Provenance integration in R seasonal-to-decadal verification workflow

Final Degree Thesis

PHYSICS ENGINEERING

Albert Puiggròs Figueras

Director: Victòria Agudetse Roures
Tutor: Gladys Urtera
Barcelona, 2024

Polytechnic University of Catalonia
Bachelor's Degree in Physics Engineering
Barcelona School of Telecommunications Engineering
C/Jordi Girona, 1-3
08034 Barcelona
Barcelona Supercommputing Center - Centro Nacional de Supercomputación
Bachelor's Degree in Physical Engineering
C/Jordi Girona, 1-3
08034 Barcelona

1 Acknowledgements

I would like to express my deepest appreciation to my tutor, Victòria Agudetse Roures (Barcelona Supercomputing Center-Centro Nacional de Supercomputación), for her guidance and unwavering support during this journey. Victòria, as my BSC tutor, played an integral role, providing consistent assistance throughout the entire Bachelor' s Thesis, addressing every issue I encountered and offering help whenever needed. I would also like to extend my deepest gratitude to my second tutor, Gladys Utrera, from the Universitat Politècnica de Barcelona (UPC).

I am extremely grateful to Bruno P. Kinoshita (BSC) for his guidance and unconditional support. Bruno, with his high expertise, provided assistance and help since day one, contributing significantly to the success of the project.

Special thanks are due to Joaquín Bedia, professor at the Dept. of Applied Mathematics and Computing Science, University of Cantabria (UC) and researcher at the Santander Meteorology Group (UC-CSIC) and Daniel San Martín, Predictia Intelligent Data Solutions SL. Their collaboration helped me understand METACLIP and facilitated our collaboration in adding new terms to their ontology.

I am also thankful to Núria Pérez-Zanón (BSC) for her feedback and to Eren Duzenli (BSC) for assisting with the implementation of metaclipR.

I would like to express my gratitude to my supervisors at the BSC, including Pierre-Antoine Bretonnière, Miguel Castrillo, and Francesco Benincasa, for their valuable contributions to the project and for the unique opportunity to be part of the BSC for the last four months.

Lastly, I would like to thank Francisco J. Doblas-Reyes (BSC) for spearheading the idea of implementing METACLIP in SUNSET.

2 Resum

En l'àmbit de les ciències de la terra i, particularment, en l'àmbit climàtic, la integritat i reproducibilitat de les dades és fonamental. Per poder assegurar la fiabilitat dels resultats, cal gestionar de manera efectiva la procedència de les dades -àrea d'estudi de software coneguda com *data provenance*. Aquest treball es centra en la implementació de *data provenance* a SUNSET (SUbseasoNal to decadal climate forecast post-processing and asSEssmenT suite) basat en el marc establert per METACLIP (METAdata for CLImate Products). L'objectiu principal és establir un sistema complet de seguiment i gestió de *data provenance* en la verificació de productes climàtics. Amb aquesta implementació busquem millorar el rastreig o traçabilitat dels productes climàtics generats, proporcionant una història clara i transparent de les transformacions i canvis que han patitit al llarg de la seva vida útil. S'espera que aquesta implementació s'estengui a tot el flux de treball de verificació climàtica de SUNSET, adequant-lo per extreure amb precisió tota la informació sobre la procedència de les dades en base al llenguatge establert per METACLIP.

Aquest projecte final de grau d' Enginyeria Física s'ha desenvolupat com a part d'un pràcticum al Barcelona Supercommputing Center (BSC) de març a juny de 2024, juntament amb un projecte preliminar realitzat entre desembre de 2023 i febrer de 2024 com a lliurament per l'assignatura de 4t any CPIA (Computing Programming and its Applications). El projecte inclou la capacitat de concebre, dissenyar, implementar i operar sistemes complexos en un entorn de programació. També, utilitza un ampli espectre de llenguatges de programació, focalitzant-se en R. En la seva totalitat, el projecte combina diferents camps d'estudi per crear un sistema robust per a la resolució de problemes complexos d'enginyeria, facilitant el desenvolupament de solucions innovadores i pràctiques en un camp de ràpida evolució com el de l' enginyeria física.

Keywords: procedència, METACLIP, SUNSET, gràfic de procedència

3 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 implementation of SUNSET (SUbseasoNal to decadal climate forecast post-processing and asSEssmenT suite) based on the METACLIP (METAdata for CLImate Products) framework, aiming to establish a complete system for tracking and managing data provenance in climate product verification. By implementing this, 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 extend comprehensively over the entire SUNSET climate verification workflow, making it suitable for accurately extracting all data provenance defined by METACLIP.

This Physics Engineering final degree project was developed as part of a practicum at the Barcelona Supercomputing Center (BSC) from March to June 2024, along with a preliminary project conducted between December 2023 and February 2024 as a deliverable for the Computing Programming and its Applications (CPIA) 4th year course. Moreover, the project encompasses the ability to conceive, design, implement, and operate complex systems within a programming environment. The project employs a broad spectrum of programming languages, with a primary focus on R. This approach combines different fields of study to create a robust system for solving complex engineering problems, facilitating the development of innovative and practical solutions in the rapidly evolving field of physics engineering.

Keywords: provenance, METACLIP, SUNSET, provenance graph

4 Resumen

En el campo de las ciencias de la tierra, la integridad y la reproducibilidad de los datos son fundamentales. Asegurar resultados fiables es crucial para gestionar eficazmente la procedencia de los datos -área dentro del desarrollo de software y tecnologías semánticas conocido como *data provenance*-. Este trabajo se centra en la implementación de *data provenance* en SUNSET (SUbseasoNal to decadal climate forecast post-processing and asSEssmenT suite) basado en el marco METACLIP (METAdata for CLImate Products), con el objetivo de establecer un sistema completo para el seguimiento y la gestión de la procedencia de datos en la verificación de productos climáticos. El proyecto busca mejorar el rastreo o trazabilidad de los datos climáticos generados tras el post-procesado, proporcionando una historia clara y transparente de las transformaciones y cambios sufridos a lo largo de su vida útil. La intención es extender la implementación a todo el flujo de trabajo de verificacián climática de SUNSET, adaptándolo para extraer con precisión toda la información sobre la procedencia de los datos en base a la estructura conceptual establecida por METACLIP.

Este proyecto de final de grado de Ingeniería Física se ha desarrollado como parte de un prácticum en el Barcelona Supercommputing Center (BSC) de marzo a junio de 2024, juntamente con un proyecto preliminar realizado entre diciembre de 2023 y febrero de 2024 como entregable de la asignatura de 4to año CPIA (Computier Programming and its Applications). El proyecto incluye la capacidad de concebir, diseñar, implementar y operar sistemas complejos en un entorno de programación. También, utiliza un amplio espectro de lenguajes de programación, principalmente R. En su totalidad, el proyecto combina diferentes áreas de estudio para crear un sistema robusto para la resolución de problemas complejos de ingeniería, facilitando el desarrollo de soluciones innovadoras y practicas en un campo de rápida evolución como el de la ingeniería física.

Keywords: procedencia, METACLIP, SUNSET, gráfico de procedencia

Contents

List of Figures

Listings

List of Tables

5 Introduction

Provenance, as defined by the Provenance Working Group in 2013 [1], 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 usage significantly increases the value of both the final product and original data by making the influence within them clear and traceable. Consequently, a recognized metadata scheme is vital when intending to extract provenance from data product generation. Also, official standards and vocabularies play a key role in properly describing the comprehensive metadata and schemes representing the lifecycle of data.

In times 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 an international standard for data provenance. Specific to the climate scene, some initiatives such the Climate and Forecast Metadata Convention (CF) [2] [3] 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)[4] as a solution for identifying, extracting, linking, and assembling the information needed to fully characterize a climate product. It is based on RDF (Resource Data Framework) and focuses on semantic description of climate products of all types, so that each of them and its provenance is inseparable and jointly delivered. This report analyzes the possibility of a suitable integration of METACLIP into the SUNSET (SUbseasoNal to decadal climate forecast post-processing and asSEmenT suite) workflow [5], enhancing provenance tracking. SUNSET is an advanced R-based modular tool developed in the BSC Earth Sciences department designed for comprehensive forecast verification. It handles data from several 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 data produced.

6 Theoretical Foundations and State of the Art

In this section, we delve into the two main foundations of this project: climate data -modelling and processing- and data provenance. Climate data modelling and post-processing involve the use of advanced computational techniques to simulate and analyze climate phenomena. We present an introduction to this field, which sets the stage for introducing SUNSET, focusing on its structure and functionalities. Then, we introduce data provenance as the detailed record of data origin. After analyzing several frameworks that aim to implement this concept, we explore METACLIP, a framework that establishes data provenance standards for climate product assessment.

6.1 Climate Modelling

6.1.1 Introduction to Climate Modelling

A forecast is a prediction or estimation of future events based on data and comprehension of past and present trends. In several fields such as meteorology, economics or finances, forecast assessment plays a crucial role in decision making [6] [7]. In earth sciences, for example, weather and climate forecasts aim to predict various atmospheric conditions ranging from hours to decades. On the one hand, weather refers to short-term forecasts ranging from hours to weeks that predict changes in the lower atmospheric conditions, usually over a reduced region. On the other hand, climate describes long-term average weather conditions and patterns over extended periods, typically decades, and across larger areas. Then, weather forecasts are required for immediate or daily decision-making, while climate forecasts help planning for future events by anticipating long-term changes and potential impacts [6]. These kind of forecasts are particularly based on complex models that analyze data from instrumental sources such as satellites or weather stations, allowing the prediction of future atmospheric conditions with a varying degree of accuracy. Additionally, climate models utilise a broad spectrum of variables including temperature, humidity, wind speed, atmospheric pressure, and precipitation patterns [8] [9].

