ways:

1) Use experiment, ensemble member and domain independently

```
experiments = cams61_monarch_ph3
domain = eu
ensemble_options = allmembers
```

```
experiments = cams61_monarch_ph3
ensemble_options = allmembers
```

```
experiments = cams61_monarch_ph3
domain = eu
```

```
experiments = cams61_monarch_ph3
```

2) Combine experiment and domain

```
experiments = cams61_monarch_ph3-eu
ensemble_options = allmembers
```

```
experiments = cams61_monarch_ph3-eu
```

Defining experiments

3) Combine experiment and ensemble

```
experiments = cams61_monarch_ph3-allmembers
domain = eu
```

```
experiments = cams61_monarch_ph3-allmembers
```

4) Use experiment field only

```
experiments = cams61_monarch_ph3-eu-allmembers
```

Aliases can be used to avoid long experiment names:

```
experiments = cams61_monarch_ph3-eu-allmembers (MONARCH)
```

Note: Aliases only work in this specific way, and also when there is only a single experiment, a single ensemble member, and a single domain.

Set up

HPC | **Local**

Connect to HPC machines

Clone Providentia

Create a configuration file

Download data

Interpolate your experiments

Launch the tool

Download data

If you clone Providentia in your local machine, initially you won't have any data to work with. To easily download the data **from esarchive and Zenodo (GHOST) onto your machine**, you can use the download mode. In this way you can have observations, experiments to interpolate, and interpolated experiments data.

The download mode can also be used when needing to run the interpolation from MN5, as the machine does not have access to the experiments that are in esarchive. These can be **downloaded into gpfs** using this mode from the transfer machine or Nord3v2. This should be done only occasionally, as experiment data is heavy and we don't want to waste resources.

Download data

To use the download mode you should run:

```
./bin/providentia --config=your_configuration_name.conf --download
```

This will download the data in the paths defined in settings/data_paths.yaml, by default in /home/{username}/data/providentia/ in local machines and /gpfs/projects/bsc32/AC_cache/exp/ in HPC machines.

We will explain this with more detail shortly.

Set up

HPC
Local
Connect to HPC machines
Clone Providentia
Create a configuration file
Download data
Interpolate your experiments
Launch the tool

Greasy and multiprocessing

Greasy is a package used by Providentia to manage very large submissions of interpolation jobs in parallel and historically has given us issues. This can now be avoided by using the multiprocessing package.

Specifically this was implemented to be able to run the interpolation locally during those (quite recurrent) times when the HPC machines are down or the space is full, and for users outside BSC who do not have access to our systems.

This option is turned on by default in local runs, but if you want to use it in HPC you can just add **multiprocessing=True** to your configuration file.

Interpolate your experiments

Make sure your experiment is set in settings/interp_experiments.yaml before interpolating.

configurations/configuration_name.conf

```
[A005]
ghost_version = default
interp_n_neighbours = default
start_date = 202403
end_date = 202411
species = sconcco
experiments = a005_cirrus
domain = regional
ensemble_options = default
network = meteofrance/eionet-cams2_40
resolution = hourly
```

settings/interp_experiments.yaml

```
"monarch": {
    "experiments": [ "a005_cirrus"],
    "paths": [
        "/esarchive/exp/monarch/"
    ]
}
```

Interpolate your experiments

If --interp, --interpolate or --interpolation is added as a launch option together with the mandatory configuration file in the command line, the interpolation will start:

```
./bin/providentia --config='configuration_name.conf' --interp
```

This will generate logs in different folders inside the logs/interpolation folder:

- **Submission logs**: To give information on the slurm and Greasy submissions to the HPC machines.
- **Management logs**: To inform about the interpolation overall. Most errors will appear here.
- **Interpolation logs**: To track the details and duration of individual interpolations. Specific errors appear here.

Set up

HPC | **Local**

Connect to HPC machines

Clone Providentia

Create a configuration file

Download data

Interpolate your experiments

Launch the tool

Launch the tool

- Opening the dashboard:

```
./bin/providentia --config='configuration_name.conf'
```

- Creating reports:

```
./bin/providentia --config='configuration_name.conf' --offline
```

- Opening a Jupyter notebook:

```
./bin/providentia --config='configuration_name.conf' --notebook
```

Launch options

	Main purpose	Argument	Configuration file	HPC	Local
Dashboard	On-the fly analysis		✓	✓	✓
Report	Detailed analysis in PDFs format	--offline	✓	✓	✓
Notebook	Open a Jupyter notebook	--notebook	✓	✓	✓
Interpolation	Interpolate experiments to observation networks	--interp, --interpolate, --interpolation	✓	✓	✗
Download	Download data to local machine	--dl, --download	✓	✓	✓
Debug	Avoid waiting in queue to relaunch Providentia	--debug	✗	✓	✗
Clean	Clean logs and new files	--clean	✗	✓	✓
Generate file tree	Check for existing directories to make launch faster	--gft, --generate_file_tree	✗	✓	✓

Download

Download

If --dl or --download is added as a launch option together with the mandatory configuration file in the command line, the download will start:

```
./bin/providentia --dl --config='/path/to/file/example.conf'
```

```
./bin/providentia --download --config='/path/to/file/example.conf'
```

The download functionality downloads all the content specified in your configuration file across all sections.

