Call: H2020-SC5-2014-two-stage

Topic: SC5-01-2014

PRIMAVERA

Grant Agreement 641727

PRocess-based climate sIMulation: AdVances in high resolution modelling and European climate Risk Assessment

Deliverable D1.1

First examples of common process-based metrics application

Deliverable Title	*First examples of the application of common process-based metrics to existing climate experiments with a focus on European climate variability and change*	
Brief Description	*Discussion of the strategic options for development of individual metrics and how these might be incorporated into existing packages*	
WP number	1	
Lead Beneficiary	UCL	
Contributors	David Docquier (UCL) Javier Vegas Regidor (BSC) Francisco J. Doblas-Reyes (BSC) Panos Athanasiadis (CMCC) Etienne Tourigny (BSC) Eleftheria Exarchou (BSC) Louis-Philippe Caron (BSC) Neven Fuckar (BSC) François Massonnet (UCL, BSC) Thierry Fichefet (UCL) Reinhard Schiemann (UREAD) Irene Mavilia (CNR) Torben Koenigk (SMHI) Ramon Fuentes Franco (SMHI) Tido Semmler (AWI)	
Creation Date Version Number Version Date	07.04.2017 02 19.05.2017	
Deliverable Due Date Actual Delivery Date	01.05.2017 30.06.2017	
Nature of the Deliverable	X	*R - Report*
		P - Prototype
		D - Demonstrator
		O - Other
Dissemination Level/ Audience	X	*PU - Public*
		PP - Restricted to other programme participants, including the Commission services
		RE - Restricted to a group specified by the consortium, including the Commission services
		CO - Confidential, only for members of the consortium, including the Commission services

Version	Date	Modified by	Comments
0.5	10.05.2017	Javier Vegas Regidor, David Docquier and Francisco J. Doblas-Reyes	
1	19.05.2017	All contributors	
2	30.06.2017	Paul van der Linden	Submission to EC

Table of Contents

1. Executive Summary

Metrics allow the assessment of model fidelity and comparisons between different models. In some projects, substantial resources are often spent in developing these metrics, however once projects end, metrics and codes are often lost to the community or, at best, not maintained.

Here we attempt a strategy for enabling individual metrics to be incorporated into larger, publicly-available metric packages, which themselves are maintained by communities for the longer term. This requires that the metrics themselves are written in the best way possible (e.g. using open source software, with well-documented functions), and also that the metric packages are clear on how additional metrics might be incorporated into their workflow. Metric packages should also provide versioning to ensure reproducibility.

A wide range of metrics related to the atmosphere, ocean and sea ice components has been developed by the PRIMAVERA partners since the beginning of the project. The aim of this report is not to present an exhaustive view of all the metrics developed but rather to provide an illustrative example with detailed scientific and technical explanations, as well as an example of code redundancy and the way to dealt with it, in order to facilitate future implementation of metrics into the Earth System Model eValuation Tool (ESMValTool), which is the package chosen to integrate the developments carried out in PRIMAVERA to ensure the legacy of the project.

This report focuses mainly on the sea ice drift-strength feedback metric as an example of the strategy that will be followed in WP1 during the rest of the project. The metric is described both scientifically (Section 3.1) and technically (Section 3.2). The information provided here will serve as a basis for future implementation of other process-based metrics into the ESMValTool, and also allows sharing the metric functions with PRIMAVERA partners. Unfortunately, the process of incorporating the metrics into ESMValTool cannot currently be described until the very last step because ESMValTool is currently (May 2017) being rewritten to improve its performance and enhance its capabilities. Section 3.3 explains how to deal with redundancy in metrics from different partners by taking the example of the atmospheric blocking. Section 3.4 provides a list of the other metrics that have been developed by PRIMAVERA partners and tested on the JASMIN platform. These metrics will be incorporated into ESMValTool at a later stage. Section 3.5 provides the references used. Finally, Sections 4 and 5 relate to the lessons learnt from the work undertaken and the links created by WP1 respectively.

