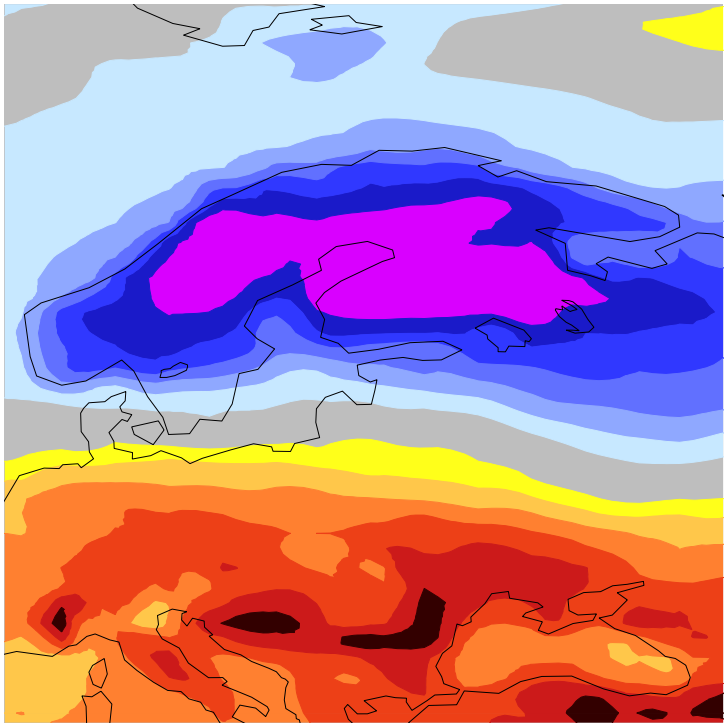
ECMWF Feature article

from Newsletter Number 179 – Spring 2024

COMPUTING

Introducing earthkit



www.ecmwf.int/en/about/media-centre/media-resources

doi: 10.21957/ir44tf392f

This article appeared in the Computing section of ECMWF Newsletter No. 179 – Spring 2024, pp. 48–53.

Introducing earthkit

Iain Russell, Tiago Quintino, Baudouin Raoult, Sándor Kertész, Pedro Maciel, James Varndell, Corentin Carton de Wiart, Edward Comyn-Platt, Olivier Iffrig, James Hawkes, Simon Smart

Earthkit is an exciting new open-source Python development led by ECMWF, providing powerful tools for speeding up weather and climate science workflows by simplifying data access, processing, analysis, visualisation, and much more. Earthkit is at the start of a multi-year development, but many aspects are already open for testing and feedback.

Earthkit in context

In 2023, ECMWF published its Software Strategy and Roadmap for 2023–2027 (Quintino et al., 2023), outlining the plan for software development over the next five years. Some of the core principles which underpin the software strategy are:

- improving reusability and componentisation of software
- further use of external software
- adoption of open development
- ensuring data scalability to new higher resolutions
- modernisation to new standards for higher interoperability.

Earthkit helps us realise this vision by offering multiple interoperable software components built on top of well-established open-source Python libraries, such as NumPy, pandas and Matplotlib. It also integrates and leverages the robust and operations-ready software stack that is familiar to ECMWF production systems (e.g. ecCodes, the Fields DataBase (FDB), etc.).

ECMWF has decades of experience in developing software that handles high data volumes with timeliness as an imperative. With this in mind, we have designed earthkit with certain principles, such as the ability to perform an entire workflow without reading any data from or writing any data to storage. We are also responding to the trend towards GPU architectures in high-performance computing facilities (HPCF), by ensuring that processing pipelines are GPU-ready.

The breadth of scope of earthkit is such that it is intended to eventually replace Metview, ECMWF's meteorological workstation software, as ECMWF's all-in-one environment for retrieving, inspecting, processing and visualising meteorological data. This is no small task, and we acknowledge that it will take several years, considering that Metview has been developed continually over the past three decades (for a look back at the beginning, see: Daabeck et al., 1995). Another major software package that will eventually be replaced by earthkit will be Magics, ECMWF's plotting package with a 35-year history of development (Lamy-Thépaut et al., 2023). Its functionality will be covered by the components earthkit-maps and earthkit-plots. Developments of Metview and Magics have already slowed in order to put more focus on earthkit, and user migration paths will be planned as more earthkit functionality becomes available.