However, if you want to download content from only one specific section, you can pass the section argument when launching Providentia.

Download options

The table below summarises the available options to download data using Providentia:

Download type	Available from local	Available from HPC machines	Need to have a BSC account to an HPC machine
Network from HPC machines	✓	✗	✓
Network from Zenodo	✓	✗	✗
Interpolated experiment from HPC machines	✓	✗	✓
Non-interpolated experiment from HPC machines	✓	✓	✓

Download to local machine

Once a download is started, the data specified in your configuration file will be downloaded to the directories defined under the **local** key in settings/data_paths.

Downloads from HPC to local machines, including GHOST, Non-GHOST networks and interpolated and non-interpolated experiment files are done via the **Transfer5** machine (or **MN5** if Transfer5 is unavailable).

For those without access to the BSC transfer machines, GHOST files will be downloaded from the Zenodo repository.

⚠️

BSC users are strongly encouraged to use the Transfer5/MN5 option instead of Zenodo as it offers faster download speeds and access to more networks.

Transfer from esarchive to gpfs

A new functionality has been added that allows you to move non-interpolated experiments from esarchive to gpfs, making them available for interpolation in MN5.

The experiment is moved to the directory specified in settings/data_paths.yaml under the **exp_to_interp_root** path for the machine.

You must specify the location of the original experiment in esarchive in settings/interp_experiments.yaml.

⚠️

This mode operates from Transfer5 and Nord3v2, but it does not work on MN5!!!

How to customise the download

Local download from HPC Machines

After running the tool on the command line:

- Answer **y** to the prompt: *Do you want to download from BSC remote machine?*
- Change the value of BSC_DL_CHOICE to **y** in the .env file.

Local download from Zenodo

After running the tool on the command line:

- Answer **n** to the prompt: *Do you want to download from BSC remote machine?*
- Change the value of BSC_DL_CHOICE to **n** in the .env file.

Local download of interpolated experiments

To download interpolated experiments locally:

- Ensure the *interpolated* field in the configuration file is set to **True** (or leave it unset, as this is the default value).

Local and gpfs download of non-Interpolated experiments

For local and GPFS downloads of non-interpolated experiments:

- Ensure the *interpolated* field in the configuration file is set to **False**.

.env file

The .env file is a recent addition designed to **store** specific **user settings.**

It will be **automatically** generated the first time Download is used with user input, provided the user chooses to save their preferences.

There are four possible fields that can be added, here an example of an .env file with all of them:

```
BSC_DL_CHOICE=y
OVERWRITE=y
PRV_USER=bsc000000
PRV_PWD=example_pwd
```

Once created, the .env file can be accessed and modified manually from the **Providentia root directory**.

Additionally, if a field is removed, it will be generated with the user input on the next execution if the user opts to restore it.

.env file

BSC_DL_CHOICE

This field indicates the method of downloading GHOST networks.

It is recommended to download from the remote machine if the user has access to BSC machines.

Possible Values:

- **y** – Enables BSC remote machine SFTP download of GHOST.
- **n** – Enables GHOST download from the Zenodo webpage GHOST publication.

OVERWRITE

This field determines whether to overwrite an existing file if a download is attempted for a file that is already present.

Possible Values:

- **y** – Allows overwriting of the file, so it will be downloaded again.
- **n** – Prevents downloading if the file is already present.

.env file

PRV_USER

This field stores the username used to connect via SSH to remote machines.

Possible Values:

- **Any valid username** – Ensure you have an account and can successfully connect using **ssh {username}@transfer1.bsc.es** or **ssh {username}@glogin4.bsc.es** in download mode.

PRV_PWD

This field stores the password for logging into the remote machine.

Note that the password is not required if you have configured the passwordless connection to different servers/machines. Tutorial here.