2. Project Objectives

With this deliverable, the project has contributed to the achievement of the following objectives (DOA, Part B Section 1.1) WP numbers are in brackets:

No.	Objective	Yes	No
A	To develop a new generation of global high-resolution climate models. *(3, 4, 6)*		X
B	To develop new strategies and tools for evaluating global high-resolution climate models at a process level, and for quantifying the uncertainties in the predictions of regional climate. *(1, 2, 5, 9, 10)*	X	
C	To provide new high-resolution protocols and flagship simulations for the World Climate Research Programme (WCRP)'s Coupled Model Intercomparison Project (CMIP6) project, to inform the Intergovernmental Panel on Climate Change (IPCC) assessments and in support of emerging Climate Services. *(4, 6, 9)*		X
D	To explore the scientific and technological frontiers of capability in global climate modelling to provide guidance for the development of future generations of prediction systems, global climate and Earth System models (informing post-CMIP6 and beyond). *(3, 4)*		X
E	To advance understanding of past and future, natural and anthropogenic, drivers of variability and changes in European climate, including high impact events, by exploiting new capabilities in high-resolution global climate modelling. *(1, 2, 5)*	X	
F	To produce new, more robust and trustworthy projections of European climate for the next few decades based on improved global models and advances in process understanding. *(2, 3, 5, 6, 10)*		X
G	To engage with targeted end-user groups in key European economic sectors to strengthen their competitiveness, growth, resilience and ability by exploiting new scientific progress. *(10, 11)*		X
H	To establish cooperation between science and policy actions at European and international level, to support the development of effective climate change policies, optimize public decision making and increase capability to manage climate risks. *(5, 8, 10)*		X

3. Detailed Report

In this section, the sea ice drift-strength feedback metric is described both scientifically (Section 3.1) and technically (Section 3.2). The main purpose is to provide guidance for future implementation of process-based metrics into ESMValTool. Section 3.3 relates to the way to cope with redundancies between metrics. Section 3.4 describes several other metric examples that have been developed by PRIMAVERA partners and will be incorporated into ESMValTool. Section 3.5 provides the references used.

3.1. Scientific perspective for metrics development

The sea ice drift-strength feedback metric (further named 'drift-strength metric') is a process-based metric that has been developed at Université catholique de Louvain (UCL) in Belgium since the start of the PRIMAVERA project. Its goal is to support the improvement of the representation of sea ice processes in global climate models (GCMs), as sea ice is a key component of the climate system through its interactions with the ocean and atmosphere. The Python scripts for computing this metric are presented in the next section.

The drift-strength feedback is the positive feedback by which an initial decrease (increase) of concentration or thickness leads to reduced (increased) ice strength and internal stress, allowing more (less) deformation and fracturing within the ice, hence higher (lower) sea ice drift speed. This in turn provides higher (lower) export of sea ice out of the Arctic Basin, resulting in lower (higher) sea ice concentration and further thinning (thickening) (Rampal *et al.*, 2011).

A metric is a scalar number that compares a couple of model diagnostics to some reference (typically observationally-based). In the context of the drift-strength metric, the first diagnostic measures the relationship between sea ice drift speed and sea ice concentration, while the second diagnostic quantifies the relationship between sea ice drift speed and sea ice thickness, over the mean seasonal cycle. These diagnostics were built based on the study from Olason and Notz (2014). In our case, we take multi-year monthly mean quantities (sea ice drift speed, concentration and thickness) over the period 1979-2013, which is a sufficiently long period to extract interannual variability and is well covered by observations and reanalysis datasets. We also spatially average over two different domains in order to test the influence of the region choice, the first domain corresponding to the Scientific Ice Expeditions (SCICEX) box from US Navy submarine cruises (Rothrock *et al.*, 2008) and the second one taking into account all grid cells north of 50ºN with a sea ice concentration of at least 0.15. A weight is given to each grid cell proportional to the grid cell area. The two diagnostics are computed for both models and observations.

The drift-strength metric is divided into two sub-metrics that are computed from the two diagnostics and over the two different domains described above:

- slope ratio: ratio between the modelled and observed drift-concentration and drift-thickness slopes (the closer the ratio to 1, the better the agreement between both the modelled and observed relationships)
- normalised distance: mean distance (in %) between the model and observations for

both the drift-concentration and drift-thickness relationships (the lower the distance, the closer the model to observations).

Further information about the metric computation can be found in Docquier *et al.* (2017).

The analysis has been performed using the Nucleus for European Modelling of the Ocean coupled to the Louvain-la-Neuve sea Ice Model (NEMO-LIM3.6) run at 1º resolution and forced by atmospheric reanalysis (Drakkar Forcing Set [DFS] 5.2). Different observational and reanalysis products have been used to evaluate model outputs:

- sea ice concentration from EUMETSAT Ocean and Sea Ice Satellite Application Facility (EUMETSAT OSI SAF, 2015)
- sea ice thickness from Pan-Arctic Ice-Ocean Modeling and Assimilation System (PIOMAS; Zhang and Rothrock, 2003)
- sea ice drift speed from International Arctic Buoy Programme (IABP; Tschudi *et al.*, 2016).