Earthkit architecture

Earthkit consists of several Python components, each specialising in a certain area, such as data access, regridding or plotting. Each of these may depend on several Python and binary libraries developed at ECMWF, which, in turn, may depend on established third-party Python and binary libraries. One of the driving forces behind the architecture is reusability and componentisation of code – each package should do one job and do it well.

In addition to end users directly working with earthkit, ECMWF has many applications and frameworks that will be migrated to use earthkit over time. These include ECMWF's post-processing framework PProc, its verification software Quaver, its graphical forecast web application ecCharts, its hydrological post-processing framework Danu, and the Common Data Store (CDS) Engine. This is all summarised in Figure 1.

 doi: 10.21957/ir44tf392f

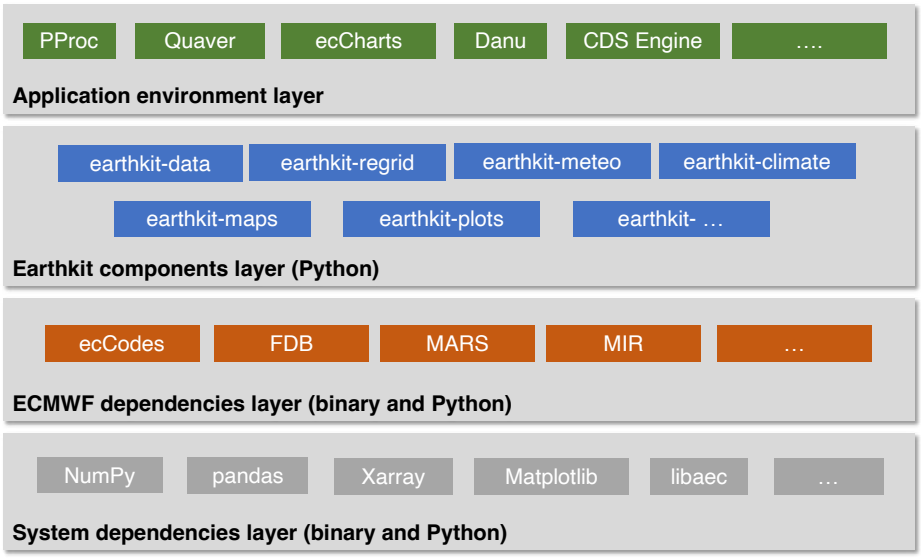


Figure 1 High-level overview of the software components that make up earthkit. The dependency chain includes many well-established third-party libraries covering, for example, data representation and processing (NumPy, pandas and Xarray), plotting (Matplotlib) and compression (libaec).

The following sections will present some detail about selected earthkit components.

Earthkit-data

The purpose of earthkit-data is twofold. Firstly, it provides an interface for accessing data, which includes loading files from disk, from URLs, directly from ECMWF's FDB, and from remote services such as ECMWF's Meteorological Archival and Retrieval System (MARS). Secondly, it represents data in a format-agnostic way and offers data extraction, inspection, slicing, filtering and conversion. By doing so, it provides a data layer making all the earthkit components interoperable. All is provided via a high-level interface, where details of data transfer, format and storage are abstracted and transparent to the user.

To simplify data access, users can invoke it via a single method `from_source()`. There are various built-in sources available, for example 'file', 'url', 'fdb' or 'mars', supporting a wide range of data formats including GRIB, NetCDF, BUFR, ODB and many more. Some sources, such as URLs and FDB, can be read as a stream straight into memory, allowing for highly effective workflows not involving any storage I/O. Additionally, users can define their own sources as plugins, which can then be published and shared with others without the need to update and release earthkit-data itself.

Earthkit-data also works comfortably with tar.gz files, returning a combination of their unpackaged contents. The code in Figure 2 shows how to load such data and display the contents of the packaged GRIB files, and the result is also shown in Figure 2.

```
import earthkit.data

path = "https://get.ecmwf.int/repository/test-data/earthkit-
data/examples/test_gribs.tar.gz"

ds = earthkit.data.from_source("url", path)

ds.ls()
```

	centre	shortName	typeOfLevel	level	dataDate	dataTime	stepRange	dataType	number	gridType
0	ecmf	2t	surface	0	20200513	1200	0	an	0	regular_ll
1	ecmf	msl	surface	0	20200513	1200	0	an	0	regular_ll
2	ecmf	t	isobaricInhPa	500	20070101	1200	0	an	0	regular_ll
3	ecmf	z	isobaricInhPa	500	20070101	1200	0	an	0	regular_ll
4	ecmf	t	isobaricInhPa	850	20070101	1200	0	an	0	regular_ll
5	ecmf	z	isobaricInhPa	850	20070101	1200	0	an	0	regular_ll

Figure 2 The ls() function displays columns of per-field information; extra keys can be specified by the user.

