Provenance Integration in R seasonal-to-decadal verification workflow

Albert Puiggròs

[a]*Universitat Politècnica de Catalunya - ETSETB, CPIA, c/ Jordi Girona, 31, Les Corts, Barcelona, 08034, Spain,*

Abstract

In the field of climate research, the integrity and reproducibility of data are paramount. Ensuring reliable outcomes is crucial to effectively manage data provenance. This report focuses on the proper integration of SUNSET (SUbseasoNal to decadal climate forecast post-processing and asSEmenT suite) and METACLIP (METAdata for CLImate Products), aiming to establish a complete system for tracking and managing data provenance in climate product verification. By implementing so, we seek to enhance useful provenance in generated climate products by providing a clear and traceable history of data sources, transformations, and final products. This integration is expected to extent comprehensively over the entire climate verification workflow, making it suitable for accurately extracting of all data provenance defined by METACLIP. The first block of the project reflects the process of familiarization and understanding of the working environment and methodology, as well as of the new software encountered and provenance data definition and implementation. Then, the second block delves into the details of the implementation and provenance extraction.

Keywords:

1. Introduction

Provenance, as defined on the Provenance Working Group definition in 2013, is a “record which specifies the people, institutions, entities, and activities involved in the creation and influence exercised on data”. Such a record turns out to be crucial, primarily to ensure reproducibility of obtained results. The proper tracking of a data’s course significantly increases the value of both the final product and original data by making the influence of one the other clear and traceable. Consequently, a recognized metadata scheme is vital when intending to extract provenance from data product generation. Also, official standard and vocabularies play a key role in properly describing the comprehensive metadata and schemes representing the lifecycle of data.

In a timed marked by rapid development of data and climate services, there exists a growing need among users and producers of an overarching complete provenance description of generated climate products. Several transnational initiatives have already been developed aiming to establish international standard for data provenance. Specific to the climate scene, some initiatives such the Climate and Forecast Metadata Convention (CF) have adopted international standards for metadata encoding. These types of conventions intend to describe physical meaning of data as well as spatial and temporal properties. The complexity of climate data processing requires a specialized and officially approved provenance framework that can adequately match the data used and generated with a recognized ontology.

To ensure the integrity and reliability of data provenance in climate research, this project relies on METACLIP (METAdata for CLImate Products) as a solution for identifying, extracting, linking, and assembling the information needed to fully characterize a climate product. It is based on RDF and focused on semantic description of climate products of all types, so that each of them and it’s provenance is inseparable and jointly delivered. These report analyses the possibility of a suitable integration of METACLIP into the SUNSET (SUbseasoNal to decadal climate forecast post-processing and asSEmenT suite) workflow, enhancing provenance tracking. SUNSET is an advanced R-based modular tool developed by the BSC Earth Science department designed for comprehensive forecast verification. It handles NetCDF files from several data sources and aims to simplify the processing, evaluation and interpretation of climate data. Its integration with METACLIP tries to embed detailed provenance information from its workflow and data sources, enhancing transparency and traceability of the climate forecasts produced. The following sections of the report delve deeper into the main components and functionalities of each system, as well as into the first approach regarding the desired implementation of the integration.

2. METACLIP

2.1. RDF-based approach for provenance description

METACLIP, standing for METAdata for CLImate Products, is a provenance framework based on semantics exploiting the web standards RDF (Resource Description Framework). It’s a framework designed to track and manage data provenance specifically of climate products. Exploiting the Resource Description Framework (RDF) METACLIP provides a

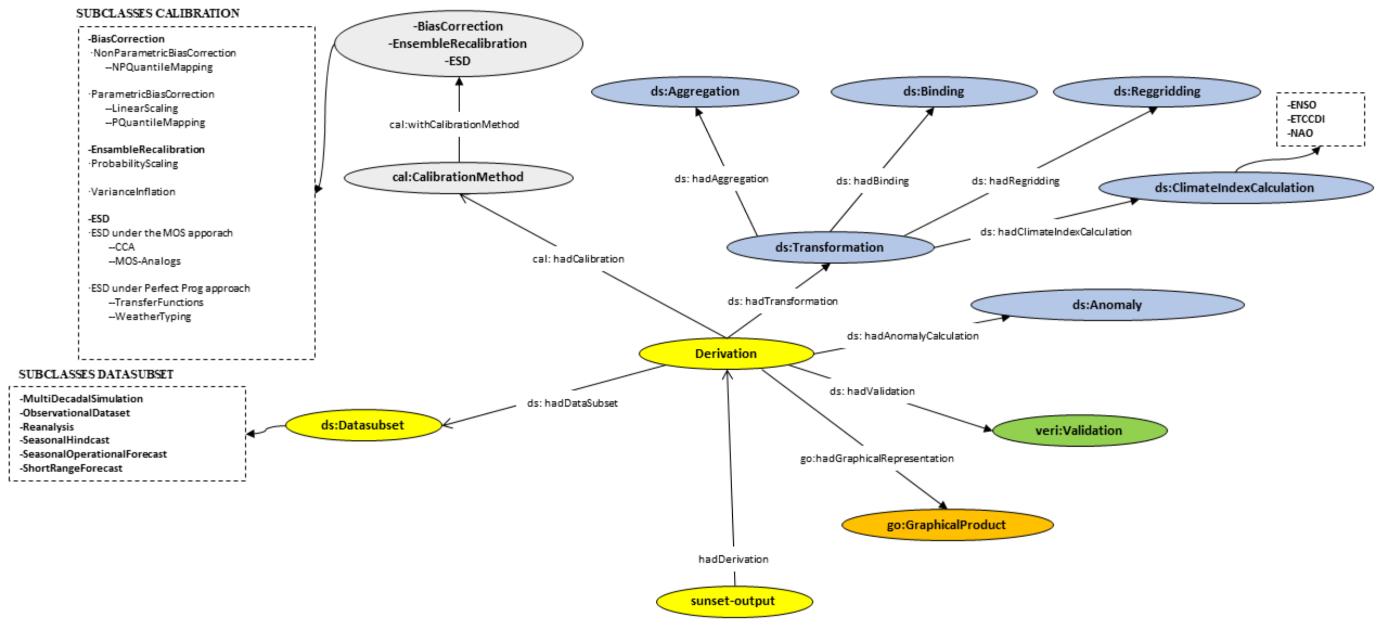

Figure 1: METACLIP vocabulary. Definition of classes and entities related to transformations undertaken by data.

language-independent solution.

RDF, developed by the World Wide Web Consortium (W3C), is a standard model originally created for metadata description that has since become a general method for describing and exchanging data. It allows the representation of information in a simple graph form, which consists of triple statements: subject, predicate, and object. Each part of the triple is uniquely identifiable by a Uniform Resource Identifier (URI), a string of characters used to recognize resources on the internet. It enables for a clear, unambiguous identification or resources in the RDF graph, easing the linking and integration of data from distinct sources. URI are the standard way to name and reference objects within RDF and broader semantic web frameworks.

Due to its versatility, RDF has been widely adopted across many fields, from geospatial features to music products. This high adaptability is achieved using ontologies, written in RDF by means of the Web Ontology Language (OWL). An ontology creates a simple model of a particular area of knowledge, listing types of objects, how they relate to each other and rules about them. As a result, the fact that RDF can handle many different types of vocabularies and models makes it particularly suitable for METACLIP, enabling it to give deep detail of climate products and their backgrounds. In the following Section more detail of the METACLIP vocabulary is given.