Unfortunately, other pre-PRIMAVERA models do not provide daily X and Y components of the sea ice velocity over the recent period. Therefore, the present analysis is limited to one ocean-sea ice model forced by atmospheric reanalysis. A full multi-model assessment of the drift-strength metric will be performed using the upcoming High Resolution Model Intercomparison Project (HighResMIP) outputs from a variety of GCMs (Haarsma *et al.*, 2016).

A paper describing the drift-strength metric has recently been published in The Cryosphere Discussions in open discussion (Docquier *et al.*, 2017). Other supplementary observational and reanalysis products are used in this paper (Bootstrap for sea ice concentration; Ice, Cloud, and land Elevation Satellite ICESat for sea ice thickness; OSI SAF and PIOMAS for sea ice drift speed).

Figure 1 shows that:

- sea ice drift speed decreases with increasing concentration (Fig. 1a) and thickness (Fig. 1b) for both NEMO-LIM3.6 and the IABP / OSI SAF pair, while the PIOMAS corrected / OSI SAF pair does not show this behaviour (more information about the PIOMAS correction can be found in Docquier *et al.* (2017))
- the drift-thickness relationship is marked by a hysteresis loop: for a given thickness, two drift speed values are obtained depending on the season, while the PIOMAS corrected / PIOMAS pair cannot reproduce this loop.

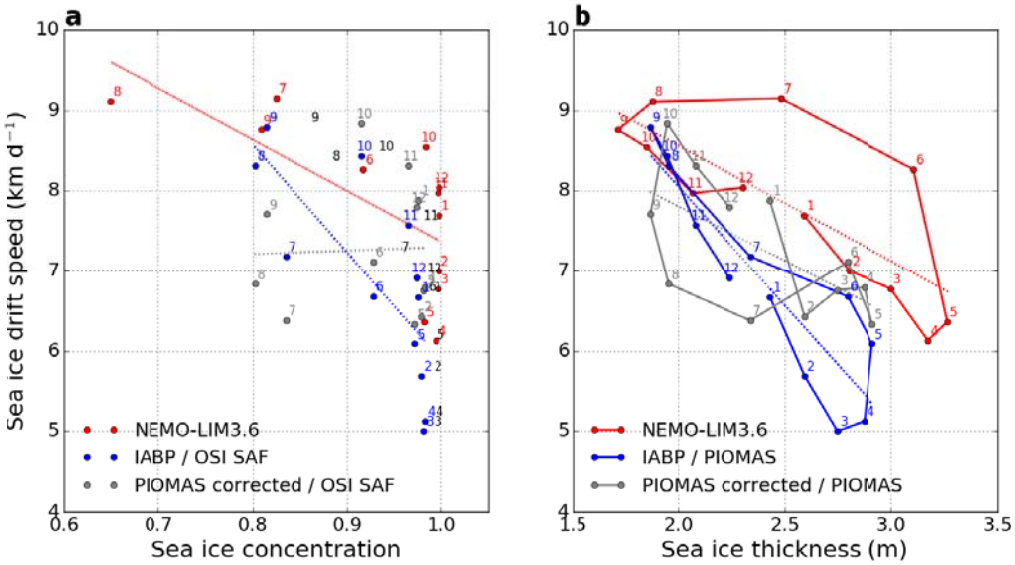

Figure 1: Sea ice drift speed against (a) sea ice concentration and (b) sea ice thickness for the NEMO-LIM3.6 model and different observation and reanalysis products. Data are temporally averaged over the 1979-2013 period and spatially averaged over the SCICEX box. Numbers denote months of the year.

Metric results are summarised in Table 1 and show that:

- the modelled drift-concentration relationship is in better agreement with the observed IABP / OSI SAF pair compared to IABP / Bootstrap and PIOMAS corrected / OSI SAF
- the modelled drift-thickness relationship is in better agreement with the PIOMAS corrected / PIOMAS pair compared to IABP / PIOMAS but the hysteresis loop is not well represented in the former pair.

Drift data / Concentration data	**Slope ratio**	**Normalised distance (%)**
IABP / OSI SAF	0.5	3.0
IABP / Bootstrap	0.3	3.5
PIOMAS corrected / OSI SAF	-15.0	2.9
Drift data / Thickness data		
IABP / PIOMAS	0.5	3.5
PIOMAS corrected / PIOMAS	1.1	3.3

Table 1: Metric results for NEMO-LIM3.6 compared to different observation / reanalysis products for drift-concentration and drift-thickness relationships.