Possible Values:

- **Any valid password**

Dashboard

Dashboard

	Mean	StdDev	p5	Median	p95
observations	27.68	14.60	4.31	27.03	54.13
EMEP	33.44	9.52	17.35	33.25	49.48
MONARCH	30.27	11.96	11.55	29.21	51.06
SILAM	26.29	12.31	7.66	25.27	49.34

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Main menu

Data Selection
EEA_AQ_eR
gas
sconco3
QA
EXPS
hourly
20180101
20190101
FLAGS
MULTI
READ
Filters
Bounds
0.0
400.0
% REP
PERIOD
META
RESET
FILTER
Statistics
Mode
Temporal|S
Aggregation
Median
Colocation
Temporal
Resampling
None
Site Selection
All
Intersect
Extent

Data selection menu

Network	Matrix	Species	Quality assurance (GHOST)	Experiments
EEA_AQ_eR	gas	sconco3	QA	EXPS

Temporal resolution	Start date	End date	Data flags (Provider)	Multispecies filtering
hourly	20180101	20190101	FLAGS	MULTI

Filters menu

	Data lower bound	Data upper bound
Bounds	0.0	400.0

Representativity filters	Time period filters	Metadata filters
% REP	PERIOD	META

Statistics menu

Statistic mode

Statistic aggregation

Colocation menu

☑ Temporal

Colocate experiments vs. observations, removing temporal gaps

	Mean	StdDev	p5	Median	p95
observations	31.66	11.26	13.05	31.71	52.00
EMEP	33.95	8.52	18.59	34.17	47.84
MONARCH	27.87	10.57	12.16	27.54	47.46
SILAM	27.43	10.58	11.04	27.24	47.62

	Mean	StdDev	p5	Median	p95
observations	32.91	11.84	13.30	32.80	54.46
EMEP	33.96	8.47	18.77	34.14	47.72
MONARCH	27.88	10.57	12.19	27.44	47.20
SILAM	27.41	10.55	11.02	27.29	47.37

Resampling menu

Temporal resolution to resample your data to (always lower than the selected resolution)

	Mean	StdDev	p5	Median	p95
observations	32.91	11.84	13.30	32.80	54.46
EMEP	33.96	8.47	18.77	34.14	47.72
MONARCH	27.88	10.57	12.19	27.44	47.20
SILAM	27.41	10.55	11.02	27.29	47.37

Providentia

Data Selection: EBAS | gas | sconco3 | QA | EXPS | hourly | 20150101 | 20200101 | FLAGS | MULTI | READ

Filters: Bounds | 0.0 | 400.0 | % REP | PERIOD | META | RESET | FILTER

Statistics: Mode: Temporal|S; Aggregation: Median

Colocation: ✓ Temporal

Resampling: daily

Site Selection: ✓ All | Intersect | Extent

(x, y) = (0.256, 0.92)

statsummar

	Mean	StdDev	p5	Median	p95
observations	32.91	9.41	17.20	33.11	48.63
EMEP	33.88	7.28	19.98	34.89	44.72
MONARCH	27.86	8.72	13.73	28.71	42.28
SILAM	27.42	8.15	14.17	27.71	40.97

Stations selection menu

Select all stations

Select intersecting stations within all model domains

Select stations on current map view

Icons menu

Upload configuration files

Back

Zoom to rectangle

Lasso

Download data and configuration files

Back to initial screen

Forward

Pan

Save canvas

Navigation tips

Individual stations can be selected by left clicking on them on the map.

Multiple stations can be selected by activating the lasso, and drawing it round all desired stations.

To unselect stations simply click on an empty space on the map.

When multiple stations are selected, to add or remove any individual station right click on the given station.

To quickly zoom on the map, the mouse or trackpad scroll wheel can be used.

Example of lasso selection

Interactive features

Legend picking

Clicking on the legend labels will remove or add data to each of the plots

Bold = Visible
Roman = Invisible

Customise the plot style

The style of the plots can be edited by clicking on the **burger menus** and changing the settings.

Choosing the statistics

The statistics in the statsummary can be updated from the burger menu.

Choosing the statistics

The statistics in other plots can also be updated from the burger menus.

Information on hover

We can see the stations details and data by hovering on the map.

Information on hover

We can also check the values of each dataset by hovering on the other plot types.

Saving plot figures

Saving plot figures is now possible by clicking on the image icons next to the burger menus

Smoothing

It is possible to add a smooth line in the timeseries plot by editing the smooth window, the data points can be then hidden by reducing their size to 0.

Statistics

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Statistics

- **Statistical modes:**
  - Temporal|Spatial (default)
  - Spatial|Temporal
  - Flattened

- **Calculation of statistics per periodic cycle, 2 modes:**
  - Independent
  - Cycle

Statistical modes

The name of each statistical mode relates to how the dimensions of the selected data are reduced to calculate the statistical metrics, e.g. mean, median etc, going from 2D to 0D.

When selecting data across multiple stations, it has 2 dimensions: **Spatial** and **Temporal**.

For **Temporal|Spatial**, aggregation is performed first temporally per station (e.g. median), going to 1D, before then calculating the desired statistic across stations (e.g. p50), going to 0D.

For some plot types the full dimensional reduction is not possible, e.g map. Therefore, they are locked in certain reduction configurations.