Similarly, a hindcast is a prediction of past events, also based on historical data. The primary goal of hindcasting is to evaluate the performance of the models, comparing the simulated outcomes with observed data from the past. By determining how the model's predictions adjust to past measurements, developers can gain insight into the models accuracy and reliability [10]. Consequently, validating climate models and evaluating their ability to accurately predict or replicate historical climate trends is crucial for the development of climate science.

There are a great number of system and model types that generate climate forecasts and hindcasts, each of which use different methods to compute the predictions. From the Climate Modelling Primer [11]: "Climate models are, likewise, idealized representations of a complicated and complex reality through which our understanding of the climate has significantly expanded. All models involve some ignoring, distorting and approximating, but gradually they allow us to build understanding of the system being modeled. A child's Lego construction typically contains the essential elements of the real objects, improves with attention to detail, helps them understand the real world, but is never confused with the real thing" [12] [13].

Moreover, climate models work with several types of climate variables, which can be classified

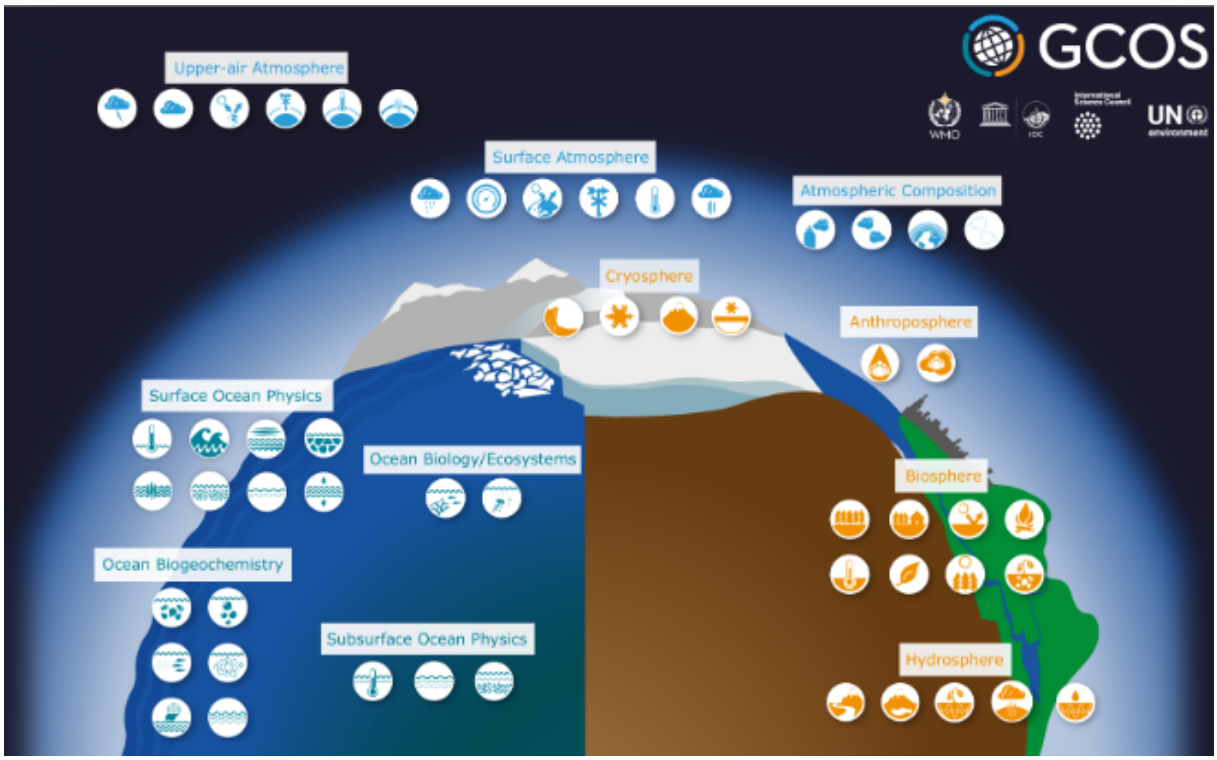

Figure 1: Representation of the different climate variables defined by GCOS[13].

into atmospheric, land and ocean variables [12] [13]. Defined by GCOS (Global Climate Observing System) [13], a climate variable is a physical, chemical or biological variable or group of related variables that critically contributes to the characterization of the Earth's climate. Then, climate variable datasets provide the information required to understand and predict climate evolution, to guide mitigation and adaptation measures, to identify the causes of certain climate events and to support climate services [12] [14].

Atmospheric variables cover the upper-air (such as clouds and upper-air temperature), the surface atmosphere (such as precipitation and surface temperature), and atmospheric composition (including aerosols and greenhouse gasses). Land climate variables describe the conditions of the Cryosphere (ice sheets and glaciers), Anthroposphere (human-related water and gas emissions), Biosphere (plant biomass and land cover), and Hydrosphere (rivers and soil moisture). Finally, ocean climate variables address ocean surface and subsurface conditions, marine biology, and ocean chemistry, including sea surface temperature, marine habitats, and nutrients [15][13].

6.1.2 Climate Forecast Verification

Forecast verification stands as a critical process in the field of climate science and meteorology. It can be defined as the set of methodologies which assess the quality of a forecast [16] [17], involving the evaluation of forecast quality based on the comparison with observed outcomes and reliable estimates of the conditions. It is essential to evaluate the reliability and accuracy of climate predictions regarding future states of weather patterns, climate trends and other phe-

nomena [12]. Whether through qualitative assessments, which focuses on visual alignment of predictions with reality, or quantitative analysis, which numerically assesses forecast accuracy, climate forecast verification intends to clarify the nature and magnitude of forecast errors. This kind of process aims to provide a deeper understanding of the complex dynamics of atmospheric and climate systems while aiding in refining forecasting methodologies[18][19]. Particularly, in the field of Climate Services, the primary aim of forecast verifications is to assess and support decision-makers by providing information on the reliability of these forecasts for particular locations and times, thereby allowing them to make informed decisions based on these predictions [16][17].

Understanding what constitutes a good forecast is crucial for evaluating the effectiveness of predictions. Allan Murphy, in the essay *What is a good forecast? An essay on the nature of goodness in weather forecasting* [20] , distinguishes between three types of elements that determine the "goodness" of a forecast: consistency, quality and value. Consistency refers to "the degree to which the forecast corresponds to the forecaster's best judgment about the situation, based upon his/her knowledge base", quality to "the degree to which the forecast corresponds to what actually happened", and value to "the degree to which the forecast helps the a decision maker to realize some incremental economic and/or other benefit" [20].

Regarding forecast verification, quality stands as the primary element to determine the "goodness" of a forecast. Murphy describes nine aspects that contribute to the quality of forecasting: bias, association, skill, reliability, resolution, sharpness, discrimination and uncertainty [20]. In this section, we delve into accuracy and skill, since they have traditionally played a crucial role in evaluating the effectiveness of a forecasting system. On the one hand, accuracy measures the agreement between the forecast and the known observations. Then, the difference between both is the error and, the lower the error, the greater the accuracy. On the other hand, skill measures the relative accuracy between the forecast and some reference forecast, defined as an unskilled or simplistic prediction such as those based on random chance. Consequently, skill quantifies the improvement in the accuracy due to the degree of sophistication or inherent capability of the forecast system. Skill takes into account the fact that some weather predictions may be accurate simply because the weather is easier to predict, rather than as a result of improvements in prediction methods and techniques [20][21].

6.1.3 Climate Data Processing

Climate data processing plays a key role in verifying climate model predictions and extracting meaningful insight. Various operations are used on climate data to ensure that both the long-term trends and short-term meteorological predictions are spatially and temporally coherent. This means that weather and long-term predictions are consistent and accurate across different locations and time instants. For example, if we want to compare a forecast with past observations we require the data to be spatially coherent. This is, a spatially coherent forecast would correctly represent a storm crossing a certain region -to be compared with- rather than predicting it in a completely different location. Some methods that account for coherence are interpolation and calibration. Moreover, other operations are used to directly extract information from the data, such as computing climatological indices or statistical parameters. This section briefly describes key climate data processing operations such as interpolation, anomaly computation, climatology analysis, index calculation, aggregation, downscaling, and probability measure [22]

[23].

Interpolation

Interpolation is a critical operation used on climate data processing. Given the fact that observational data is sparsely and irregularly distributed during retrieval, interpolation is used to help in data completeness and data gap filling. By estimating missing values based on the available information, this transformation ensures a comprehensive dataset for analysis. Furthermore, it is adopted to ensure a proper comparison of two distinct datasets. To achieve this, first, observations are interpolated to create a regularly-spaced reanalysis data. Afterwards, all datasets are interpolated to a common grid so they all have the same spatial extent. Generally used interpolation methods such as inverse distance weighting or bilinear interpolation help in maintaining spatial and temporal integrity and continuity [24][25].

In a review largely informed by Dobesch et al. (2007)[26] and Tveito et al. (2006)[27], interpolation techniques are categorized into three primary groups: deterministic, probabilistic, and other specialized methods[24]. Deterministic methods generate a continuous surface entirely based on the geometric properties of observation points. Probabilistic methods apply theories of randomness to produce one possible interpolated field out of many potential statistical realizations. Also, probabilistic methods allow to include the variance in the interpolation process and the calculation of statistical parameters to evaluate the significance of the predicted values. The category labeled "other methods" includes techniques specifically designed for meteorological applications, often integrating deterministic and probabilistic approaches[26][27].