The observational uncertainty related to the drift-strength metric is not directly computed within the Python scripts but can be estimated by calculating the temporal standard deviation of monthly mean sea ice drift speed, concentration and thickness over the mean seasonal cycle. This observational uncertainty, which is important to take into account due to the inherent error related to observations (e.g. large satellite observations errors in the summer due to the presence of melt ponds), is represented by error bars in Docquier *et al.* (2017).

The metric uncertainty is taken into account in the Python scripts by computing the standard error of the drift-concentration and drift-thickness linear regression slopes (Equation 6.18b of Wilks (2006)) and subsequently calculating the significance of these relationships at the 5% level using a Student's t-test. Results are not shown here but outputs are available when running the scripts.

Key points related to Section 3.1 (scientific perspective):

- the sea ice drift-strength feedback is an important feedback operating in the Arctic
- a metric related to this feedback has been developed and is divided into two sub-metrics (based on two different diagnostics) that quantify the ability of the model to reproduce observations
- analysis has been carried out with NEMO-LIM3.6 (forced atmospheric mode) against several observational datasets (Docquier *et al.*, 2017)
- observational and metric uncertainties are important to take into account (there is no single definition for these uncertainties but they need to be defined for each metric individually).

3.2. Technical perspective for metrics development

The Earth System Model eValuation Tool (ESMValTool) is a community diagnostics and performance metrics tool for the evaluation of Earth System Models (ESMs) that allows for routine comparison of single or multiple models, either against predecessor versions or against observations (Eyring *et al.*, 2016). ESMValTool is Open Source and has been developed using a public Git repository, allowing users to implement its own modifications with ease. Using Git version control system, users can manage versioning and ensure reproducibility, even when working with non-released or even customized versions. This tool has been chosen to insert the relevant PRIMAVERA metrics in order to reach a broader community and ensure metrics maintenance once PRIMAVERA is finished.

From a technical point of view, there are three steps that must be performed in order to develop the metric and incorporate it to ESMValTool:

- developing a prototype for JASMIN, the common platform of PRIMAVERA, which constitutes the core resource for exploiting data within the project
- refactoring
- integrating the metric into ESMValTool.

Each of these steps has its own focus and goals, which will be described in the following text using the drift-strength metric as an example.

a) Develop a prototype for JASMIN

The first step for each metric consists in developing a prototype that can be used with the data available on JASMIN, covering all the processes from gathering the data to the computation and plotting of the results. The prototype code should be made available through the PRIMAVERA Subversion repository in a branch named WP*X*/*metric_name*, where *X* is the work package number. Detailed information about how to use this Subversion repository can be found on the PRIMAVERA wiki (http://proj.badc.rl.ac.uk/primavera-private/wiki/SourceControl).

The drift-strength metric prototype is implemented as a set of Python scripts adapted for its run on JASMIN. They are available at the PRIMAVERA Subversion repository (http://proj.badc.rl.ac.uk/primavera-private/browser/WP2/sea_ice_drift_strength).

One of the most important things to bear in mind is that the prototype has to be transformed and adapted so that the metric can be incorporated into ESMValTool. This implementation work will be done by a computer scientist – who is a different person to the one who developed the original scripts. Therefore, the developer must focus on readability and ease of understanding over performance and memory consumption in order to ease the subsequent implementation into ESMValTool by the computer scientist. This focus can be seen in the following code excerpt (Code 1), where the script is thoroughly commented and empty lines are used to improve readability.

```
## 3. Load model outputs (NEMO-LIM3.6)
print('3. Load model outputs (NEMO-LIM3.6)')

## 3.1. Sea ice drift speed
print('  3.1. Sea ice drift speed')
```

```
if load_processed == False:

    # Select years for X component of sea ice velocity (daily outputs)
    siu_init = dir_model + 'siu_OIday_NEMO-LIM3-6_ORCA1_UCL_1_195801-201512.nc'
    siu = cdo.selyear('1979/2013',input=siu_init,output=dir+'siu.nc')

    # Select years for Y component of sea ice velocity (daily outputs)
    siv_init = dir_model + 'siv_OIday_NEMO-LIM3-6_ORCA1_UCL_1_195801-201512.nc'
    siv = cdo.selyear('1979/2013',input=siv_init,output=dir+'siv.nc')

    # Compute daily sea ice drift speed from X and Y components of sea ice velocity
    siu2 = cdo.sqr(input=siu,output=dir+'siu2.nc')
    os.remove(siu)
    siv2 = cdo.sqr(input=siv,output=dir+'siv2.nc')
    os.remove(siv)
    adduv = cdo.add(input=(siu2,siv2),output=dir+'adduv.nc')
    os.remove(siu2)
    os.remove(siv2)
    drift1 = cdo.sqrt(input=adduv,output=dir+'drift1.nc')
    os.remove(adduv)
    # convert from m/s to km/day
    drift2 = cdo.mulc('86.4',input=drift1,output=dir+'drift2.nc')
    os.remove(drift1)

    # Compute monthly mean sea ice drift speed
    file_drift = cdo.monmean(input=drift2,
                output = dir + drift_OImon_NEMO-LIM3-6_ORCA1_UCL_1_197901-
201312.nc')
    os.remove(drift2)

# Load monthly mean sea ice drift speed from netCDF file
fh = Dataset(file_drift, mode='r')
lon = fh.variables['nav_lon'][:] # longitude
lat = fh.variables['nav_lat'][:] # latitude
drift = fh.variables['siu'][:]
nm,nx,ny = drift.shape # number of months, X dimension, Y dimension
fh.close()
```