Statistical modes

Statistical modes

On the dashboard, in the statistics menu at the top, the mode and aggregation can be selected via the dropdown menus.

In the configuration files these can be set like so:

statistic_mode = Temporal|Spatial

statistic_aggregation = Median

Note: For the Flattened mode, there is no aggregation statistic.

Periodic statistics

The periodic plot gives statistical information for grouped data in individual timesteps. Thus it can be seen how each individual timesteps compare for observations vs experiment/s.

However, statistics can also be calculated which assess the entirety of each periodic cycle, i.e. diurnal, weekly, monthly. These statistics are available via the **statsummary**, **table** and **heatmap** plot types.

Periodic statistics

There exist 2 modes for calculating these periodic statistics:

- Independent (default)
- Cycle

Independent works by calculating the desired statistic per timestep (i.e. as seen in periodic plot), before aggregating across the timesteps.

Cycle works by aggregating the grouped data per timestep (e.g. mean), before then calculating the desired statistic across the timesteps.

Periodic statistics

On the dashboard, in the plot options of the **statsummary** plot, the periodic statistic mode and aggregation can be selected via the dropdown menus. Additionally, periodic statistics can be added to the statsummary plot also via the dropdown menus.

In the configuration file these can be set like so:

periodic_statistic_mode = Independent

periodic_statistic_aggregation = Median

Report

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Choosing the plots

Users need to define the **report_type** in the configuration file, which will be linked to the plot types that appear in settings/report_plots.yaml.

configurations/configuration_name.conf

```
[CAMS2_40]
network = ineris/eionet-cams2_40-ira
species = sconco3,sconcno2,sconcco,sconcso2,pm10,pm2p5
resolution = hourly
start_date = 20230101
end_date = 20230215
experiments = a59g-regional-000, a59j-regional-006 (Forecast, Analysis)
temporal_colocation = True
spatial_colocation = False
report_type = operational
report_summary = True
report_stations = False
report_title = CAMS2_40 Forecast and Analysis Report
report_filename = operational_report
statistic_mode = Temporal|Spatial
statistic_aggregation = Median
periodic_statistic_mode = Independent
periodic_statistic_aggregation = Mean
```

settings/report_plots.yaml

```
"operational": ["statsummary_multispecies",
"statsummary_multispecies_bias",  "map-Mean", "map-MB",
"map-RMSE", "map-r", "timeseries_annotate",
"timeseries_bias_annotate", "distribution",
"scatter_annotate", "periodic-Mean", "periodic-MB",
"periodic-RMSE", "periodic-r"]
```

Plot types and options

Available Unavailable

	-[stat]	_bias	_annotate	_regression	_multispecies	_logx	_logy	_smooth	_hidedata	_domain	_individual	_obs	_threshold
map													
timeseries													
periodic													
periodic-violin													
distribution													
scatter													
heatmap													
table													
boxplot													
statsummary													
metadata													
taylor													
fairmode-target													
fairmode-statsummary													

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Plot types

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

map-[stat]

Map Mean (Summary)
EEA_AQ_eReporting|sconco3

observations
TRAINING|All (2241 stations)

observations
TRAINING|Spain (378 stations)

timeseries

80
70
60
50
40
30
20
10
0
nmol mol-1
2018-01-01
2018-03-01
2018-05-01
2018-07-01
2018-09-01
2018-11-01
2018-12-31
Time

periodic

H
M
DoW
nmol mol-1
0
10
20
30
40
50
60
0
3
6
9
12
15
18
21
J F M A M J J A S O N D
M T W T F S S

periodic-violin

distribution

scatter

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

heatmap-[stat]

table-[stat]

	observations	EMEP	MONARCH	SILAM
Europe	19.16	19.16	19.16	19.16
Spain	19.56	19.56	19.56	19.56
France	19.29	19.29	19.29	19.29
Germany	17.41	17.41	17.41	17.41

boxplot

statsummary

	p5	Mean	StdDev	p50	p95
Observations	10.52	30.06	11.52	29.69	48.61
EMEP	24.55	35.18	6.86	34.69	46.72
MONARCH	22.39	34.14	8.51	32.87	49.91
SILAM	16.53	30.39	9.36	29.26	46.99

metadata

378 Stations
Median Measurement Altitude: 223.55 m
Median To Coast: -33.00 km
Median Population Density: 2491.1 xx/km–2
Country: Spain:99.1%, nan:0.9%
Area Classification: 7 uniques
Station Classification: 4 uniques
MODIS Land Use: 13 uniques
GHSL Classification: 8 uniques
Method: nan:0.9%, ultraviolet photometry:99.1%
Instrument: 20 uniques

taylor-[stat]