An additional benefit for users is that data retrieved from remote sources can be cached to disk, providing immediate access to it when the same retrieval is run again. The cache is highly configurable and can be switched off as per the typical requirements for operational environments, where disk access should be minimised and explicit.

All data objects returned by `from_source()` allow a few common methods, including the conversion of the underlying data into familiar scientific Python formats via `to_numpy()`, `to_pandas()` and `to_xarray()`. GRIB data and suitable NetCDF files are represented as a FieldList (i.e. a list of fields: horizontal slices of the atmosphere at a given time or period), offering a more specialised and richer functionality.

By design, memory usage in earthkit-data is kept at a minimum by implementing lazy loading whenever possible. When iterating over a FieldList created from GRIB, only one field is kept in memory at a time by default. This technique is vital when larger-than-memory datasets must be processed, and it can deal with an arbitrary number of GRIB fields. We can also represent a field as a combination of a metadata object and a NumPy array storing the values. With this we can build whole FieldLists decoupled from the original data format. This representation is particularly useful for computations, and support for other array backends, such as the GPU-ready PyTorch, is planned for the near future.

For examples, see the earthkit-data documentation, including its user guide and API reference at *https://earthkit-data.readthedocs.io/en/latest/*.

Earthkit-maps

Earthkit-maps is the geospatial visualisation component of earthkit, empowering users to produce publication-quality weather and climate maps in just a few lines of code. It is a convenient tool for taking a quick look at any geospatial data from a wide range of formats, whilst also offering the depth to fully customise every aspect of a chart for a publication or a presentation.

Earthkit-maps is built on the shoulders of giants, taking some of the features and philosophies of ECMWF's Magics visualisation software and applying them to a backend built on top of the feature-rich, open-source visualisation libraries Matplotlib and cartopy. It builds upon this strong foundation with domain-specific knowledge to offer a super simple API, which removes much of the code one would normally have to write when using either Magics or Matplotlib.

Earthkit-maps also leverages the power of earthkit-data and earthkit-regrid to greatly simplify the process of taking data from its source through to a visualisation. Users can easily write input-agnostic code that will work just the same with GRIB data as it does for NetCDF. They can also visualise data on an octahedral reduced Gaussian grid in exactly the same way as for a regular latitude–longitude grid.

Finally, earthkit-maps provides some extra convenience features, enabling:

- cropping a domain to fit a named country, region or continent, optimising (optionally) the map projection for the given region of the world

 doi: 10.21957/ir44tf392f

- automatic use of a suitable style based on input metadata
- creation and use of style libraries based on an organisation, project, or personal choice
- formatting of titles and labels using templates which read directly from the data's metadata
- built-in support for a wide range of ancillary layers from the Natural Earth cartography library, such as cities, urban areas, rivers, lakes and more.

The quickest way to get started using earthkit-maps is with `earthkit.maps.quickplot()`, which will attempt to visualise data the best it can based on its metadata. Figure 3 shows an example plot of 2-metre temperature and mean sea-level pressure.

```
import earthkit.data
import earthkit.maps
source_filename = "era5-2t-mslp-christmas-2010.grib"
earthkit.data.download_example_file(source_filename)
data = earthkit.data.from_source("file", source_filename)
earthkit.maps.quickplot(data)
```