Calibration

Calibration is used to refine the model's output, correcting the systematic errors (biases) it contains. For example, these could be errors such as the amount of seasonal rainfall or temperatures being consistently too high or too low [28]. Biases in climate models can be caused by several factors: limited spatial resolution, simplified thermodynamic modeling, incomplete understanding of the global climate system, and other factors. Consequently, these errors can lead to unrealistic and inaccurate forecasts [28]. To address the inherent biases, bias correction techniques adjust the model outputs to correct biases, either systematic or of other nature, by comparing model predictions with observed data. Common bias correction methods include quantile mapping, regression-based approaches, and empirical methods. These techniques ensure that the statistical properties, such as means and extremes, are accurately represented in the model projections, thereby maintaining the integrity of future climate projections [29] [30].

Climatology

Climatological data analysis refers to the way historical patterns are processed to analyze the past recorded data through statistical methodologies. Commonly calculated as the mean over a certain region and temporal period, climatology provides recurring patterns of natural weather events of interest, which can be determined with seasonal, annual, and multiyear historical data [14][23]. Furthermore, climatology data is a key input in climate data processing, providing meaningful patterns to analyze and compare with current climate conditions and predict future variation patterns. That is, climatological analysis not only provides valuable context of past climate behaviors, but also allows researchers to broaden their understanding of present climate conditions and predicting future changes, aiding in the development of effective adaptation and

mitigation strategies.

The climatology of a certain variable is commonly the variable's average over a period of time. Climatologies can be calculated based on different definitions. For example, we can compute the climatological average as the mean of monthly values over a specific period. Specifically, we can compute "climatological normals", monthly averages for a prolonged period of at least 30 consecutive years [14, 31].

Anomaly Computation

Anomaly computation is used to calculate deviations from expected climate patterns. By comparing data to climatology anomalies can be detected, indicating extreme climate conditions or, in other words, deviations from the mean [32]. These anomalies provide valuable information about the variability and potential impacts of the evolution of climate, helping to understand the underlying mechanisms driving environmental shifts. Then, the anomalies or deviations from the mean value of a particular variable are computed by subtracting the long-term climatology from the observed or predicted data [32][14].

Downscaling

Downscaling involves the transformation of coarse-resolution predictions produced by climate models at relatively large scales to finer scales of either space or time. Coarse-resolution global climate models cannot capture local atmospheric phenomena, which means that predictions cannot assess the climatological trends at regional or local scales. For better regional modeling, high-resolution models are employed, and techniques such as statistical downscaling are used to bridge the gap, providing valuable information on regional or local climates [33].

Commonly, there exist two different approaches to generate high-resolution climate data from larger-scale inputs: statistical downscaling and dynamical downscaling. Statistical downscaling uses several statistical techniques to determine relations between the large-scale data provided by climate models and observed local measurements. On the other hand, dynamical downscaling utilises high-resolution simulations to dynamically interpolate the climate variables from large-scale resolution to regional or local scales [34].

Index calculations

Index calculation is a crucial step in climate data processing, involving the computation of distinctive climate variables. These variables, known as climate indices, describe climate patterns and provide a foundation for understanding the local climate of a region [35]. Namely, climate indices are diagnostic values used to characterize an aspect of a geophysical system obtained by computing statistical values, such as averages and correlations, over a region and time period. Notable examples include the Arctic Oscillation Index and the Pacific Decadal Oscillation Index [36].

Aggregation

Aggregation is the process of summarizing large datasets into manageable subsets, facilitating trend analysis and pattern recognition. By aggregating data over different spatial or temporal scales, researchers can discern overarching climate trends, identify regional variations, and assess the spatial distribution of climate phenomena. Aggregation techniques enable researchers

to extract meaningful insights from complex climate datasets, aiding in the development of informed climate policies and adaptation strategies [37].

(Climate) Probability Assessment and Verification Methodology

Probability assessment involves the calculation of probabilities to determine the likelihood of specific weather events or climatic outcomes. By analyzing historical data and climate model projections, researchers can quantify the probability of weather events, such as heatwaves, storms, or floods. Probability assessments provide valuable information for risk assessment, emergency readiness, and decision-making in climate-sensitive sectors such as agriculture, water resource management, and infrastructure planning. These common operations in climate data processing are essential for verifying data integrity, extracting meaningful insights, and understanding the complex dynamics of Earth's climate system [38].

Climate verification methods are essential for assessing the performance of weather and climate models. Among several possible approaches, statistical methods play a crucial role in evaluating the accuracy and skill of forecasts [20] [39]. Forecast verification usually involves the use of metrics such as skill scores. Metrics provide information by comparing predicted values with observed outcomes, giving valuable information about the forecasts accuracy, reliability and sharpness. Particularly, skill scores quantify the percentage of improvement of the forecast over the reference or, in other words, the relative accuracy, with positive values indicating a better performance [40][41]. Several key metrics are employed in climate verification to allow for a comprehensive evaluation of prediction models. Moreover, metrics generate understandable scalar values from complex forecast data, providing clear indications of the forecast performance. Examples include Brier Score, which measures the mean squared error of forecast and Ranked Probability Score, assessing the accuracy of forecast for ordered categorical events [42][43].

6.1.4 SUNSET Workflow

SUNSET (SUbseasoNal to decadal climate forecast post-processIng and asSEssmenT suite) [44] [5] is a tool for subseasonal to seasonal to decadal forecast verification workflows, developed in the Barcelona Supercomputing Center (BSC) in collaboration between the Computational Earth Science (CES) and Earth System Services (ESS) groups. The main users are climate scientists who require a tool to evaluate models in order to provide climate services for end users. The SUNSET framework offers a modular processing of climate data products combining user-defined recipes, climate data and scripts. Its modularized structure, where each module constitutes a separate part of the code that performs a specific task, allows it to skip or reorder parts of the workflow when executed. Then, users can select any of the modules and the order in which they will be executed depending on their needs. Alongside the script that details which modules will execute and their order, the user needs to specify in the recipe what operations and methods will each module compute. The recipe is a template that determines how the execution will take place and which operations will the modules apply. It specifies datasets, forecast horizon time period, skill metrics to compute and many other parameters. Once the recipe is loaded, the data requested in the recipe is encapsulated in a list of objects which contains observational, hindcast and forecast data. Moreover, SUNSET includes several R packages developed in the BSC for climate data processing such as startR[45], CSTools[46]

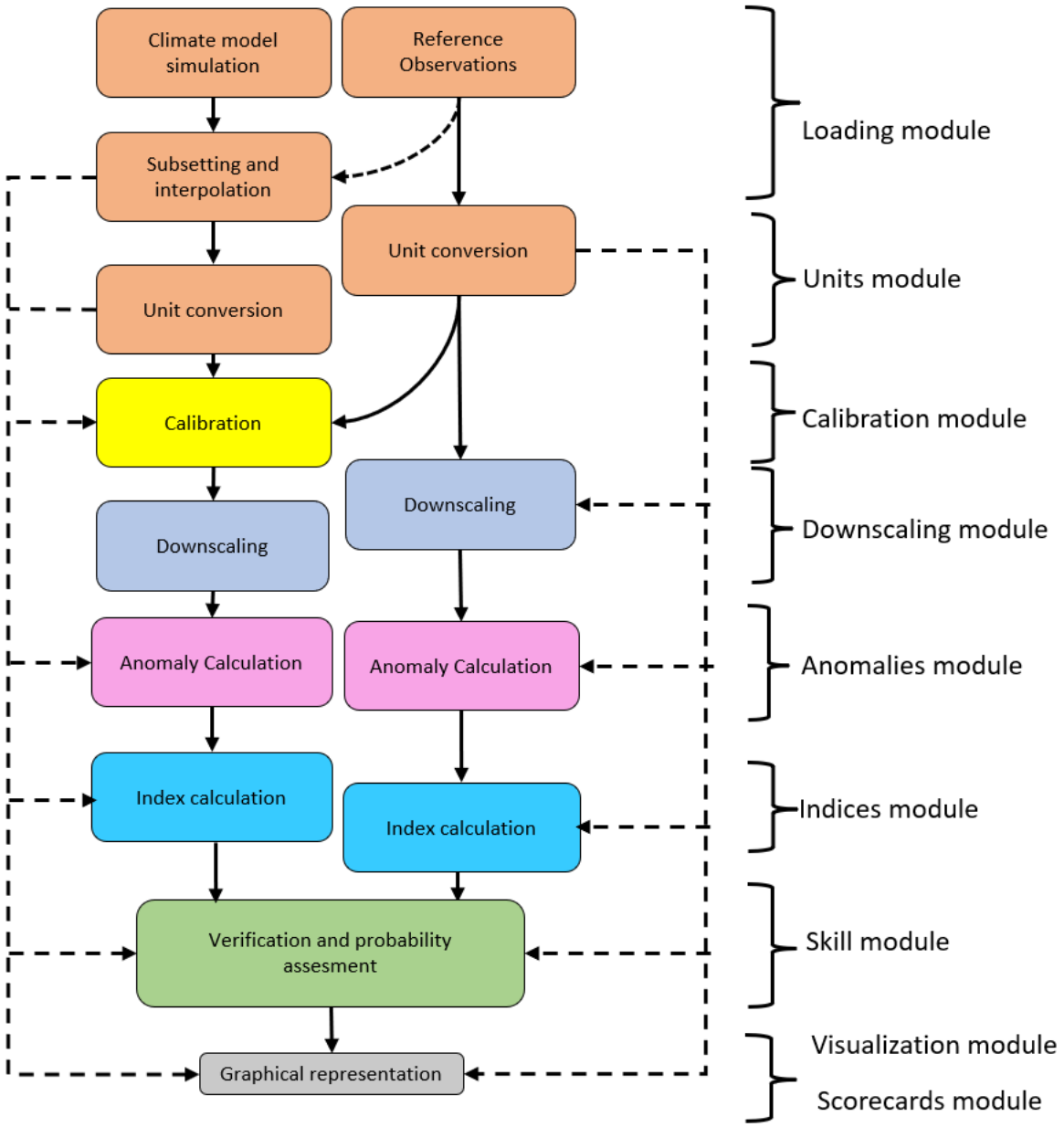

Figure 2: Example structure of SUNSET: Its modularized design allows users to dynamically execute the modules in different orders, enabling a wide range of post-processing operations.