Essentially, RDF's is well suited for an easier and more efficient use of the semantic web. It's flexible, supporting a wide variety of syntaxes and data organization methods. Among these, Turtle is particularly popular due to its readability and simplicity. In addition, other common syntaxes such as Json provide alternative ways to encode and access RDF data, offering different choices depending on distinct needs and preferences. The use of RDF by METACLIP ensures that every piece of generated climate data, whether it's climatological data, indices, plots, or other kind of digital data, is inseparably linked with its provenance information. Then, users that have access to it can always trace the data back to its origins.

2.2. *Metaclip vocabularies*

The METACLIP framework implements specialized ontology to organize and manage provenance of climate data products. In this particular context, an ontology refers to a set of vocabularies and specific definitions regarding climate operations and data to represent the climate domain comprehensively. These vocabularies are collections of terms, definitions, and concepts, either abstract or physical-related, that are unequivocally interconnected. They attempt to illustrate not only data but also the processes undertaken and relationships directly associated with climate data production and processing.

It implements four core vocabularies, which are an extension of the well-known PROV-O ontology:

- **Datasource (ds:)** The datasource vocabulary details the source of the input data. It includes dataset descriptions and transformations. Also, it establishes the proper links

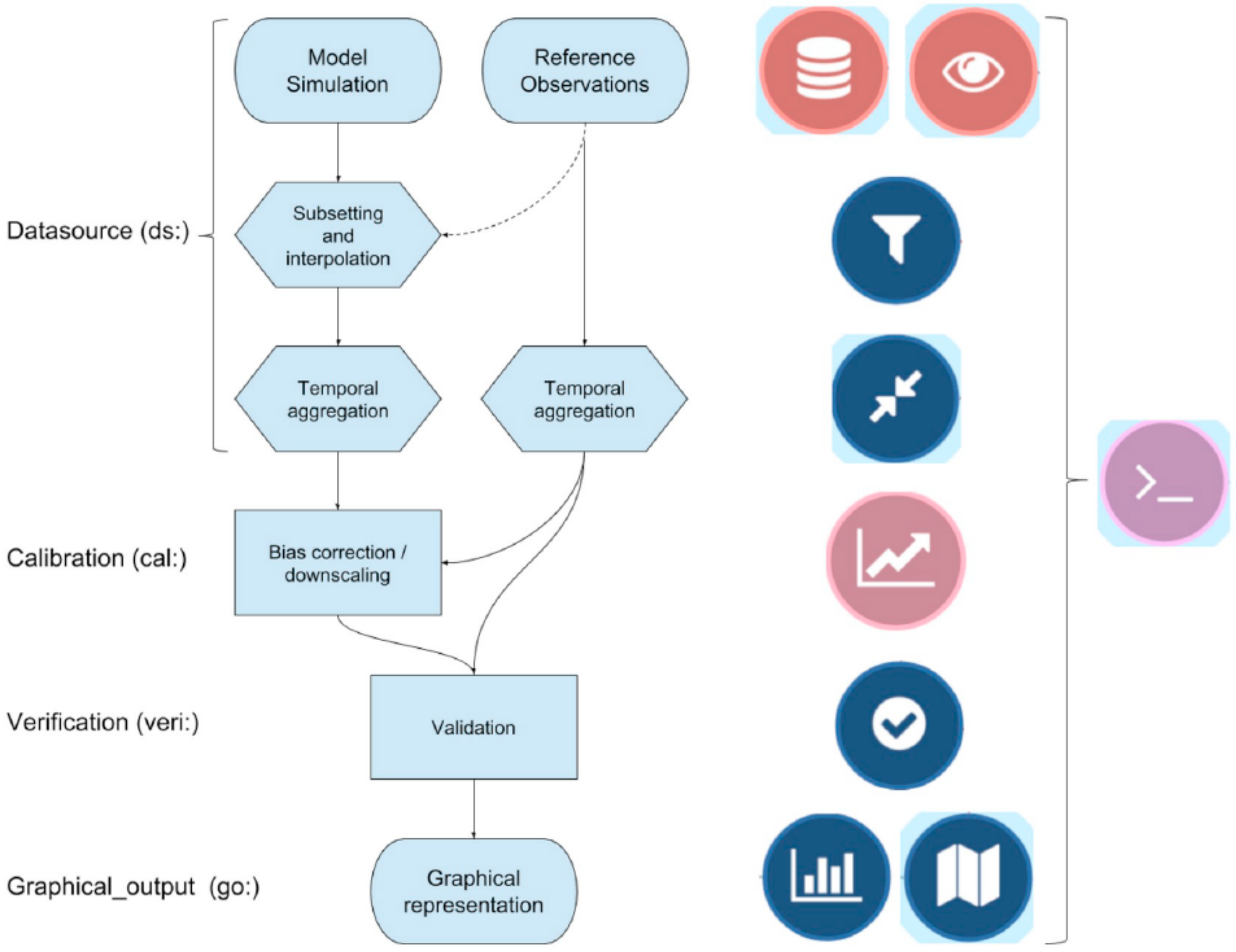

Figure 2: Schematic representation of a typical climate product generation workflow. METACLIP specifically considers the different intermediate steps and establishes a particular ontology for each of them.

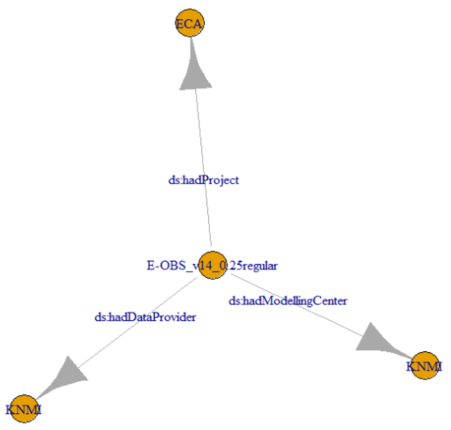

Figure 3: METACLIP-generated simple provenance graph.

between different classes and arguments at each step. Its URI is `http://metaclip.org/datasource.owl`.

- **Calibration (cal:)** Calibration encodes metadata related to bias correction, downscaling, and other statistical operations. It focuses on validating and developing statistical downscaling functionalities. Its URI is `http://metaclip.org/calibration.owl`.

- **Verification (ver:)** Aims to encode metadata for verifying seasonal forecast products. Moreover, it also includes a conceptual framework to implement other climate validation operations and forms. Its URI is `http://metaclip.org/verification.owl`.

- **Graphical Output (go:)** The graphical output ontology describes graphical products such as charts and maps. Characterizes the uncertainty types and their communication. Its URI is `http://metaclip.org/graphical_output.owl`.

METACLIP represents a significant extension of the PROV-O ontology which serves as its fundamental starting point pro-