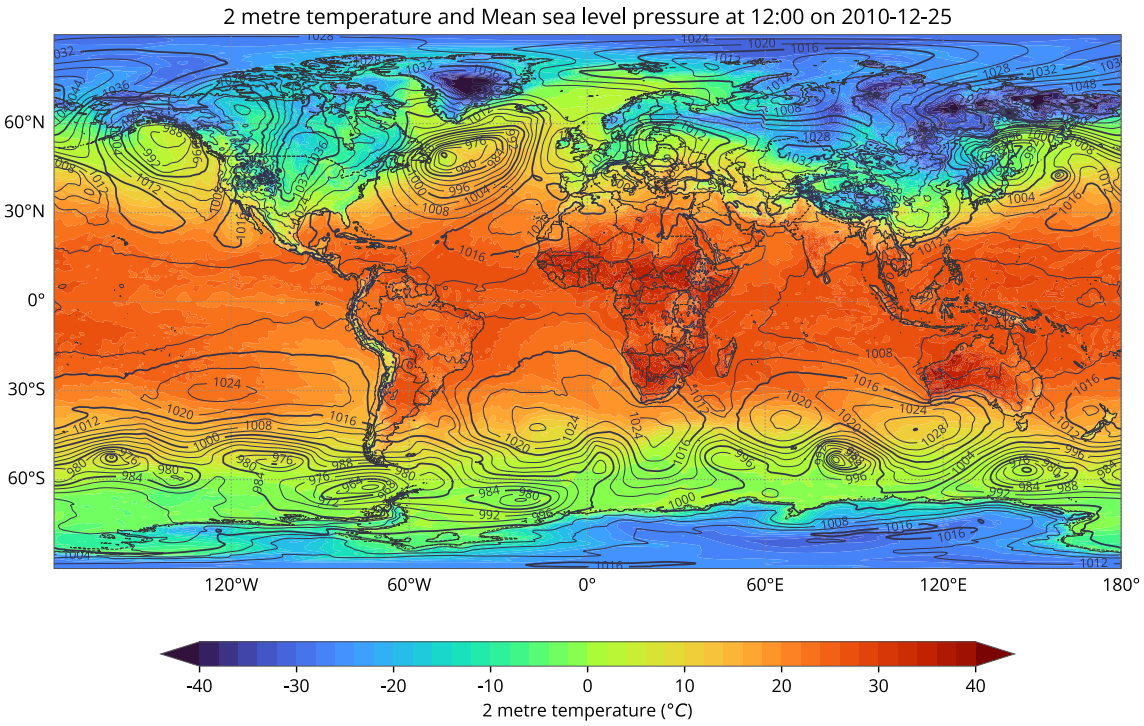


Figure 3 The quickplot() function of earthkit-maps is able to automatically assign styles and combine layers.

We have a global field here, but what if we want to zoom in on Europe? All we need to do is include a `domain="Europe"` argument to obtain the plot shown in Figure 4.

For many more examples, see the earthkit-maps documentation, including its user guide and gallery at *https://earthkit-maps.readthedocs.io/en/latest/*.

```
earthkit.maps.quickplot(data, domain="Europe")
```