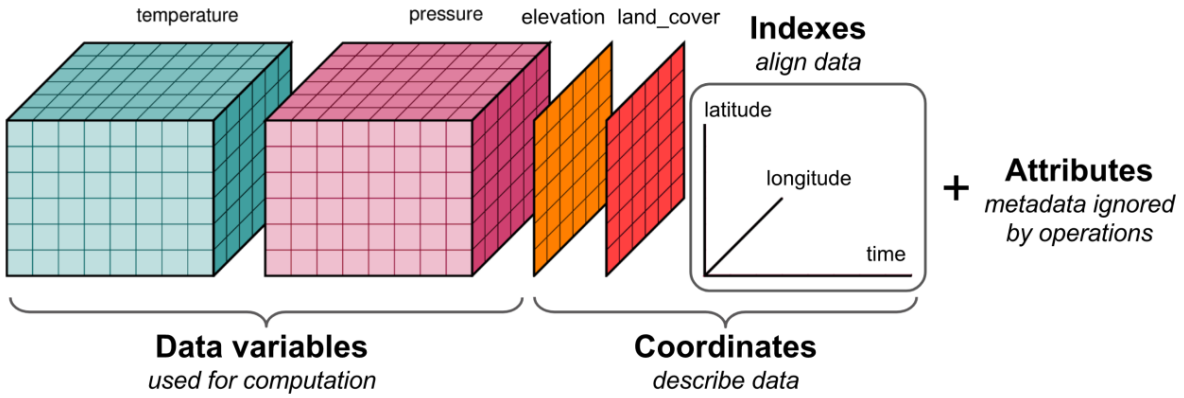

Figure 3: `s2dv_cube` object description.

and s2dv_cube[47].

By means of the CSTools R package [46], each dataset is stored as an `s2dv_cube`, an R object that contains data and metadata [47]. Its first component, labeled as `data`, contains a numerical multidimensional array indexed over several dimensions such as time, climate variables, spatial coordinates, and temporal coordinates. The elements that list the length and size of each dimension in `data` are enclosed in the `dims` element. The third main class, established as `coords`, provides detailed information of each dimension, including metadata such as standard names, units, axis types, and other attributes. Finally, the attributes, `attr` element includes information regarding additional metadata, providing the necessary elements to ensure a proper description and context of the data. Overall, the `s2dv_cube` object allows for an efficient handling and analysis of complex multidimensional climate data by means of the integration of data and metadata in a single object, as is shown in Figure 3.

In the following paragraphs, each SUNSET module is briefly explained.

Loading Module

The Loading Module starts the process by retrieving the data requested in the recipe from the directory `/esarchive/`, an NFS (Network Files System) mount used within the Earth Science department. Then, the data is processed, interpolated into the desired grid, and converted into an `s2dv_cube` object. That is, the Loading Module is capable of interpolating the data while loading it. The dataset subset selected and the methods used must specified in the recipe. An example of the portion of the recipe required to configure the Loading module is provided in the script below.

The interpolation method can be established in the recipe under `Regrid:method`. Only those methods within CDO (Climate Data Operators) [48] are available. In addition, the target grid is specified in the `Regrid` element of the recipe. The regridding can be one of the following options: `to_system`, where observations are interpolated to the system grid; `to_reference`, where the hindcast and forecast are interpolated to the reference grid; `none`, where no interpolation is performed; and a CDO-accepted grid, where both the system and observations are interpolated to a third grid.

```
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)
   # units: desired data units for each variable. Only available for
      temperature,
   # precipitation, and pressure variables.
   - {name: 'tas', freq: 'monthly_mean', units: 'C'}
 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)
```

Listing 1: SUNSET Recipe template for dataset subsetting and regridding specifications

Units Module

The Units module ensures unit consistency between system and reference datasets. It must be used after the Loading module and currently supports unit transformation for temperature, precipitation, and pressure variables. Its main function is `Units()`.

Calibration Module

This first step can be followed by the Calibration Module, which performs a bias correction to the loaded data. It requires the output of the Loading Module and the recipe as input. The bias correction method selected in the recipe is applied to both the hindcast and forecast utilising the observations as reference. The calibration method is stated in the recipe

in `Workflow:Calibration:method` and the user can only request one per recipe. Subsequently, the main function `Calibration()` returns the calibrated data and its metadata as an `s2dv_cube` object.

```
Calibration:
   method: mse_min # Calibration method. (Mandatory, str)
   save: 'all' # Options: 'all', 'none', 'exp_only', 'fcst_only' (Mandatory
      , str)
```

Listing 2: SUNSET Recipe template for calibration specifications

Anomalies Module

The Anomalies Module calculates data anomalies relative to the climatological mean, with or without cross-validation depending on the specifications in the recipe. This module utilises the CSTools function `CST_Anomaly()`. The main function of the module, `Anomalies()`, returns a list of `s2dv_cube` objects containing anomalies for the hindcast, forecast and observations, as well as the original hindcast and observation full fields. If cross-validation is requested leave-one-out cross-validation [49] [50] will be applied. For SUNSET, cross-validation is only available when observations and hindcast share the same grid.

```
Anomalies:
   compute: yes # 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 if 'compute: yes', str)
```

Listing 3: SUNSET Recipe template for anomalies specifications

Downscaling Module

The Downscaling Module conducts downscaling operations on the anomalies utilising functions in the CSDownscale package [51]. It takes the output from the Anomalies module and the recipe as input. The module applies the specified downscaling method to the hindcast, taking the observations anomalies as the reference. Finally, it returns the downscaled data and metadata in `s2dv_cube` form. The main function `Downscaling()` outputs a list containing the downscaled hindcast and forecast. In the latest released version of SUNSET (2.0.0) it is not possible to apply downscaling methods to the forecast.

The methods available need to be defined in `Workflow:Downscaling:type`. Also, the method can be further specified through `method` options. Other options in the recipe are `Workflow:Downscaling: target_grid`, which indicates the target grid to which the system will be downscaled, and `Workflow:Downscaling:size`, only applicable when the downscaling type is `analogs` and the dataset has `daily` or `daily_mean frequency`.

```
Downscaling:
  # Assumption 1: leave-one-out cross-validation is always applied
  # Assumption 2: for analogs, we select the best analog (minimum distance)
```

```
  type: intbc # mandatory, 'none', 'int', 'intbc', 'intlr', 'analogs', '
     logreg'.
  int_method: conservative # regridding method accepted by CDO.
  bc_method: bias # If type intbc. Options: 'bias', 'calibration', '
     quantile_mapping', 'qm', 'evmos', 'mse_min', 'crps_min', 'rpc-based'.
  lr_method: # If type intlr. Options: 'basic', 'large_scale', '9nn'
  log_reg_method: # If type logreg. Options: 'ens_mean', 'ens_mean_sd', '
     sorted_members'
  target_grid: /esarchive/recon/ecmwf/era5/monthly_mean/tas_f1h/tas_200002.
     nc # nc file or grid accepted by CDO
  nanalogs: # If type analgs. Number of analogs to be searched
  save: 'all' # 'all'/'none'/'exp_only'
```

Listing 4: SUNSET Recipe template for downscaling specifications

Indices Module

The Indices Module aggregates the hindcast and reference data to calculate climatological indices such as the North Atlantic Oscillation (NAO) or El Niño indices. Upon the execution of the main function, `Indices()`, the module returns the hindcast and observations in `s2dv_cube` form for each requested index. During the process the latitude and longitude dimensions of the original arrays are aggregated into a single `region` dimension. The available indices are shown in Table 1.

```
Indices:
  ## Indices available: NAO, Nino1+2, Nino3, Nino3.4, Nino4.
  ## Each index can only be computed if its area is within the selected
     region.
  # obsproj: NAO computation method (see s2dv::NAO()) Default is yes/true. (
     Optional, bool)
  # save: What to save. Options: 'all'/'none'. Default is 'all'.
  # plot_ts: Generate time series plot? Default is yes/true. (Optional, bool
     )
  # plot_sp: Generate spatial pattern plot? Default is yes/true. (Optional,
     bool)
  # alpha: Significance threshold. Default value is 0.05 (Optional, numeric)
  NAO: {obsproj: yes, save: 'all', plot_ts: yes, plot_sp: yes}
  Nino1+2: {save: 'all', plot_ts: yes, plot_sp: yes, alpha: 0.05}
  Nino3: {save: 'all', plot_ts: yes, plot_sp: yes, alpha: 0.05}
  Nino3.4: {save: 'all', plot_ts: yes, plot_sp: yes, alpha: 0.05}
  Nino4: {save: 'all', plot_ts: yes, plot_sp: yes, alpha: 0.05}
```