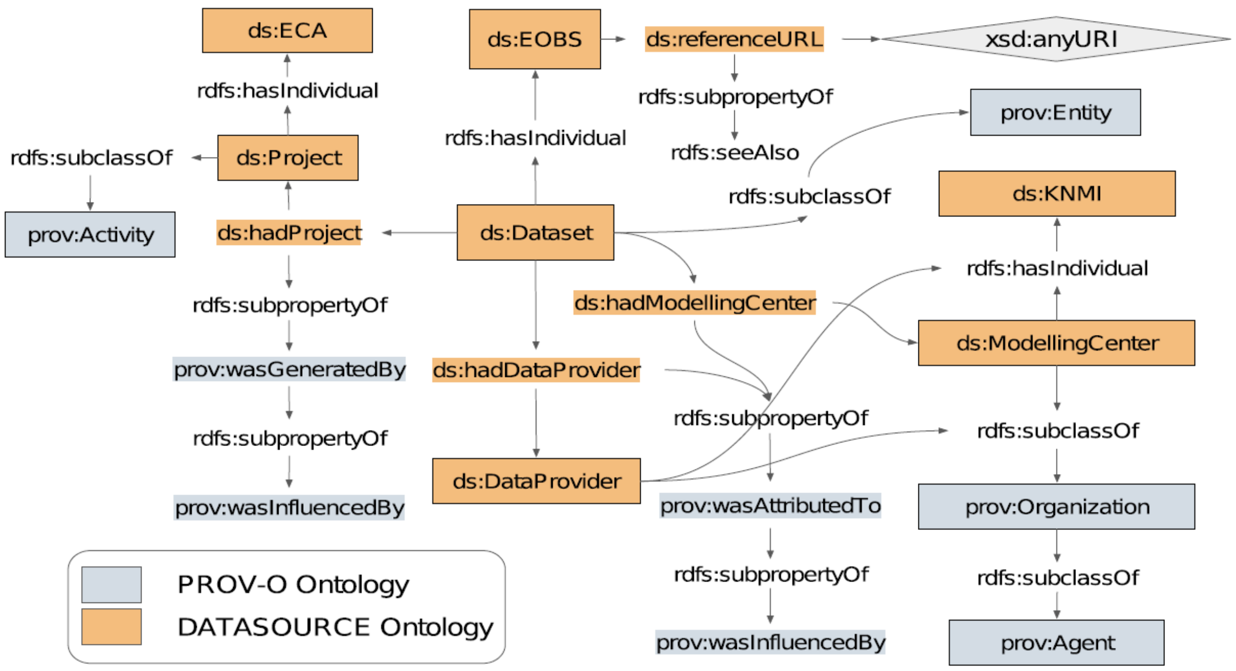

Figure 4: Schematic example showing the re-use of the PROV-O ontology by the METACLIP's datasource ontology.

viding key terms and concepts. The primordial terms from PROV-O are Entities, Activities and Agents. It's proper understating is highly important to achieve a correct definition of METACLIP terms. The "prov:Entity" class defines any physical, digital or conceptual objects -either real or imaginary-. The "prov. Activity" class describes something that occurs over a particular period of time and acts on entities. It includes data transformations, processing, relocation or modifying. Lastly, the "prov:Agent" class defines what has some form of responsibility for an activity taking place, either for the existence of an entity or for an activity being conducted. To accurately understand the METACLIP vocabularies and the way they establish an extension of the PROV-O data model, a brief explanation regarding Figure 4 is next given. In METACLIP, the first setp in provenance extraction involves defining the input data by means of the datasource(ds:) vocabulary. It is defined with the entity "ds:Dataset" (ds:Dataset) which can be identifies as an extension of the "prov:Entity". It's subdivided in other six subclasses based on the data nature. For example, "ds:ModellingCenter" and "ds:DataProvider", both subclasses of "prov:Agent" define, though in a different scope, the data's origin. Several other specific details unique to each "ds:Dataset" are recorded. When generally implemented, METACLIP can also encode the data's various transformations by means of the "ds:Step" class or even the construction of multi-model ensembles using the "ds:Ensemble" class. Many more operations can be described due to the large extension and specificity of the classes and relations.

2.3. *metaclipR: Climate4R extension*

The METACLIP framework is implemented through the programming environment R in the package *metaclipR*. The primary function of *metaclipR* is to track and extract the provenance data from various operations during data workflows and convert that information into RDF, based on the METACLIP ontology. Even though *metaclipR* is capable of handling different types of data, it is particularly customized for the *climate4R* framework, a tool designed for processing climate data within the R environment. It's structured approach offers the handling of various types of data: observations, seasonal forecasts, graphical representations, etc. It is subdivides in four core packages: *loadR*, *transformR*, *visualizeR* and *downscaleR*. The specialization of *metaclipR* on *climate4*R allows to efficiently deal with all the packages and their execution. Thus, it is built to specifically extract provenance information from the way *climate4R* handles and analyses climate data. Furthermore, *metaclipR* includes functions from *climate4R* packages. Then, in order to properly run *metaclipR* functionalities all the packages prior mentioned have to be installed. In summary, *metaclipR* generates metadata by mapping function calls or input arguments onto the METACLIP ontology, creating an RDF graph representation of the provenance data. The graph is contructed using the *igraph* package. A preliminary overview of the *metaclipR* functions lead to an output shown in Figure 5 where some simple dataset provenance is stored from an established data source.

3. SUNSET

3.1. *SUNSET: Workflow*

The SUNSET framework offers a modular processing of climate data products combining user-defined recipes, climate

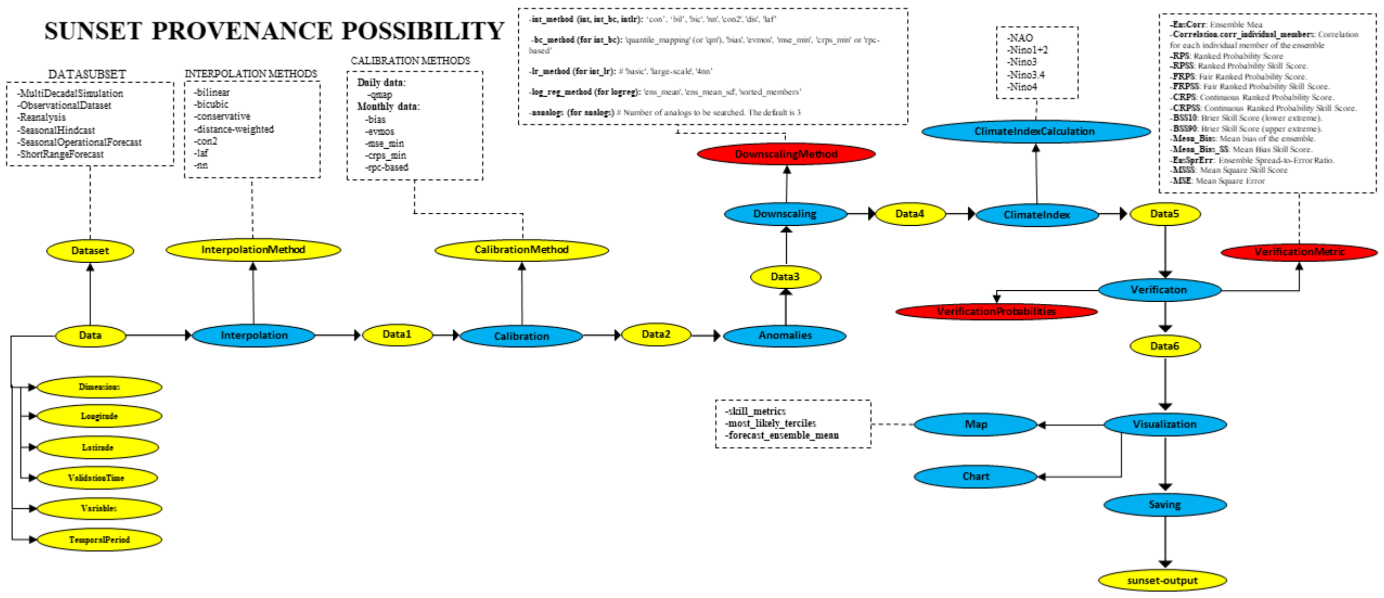
SUNSET PROVENANCE POSSIBILITY
DATASUBSET
-MultiDecadalSimulation
-ObservationalDataset
-Reanalysis
-SeasonalHindcast
-SeasonalOperationalForecast
-ShortRangeForecast
INTERPOLATION METHODS
-bilinear
-bicubic
-conservative
-distance-weighted
-con2
-laf
-nn
CALIBRATION METHODS
Daily data:
-qmap
Monthly data:
-bias
-evmos
-mse_min
-crps_min
-rpc-based
-int_method (int, int_bc, intlr): 'con', 'bil', 'bic', 'nn', 'con2', 'dis', 'laf'
-bc_method (for int_bc): 'quantile_mapping' (or 'qm'), 'bias', 'evmos', 'mse_min', 'crps_min' or 'rpc-based'
-lr_method (for int_lr): # 'basic', 'large-scale', '4nn'
-log_reg_method (for logreg): 'ens_mean', 'ens_mean_sd', 'sorted_members'
-nanalogs (for analogs) # Number of analogs to be searched. The default is 3
-NAO
-Nino1+2
-Nino3
-Nino3.4
-Nino4
-EnsCorr: Ensemble Mea
-Correlation.corr_individual_members: Correlation for each individual member of the ensemble
-RPS: Ranked Probability Score
-RPSS: Ranked Probability Skill Score.
-FRPS: Fair Ranked Probability Score.
-FRPSS: Fair Ranked Probability Skill Score.
-CRPS: Continuous Ranked Probability Score.
-CRPSS: Continuous Ranked Probability Skill Score.
-BSS10: Brier Skill Score (lower extreme).
-BSS90: Brier Skill Score (upper extreme).
-Mean_Bias: Mean bias of the ensemble.
-Mean_Bias_SS: Mean Bias Skill Score.
-EnsSprErr: Ensemble Spread-to-Error Ratio.
-MSSS: Mean Square Skill Score
-MSE: Mean Square Error
DownscalingMethod
ClimateIndexCalculation
Dataset
InterpolationMethod
CalibrationMethod
Downscaling
Data4
ClimateIndex
Data5
VerificationMetric
Data3
Verificaton
VerificationProbabilities
Data
Interpolation
Data1
Calibration
Data2
Anomalies
Data6
Dimensions
Longitude
Latitude
ValidationTime
Variables
TemporalPeriod
-skill_metrics
-most_likely_terciles
-forecast_ensemble_mean
Map
Visualization
Chart
Saving
sunset-output