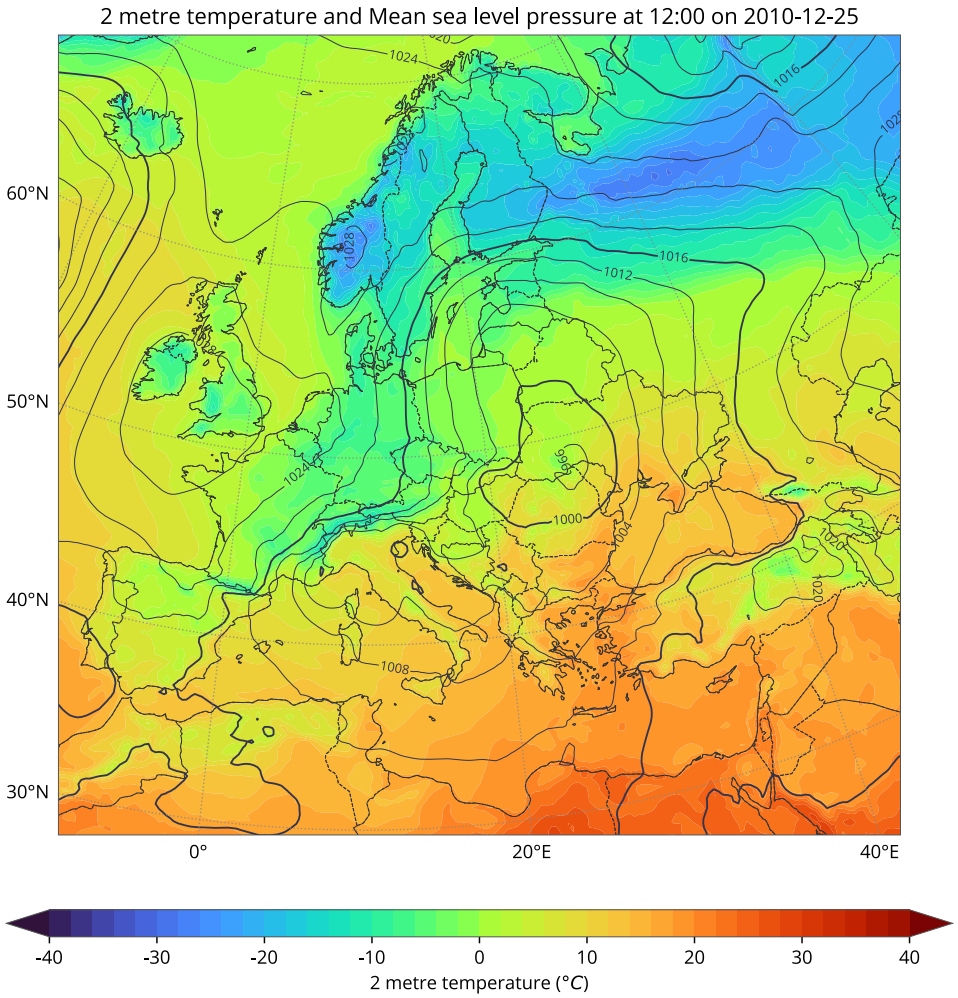


Figure 4 The result of specifying one of the pre-defined domains as an argument to the quickplot() function.

Earthkit-regrid

Earthkit-regrid is a component that specialises in mapping one arrangement of geographical points onto another, for example transforming an octahedral reduced Gaussian grid onto a regular latitude-longitude grid at a specific spatial resolution.

Earthkit-regrid leverages the principle that interpolation is fundamentally a geometrical relationship between input and output grid resolutions and formats. This relationship is modelled as a linear operation $\mathbf{y} = \mathbf{Ax}$, where **x** and **y** represent the input and output fields respectively, and **A** denotes the interpolation operator as a matrix that maps the input to the output. This matrix is generally sparse (lightweight) and multiplication can be efficiently implemented by a variety of optimized backends, including both CPU- and GPU-based architectures. The advantage of separating the matrix computation from the interpolation process is that it allows for a simpler interpolation package that can be deployed anywhere with minimal Python-only libraries.

The concept of calculating interpolation weights, caching, and performing multiplication was pioneered with the Meteorological Interpolation and Regridding (MIR) software package (Maciel et al., 2017). This is the post-processing engine of ECMWF's archive and product generation services (MARS/pgen), and it is key to the services' scalability and efficiency. Earthkit-regrid utilises pre-calculated MIR interpolation weights for a large selection of input/output grid combinations and a limited number of interpolation methods, which are fetched from a designated online source and cached upon the first request. MIR supports many advanced interpolation methods (e.g. conservative interpolation or methods that support climate files of ECMWF's Integrated Forecasting System), is extensively configurable, and has followed the ever-increasing requirements of high-resolution model runs. By sharing interpolation weights, MIR's functionality can be made increasingly accessible to earthkit-regrid over time.

 doi: 10.21957/ir44tf392f

The strategy of separating the generation of interpolation weights from performing interpolations through a high-level Python package is highly advantageous for both users and ECMWF as a data provider. Future iterations of earthkit-regrid will be able to directly call the MIR engine in order to facilitate regriddings that have not been precomputed. There will thus be facilitated support for new input/output grid combinations and more sophisticated interpolation methods, without necessitating changes in the user environment. The synergy between earthkit and MIR is a testament to the transformative potential of collaborative software development.

For more information and examples, see the earthkit-regrid documentation at *https://earthkit-regrid.readthedocs.io/en/latest/*.

Earthkit-meteo

The earthkit-meteo component is in its design phase and will contain a series of meteorological and mathematical functions that are widely used in our field. It aims to offer clear APIs coupled with extensive testing. This is to ensure their safe application across various contexts, such as operational forecast post-processing, integration into user applications, and usage within Jupyter notebooks, among others.

The earthkit-meteo component will be organised into several sub-modules, each dedicated to a specific post-processing category:

- **thermo:** contains thermodynamic functions, including computation of humidity, temperature conversions, etc.
- **wind:** calculates wind-related quantities, such as wind directions, and conversions between polar coordinates and Cartesian coordinates
- **stats:** offers essential mathematical and statistical functions, including quantiles and extensions of NumPy functions
- **extreme:** includes Extreme Forecast Index (EFI) and Shift of Tails (SOT)
- **score:** provides metrics such as the Continuous Ranked Probability Score (CRPS)
- … and many more coming!