Listing 5: SUNSET Recipe template for downscaling specifications

Skill Module

The Skill Module computes the metrics that indicate the quality of the forecast. The main function, `Skill()`, computes the verification metrics specified in the recipe under `Workflow: Verification:metric`, as shown in Listing 6. Details about the existing metrics that can be currently requested can be found in the SUNSET Wiki [44]. The second main function, `Probabilities()`, computes the probability of certain thresholds specified in the recipe as `Workflow: Probabilities: percentiles`, `Workflow: Probabilities: probs`, or `Workfl`

Index	Recipe name
North Atlantic Oscillation	NAO
Niño 1+2	Nino1+2
Niño 3	Nino3
Niño 3.4	Nino3.4
Niño 4	Nino4

Table 1: Climate indices available in SUNSET

ow: Probabilities:probs _fcst.

```
Skill:
   metric: RPSS CRPSS # Skill metrics separated by spaces or commas. (
      Mandatory, str)
   save: 'all' # Options: 'all', 'none' (Mandatory, str)
```

Listing 6: SUNSET Recipe template for Skill specifications

```
Probabilities:
   percentiles: [[1/3, 2/3], [1/10, 9/10]] # Thresholds for quantiles and
      probability categories. Each set of thresholds should be enclosed
      within brackets.
   save: 'percentiles_only' # Options: 'all', 'none', 'bins_only', '
      percentiles_only' (Mandatory, str)
```

Listing 7: SUNSET Recipe template for probability assessment specifications

Visualization and Scorecards Module

Both the Visualization and Scorecards modules provide tools to visualize the data loaded and processed using the previous modules, either with basic plots or in a scorecard format. The structure of the plots and scorecards depends on the specifications requested in the recipe. More information can be found in the SUNSET Wiki [44].

Saving Module

The outputs from the workflow are organized and saved in a folder, previously defined in the recipe, by means of the Saving Module as a NetCDF file. 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.

To facilitate a comprehensive analysis, SUNSET developers utilise GitLab, a platform for software development and collaboration [52]. 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 us 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 the same project without altering the main code changes, using separate branches which can later be merged seamlessly.

6.2 Data Provenance

This section aims to introduce the concept of data provenance and analyze the tools utilised for its generation. Data provenance can be understood as a documented record that describes the origin and transformations of a piece of data. This concept aims to achieve a detailed record of the data creation, modification, and movement between various systems, offering transparency and accountability. According to Werder et al. (2022) [53], establishing data provenance is vital for responsible software systems, as it promotes transparency, reproducibility, and trustworthiness. Furthermore, data provenance plays a pivotal role in data preservation and curation. It ensures that the complete history and relationships between datasets is documented, providing researchers and organizations with the necessary elements to verify data authenticity and consistency [54].

Viglas (2013) [55] explored the relationship between data provenance and trust, noticing that provenance could contribute significantly to the trustworthiness of data. Organizations can better user confidence in their data systems by keeping a detailed record of data origins and transformations. This helps ensure data integrity, while also improving decision-making by clarifying data quality and lineage. Overall, as data systems become more complex and its origin harder to identify, maintaining data provenance is crucial for responsible data management and fostering a culture of trust and integrity.

Several initiatives aim to integrate the principles of data provenance through a structured and organized approach, enhancing its effectiveness and reliability. Generating data provenance requires a specific language that defines and describes elements. This section reviews the provenance frameworks driven by the W3C (World Wide Web Consortium) in the Semantic Web based on formal standards such as RDF (Resource Description Framework) [56], OWL (Web Ontology Language)[57] and PROV (W3C provenance)[58]. The W3C is an international community that develops open standards to ensure the long-term growth of the Web and is responsible for the creation and maintenance of these languages. Moreover, the Semantic Web is an extension of the current web promoted and standardized by the W3C. By means of these standards, it aims to achieve a more efficient data integration and enhanced search capabilities. In addition, RDF is the foundation of the Semantic Web establishing the most basic definitions and relationships. Building upon it, the Web Ontology Language (OWL) and PROV allow for the representation of more complex relations and metadata definitions.

6.2.1 Semantic Web and Ontologies

The Semantic Web [59] is an extension of the current web proposed by the W3C that attempts to make machine-readable the vast majority of information available in the Web [60]. It is an extension of the World Wide Web established by the W3C standards [61]. Its intention is to define information in an organized and structured manner to guarantee data to be shared and rescued across applications and community boundaries, allowing it to be automatically processed by machines. Then, a common framework is necessary to increase the ability of optimizing the consumption and trustworthiness of data.

The Semantic Web uses as a unique identifier the Internationalized Resource Identifier (IRI)[62], an internet protocol standard with format of string of characters used to recognize resources on the internet. Defined by the Internet Engineering Task Force (IETF), IRI enables for a clear, unambiguous identification of resources, easing the linking and integration of data from distinct sources. Additionally, IRI builds on the Uniform Resource Identifier (URI), a unique character string that identifies a resource, considerably expanding its permitted characters [63]. Overall, IRI is the standard way to name and reference objects within semantic web frameworks [64].

Beyond the use of IRIs, the Semantic Web allows information elements or concepts to be classified and associated, making it possible to infer information from the element's relationships and categorization. This is achieved by means of ontologies[1]. Ontologies are collections of terms, definitions, and concepts, either abstract or physical-related, that are unequivocally interconnected and create a simple model of a particular area of knowledge, listing types of objects, how they relate to each other, and rules about them [65] [66]. An ontology, according to OWL standards [67], must include Classes, which represent entities within the specific domain of knowledge, Properties or Relations, describing relationships between elements, Individuals, which are specific elements that relate to a particular concept, and Axioms or Constraints, which define logical statements that build a set of rules and constraints within the entities of the ontology [57]. Consequently, when a particular piece of information is classified within an ontology, for example as a Class, all information can be automatically inferred based on the defined relationships and constraints. This inference capability is crucial in data provenance, providing richer and more meaningful data integration and retrieval. Lastly, ontologies can be written in formal languages that support the definition of data and their relationships or constraints such as RDF or OWL [68] [69].

6.2.2 RDF (Resource Data Framework)

In the first place, the Semantic Web relies on RDF [56] (Resource Description Framework) to structure data. RDF, developed by the 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 (rdf:subject), predicate (rdf:predicate), and object (rdf:object), as shown in Figure 4. Each part of the triple is uniquely identifiable by an IRI. Then, RDF graphs are a set of triples, and the way RDF files are represented. Moreover, the RDF framework is strucutred and defined by the RDF vocabulary [56], a set of IRIs utilised to build RDF graphs, providing additional definitions and relations. This vocabulary is primarily based on RDF [70], RDFS [71], and XSD [72] [2]. Overall, due to its versatility, RDF has been widely adopted across many fields, from geospatial features to music products [4], enabling the linking and interconnecting data in a human and machine readable language [73] [56].

RDF is well suited for an easier and more efficient use of the Semantic Web. It is flexible, sup-

[1] Even though ontology and vocabulary share similar definitions, in this report, the term "ontology" will describe the group of definitions and relations that define a common framework, while "vocabulary" refers to the single defined elements of an ontology.

[2] In other words, this schemas provide additionall elements to describe the RDF framework. This helps broaden its descriptive capabilities. More information can be found in Section 1.4 of the RDF 1.1 Concepts and Abstract Syntax manual [70]

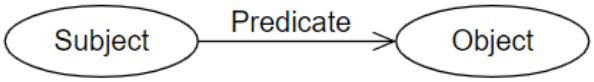

Figure 4: RDF graph with two nodes (Subject and Object) and a triple connecting them (Predicate).

porting a wide variety of syntaxes [74], and it can be serialized in several formats. In this case, serialization refers to the process of converting data structures into a format that can be stored. Among all syntaxes, TURTLE (Terse RDF Triple Language) [56] is particularly popular due to its readability and simplicity. In addition, other common syntaxes such as JSON (JavaScript Object Notation)[75] provide alternative ways to encode and access RDF data, offering different choices depending on distinct needs and preferences. This syntax makes RDF data more accessible to web developers, facilitating the integration of Semantic Web technologies into the web development. In Listing 8 we can identify an RDF structure in JSON syntax where `@context` describes context for the data, defining a prefix, `ex`, that associates with an URI, `@id` represents the subject uniquely identified by the URI, and `ex:hasEmail`, `ex:hasphoneNumber`, `ex:hasAddress` are properties with their corresponding values. A similar structure is shown in Listing 9 with the same RDF representation in TURTLE syntax [56]. These examples are merely representative and do not adhere to a perfectly accurate structure, for simplicity and illustrative purposes.

```
{
  "@context": {
    "ex": "http://example.org/"
  },
  "@graph": [
    {
      "@id": "#JohnDoe",
      "@type": "ex:Person"
   ,
  "ex:hasEmail": "john.doe@example.org"
  ,
  "ex:hasPhoneNumber": "123-456-7890"
  ,
  "ex:hasAddress": "123 Main St, Anytown, USA"
    }
  ]
}
```

Listing 8: RDF in JSON syntax

```
@prefix ex: <http://example.org/> .

ex:JohnDoe
   ex:hasEmail "john.doe@example.org" ;
   ex:hasPhoneNumber "123-456-7890" ;
   ex:hasAddress "123 Main St, Anytown, USA"
```