Code 1: Extract from the compute_metric.py file showing the data loading for NEMO sea ice drift speed. Some minor formatting changes have been done due to the lower line length available at the document.

Also important for the integration and code sharing is the maximum reduction of the code dependencies so that the final piece of software does not depend on other pieces of code. The dependencies can be further reduced by using the most common libraries available, as probably those will already be required by the software. The drift-strength metric required non-standard libraries are numpy, matplotlib, basemap and CDO, which are some of the most used by the climate community when developing code with Python.

Keeping only a few and fairly standard dependencies also helps other programmers working on the metric. In this way they do not need to know how lots of libraries work together and chances are that they already have experience with those tools. Even if they do not have any experience and have to learn how they work, there is a high probability that this learning

benefits the development of other metrics.

b) Refactoring

Once the metric prototype is ready, preparations for the integration into ESMValTool can start. In order to simplify the integration, the prototype is refactored with a focus on three main aspects:

- porting the code so it is based in the Iris library
- improve the overall code quality
- improve performance.

Depending on the prototype, the effort needed for each aspect can change, or even be unnecessary.

This refactoring is not done by the prototype developer (usually a scientist). Instead, it is done by the person in charge of the final integration into ESMValTool, which usually has a stronger background in computer science. The refactored code should be able to run using the same data as the prototype, simplifying comparisons between results.

The result of this refactoring for the drift-strength metric is available at PRIMAVERA's Subversion repository (http://proj.badc.rl.ac.uk/primavera-private/browser/WP2/sea_ice_drift_strength_refactor).

Porting the code to Iris

The Iris library is a Python library developed by the MetOffice that allows Python programmers to read, write and operate easily on climate data. The next version of the ESMValTool will use this as the main library for its backend. This new version will have better performance and more capabilities in the preprocessing step, and also will allow Python diagnostics to communicate with the tool by passing Iris cubes, instead of the old netCDF file-based communication used by NCL, R and old Python diagnostics. Any diagnostic using this new method will reduce its I/O operations, improving its performance. For these reasons, the drift-strength metric has been adapted to use Iris. Note that this is possible only if the final diagnostic is going to be implemented in Python.

As a bonus, the sea-ice drift adaptation has also lead to the reduction of the usage of CDO in its code, as most of the CDO calls used can also be implemented using Iris. Only one interpolation between two irregular grids is still using CDO, as Iris cannot do it for now.

Code quality improvements

Usually, the first draft of the metric is developed by scientists with no background in computer sciences. Due to this, there is a high probability that there is ample room for code cleaning and generalization. For example, most scientists use Python as a scripting language for procedural programming, defining a few long functions or even none at all. Modifying the code to use the object-oriented capabilities of Python, at the same time that the functionality is divided into small focused methods, greatly improves the generalization and maintainability of the code.