Figure 5: Schematic example showing a possible provenance implementation in the Sunset workflow -without considering METACLIP ontology-.

R Package	GitHub Repository URL
devtools	`https://devtools.r-lib.org/`
transformR	`https://github.com/SantanderMetGroup/climate4R.git`
visualizeR	`https://github.com/SantanderMetGroup/visualizeR.git`
loadeR	`https://github.com/SantanderMetGroup/loadeR.git`
metaclipR	`https://github.com/metaclip/metaclipR.git`
climate4R	`https://github.com/SantanderMetGroup/climate4R.git`

Table 1: List of R packages and their GitHub repositories

data and scripts. The Recipe template for the workflow determining how the execution will take place and which operations will the modules within sunset apply. It specifies datasets, forecast horizon time period and skill metrics to compute and many other parameters.

Once the recipe is loaded, the data from the recipe -that also contains the directory of the climate data- is encapsulated within a *s2dv$_c$ube* object which contains observational, hindcast and forecast raw data. It's execution is distributed among several modules. The Loading Module starts the process by extracting and processing the data, interpolating it into *s2dv$_c$ube*. This first step is followed by the Calibration Module and Anomalies Module, which apply a bias correction and where anomalies are computed. Other existing modules specific to other climate processing actions are the Downscaling Module, the Indices Module, the Skill and Probabilities Modules, the Scorecard Module and the Saving module, all of them executing a particular transformation according to how is outlined in the Recipe. The outputs from the workflow are organized and saved in the designated folder, previously defined in the Recipe, in a NetCDF file. Remark the fact that NetCDF (Network Common Data Form) files are widely used in climate data processing for storing and distributing climate and forecast data and are particularly suited for handling large, multi-dimensional datasets.

3.2. *SUNSET: Environment, Gitlab and Conda*

To facilitate a comprehensive analysis of the SUNSET project, we utilized GitLab, a web-based DevOps lifecycle that provides a platform for software development and collaboration. The Sunset repository in GitLab integrates several stages of the development process, the source code, numerous example scripts and execution instructions. This approach not only allowed to access all relevant information of the sunset workflow and develop the code but also enabled us to collaborate more effectively, ensuring a detailed evaluation of the project's advancement. In this sense, it includes issue tracking systems, code review tools and merge requests, which promotes constant team interaction and collaboration. Moreover, it also allows multiple developers to work on a same project without altering the main code and file branch as changes can be merged seamlessly.

Although the sunset GitLab repository was used to understand the workflow and develop a first code implementation, it became necessary to establish a local environment to run and test it effectively. Among several options, we opted to install a Conda environment. Conda is an open-source package management system that allows the creation of isolated environments such the one of Sunset. This enabled us to mirror the project's repository functionalities into a local environment; then, to prevent conflicts with specific software requirements. Further details of its installation are given in the following section.

4. Provenance integration

4.1. *First approach: Familiarization and identification*

The initial phase of the research involved the analysis and comprehension of the METACLIP vocabularies and their implementation. For this purpose, we utilized Protégé, a free, open-source ontology editor developed by the Stanford Center for Biomedical Informatics Research. Protégé serves as a platform for constructing and manipulating ontology models. Its application was key to understand METACLIP's classification of each class and relation, and how the interconnections between classes are stablished. Protégé is designed to work with several formats, including .owl files. OWL stands for Web ontology Language. It is a semantic markup language used to publish and share data using specific ontologies and contains all definitions of classes, properties, individuals and data values, according to the constraints stablished by a particular ontology.

Listing 1: RDF file outcome

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ds: <http://metaclip.predictia.es/datasource/datasource.owl> .
@prefix prov: <http://www.w3.org/ns/prov> .

ds:Dimension
    ds:hasDimensions "Latitude", "Longitud" ;
    a ds:Entity .

ds:ERA5
    ds:SubClassOf "Dimension" ;
    ds:hadDataProvider "Meteo-France-System7" ;
    ds:hadModellingCenter "Meteo-France-System7" ;
    a ds:Dataset .