Listing 9: RDF in TURTLE syntax

Overall, RDF's flexibility, alongside its support for multiple syntaxes, makes it a good technology for achieving the goals of the Semantic Web. Whether for encoding data in a human-readable format, integrating with modern web development practices, or handling large datasets efficiently, RDF offers a robust and adaptable framework for data representation, data exchange and data provenance[74]. Finally, a crucial aspect of RDF lies in its ability to serialize in XML(Extensive Markup Language) format [76], known as RDF/XML [77]. This feature exploits the flexibility of XML to provide machine-readable format and eases the storage and transport of data. An example of XML structure is shown in Listing 10 following the previous example.

```
<Person>
  <Name>JohnDoe</Name>
  <Email>john.doe@example.org</Email>
  <PhoneNumber>123-456-7890</PhoneNumber>
  <Address>123 Main St, Anytown, USA</Address>
</Person>
```

Listing 10: RDF in XML syntax

6.2.3 OWL (Web Ontology Language)

Within the Semantic Web, ontologies play a crucial role in helping assess the interpretation of structured information elements and can be built in OWL format (as well as in RDF format) [67]. In this report we focused on the description and understanding of OWL since it is the format the METACLIP ontology [78] [79] is defined in. OWL builds upon RDF and provides a more expressive vocabulary to describe complex relationships between elements. It supports the definition of classes, subclasses, and instances, as well as relationships between individuals via object and data properties. Furthermore, OWL organizes classes in a subclass hierarchy, allows for logical combinations of classes, and supports other logical constraints. OWL's sophistication relates to its ability to define more complex relationships and properties. These features are a direct response to a variety of conflicting requirements, ensuring that OWL meets the diverse needs of different applications in the Semantic Web [67].

In Listing 11 an example of an OWL format in RDF/XML syntax is shown. First, the file defines the prefixes as a shorthand to refer to IRIs or URIs. Second, the ontology is described with a comment describing it. Then, the Class `Person` and the datatype property, with domain `Person` and range `xsd:string` are defined. Lastly, the RDF/XML syntax defines an individual, `JohnDoe` of type `Person` and property `hasEmail` linking to `john.doe@example.org`. With this example, we can observe how by means of RDF/XML syntax and OWL standards one can create an ontology with a class, a property and an individual.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:owl="http://www.w3.org/2002/07/owl#"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
       xmlns:ex="http://example.org/ontology#">

```

```
  <!-- Ontology Definition -->
  <owl:Ontology rdf:about="http://example.org/ontology">
    <rdfs:comment>A simple example ontology.</rdfs:comment>
  </owl:Ontology>

  <!-- Class Definition -->
  <owl:Class rdf:about="http://example.org/ontology#Person"/>

  <!-- Datatype Property Definition -->
  <owl:DatatypeProperty rdf:about="http://example.org/ontology#hasEmail">
    <rdfs:domain rdf:resource="http://example.org/ontology#Person"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>

  <!-- Individual Definition -->
  <rdf:Description rdf:about="http://example.org/ontology#JohnDoe">
    <rdf:type rdf:resource="http://example.org/ontology#Person"/>
    <ex:hasEmail>john.doe@example.org</ex:hasEmail>
  </rdf:Description>

</rdf:RDF>
```

Listing 11: OWL language example in XML syntax

6.2.4 PROV

PROV [80] is a standardized model developed by the World Wide Web Consortium (W3C) Provenance Working Group. It aims to represent data provenance information in a way that it facilitates sharing and interchange across different systems and applications. PROV-O is part of a broader family known as PROV standard, which includes conceptual data models, vocabularies and technical notes. PROV standard aims to provide a comprehensive guide for modeling, serializing and reasoning about provenance data. Its main goal is to achieve a unified framework for describing data provenance based on the notions of entities, activities and agents involved in manipulating, producing, influencing and delivering a piece of data [80] [81]. The conceptual framework of the PROV model is established by PROV-DM (Provenance Data Model) [82], defining the core concepts and relationships to capture the provenance data in a structured manner. This data model provides the foundation upon which other PROV components, such PROV-O [83], are build.

PROV-O (Provenance ontology) [83] is a specific component of the PROV standards that provides an OWL ontology to represent provenance information. It enables the encoding of provenance data in RDF and is designed to describe how data is generated, by whom, and through what process, making it easier to trace its the origin and lifecycle across platforms and applications. The core elements of PROV-O are three: entities, activities and agents. The `prov:Entity` class defines any physical, digital or conceptual objects -either real or imaginary. The `prov:Activity` class denotes something that occurs over a particular period of time and acts on entities. It includes data transformations, processing, relocation or modifying. Lastly, the class `prov:Agent` 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. These concepts are the backbone of any provenance graph that aims to illustrate a data's path [80].

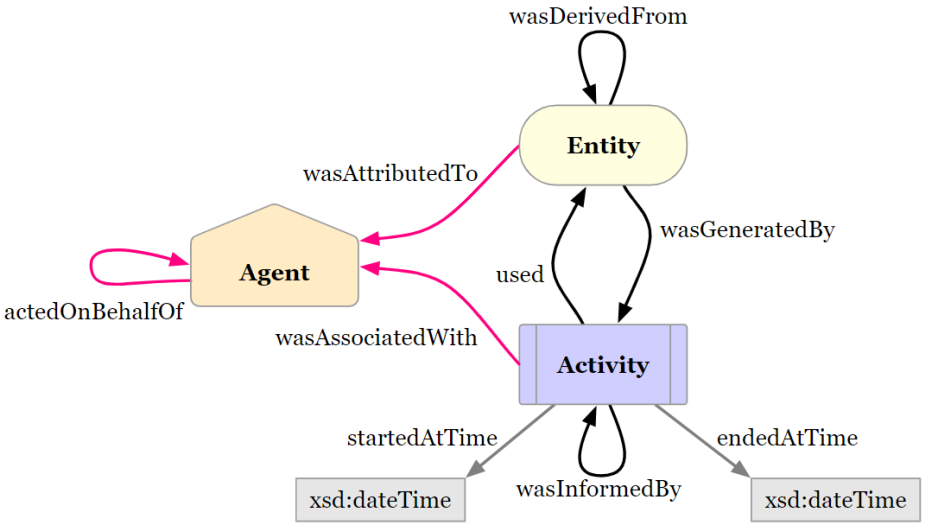

Figure 5: PROV Core Structure. Image from "*The METACLIP semantic provenance framework for climate products*" [4].

Furthermore, PROV-O offers various forms of influence to capture the dependencies and causal relationships among entities, activities and agents. For example, an activity can use an entity as input (`prov:used`), generate a new entity as output (`prov:wasGeneratedBy`), or derive a new entity from an existing entity (`prov:wasDerivedFrom`). Similarly, agents can be associated with activities (`prov:wasAssociatedWith`) and attributed to entities (`prov:wasAttributedTo`). By combining all the available relationships together, provenance graphs can be assembled, allowing users to trace the lineage of a particular data structure and understand its present state [83]. This relations are illustrated in Figure 5.

Since PROV-O aims to be versatile and suit various needs, it provides additional flexibility on its vocabularies by means of mechanisms that allows the grouping of related provenance statements. These mechanisms, called bundles, enable encapsulation of related statements into logical groups. Bundles are sub-graphs of larger provenance graphs, which facilitate management, modularity and reuse of the ontologies. This feature empowers the extension of PROV-O, allowing other ontologies to add new terms and relationships that are specific to a particular field or domain of study while still complementing the PROV-O core model [83].

Essentially, PROV is designed to model data provenance, tracking the lineage and history of entities, agents, and activities. It defines entities as things that are created, transformed, or used; activities as processes that act upon entities; and agents as individuals or organizations responsible for those activities. Relationships like derivation, attribution, and usage enable comprehensive provenance records. Combined, RDF, OWL, and PROV form a cohesive framework for building rich, interconnected datasets on the Semantic Web [82].

6.2.5 METACLIP

METACLIP, standing for METAdata for CLImate Products [4], is a provenance framework based on semantics exploiting the web standards of RDF (Resource Description Framework).

Exploiting the Resource Description Framework (RDF), METACLIP is designed to track and manage data provenance specifically of climate data. Its framework implements specialized ontology to organize and manage provenance of climate data products. The use of RDF by METACLIP ensures that every piece of generated climate data, whether it is climatological data, indices, plots, or other kinds of digital data, is inseparably linked with its provenance information. [4] Then, users that have access to it can always trace the data back to its origins. In this particular context, the METACLIP ontology refers to a set of defined terms and specific definitions regarding climate operations, relationships, and data to represent the climate domain comprehensively. It attempts to illustrate not only data but also the processes undertaken and relationships directly associated with climate data production and processing. It achieves so by embedding several other known ontologies into its own [4], such as Dublin Core [84], RDFS and others. All standards or ontologies integrated or used by METACLIP are shown in Table 2.

Vocabulary name	Prefix
Dublin Core Metadata Element Set [85]	ds
GeoSPARQL [86]	geosparql
PROV Ontology (PROV-O) [1]	prov
RDF Schema [87][3]	rdfs
Simple Knowledge Organization System (SKOS) [88]	skos
Dublin Core Metadata Initiative Metadata Terms [89]	terms
eXtensible Markup Language Schema [90]	xsd

Table 2: Pre-existing ontologies used by METACLIP

The METACLIP framework implements four core ontologies, which are an extension of the well-known PROV-O and can be publicly accessed:

- **Datasource ("ds")**: The datasource vocabulary details the source of the input data. It includes dataset descriptions and transformations. It also establishes the proper links between different classes and arguments at each step [91].
- **Calibration ("cal")**: Calibration encodes metadata related to bias correction, downscaling, and other statistical operations. It focuses on validating and developing statistical downscaling functionalities [92].
- **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 [93].
- **Graphical Output ("go")**: The graphical output ontology describes graphical products such as charts and maps. Characterizes the uncertainty types and their communication [94].

[3] The difference between a schema and an ontology in the context of provenance can be summarized as follows: a schema, such as RDF Schema (RDFS)[71], defines the basic structure and constraints of certain data. An ontology, on the other hand, offers a richer and more expressive framework for modeling domain knowledge, allowing for complex relationships and inferencing capabilities.

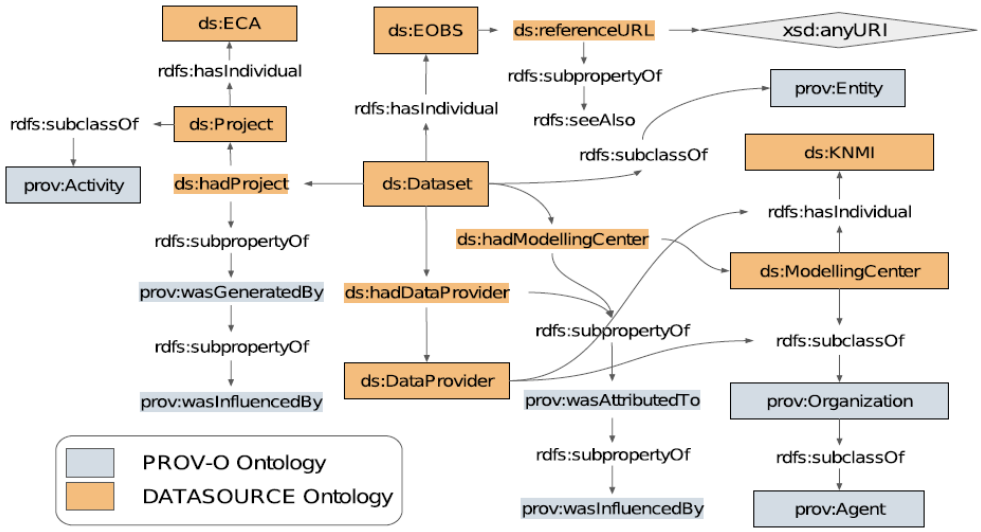

Figure 6: Example demonstrating how the METACLIP's datasource ontology integrates and utilises the PROV-O ontology.

METACLIP represents a significant extension of the PROV-O ontology, which serves as its fundamental starting point, providing key terms and concepts [4]. To accurately understand the METACLIP vocabularies and the way they establish an extension of the PROV-O data model, a brief explanation regarding Figure 6 is given next. In METACLIP, the first step in provenance extraction involves defining the input data by means of the datasource(`ds:`) vocabulary. It is defined with the entity `ds:Dataset` which can be identified as an extension of the `prov:Entity`. It is 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 origin of the data. Several other specific details unique to each `ds:Dataset` are recorded. When generally implemented, METACLIP can also encode the various transformations of the data 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.

6.2.6 metaclipR: Climate4R extension

The METACLIP framework was first implemented through the programming environment R in the package metaclipR[4][95]. The primary functionality 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. metaclipR is capable of handling different types of data but it is particularly customized for the climate4R framework[96], which makes use of several R packages different from the ones SUNSET works with, such as transformeR[97] or visualizeR[98]. Its structured approach offers the handling of various types of data: observations, seasonal forecasts, graphical representations, and others. It is subdivided in four core packages: loadR[99], transformR[97], visualizeR[98] and downscaleR[100][101]. The specialization of metaclipR on climate4R allows it to efficiently deal with all the packages and their execution. Thus, it is built to specifically extract provenance information from climate4R objects and the

way that particular climate data structure is processed.

metaclipR provenance generation functions take as input manually added strings with information and the previously generated provenance graph or the climate4R object. For example, as shown in Listing 12, `metaclipR.Interpolation()` takes several inputs: a list with the prior graph and the last nodename, contained in the `graph` object, the package, `package` and version of the executed function, `version`, which must be manually added, a graph object with a `ds:SpatialExtent` node, `RefSpatialExtent`, and the interpolation method used, which also must be added manually. Provenance of the function that computes the interpolation, `fun`, is set to default to the climate4R function `interpGrid()`. Additionally, `metaclipR.Interpolation()` takes a list of arguments, `arg.list`, set to `NULL`, an option to disable the command call provenance, and `disable.command`, default to `FALSE`, and a description `dc.description` also set to `NULL`. As output, metaclipR functions return a list with the built provenance graph with the added nodes and relations and the name of a particular node. In the case of `metaclipR.Interpolation()`, it returns a list containing the provenance graph with the added interpolation node and its relations with the existing elements of the graph and the name of the interpolation node.