In the following code excerpt (Code 2), the same functionality of Code 1 is shown after porting to Iris and cleaning the code, mostly by using object oriented programming. By using

short functions and meaningful names for variables and functions, the need for comments has been eliminated, assuming that code readers have a basic knowledge of the Iris library. This code is also more powerful than the original: it can work with data split in different files, with different models and for different time intervals. Also, by using Iris instead of CDO, all calculations are done in memory without intermediate writes, making it more efficient.

```
def _load_drift_speed(self, model):
    self.log_subsubsection('Sea ice drift')
    if not self.recalculate and os.path.isfile(model.drift_file):
        model.drift[RAW] = iris.load_cube(model.drift_file)
        return

    siu = self._load_velocity(model, 'u')
    siv = self._load_velocity(model, 'v')

    cube = ((siu ** 2 + siv ** 2) ** 0.5)
    iris.coord_categorisation.add_day_of_year(cube, 'time')
    iris.coord_categorisation.add_year(cube, 'time')
    iris.coord_categorisation.add_month_number(cube, 'time')
    model.drift[RAW] = cube.aggregated_by(('year', 'month_number'),
                                          Iris.analysis.MEAN)
    model.drift[RAW].short_name = 'sivel'
    model.drift[RAW].long_name = 'Ice velocity'
    model.drift[RAW].convert_units('km day-1')
    iris.save(model.drift[RAW], model.drift_file, zlib=True)

def _load_velocity(self, model, component):
    fname = 'si{0}_OIday*.nc'.format(component)
    with iris.FUTURE.context(cell_datetime_objects=True):
        cubes = iris.load(os.path.join(model.path, fname), self.years_constraint)
    equalise_attributes(cubes)
    iris.util.unify_time_units(cubes)
    cube = cubes.concatenate_cube()
    return cube
```

Code 2: Extract from the compute_metric.py refactored, showing the data loading for NEMO sea ice drift speed. Some minor formatting changes have been done due to the lower line length available at the document.

This code excerpt also shows that this code cleaning concept is subject to interpretation. For example, at the _load_drift_speed it is possible to extract extra methods for the drift calculation and data saving, but in this case the programmer chooses not to extract them.

Finally, the code excerpt includes the functionality to allow the user to preprocess the data at will. It is mandatory at this stage that preprocessing and computation are clearly separated, even allowing to run without preprocessing the data: this helps the integration procedure in which the pre-process should be done by the backend. This functionality was already present in the original prototype of the drift-strength metric.

In the case of the drift-strength metric, another code quality improvement was achieved by changing some calculations that were done iterating through the data arrays to being computed using numpy matrix functions. This change also improves performance, as numpy functions are implemented by calling C code, which is much faster than pure Python.

Performance improvements

The last step in the refactoring is the work on the performance improvement. Before starting any work on it, the programmer must realize that preprocessing the data (time averaging, regridding, region extraction...) will be done in the final version by the ESMValTool backend, so no effort should be spent to improve performance in those parts of the code.

At this step, it is very likely that some of the work that has been done to improve the quality of the code has already given some performance improvement. For the drift-strength metric, the time consumed can be seen in Table 2. Note that the 'with preprocessing column' shows the time consumed including all preprocessing on raw data (e.g. time averaging, regridding), while the 'without preprocessing' uses the already pre-processed data. Therefore, the time consumed by the former is longer than the time used by the latter.

Version	With preprocessing	Without preprocessing
Original	~ 22 min	~ 11 min
Refactored	~ 16 min	~ 30 sec

Table 2: Time consumed by the metric for NEMO-LIM3.6 1979-2013 for the original and refactored code, using raw outputs (column 1) and using already preprocessed files (column 2).

The overall performance increases due to the refactoring. This can be explained by the intermediate writes removed while porting the code to Iris and the more extensive use of the numpy library for calculations. The final performance is more than acceptable, so no specific work to improve computation performance is required.

c) Integration into ESMValTool

The final step will integrate the metric into ESMValTool, but at this moment it is not advisable to do it. As explained before, the tool backend has been under a total rewrite and any metrics incorporated to the current version will need to be adapted again to the new backend in order to get the maximum profit from it. Therefore, our decision has been to go a bit further than necessary in the preparations step, trying to optimize and generalize the code as much as possible. The objective is to make the integration as straightforward as possible when the new ESMValTool backend is ready (expected for this summer).

This final integration step will also include final tweaks for code quality and performance. At this stage, working on performance can include trying to improve the backend, with the added advantage that any enhancement will probably benefit other metrics as well.

The scientist who developed the initial prototype is also involved in this stage. Technical and scientific documentation of the metric have to be added to ESMValTool documentation by the prototype developer and computer scientist in collaboration. This includes references to the peer-reviewed papers on which the metric is based and that should be cited whenever results coming from this metric are to be published. The prototype developer should also test the final metric to ensure that it fulfils all the initial requirements.

Key points related to Section 3.2 (technical perspective):

- the procedure involves three steps: prototype development, refactoring and ESMValTool integration
- it is important that the developer concentrates on code readability during the development phase
- before integration into ESMValTool, the codes are ported to Iris, which will be used in the next version of ESMValTool
- code performance can greatly increase following refactoring by the computer scientist
- integration of the drift-strength metric into ESMValTool has not already started since the new backend is not ready yet.