ds:Meteo-France-System7
    a ds:ModellingCenter .
```

Similarly, the research also involved the meticulous comprehension of the sunset workflow, as well as of the new environments it is established on. Alongside it, we took a first approach of provenance extraction from sunset, thus identifying possible integration points and defining individual pieces information to be propagated. Without considering the METACLIP vocabularies at first, we created a provenance scheme, as shown in Figure 5. In it we can identify all steps regarding the transformations the data has undertaken, as well as several provenance information that directly identify the dataset. All research done is reflected on the first sections of this report.

4.2. *RDF generation using R locally*

Due to the complexity and extensive nature of the sunset workflow and the METACLIP ontologies, our initial goal was to extract provenance information from the Recipe, which, in fact, contains a substantial portion of it, by means of the established METACLIP ontologies. Some difficulties with the local execution of the METACLIP R packages led us to develop a function that implements from scratch some defined ontology definitions to elements extracted and translated from the recipe. Two specific packages were required to use: "yml", for the efficient handling of YML data such the recipe, and "redland", for a solid RDF graph management within R. By combining the functionalities of these two packages, we were able to create a function that integrated minor provenance data from the recipe. The outcome data was saved in RDF format and serialized into turtle format(.ttl). The outcome is shown in Listing 1.
Even though the provenance extraction is brief and simple, the results obtained have proven to be insightful and valuable. The RDF data captured provides a structured representation of a simple provenance model of a particular dataset established in the Recipe. Remark the fact that this particular model could be largely extended by including the definition of more classes and relations from the METACLIP ontology. No further development is intended to be made on this code as I does not achieve the desired implementation into the sunset workflow. Also, it presents some minor errors when defining several classes. However, it's simply been created for research purposes and allowed us to gain a deeper comprehension on the structure of this type of provenance data. Finally, the results can be graphically visualized by means of a proper RDF interpreter. In this particular case, we have used an online interpreter with URL`https://www.ldf.fi/service/rdf-grapher`. The obtained graph is Shown in Figure 5. Remark the fact that many other libraries related to the provenance project were also analysed and tested, such as *rdflib* or *jsonld*.

4.3. *METACLIP integration in Sunset: environment setting*

As previously stated, the aim of establishing a conda environment is to create an isolated workspace where specific software packages can be managed. In the first place, we had to install WSL (Windows Subsystem for Linux), a compatibility layer in Windows that allows Linux distribution to run. That enabled us to use a full-based Linux environment within our Windows operating system. Once the installation is complete

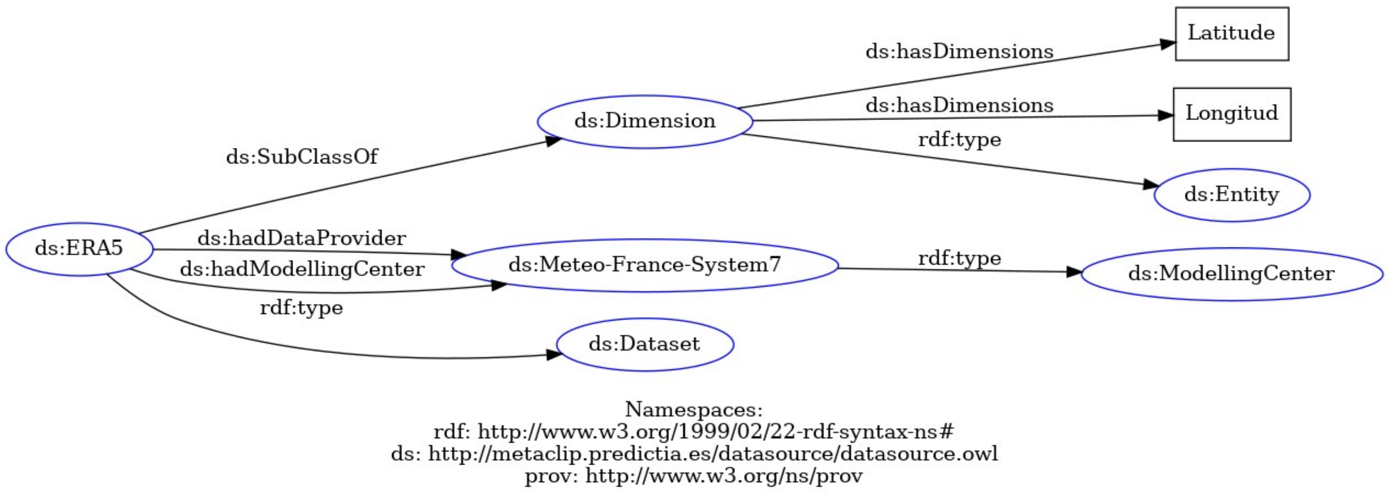

Figure 6: RDF graph

R Package	CRAN Repository URL
redland	`https://github.com/ropensci/redland-bindings/tree/master/R/redland`
yml	`https://github.com/vubiostat/r-yaml/`
jsonld	`https://docs.ropensci.org/jsonld`
rdflib	`https://github.com/ropensci/rdflib`

Table 2: List of R packages and their GitHub repositories

we can implement it by typing in the terminal `wsl`. This simple command initiates the Linux environment and grants access to its utilities.
Before setting up the sunset conda environment in our WSL workspace, we have to install either Miniconda (`https://docs.conda.io/en/latest/miniconda.html`)) or Anaconda (`https://www.anaconda.com/products/individual`). After the installation, we use the following command to create the environment:

```
> conda env create --file
environment-sunset.yml --prefix
User/conda-sunset/
```

or we can also use:

```
> mamba env create --file
environment-sunset.yml --prefix
User/conda-sunset/
```

This command uses either ”mamba” or ”conda” package manager to create a conda environment based on the specifications defined in an the environment configuration file `environment-sunset.yml`, which can be downloaded from the Sunset GitLab repository. Afterwards, we only need to activate the environment and the workspace setting will be completed:

```
> conda activate /home/albertpuiggros/User/
conda-sunset
```

Once the environment is properly set, we proceed to copy the sunset branch repository we have been working on in GitLab and install all the R packages via the R console to ensure a proper functioning of the code we desire to implement. First, we make a copy of the sunset branch:

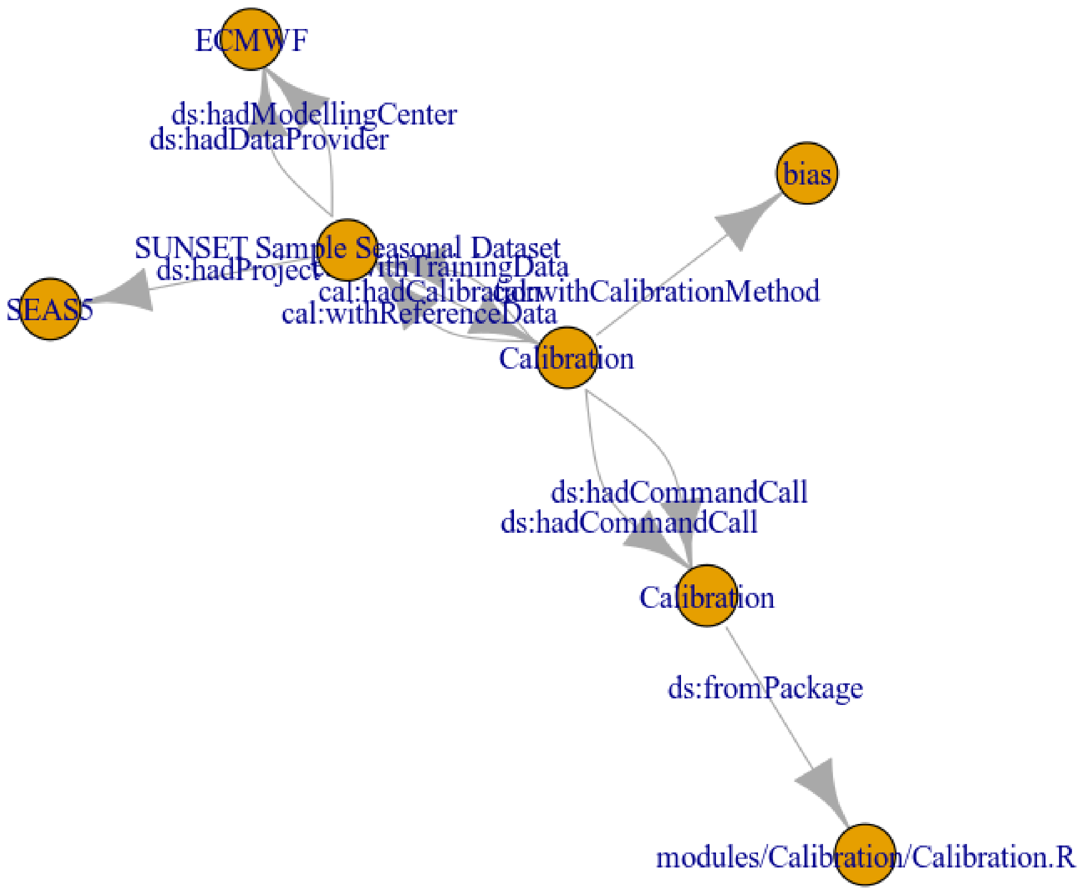

Figure 7: Final output