0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0.95
0.99
r
25.00
20.00
15.00
5.00
0.00
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
StdDev

fairmode-target

α=0.79
β=2
RV=60.12
$U_{95,r}^{RV}$=0.18

365 stations with coverage above 75%

EMEP
MQI_{90} = 0.57
0 stations with MQI > 1

MONARCH
MQI_{90} = 0.62
1 station with MQI > 1

SILAM
MQI_{90} = 0.48
0 stations with MQI > 1

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

fairmode-statsummary

Plot options

-[stat]

It must be added to create maps, periodic plots, heatmaps, tables and Taylor diagrams

Basic statistics

Statistic	Meaning
Mean	Mean
StdDev	Standard deviation
Var	Variance
Min	Minimum
Max	Maximum
Data%	Data availability
Exceedances	Number of exceedances
p1, p5, p10, p25, p50, p75, p90, p95, p99	Percentiles
NStations	Number of stations
MDA8	Daily maximum 8 hour average

Experiment bias statistics

Statistic	Meaning
MB	Mean bias
NMB	Normalised mean bias
ME	Mean error
NME	Normalised mean error
MNB	Mean normalised bias
MNE	Mean normalised error
MFB	Mean fractional bias
MFE	Mean fractional error
RMSE	Root mean square error
NRMSE	Normalised root mean square error
COE	Coefficient of efficiency
FAC2	Fraction of experiment values within a factor of two of observed values
IOA	Index of agreement
R	Pearson correlation coefficient
R^2	Coefficient of determination
UPA	Unpaired peak accuracy

-[stat]

map-Mean

timeseries-r2-daily

-[stat]

periodic-RMSE

heatmap-Mean

-[stat]

table-Data%

	observations	EMEP	MONARCH	SILAM
Europe	19.16	19.16	19.16	19.16
Spain	19.56	19.56	19.56	19.56
France	19.29	19.29	19.29	19.29
Germany	17.41	17.41	17.41	17.41

taylor-r

_bias

timeseries_bias

distribution_bias

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

_annotate

The statistics to annotate are defined per plot type in settings/plot_characteristics.yaml.

```
"scatter":
  {
    "grid": { "axis": "both", "color": "lightgrey", "alpha": 0.8 },
    "annotate_stats": ["r2", "RMSE"],
    "annotate_offset": { "loc": "upper left" },
    "annotate_bbox":
     {
       "facecolor": "white",
       "edgecolor": "gainsboro",
       "alpha": 1,
       "zorder": 100,
     },
    …
```

scatter_annotate

_individual

Sometimes you might want to show the data per experiment to see it with more detail.

scatter

scatter_individual

_obs

There are occasions when you might only want to isolate the observations data.

timeseries

timeseries_obs

_multispecies

The report mode is the only one able to plot multiple species per plot.

boxplot_multispecies

statsummary_multispecies

		p5	Mean	StdDev	p50	p95
sconco3	Observations	50.0	71.03	14.59	69.42	96.0
	Forecast	40.44	60.34	13.72	60.12	83.31
	Analysis	47.14	67.25	13.7	65.81	89.43
sconcno2	Observations	5.64	8.5	2.21	8.0	13.08
	Forecast	1.66	4.59	2.34	4.31	9.09
	Analysis	2.92	5.65	2.09	5.34	9.69
sconcco	Observations	0.2	0.24	0.02	0.24	0.28
	Forecast	0.13	0.15	0.01	0.14	0.17
	Analysis	0.16	0.18	0.01	0.18	0.2
sconcso2	Observations	2.0	2.1	0.16	2.0	2.49
	Forecast	0.45	0.81	0.24	0.82	1.2
	Analysis	0.95	1.45	0.29	1.52	1.82
pm10	Observations	12.01	15.91	2.43	16.19	19.46
	Forecast	5.99	8.44	1.6	8.17	10.96
	Analysis	10.43	13.91	2.27	14.06	17.3
pm2p5	Observations	5.7	8.83	1.84	9.02	11.55
	Forecast	4.18	6.17	1.33	6.03	8.52
	Analysis	5.12	8.15	1.72	8.29	10.84

_logx and _logy

scatter_logx

scatter_logy

_regression

scatter

scatter_regression

_smooth

The smooth line in the timeseries plot can be adjusted by setting the window and minimum points percentage in settings/plot_characteristics.yaml.

timeseries-Mean-daily_smooth (window=10)

timeseries-Mean-daily_smooth (window=30)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

_hidedata

You can hide the points to only show the smooth or regression lines by using this option.

scatter_regression_hidedata

timeseries_smooth_hidedata

_domain

map-RMSE

map-RMSE_domain

_threshold

timeseries_threshold (sconco3: 30.07)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

_threshold

The **Exceedances** statistic gives the number of instances above a threshold. The threshold values can now be set in the file settings/exceedances.yaml per component, or network-component pair, as so:

settings/exceedances.yaml

```
sconco3: 30.07
sconcno2: 5.23
sconcso2: 15.02
sconcco: 3435.18
pm2p5: 5.0
pm10: 15.0
```

In the case a threshold is set for a specific component, and per network-component, then the threshold for network-component is taken preferentially.