```
metaclipR.Interpolation <- function(graph,
                            package = "transformeR",
                            version = as.character(packageVersion(package)),
                            RefSpatialExtent = NULL,
                            InterpolationMethod,
                            fun = "interpGrid",
                            arg.list = NULL,
                            disable.command = FALSE,
                            dc.description = NULL)
```

Listing 12: metaclipR.Interpolation() function

Moreover, the metaclipR functions are designed and structured to be included in the execution script so each specific function runs after its corresponding climate4R function. `metaclipR.Climatology()`, for example, as shown in Listing 13, has to be executed after the climate4R function `climatology()`. That is, the provenance is generated after the execution of a particular transformation and has to be added manually in the script.

```
SU.clim <- climatology(SU, clim.fun = list(FUN = "mean", na.rm = TRUE))

metadata.EOBS.SU <- metaclip.Climatology(graph = metadata.EOBS.SU,
                                  arg.list = list(clim.fun =
                                    list(FUN = "mean",
                                  na.rm = TRUE)))
```

Listing 13: metaclipR.Climatology() execution example

metaclipR includes functions from climate4R packages [101]. Then, all the aforementioned packages have to be installed in order to use it. 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 provenance graphs are constructed using the

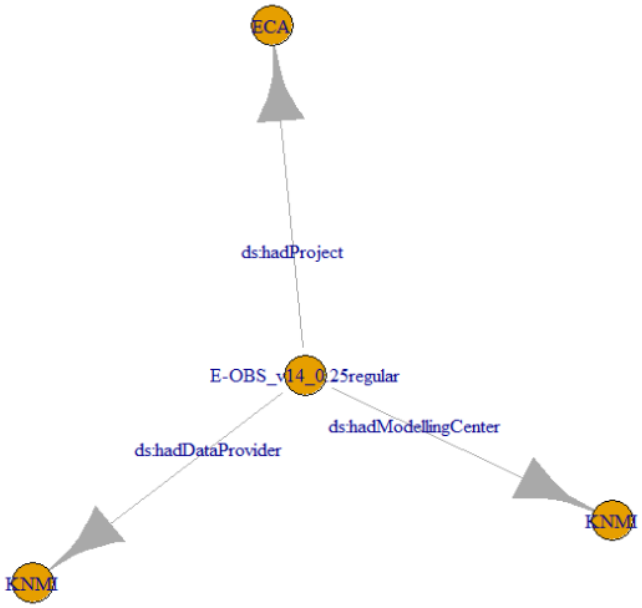

Figure 7: metaclipR generated simple provenance graph.

igraph package [102]. A preliminary overview of the metaclipR functions lead to an output shown in Figure 7 where some simple dataset provenance is stored from an established data source.

6.2.7 metaclipR internal structure: igraph package

metaclipR uses the igraph package [102] to generate and manage provenance graphs, with core functions `make_empty_graph()`, `add_vertices()` and `add_edges()`. First, the function `make_empty_graph()` generates the `igraph` object that will be updated after each execution of the metaclipR functions. Then, `add_vertices()` is utilised to include a node or vertex to the object. This function requires an `igraph` object as input. Additionally, it establishes elements related to provenance graphs such as the `name` to unequivocally identify the node, the `label`, to define the displayed title of the node, and the `className`, to define the ontology class. Finally, `add_edges()` establishes the relation between the two selected nodes, either an Object property, Data property or Annotation property [67]. Based on this procedure metaclipR generates the provenance graph taking the required inputs to generate nodes and relations, either by manually adding them prior to function execution or directly from the climate4R object. An example is provided in Listing 14.