```
  > git <https://earth.bsc.es/gitlab/es/
sunset/dev-test-provenance.git>
```

Then we open an R session and install all the missing R packages that were mentioned in previous Sections through its console:

```
  > install.packages("CSTools")
  > install.packages("docopt")
  > install.packages("devtools")
  > install.packages("BiocManager")
  > BiocManager::install("Rgraphviz")
  > devtools::install-github("SantanderMetGroup/
transformeR")
  > devtools::install-github("metaclip/metaclipR")
  > devtools::install-github("SantanderMetGroup/
startR")
  > devtools::install-github("SantanderMetGroup/
loadeR")
  > devtools::install-github("SantanderMetGroup/
visualizeR")
```

After all the packages are installed, the environment is ready to run the sunset workflow locally. Even if we could manage to execute the workflow with some sample data, further test are required to achieve more certainty about its correct execution.

4.4. METACLIP integration in Sunset: Developed code

This section aims to entirely describe the implemented script. After a long familiarization process with the METACLIP and Sunset R functions and their use, we cooperatively created a simple code that aims to merge in a same execution both their functionalities. The created code is organized into several sections which simultaneously include METACLIP and Sunset functions, thus illustrating how they can be merged. In the first place, the scrpit begins by sourcing several Sunset modules such "Loading", "Calibration", "Skill", "Saving" and "Visualization" from the copied repository branch. It then loads the *metaclipR* library which implements all METACLIP functions. Then each line calls an R script from their specific path and load's it into the environment.

```
  > source("modules/Loading/Loading.R"))
  > source("modules/Calibration/Calibration.R")
  > source("modules/Skill/Skill.R")
  > source("modules/Saving/Saving.R")
  > source("modules/Visualization/Visualization.R")
  > library(metaclipR)
```

Then, the recipe is specified from its path. It is loaded and properly established in R by means of the `prepare-outputs()` function from Sunset.

```
  > recipe-file <- recipes/atomic-recipes/
recipe-seasonal-provenance.yml
> recipe <- prepare-outputs(recipe-file)
```

The first provenance implementation occurs after Sunset's Loading module is executed. This module retrieves the requested data and interpolates it into a desired grid based on the recipe's execution instructions. Afterwards, by means of the `metaclipR.Dataset()` function a metadata object is created detailing its source, type and several other possible characteristics.

```
  > data <- Loading(recipe)
  > metadata.seas5 <- metaclipR.Dataset(
Dataset.name = "SUNSET Sample Seasonal Dataset",
DataProvider = "ECMWF",  DataProvider.URL = url,
Dataset.subclass = "SeasonalHindcast",   Project
= "SEAS5", ModellingCenter = "ECMWF")
```

A similar procedure takes place when the Calibration module is executed. The data is calibrated using the Sunset function `Calibration()` with the method established in the recipe. Subsequently, provenance metadata is added to the prior metadata recorded by means of the METACLIP function `metaclipR.BiasCorrection`, which includes information about the method used for the bias correction, the arguments passed and the references for graphs and datasets.

```
  > calibrated-data <- Calibration(recipe, data)
  > metadata-calibrated <-
metaclipR.BiasCorrection(
package="modules/Calibration/Calibration.R",
"1.0", BC.method = "bias", fun = "Calibration",
arg.list = list( recipe data ), TrainingGraph =
metadata.seas5, ReferenceGraph = metadata.seas5,
graph = metadata.seas5)
```

Afterwards, skill metrics and probabilities, from Sunset's Skill module, are computed depending on the specific recipe instructions. Both functions execute taking as input the recipe and the calibrated data. However, no provenance data is extracted from this part of the script.

```
  > skill-metrics <- Skill(recipe,
calibrated-data)
> probabilities <- Probabilities(recipe,
calibrated-data)
```

Finally, the open graphics devices are closed by means of `dev.of()`. Then, the graph representing the entire defined provenance regarding dataset and calibration is plotted and saved as a JSON file.

```
  > dev.off()
> plot(metadata-calibrated/graph)
> graph2json(metadata-calibrated/graph,
"/tmp/graph.json")
```

The entire script is showed in Listing 4 in the Annexes. Overall, the code executes part of the sunset workflow for a sample data and simultaneously extracts dataset and calibration metadata by means of the functions of the *metaclipR* package.

4.5. METACLIP integration in Sunset: Final execution

Once all the installation is over, the environment set up and the code constructed, we proceed to execute the provenance integration code. To do so, we type the following command in the environment terminal:

```
  > Rscript ./example-scripts/test-decadal-
provenance.R
```

Upon successful execution of the command, the script processes data as stated before and generates two outputs: a JSON graph and a visual plot that represent the data's provenance and relationships. The JSON file is shown in Listing 5 in the Annexes and the visual plot in Figure 7. The JSON comprehensively encapsulates the relationships and dependencies among various provenance entities defined based on the METACLIP ontology. Thus, the METACLIP functions properly established the connection with their ontology. The dataset is identified as "SUNSET Sample Seasonal Dataset", categorized as a "ds:SeasonalHindcast" and is central in the graph's structure. From it, all metadata is added along the execution of the workflow. The metadata is well defined in the `@context` section where each used ontology, either "ds:", "cal:" or "prov:", is unequivocally identified with their URI. In the `@graph` section all the provenance information regarding the dataset -modeling center, data provider, project, type of dataset, etc- and the calibration -calibration method, command call, package used,etc- is recorded in triple form. The visual representation of the graph effectively translates the detailed provenance information from the JSON file into a more accessible format.

5. Conclusions

In conclusion, this report effectively demonstrated the possibility of a suitable integration of the METACLIP ontologies and its environment into the Sunset workflow. By embedding METACLIP specific R functions that detail provenance information within a Sunset execution script, we have reached transparency and traceability of a particular type climate products. This project has not only facilitated a deeper understanding of the ontology behind the definition of data, transformations and relations provenance, but also paved a way for a further more accurate implementation.

Moreover, this project implied the use of various new development environments: GitHub, GitLab, R programming languaje, Conda environment, etc. We have improved our technical skills regarding the new software encountered and learned how to address the managing and development of code cooperatively. The development of the project demonstrates how collaboration and the integration of diverse expertise can lead to rapid advancement and a suitable implementation in a relatively short period of time. Finally, further advancements need to be made regarding the implementation of the developed code with real data, the expansion of the implementation throughout a real workflow execution and a wider understanding of provenance definition and management within the climate scenario.

6. Annexes

6.1. Example Recipe

Listing 2: Example Recipe

```yaml
Description:
  Author: V. Agudetse
  Info: ECMWF System5 Seasonal Forecast Example recipe (monthly mean, tas)