Library

Library

The library functionality allows Providentia to be used as a Python module, with access to its backend functions, allowing different parts of Providentia to be used in scripts, as required.

In order to allow Python to see Providentia as a module, the following lines must be added first to your Python script:

```
import sys
sys.path.append("path/to/your/Providentia/directory")
```

This will append the directory of your Providentia repository to your system's paths. In the future we plan to install Providentia as a system wide module, and add it as a module to PyPi, thus avoiding the need for appending to system paths.

Opening Jupyter notebook session

One typical use case is the interactive evaluation in a Jupyter notebook. Providentia has an in-built manner of opening a Jupyter notebook.

If we add --notebook as a launch option in the command line, a Jupyter session will start:

```
./bin/providentia --notebook
```

If you are on a HPC machine this this will launch an associated slurm job, and a file will be created (notebook.out) containing the command to create an SSH tunnel on your local machine, which should be pasted into the terminal, and the path to open Jupyter in your browser. Running locally, Jupyter should open automatically.

The Jupyter session can make use of the parallelisation available in Providentia, as long as the --cpus-per-task=N argument is passed, where N equals the number of cores.

Loading a configuration file

Providentia can be imported as a module as follows:

```
import providentia as prv
```

Once imported the first step to start using the module is to load a configuration file, using the *load* method. This reads the conf file, reads the appropriate data and filters it:

```
provi = prv.load("interactive_template.conf")
```

In this process a class instance instance is created (*provi*), with access all of the Providentia library class variables and methods. However, only one *section-subsection* pair from a conf file can be loaded at one time. Where there is more than one available, the specific pair wished to be read can be set as follows, otherwise the first available *section-subsection* is taken preferentially:

```
provi = prv.load("interactive_template.conf",
section="SECTIONNAME", subsection="SECTIONNAME·SUBSECTIONNAME")
```

Overwriting configuration parameters

If wanting to overwrite any overwrite any variables defined in the .conf file, each variable can be simply passed as an argument when loading the data, e.g.:

```
provi = prv.load("interactive_template.conf", network="EANET")
```

These variables can be any variable that can be set in a .conf file.

See here for more information:

https://earth.bsc.es/gitlab/ac/Providentia/-/wikis/Configuration-files.

Viewing active configuration

It is possible to view the active configuration variables in 2 different ways. The first is using the method *config*:

```
provi.config()
```

Note, this method can also be used to view the variables set in another given conf file, by passing the filename:

```
provi.config("important.conf")
```

The other way of viewing the active configuration variables is by printing the active class instance:

```
print(provi)
```

Access to other functionalities

Rather than loading data from a configuration file, some of the other functionalities of Providentia can also be run directly from within scripts. This means specific parts of Providentia can be integrated in automatic workflows with little effort. These are namely the *download*, *interpolate* and *report* functionalities. These are used in the same way as the *load* method, passing a configuration filename:

```
prv.download("interactive_template.conf")
```

```
prv.interpolate("interactive_template.conf")
```

```
prv.report("interactive_template.conf")
```

Some of these functionalities can utilise multiple sections / subsections at once (i.e. download, report). The section / subsection can be passed as arguments if wanting to to be more specific:

```
provi = prv.download("interactive_template.conf",
section="SECTIONNAME", subsection="SECTIONNAME·SUBSECTIONNAME")
```

Accessing loaded data

Once a configuration file has been loaded, the loaded metadata / data can be easily accessed by 2 methods. The first is to retrieve a specific variable, by providing a variable name to the *variable* method. The variable data is returned in a numpy array. If more than one network or component has been loaded, then the specific network and component must also be passed together with the variable name, with the syntax *network|component_variablename.* If not, just the variable name is needed:

```
var_data = provi.variable("network|component_variablename")
```

The second is to retrieve all metadata / data variables by using the *data* method. The returned data can be returned in 3 different formats: nc (netCDF), np (numpy) or xr (xarray) by passing the format argument:

```
data = provi.data(format="xr")
```

Applying filters

To apply a filter not set in the configuration file, the *filter* method is used. The fields to filter by can be any data /metadata variable standardised in GHOST, and can be textual or numeric.

- If the field is numeric, use *lower* and/or *upper*, e.g.:

```
provi.filter(field, lower=28, upper=31)
```

- If the field is textual, use *keep* and/or *remove*, e.g.:

```
provi.filter("country", keep="Spain")
```

```
provi.filter("country", remove=["Spain", "France"])
```

- If the field is a representativity field, use *limit*:

```
provi.filter(rep_field, limit=20)
```

Reset filters

If at any time any active data / metadata filters are wished to be reset, this can be done by using the *reset* method.