3.3. Homogenising metrics

One of the opportunities offered by a project as broad as PRIMAVERA is to homogenise the metrics used by the project partners. It is natural that different PRIMAVERA partners may have expertise in the same processes, and that they may therefore have developed similar or in some cases the same metrics. It is also possible that the same techniques may be implemented by different partners in metrics characterizing different phenomena.

As these metrics are provided for integration into ESMValTool, it is important to ensure that ESMValTool incorporates the joint capability of the community on the one hand, but avoids redundancy at code level on the other hand. Note that having two different metrics for the same process is not a problem; it is even a bonus for the users who can choose whichever fulfil better their needs or even use both at will.

A case in point is the “1D Blocking” and the “2D Blocking” (Scherrer et al., 2006; Schiemann et al., 2017) metrics, developed at Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC) and at the University of Reading (URead) respectively.

When both prototypes were ready and submitted, and the refactoring work started for both, it was easy to realize that much of the code was very similar in both cases. There were some parts that diverged and even the programming language used was different (Python for the 1D, R for the 2D) but the main algorithm was much the same with different configuration options and a different results presentation, and it was therefore not necessary to implement both these metrics into ESMValTool separately. While this point was obvious to the climate scientists developing these metrics, it should have been communicated more clearly to the computer scientist at the point of submitting the prototypes.

When these redundancies were detected, the refactoring was stopped and both prototype developers were contacted to discuss the strategy to be followed to avoid redundant work in the future. The decision was to merge both codes in a general blocking metric class, and use the example namelist that must be provided in ESMValTool to show how to configure it to launch two executions that exactly match what is done in the original prototypes.

Once the decision was made, refactoring restarted in the 1D metric code, as this was chosen to be the starting point for the final metric. Once this refactoring is advanced enough, the missing functionalities required by the 2D metric will be incorporated and a final code cleaning and performance work will be done. No more redundant work will be necessary. It is worth mentioning that the very fact that both partners provided their metric prototypes still proved to be useful. Two different implementations (here in R and Python) provided two possible starting points for refactoring, the more convenient of which has been chosen as the starting point for implementation.

Ideally, no redundancies should appear if coordination between partners works perfectly. Nevertheless, as redundancies can appear even in metrics characterizing completely different phenomena, it is probable that new redundancies will appear. There can be also cases in which redundancy is expected and even desirable (because metrics are based on already existing code or for developments in which the coordination will require more resources than doing it separately and then merge).

This example made clear that redundant code can be dealt with at the refactoring process easily. For that reason, when a metric prototype is submitted, a first check looking for redundancies with other metrics must be done and, if detected, affected metric developers must be contacted to decide how to solve the issue. The strategy presented in the example above may not be desirable or even possible for all the cases, so each case should be dealt with independently.

Key points related to Section 3.3 (metric homogenisation):

- code redundancy between two metrics related to the atmospheric blocking has been detected
- the two metrics are merged during the refactoring process
- effective communication between partners and across subject boundaries is crucial to avoid unexpected redundancies.

3.4. Other metric examples

Besides the sea ice drift-strength feedback metric (UCL) and the blocking metrics (CMCC, UREAD) presented in the previous sections, the following metrics have been submitted and tested on the JASMIN platform:

- land-atmosphere interactions in summer (CERFACS)
- cloud-temperature interactions (CERFACS)
- jet latitude index (CMCC)
- frequency of occurrence, pattern correlation and significance of cluster partition for weather regimes (CNR)
- cyclone tracking using the TRACK tool (Hoskins and Hodges, 2002) (UREAD)
- e-vectors (UREAD)
- energy budget and hydrological cycle (Demory *et al.*, 2014) (UREAD)
- deep water formation in the North Atlantic Ocean (SMHI)
- regional Arctic sea ice areas (SMHI)
- relation between sea ice variations and lower latitude climate (SMHI)
- Arctic freshwater exports (SMHI)
- profiles of temperature and salinity biases and mean absolute errors by ocean basin (AWI)
- ocean heat content in different layers (BSC)
- formation and propagation of tropical cyclones using GFDL Vortex Tracker (BSC)

Additionally, other metrics have been developed and will be tested on JASMIN, such as the sea ice cluster analysis (BSC; Fučkar *et al.*, 2016) and NCAR's CVDP diagnostics (SMHI, BSC).

Key points related to Section 3.4 (other metric examples):

- a wide range of metrics has been developed and will be incorporated into ESMValTool.