Analysis:
  Horizon: seasonal # Mandatory, str: 'subseasonal', 'seasonal', or 'decadal'
  Variables:
    # name: variable name(s) in the /esarchive (Mandatory, str)
    # freq: 'monthly_mean', 'daily' or 'daily_mean' (Mandatory, str)
    - {name: 'tas', freq: 'monthly_mean'}
  Datasets:
    System:
      # name: System name (Mandatory, str)
      # member: 'all' or individual members, separated by a comma and in quotes (decadal only, str)
      - {name: 'ECMWF-SEAS5', member: 'all'}
    Multimodel: no # Either yes/true or no/false (Mandatory, bool)
    Reference:
      - {name: 'ERA5'} # Reference name (Mandatory, str)
  Time:
    sdate: '1101' # Start date, 'mmdd' (Mandatory, int)
    fcst_year: '2020' # Forecast initialization year 'YYYY' (Optional, int)
    hcst_start: '1993' # Hindcast initialization start year 'YYYY' (Mandatory, int)
    hcst_end: '2016' # Hindcast initialization end year 'YYYY' (Mandatory, int)
    ftime_min: 1 # First forecast time step in months. Starts at    1   . (Mandatory, int)
    ftime_max: 6 # Last forecast time step in months. Starts at    1   . (Mandatory, int)
  Region:
    latmin: -90 # minimum latitude (Mandatory, int)
    latmax: 90  # maximum latitude (Mandatory, int)
    lonmin: 0   # minimum longitude (Mandatory, int)
    lonmax: 359.9  # maximum longitude (Mandatory, int)
  Regrid:
    method: bilinear # Interpolation method (Mandatory, str)
    type: to_system # Interpolate to: 'to_system', 'to_reference', 'none',
                    # or CDO-accepted grid. (Mandatory, str)
  Workflow:
    # This is the section of the recipe where the parameters for each module are specified
    Calibration:
      method: mse_min # Calibration method. (Mandatory, str)
      save: 'all' # Options: 'all', 'none', 'exp_only', 'fcst_only' (Mandatory, str)
    Anomalies:
      compute: no # Either yes/true or no/false (Mandatory, bool)
      cross_validation: no # Either yes/true or no/false (Mandatory if 'compute: yes', bool)
      save: 'fcst_only' # Options: 'all', 'none', 'exp_only', 'fcst_only' (Mandatory, str)
    Skill:
      metric: RPSS CRPSS # List of skill metrics separated by spaces or commas. (Mandatory, str)
      save: 'all' # Options: 'all', 'none' (Mandatory, str)
    Probabilities:
      percentiles: [[1/3, 2/3], [1/10, 9/10], [1/4, 2/4, 3/4]] # Thresholds
      # for quantiles and probability categories. Each set of thresholds should be
      # enclosed within brackets. (Optional)
      save: 'percentiles_only' # Options: 'all', 'none', 'bins_only', 'percentiles_only' (Mandatory, str)
    Visualization:
      plots: skill_metrics, most_likely_terciles, forecast_ensemble_mean # Types of plots to generate (Optional, str)
      multi_panel: yes # Multi-panel plot or single-panel plots. Default is 'no/false'. (Optional, bool)
      projection: 'cylindrical_equidistant' # Options: 'cylindrical_equidistant', 'robinson', 'lambert_europe'. Default is cylindrical equidistant. (Optional, str)
      mask_terciles: no # Whether to mask the non-significant points by rpss in the most likely tercile plot. yes/true, no/false or 'both'. Default is no/false. (Optional, str)
      dots_terciles: yes # Whether to dot the non-significant by rpss in the most likely tercile plot. yes/true, no/false or 'both'. Default is no/false. (Optional, str)
  ncores: 10 # Number of cores to be used in parallel computation.
             # If left empty, defaults to 1. (Optional, int)
  remove_NAs: yes # Whether to remove NAs.
                  # If left empty, defaults to no/false. (Optional, bool)
  Output_format: 'S2S4E' # 'S2S4E' or 'Scorecards'. Determines the format of the output. Default is 'S2S4E'.
Run:
  Loglevel: INFO # Minimum category of log messages to display: 'DEBUG', 'INFO', 'WARN', 'ERROR' or 'FATAL'.
                 # Default value is 'INFO'. (Optional, str)
  Terminal: yes  # Optional, bool: Whether to display log messages in the terminal.
                 # Default is yes/true.
  output_dir: /esarchive/scratch/vagudets/repos/sunset-outputs/ # Output directory. Must have write permissions. (Mandatory, str)
  code_dir: /esarchive/scratch/vagudets/repos/sunset/
```

6.2. RDF code

Listing 3: RDF simple extraction

```r
library(redland)
library(yaml)
library(igraph)
library(rdflib)
library(jsonld)

read_recipe <- function(file_path) {
  recipe <- yaml::yaml.load_file(file_path)
  return(recipe)
}

extract_data <- function(recipe) {
  components <- list()
  components$reference_datasets <- recipe$Analysis$Datasets$System  # Extracting the system datasets name
  components$system_datasets <- recipe$Analysis$Datasets$Reference  # Extracting the reference datasets name
  components$time_datasets<-recipe$Analysis$Time
  components$region_datasets<-recipe$Analysis$Region
  return(components)
}

recipe<-read_recipe("C:/Users/User/Desktop/RECIPE.yml")
raw_metadata<-extract_data(recipe)

world <- new("World")
storage <- new("Storage", world, "hashes", name="", options="hash-type='memory'")
model <- new("Model", world=world, storage, options="")

ds<-"http://metaclip.predictia.es/datasource/datasource.owl"
prov<-"http://www.w3.org/ns/prov"
rdf<-"http://www.w3.org/1999/02/22-rdf-syntax-ns#"
cal<-"http://metaclip.predictia.es/calibration/calibration.owl#"
go <-"http://metaclip.predictia.es/graphical_output/graphical_output.owl#"
skos<-"http://www.w3.org/2004/02/skos/core#"

#DATASET
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$system_datasets$name),
            predicate=paste0(rdf,"type"),
            object=paste0(prov,"Entity"))
#addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$system_datasets$name),
            predicate=paste0(rdf,"type"),
            object=paste0(ds,"Dataset"))
addStatement(model, stmt)

#ModellingCenter
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$system_datasets$name),
            predicate=paste0(ds, "hadModellingCenter"),
            object=raw_metadata$reference_datasets$name)
addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$reference_datasets$name),
            predicate=paste0(rdf,"type"),
            object=paste0(prov,"Agent"))
#addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$reference_datasets$name),
            predicate=paste0(rdf,"type"),
            object=paste0(ds,"Organization"))
#addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$reference_datasets$name),
            predicate=paste0(rdf,"type"),
            object=paste0(ds,"ModellingCenter"))
addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$system_datasets$name),
            predicate=paste0(ds, "hadDataProvider"),
            object=raw_metadata$reference_datasets$name)
addStatement(model, stmt)

#HasDimensions
stmt <- new("Statement", world=world,
            subject=paste0(ds,"Dimension"),
            predicate=paste0(rdf,"type"),
            object=paste0(ds,"Dataset"))
#addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,"Dimension"),
            predicate=paste0(rdf,"type"),
            object=paste0(ds,"Entity"))
#addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,"Dimension"),
            predicate=paste0(rdf,"type"),
            object=paste0(ds,"Entity"))
#addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,raw_metadata$system_datasets$name),
            predicate=paste0(ds, "SubClassOf"),
            object="Dimension")
addStatement(model, stmt)

#Latitude(Dimension)
addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,"Dimension"),
            predicate=paste0(ds, "hasDimensions"),
            object="Latitude")
addStatement(model, stmt)
stmt <- new("Statement", world=world,
            subject=paste0(ds,"Dimension"),
            predicate=paste0(ds, "hasDimensions"),
            object="Longitud")
addStatement(model, stmt)


# Serialize the model to a TTL file
serializer <- new("Serializer", world, name="turtle", mimeType="text/turtle")
```

```
status <- setNameSpace(serializer, world, namespace="http://www.w3.org/2000/01/
       rdf-scheme", prefix="rdf")
status <- setNameSpace(serializer, world, namespace="http://metaclip.predictia.
       es/datasource/datasource.owl", prefix="ds")
status <- setNameSpace(serializer, world, namespace="http://www.w3.org/ns/prov"
       , prefix="prov")
filePath <- filePath <- "C:/Users/User/Desktop/recipe_output.ttl"
status <- serializeToFile(serializer, world, model, filePath)
readLines(file(filePath))
```

6.3. *Final code*

Listing 4: Final implementation script