To reset all filters to the Providentia defaults, including those set from the original read configuration file:

```
provi.reset()
```

To keep the filters defined in the configuration file, but reset the ones added later, pass the *initialise* argument as True:

```
provi.reset(initialise=True)
```

Calculating statistics

All defined basic and experiment bias statistics defined in Providentia can be calculated from the loaded data, using the *statistic* method:

```
stat_calc = provi.statistic(stat, labela="OBS")
```

where *stat* is the statistic wished to be calculated, and *labela* is “observations”/experiment name or the alias set in the conf file. If wanting to calculate a bias statistic, you need to set *labelb:*

```
stat_calc = provi.statistic(stat, labela="OBS", labelb="EMEP")
```

For bias statistics for which a subtraction is involved, it is always done as *datab* - *dataa*.

If wanting to calculate statistics per station, *per_station* can be passed as True:

```
stat_calc = provi.statistic(stat, labela="OBS", labelb="EMEP", per_station=True)
```

Plotting

You can use the plotting functions of Providentia with the *plot* method. Any of the available plot types in Providentia can be made by passing the desired type, with all types listed here: https://earth.bsc.es/gitlab/ac/Providentia/-/wikis/Plot-types-and-options

```
provi.plot(plot_type)
```

By default, plots will attempt to be made interactively, immediately plotting them and sending them to the screen, however plots can also be saved to file, by passing *save* and a filename with the wanted extension (e.g. png, jpg etc.):

```
provi.plot(plot_type, save="myplot.png")
```

The plotting object can also be returned for additional tinkering, by setting *return_plot* to True:

```
plot_obj = provi.plot(plot_type, return_plot=True)
```

Plot customisation

Each plot can be customised in a number of ways. This is done by passing a series of arguments to the plot method. Some of most important ones are as follows:

To remove the colourbar from the plot, set *cb* to False:

```
provi.plot(plot_type, cb=False)
```

To remove the legend from the plot, set *legend* to False:

```
provi.plot(plot_type, legend=False)
```

To limit which data is plotted, pass the *data_labels* argument and wanted data, which is the "observations"/experiment name or the alias set in the conf file:

```
provi.plot(plot_type, data_labels=["EMEP", "MONARCH"])
```

Plot customisation

To set the title, x or y axis labels:

```
provi.plot(plot_type, title="My custom title", xlabel="My custom xlabel", ylabel="My custom ylabel")
```

Any plot options that can be used in the reports can also be used within the library. These can be set by passing the *plot_options* argument and a list of plot options:

```
provi.plot(plot_type, plot_options=['annotate', 'bias’])
```

Or, individual plot options can be passed, for example:

```
provi.plot(plot_type, annotate=True, bias=True)
```

All other available customisation arguments can be found here:

https://earth.bsc.es/gitlab/ac/Providentia/-/wikis/Library

Customisation

Plot customisation

To update the style of plots in the dashboard, reports and library, edit the plot characteristics in settings/plot_characteristics.yaml.

settings/plot_characteristics.yaml

All

Dashboard

Report

Library

Tests

```
"timeseries":
  {
    "grid": { "axis": "both", "color": "lightgrey", "alpha": 0.8 },
    "threshold_line":
      { "linestyle": "--", "linewidth": 1.0, "color": "red", "zorder": 0 },
    …
    "dashboard":
      {"remove_spines": ["top", "right"], ...},
    "offline":
      {"orientation": "landscape", ...},
    "interactive":
      {"figsize": [17, 6], ...},
    "tests":
      {"figsize": [17, 6], ...},
  },
```

Colorbar customisation - Bounds

There are two ways to change the colorbar bounds:

- If you want to set the same bounds for all statistics, you can edit the parameters *vmin_absolute*, *vmax_absolute*, *vmin_bias* and *vmax_bias* under *map* in the plot characteristics files.
- If you want to set the bounds for each statistic (recommended), you should edit the same parameters in settings/basic_stats.yaml and settings/experiment_bias_stats.yaml.

Colorbar customisation - Color

To change the colors, you need to edit the cmap:

- If you want to set the same colors for all statistics, edit the parameters *cmap_absolute* and *cmap_bias* under *map* in the plot characteristics files.
- If you want to set the colors for each statistic (recommended), you should edit them in settings/basic_stats.yaml and settings/experiment_bias_stats.yaml.

Colorbar customisation - Color

Now the legend and colorbar colormaps can be customized by adding a new schema in settings/color_palettes.yaml.

```
"cams": ["#ED1B23", "#2D2F92", "#7CFC00", "#009B55"]
```

To update the legend colors, you will need to set the custom name in *legend_color_palette* in the plot characteristics files.