3.5. References

Demory, M.-E., P. L. Vidale, M. J. Roberts, P. Berrisford, J. Strachan, R. Schiemann, M. S. Mizielinski (2014). The role of horizontal resolution in simulating drivers of the global hydrological cycle, Climate Dynamics, doi:10.1007/s00382-013-1924-4.

Docquier, D., F. Massonnet, N. F. Tandon, O. Lecomte, T. Fichefet (2017). Arctic sea ice drift-strength feedback modelled by NEMO-LIM3.6, The Cryosphere Discussions, doi:10.5194/tc-2017-60, in review.

EUMETSAT OSI SAF (2015). Global sea ice concentration reprocessing dataset 1979-2015 (v1.2), http://www.osi-saf.org/.

Eyring, V. *et al.* (2016). ESMValTool (v1.0) - a community diagnostic and performance metrics tool for routine evaluation of Earth system models in CMIP, Geoscientific Model Development, doi: 10.5194/gmd-9-1747-2016.

Fučkar, N. S., V. Guemas, N. C. Johnson et al. (2016). Clusters of interannual sea ice variability in the northern hemisphere, Climate Dynamics, doi:10.1007/s00382-015-2917-2.

Haarsma, R. J. *et al.* (2016). High Resolution Model Intercomparison Project (HighResMIP v1.0) for CMIP6, Geoscientific Model Development, doi:10.5194/gmd-9-4185-2016.

Hoskins, B. J., K. I. Hodges (2002). New Perspectives on the Northern Hemisphere Winter Storm Tracks, Journal of the Atmospheric Sciences, doi: 10.1175/1520-0469(2002)059<1041:NPOTNH>2.0.CO;2.

Olason, E., D. Notz (2014). Drivers of variability in Arctic sea-ice drift speed, Journal of Geophysical Research, doi: 10.1002/2014JC009897.

Rampal, P., J. Weiss, C. Dubois, J.-M. Campin (2011). IPCC climate models do not capture Arctic sea ice drift acceleration: Consequences in terms of projected sea ice thinning and decline, Journal of Geophysical Research, doi: 10.1029/2011JC007110.

Rothrock, D. A., D. B. Percival, M. Wensnahan (2008). The decline in Arctic sea-ice thickness: Separating the spatial, annual, and interannual variability in a quarter century of submarine data, Journal of Geophysical Research, doi: 10.1029/2007JC004252.

Scherrer, S. C., M. Croci-Maspoli, C. Schwierz, C. Appenzeller (2006). Two-dimensional indices of atmospheric blocking and their statistical relationship with winter climate patterns in the Euro-Atlantic region, International Journal of Climatology, doi:10.1002/joc.1250.

Schiemann, R., M.-E. Demory, L. C. Shaffrey, J. Strachan, P. L. Vidale, M. S. Mizielinski, M. J. Roberts, M. Matsueda, M. F. Wehner, T. Jung (2017). The Resolution Sensitivity of Northern Hemisphere Blocking in Four 25-km Atmospheric Global Circulation Models, Journal of Climate, doi:10.1175/JCLI-D-16-0100.1.

Tschudi, M., C. Fowler, J. Maslanik, J. S. Stewart, W. Meier (2016). Polar Pathfinder Daily 25 km EASE-Grid Sea Ice Motion Vectors, Version 3 [IABP buoys], doi:10.5067/O57VAIT2AYYY, http://nsidc.org/data/NSIDC-0116/versions/3.

Wilks, D. S. (2006). Statistical methods in the Atmospheric Sciences (Second Edition), International Geophysics Series, Vol. 91.

Zhang, J., D A. Rothrock (2003). Modeling global sea Iie with a thickness and enthalpy distribution model in generalized curvilinear coordinates, Monthly Weather Review, 131, 845–861.

4. Lessons Learnt

- A strong interaction between the metric developers and the software expert has been developed, especially in the case of the examples provided in this report (drift-strength metric and blocking metrics). This interaction has allowed substantial code improvements and exchanges of views on scientific and technical aspects of the metrics developed.
- Coordination within partners is critical to avoid potential overlaps in metric development.

5. Links Built

- Strong collaboration with WP2 has been carried out to design process-based metrics. Some of the metrics described/listed in this report are also described in the deliverable D2.1 that was submitted by WP2.
- A solid link has been established with the team in charge of the redevelopment of ESMValTool. This is a key activity given that this tool has been chosen as the framework in which all the relevant PRIMAVERA metrics will be inserted and made publicly available.