```
source("modules/Loading/Loading.R")
source("modules/Calibration/Calibration.R")
source("modules/Skill/Skill.R")
source("modules/Saving/Saving.R")
source("modules/Visualization/Visualization.R")

library(metaclipR)

recipe_file <- "recipes/atomic_recipes/recipe_seasonal_provenance.yml"
recipe <- prepare_outputs(recipe_file)
# archive <- read_yaml(paste0(recipe£Run£code_dir, "conf/archive_decadal.yml"))
       £archive

# Load datasets
data <- Loading(recipe)

metadata.seas5 <- metaclipR.Dataset(
  Dataset.name = "SUNSET␣Sample␣Seasonal␣Dataset",
  DataProvider = "ECMWF",
  # DataProvider.URL = url,
  Dataset.subclass = "SeasonalHindcast",  # Right?
  Project = "SEAS5", # ?
  ModellingCenter = "ECMWF"
)
# TODO: Create ensemble metaclipR.Ensemble()

# Calibrate datasets
calibrated_data <- Calibration(recipe, data)

# Register the calibration...
metadata_calibrated <- metaclipR.BiasCorrection(
  package="modules/Calibration/Calibration.R",
  "1.0",
  BC.method = "bias",  # from recipe
  fun = "Calibration",
  arg.list = list(
    recipe,
    data
  ),
  TrainingGraph = metadata.seas5,  # ?
  ReferenceGraph = metadata.seas5,  # ?
  graph = metadata.seas5
)

# Compute skill metrics
skill_metrics <- Skill(recipe, calibrated_data)

# Compute percentiles and probability bins
probabilities <- Probabilities(recipe, calibrated_data)

# Plot data
# Visualization(recipe, calibrated_data, skill_metrics, probabilities,
#              significance = T)

dev.off()
plot(metadata_calibrated$graph)
graph2json(metadata_calibrated$graph, "/tmp/graph.json")

print("OK!")
```

6.4. *JSON file*

Listing 5: Final implementation script

```
{
        "@context": {
                "ds": "http://www.metaclip.org/datasource/datasource.owl#",
                "ipcc": "http://www.metaclip.org/ipcc_terms/ipcc_terms.owl#",
                "veri": "http://www.metaclip.org/verification/verification.owl#
                      ",
                "cal": "http://www.metaclip.org/calibration/calibration.owl#",
                "go": "http://www.metaclip.org/graphical_output/
                      graphical_output.owl#",
                "prov": "http://www.w3.org/ns/prov#",
                "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
                "dc": "http://www.w3.org/2002/07/owl",
                "skos": "http://www.w3.org/2004/02/skos/core#"
        },
        "@graph":[{
        "@id":  "#Dataset.moijnv",
        "@type": "ds:SeasonalHindcast",
        "rdfs:label": "SUNSET␣Sample␣Seasonal␣Dataset"
,
"cal:hadCalibration": {
        "@id":  "#Calibration.rjtqna",
        "@type": "cal:Calibration",
        "rdfs:label": "Calibration"
,
"cal:withCalibrationMethod": {
        "@id":  "#bias.fdsliz",
        "@type": "BiasCorrection",
        "rdfs:label": "bias"},
"cal:withTrainingData": {       "@id":  "#Dataset.moijnv"
},
"cal:withReferenceData": {      "@id":  "#Dataset.moijnv"
},
"ds:hadCommandCall": [{
        "@id":  "#Calibration.dknwmb",
        "@type": "ds:Command",
        "rdfs:label": "Calibration",
"prov:value": "Calibration"
,
"ds:fromPackage": {
        "@id":  "#modules/Calibration/Calibration.R.ralbwe",
        "@type": "ds:Package",
        "rdfs:label": "modules/Calibration/Calibration.R"}},{   "@id":  "#
              Calibration.dknwmb"
}]},
"ds:hadProject": {
        "@id":  "#Project.fepswo",
        "@type": "ds:Project",
        "rdfs:label": "SEAS5"},
"ds:hadModellingCenter": {
        "@id":  "ds:ECMWF",
        "@type": "ds:DataProvider",
        "rdfs:label": "ECMWF"},
"ds:hadDataProvider": { "@id":  "ds:ECMWF"
}}]}
```

References

[1] Bedia J. (2018). climate4R.climdex: Climate Change Index Calculation for climate4R Data. R package version 0.1.3. http://meteo.unican.es/climate4R.

[2] Brickley D., Guha R.E. (2014). RDF Schema 1.1. W3C Recommendation World Wide Web Consortium. https://www.w3.org/TR/rdf-schema/.

[3] Bronaugh D. (2015). climdex.pcic: PCIC Implementation of Climdex Routines. R Package Version 1.1-6. https://CRAN.R-project.org/package=climdex.pcic.

[4] Candan K.S., Liu H., Suvarna R. (2001). Resource description framework: metadata and its applications. ACM SIGKDD Explorations Newsletter, 3 (1), 6–19.

[5] GitHub. (2023). Sunset: Topic on GitHub. Available at: https://github.com/topics/sunset.

[6] Santander Met Group. (2023). Climate4R: A suite of R packages for climate data access, post-processing, downscaling, and visualization. Available at: https://github.com/SantanderMetGroup/climate4R.

[7] W3C. (2024). PROV-O: The PROV Ontology. World Wide Web Consortium. Available at: https://www.w3.org/TR/prov-o/.

[8] Metaclip. (2023). metaclipR: [Software]. Available at: https://github.com/metaclip/metaclipR.

[9] CRAN. (2023). The Comprehensive R Archive Network: Available R Packages. Available at: https://cran.r-project.org/web/packages.html.

[10] Hassell D., Gregory J., Blower J., Lawrence B.N., Taylor K.E. (2017). A data model of the climate and forecast metadata conventions (cf-1.6) with a software implementation (cf-python v2.1). Geosci. Model Dev. (GMD) 10(12), 4619–4646.

[11] Hewitt C., Mason S., Walland D. (2012). The global framework for climate services. Nat. Clim. Chang. 2(12), 831–832.

[12] Hills D.J., Downs R.R., Duerr R., Goldstein J.C., Parsons M.A., Ramapriyan H.K. (2015). The importance of data set provenance for science. Eos 96.

[13] Hogan A., Harth A., Polleres A. (2009). Scalable authoritative OWL reasoning for the web. International Journal on Semantic Web and Information Systems 5(2).

[14] Iturbide M., Bedia J., Herrera S., Baño-Medina J., Fernández J., Frías M., Manzanas R., San-Martín D., Cimadevilla E., Cofiño A., Gutiérrez J. (2019). The R-based climate4R open framework for reproducible climate data access and post-processing. Environ.

[15] Ma X., Zheng J.G., Goldstein J.C., Zednik S., Fu L., Duggan B., Aulenbach S.M., West P., Tilmes C., Fox P. (2014b). Ontology engineering in provenance enablement for the national climate assessment. Environ. Model. Softw 61, 191–205.

[16] Gutiérrez, J.M., Bedia, J., Iturbide, M., Herrera, S., Manzanas, R., Medina, J.B., Frías, M.D., San-Martín, D., Fernández, J., Cofiño, A. (2018). Climate Research Reproducibility with the Climate4R R-based Framework. In Proceedings of the 8th International Workshop on Climate In-

formatics (CI2018). NCAR Technical Note NCAR/TN-550+PROC, doi: 10.5065/D6BZ64XQ.

[17] Moreau, L., Groth, P., Cheney, J., Lebo, T., Miles, S. (2015). The Rationale of PROV. Web Semantics: Science, Services and Agents on the World Wide Web, 35, 235–257. Elsevier.

[18] Agudetse, V., Rifà, E., Ho, A-C. (2023). Climate Forecast Analysis Hands-On Tutorial: R Tools. BSC Training Course 2023.