The earthkit-meteo modules will be available through two main interfaces, each designed for different use cases:

- **A NumPy low-level Python API.** This interface directly processes NumPy arrays, facilitating efficient post-processing without the complexities of data management.
- **An earthkit high-level Python API**. This interface maintains metadata throughout the processing, ensuring outputs are valid earthkit-data objects with appropriate metadata, constructed from the inputs.

Earthkit-meteo will undergo some consolidation before its initial release, which will initially introduce the NumPy-level API.

Collaboration

All source code for earthkit is openly available on GitHub, and we have avidly embraced open development. With this approach, we mean to develop the codebase publicly. This means making the process transparent and, from the outset, available for interaction with the community, and in particular with ECMWF Member and Co-operating States. We seek early feedback on the design and features of the earthkit component system.

Early collaborations have already been set in motion. For example: (1) MeteoSwiss is seeking to adopt the earthkit components as a basis for its new generation post-processing system; (2) the German Meteorological Service (DWD) has provided a design and requirements for earthkit-data to support accessing data from the Amazon Simple Storage Service (S3); and (3) the Australian Bureau of Meteorology (BoM) is contributing new plot styles to earthkit-maps.

Through these and future collaborations, we believe that earthkit will strengthen our operational workflows as it becomes a building block for applications such as PProc (for post-processing), ecCharts, or new machine learning frameworks.

Conclusion and future

Earthkit is a large part of ECMWF's software strategy published in 2023, and it embodies all its principles. As more services and applications migrate to use earthkit, we will create an ever-more efficient and consistent software stack. The immediate priorities will be to allow basic and scientific computations via earthkit-meteo and the planned earthkit-hydro, and to bring earthkit-regrid to a level that more fully leverages the power of the MIR regridding engine. Developments have a robust future with further earthkit components planned or in early development stages. Earthkit-plots will complement earthkit-maps by providing ways to generate non-map plots, such as time series, meteograms and cross sections. Earthkit-climate will provide methods for calculating climate metrics and risk factors. It is expected to be highly community-driven. This is also because contractors of the EU's Copernicus Climate Change Service (C3S), which is implemented by ECMWF, will contribute their expertise whilst producing fully traceable project outputs, e.g. derived datasets and web applications.

We envisage a future where earthkit is widely adopted and contributed to, from both inside and outside ECMWF, and where it serves as a platform for stronger interaction with our Member States. Earthkit is still under development, but beta versions of earthkit-data, earthkit-maps, earthkit-plots and earthkit-regrid are available through PyPI. We welcome feedback on these packages through issues raised on GitHub.

A general introduction to earthkit is available here: *https://earthkit.readthedocs.io/en/latest/*, with installation instructions here: *https://earthkit.readthedocs.io/en/latest/install.html*. The GitHub code repositories for earthkit-data, earthkit-maps and earthkit-regrid are available here: *https://github.com/ecmwf/earthkit-data; https://github.com/ecmwf/earthkit-maps*; *https://github.com/ecmwf/earthkit-regrid*.

Further reading

Quintino, **T.**, **U. Modigliani**, **F. Pappenberger**, **S. Lamy-Thépaut**, **S. Smart**, **J. Hawkes**, **et al.**, 2023: Software Strategy and Roadmap 2023–2027. *ECMWF Technical Memorandum* **No. 904**.

Daabeck, **J.**, **B. Norris** & **B. Raoult**, 1995: Metview – Interactive Access, Manipulation and Visualisation of Meteorological Data on UNIX Workstations. *ECMWF Newsletter* **No. 68**, 9–28.

Lamy-Thépaut, **S.**, **S. Siemen** & **J. Varndell**, 2023: What next for Magics visualisation? *ECMWF Newsletter* **No. 177**, 22–26.

Maciel P., **T. Quintino**, **U. Modigliani**, **P. Dando**, **B. Raoult**, **W. Deconinck**, **et al.**, 2017: The new ECMWF interpolation package MIR. *ECMWF Newsletter* **No. 152**, 36–39.

 doi: 10.21957/ir44tf392f