For the colorbar, you should define *cmap_absolute* or *cmap_bias*, depending on the statistic type. The colors will interpolate depending on n_discrete. Below n_discrete is 15.

Colorbar customisation - Discretisation

You might also want to change the number of breaks if you have a discrete colorbar. For this you will need to change the number of ticks (*n_ticks*) and number of discrete colors (*n_discrete*) under *map* in the plot characteristics files.

In this example, we have set *n_ticks* to be 8 and *n_discrete* to be 15.

If you prefer to have a continuous colorbar and want to remove the breaks, you need to set *discrete* to be false.

Colorbar customisation - Per statistic

Users can define the color and bounds of the colorbar per species using a dictionary.

settings/basic_stats.yaml

```
"Mean": {"function": "calculate_mean",
         "order": 0,
         "label": "Mean",
         "arguments": {},
         "units": "[measurement_units]",
         "minimum_bias": [0.0],
         "vmin_absolute": {"sconco3": 0, "sconcno2": 0},
         "vmax_absolute": {"sconco3": 20, "sconcno2": 5},
         "vmin_bias": {},
         "vmax_bias": {},
         "n_ticks": {},
         "n_discrete": {},
         "cmap_absolute": {"sconco3": "viridis", "sconcno2": "viridis"},
         "cmap_bias": "RdYlBu_r"}
```

Legend customisation

If you want to change the color of the legend, you can edit the *legend_color_palette* under *general* in the plot characteristics files.

The default palette is *husl*:

Labels customisation

By defining **observations_data_label** in our configuration files, we can customise the name of the observations in the legend, plots and maps.

observations_data_label = EBAS

If you want to edit the names of the experiments, you can only do it launching Providentia with a configuration file, where you will define the alternative names as in:

experiments = cams61_emep_ph2-eu-000, cams61_monarch_ph2-eu-000 (EMEP, MONARCH)

Labels customisation

Remove stations by statistics

Remove stations by statistics

If you want to automatically remove stations that have certain statistical values, you will need to add your criteria in the file settings/remove_extreme_stations.yaml. An example of this exists for CAMS:

configurations/configuration_name.conf

```
[CAMS2_40]
network = ineris/eionet-cams2_40-ira
species = sconco3,sconcno2,sconcco,sconcso2,pm10,pm2p5
resolution = hourly
start_date = 20230101
end_date = 20230215
experiments = a59g-regional-000, a59j-regional-006 (Forecast, Analysis)
temporal_colocation = True
spatial_colocation = False
report_type = operational
report_summary = True
report_stations = False
report_title = CAMS2_40 Forecast and Analysis Report
report_filename = operational_report
statistic_mode = Temporal|Spatial
statistic_aggregation = Median
periodic_statistic_mode = Independent
periodic_statistic_aggregation = Mean
remove_extreme_stations = CAMS
```

settings/remove_extreme_stations.yaml

```
"CAMS": {"r": ["<0.3"],
         "NMB": ["<-100.0", ">100.0"],
         "NRMSE": [">100.0"]}
```

Remove stations by statistics

Any absolute statistic can be set to be a bias statistic by adding _bias e.g.:

"p95_bias": ["<10",">20"]

The statistics can be general, across all components, or they <u>can be specific per component</u>, for example:

settings/remove_extreme_stations.yaml

```
"CAMS": {"r": {"sconco3": ["<0.3"],
               "sconcno2":[<0.55]},
         "NMB": {"sconco3": ["<-100.0", ">100.0"],
                 "sconcno2": ["<-20.0", ">20.0"]},
         "NRMSE": {"sconco3": [">100.0"],
                   "sconcno2":[">200.0"]}}
```

Map background

Users have the ability to plot a background image on the map axis by changing the ***background*** variable under the ***map*** section in the settings/plot characteristics.yaml.

There are 3 available standard options:

1. ***providentia*** (default)
2. ***blue_marble*** (NASA's blue marble product)
3. ***shaded_relief*** (imagery showing changes in evolution)

However, any image can be set. Check the Plot Customisation wiki page for all the details.

Related tools

GHOST

Details: https://earth.bsc.es/gitlab/ac/GHOST

Project dedicated to the harmonisation of global surface observations (most notably air quality pollutants) and metadata.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

NES

Details: https://earth.bsc.es/gitlab/es/NES

Used to create observational networks from scratch that can be read by Providentia (XVPCA, CSIC, Port Barcelona, etc.)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

MAPIES

Details: https://earth.bsc.es/gitlab/ac/MAPIES

MAPIES stands for Mapper, Aggregator and Preprocessor for Integration of Earth observation Satellites to be used in our corresponding tools and research.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you for your attention!

More information at:
https://earth.bsc.es/gitlab/ac/Providentia

Join the #providentia Slack channel!

paula.serrano@bsc.es | alba.vilanova@bsc.es | dene.bowdalo@bsc.es