```
graph <- make_empty_graph(directed = TRUE)

graph <- add_vertices(graph,
                      nv = 1,
```

```
                  name = DatasetSubset.nodename,
                  label = "DatasetSubset",
                  className = "ds:DatasetSubset")

graph <- add_edges(graph,
               c(getNodeIndexbyName(graph, parent.node),
                 getNodeIndexbyName(graph, DatasetSubset.nodename)),
               label = paste0("ds:hadDatasetSubset"))
```

Listing 14: igraph core functions execution example. To easily manage the igraph functions we utilised several functions from the metaclipR package

6.2.8 METACLIP Interpreter

After producing a provenance product and embedding it in the output, the METACLIP interpreter allows for an interactive exploration of the provenance information. The METACLIP interpreter is a web tool designed to handle RDF representations in several serialization formats, particularly JSON. It has been designed as an interactive visualization tool that provides a user-friendly interpretation of the provenance information generated during the workflow. The interpreter links the provenance-generated file with the ontology, displaying all the hierarchical relations established by it. Then, simply by classifying an element within a class, information can be inferred from it.

By means of the interpreter, the user can visualize provenance at different levels, from technical and complex details to the general aspects retrieved. It includes several functionalities: users can click nodes or use a topic selector to illustrate details like calibration, dataset subset, and command calls, including full source code descriptions if needed. It has a two-component architecture: a back-end service using Apache Jena [103] for parsing and extracting METACLIP provenance information, and a front-end component using D3.js [104] for interactive visualization. Furthermore, the interpreter can extract and decompress metadata embedded in product files such as images or graphs, easing the manipulation and visualization of metadata. More information and examples can be found at metaclip.org [105].

7 Work performed and results

The main objective of the project is to incorporate METACLIP's provenance framework into SUNSET by embedding all the provenance information in the R code. To achieve this, it is necessary to use a set of functions capable of extracting and properly defining provenance during a SUNSET execution. Initially, three options were considered: using the metaclipR functions, developing in-house functions, or a hybrid approach. Therefore, a key part of this objective is to identify how to implement provenance more efficiently in SUNSET using the existing tools. This requires evaluating the compatibility of SUNSET and the metaclipR package and recognizing possible integration methodologies.

This objective presents several other challenges. It requires developing and familiarizing with the R environment, with SUNSET's framework and the packages it includes. Additionally, it is essential to acquire an understanding of climate data processing and climate verification to properly integrate METACLIP's provenance framework. Moreover, understanding provenance data semantics is a significant challenge. This requires familiarization with METACLIP and its conventions.

The second objective consists of adapting SUNSET outputs to the METACLIP web interpreter (web-based interactive front-end for metadata visualization) to better display the provenance information produced by the workflow. The use of the interpreter can make the provenance data generated more accessible and intuitive to visualize, allowing a user-friendly interface. The integration aims to provide a clear and highly interactive visualization experience for users to engage with, and to understand the climate product provenance.

7.1 Methodologies

On the one hand, the methodology mainly involved a deep understanding of SUNSET's workflow structure and the operations that take place within it, the fundamentals behind climate data processing, and the upcoming tools to generate provenance in the climate scenario, particularly focusing on METACLIP and metaclipR. In other words, we aimed to comprehensively integrate the previously presented Sections 6.1 and 6.2 to create a unique framework for data provenance that incorporates all aspects of the theoretical foundations. On the one hand, we identified each operation and transformation step involved in the workflow execution for atomic recipes -simple executions with one climate variable and a single climate model. This analysis included mapping data inputs, intermediate processes and final outputs. By breaking down the workflow into parts, we could identify possible provenance integration points and determine the proper structure that the provenance graph should have.

On the other hand, it was crucial to understand the physical, mathematical and statistical procedures underlying the main operations and transformations within the workflow. This involved delving into the specific nature of the climate data briefly presented in Section 6.1, its structure and the methods applied to it, primarily focusing on the methods used for verification. By gaining a deeper understanding of these statistical and computational processes, we could design a set of provenance functions that accurately captured the details of each process. That is, the provenance functions are not only designed to capture exclusively the data transformations but also the contextual information about statistical procedures and physical principles involved.

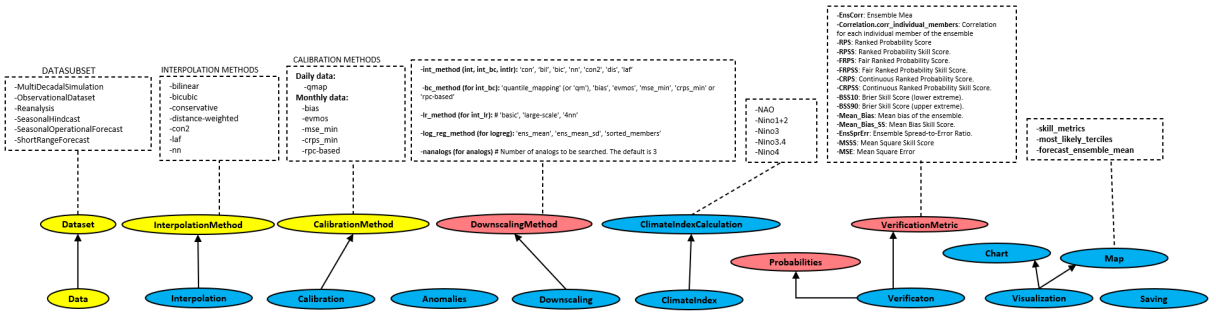

Figure 8: Schematic example showing possible provenance elements in the SUNSET workflow based on the information specified in the recipe-without considering METACLIP ontology.

Notably, this methodology ensures that the developed provenance functions provide reliable and comprehensive provenance data.

7.2 Preliminary Project

This section aims to briefly describe the project conducted during December 2023 and February 2024 as a deliverable for the Computing Programming and its Applications (CPIA) class. This initial analysis was essential for identifying the critical elements required for integrating METACLIP and SUNSET, setting a robust foundation for the subsequent phases of the project. This stage not only enhanced our understanding of the individual elements and tools available, but also facilitated a more strategic approach to their integration.

7.2.1 First approach: familiarization and identification

The initial phase of the research involved the analysis and comprehension of the METACLIP vocabularies and their possible implementation into SUNSET. For this purpose, we utilised Protégé [106], 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 understanding METACLIP's hierarchy and classification of each class and relation, and how the interconnections between classes are established. Protégé is designed to work with several formats, including OWL files [106].

Similarly, the research also involved the meticulous comprehension of the SUNSET workflow, as well as of the new environments it is established on. In addition, we started a preliminary approach to provenance extraction and generation from SUNSET, identifying potential integration points and defining the specific pieces of information to be propagated. Without considering the METACLIP vocabularies at first, we created a provenance scheme, as shown in Figure 8. In it, we can identify several steps regarding the transformations the data can undertake, as well as several provenance pieces of information based on the recipe specifications that directly identify the datasets.

7.2.2 RDF Generation Using R Locally

Due to the complexity and extensive nature of the SUNSET workflow and the METACLIP ontologies, the initial approach was to extract provenance information from the recipe and generate a simple RDF graph by means of the established METACLIP ontologies, but without utilising metaclipR. This was due to some difficulties with the local execution of the metaclipR package [95]. That led us to develop a function that implements from scratch some ontology definitions to elements extracted and translated from the recipe. Two specific packages were required to use: yaml [107], for the efficient handling of YML files such the recipe, and redland [108], for 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 and serialized into TURTLE. The outcome is shown in Listing 15.

```
@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", "Longitude" .

ds:Dataset
   ds:SubClassOf prov:Entity .

ds:DatasetSubset
   ds:SubClassOf ds:Dataset ;
   ds:hadDataProvider "Meteo-France-System7" ;
   ds:hadModellingCenter "Meteo-France-System7" ;
   ds:hasSpatialExtent ds:Dimension ;
   a ds:Dataset .
```

Listing 15: RDF file outcome

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. Note the fact that this particular model could be largely extended by including the definition of more classes and relations from the METACLIP ontology. However, no further development is intended to be made on this code as it does not achieve the desired implementation into the SUNSET workflow, or, at least, it did not prove to be the best way to achieve the proposed objectives. This is, it does not generate full provenance for SUNSET, it does not define properly the terms according to METACLIP ontologies and its use is complex to implement, resulting in difficulties defining a large number of elements and relations. Also, it presents some minor errors when defining several classes. It was simply created for research purposes and to gain a deeper comprehension of the structure of this type of data. Finally, the results can be graphically visualized by means of an RDF interpreter. In this particular case, we have used an online interpreter with URL 4096 https://www.ldf.fi/service/rdf-grapher. The obtained graph is shown in Figure 9 and the code in the Appendices B.1. Note the fact that many other libraries related to the provenance project, such as rdflib or jsonld, were also analyzed and tested.

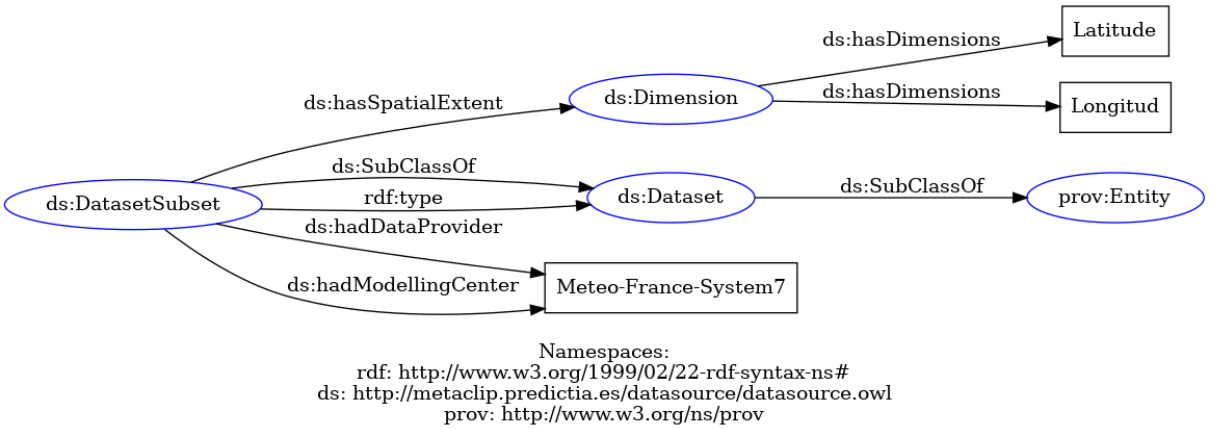

Figure 9: Graphical representation of the first provenance generation. We can identify the nodes describing classes and the edges describing either object properties such as `ds:SubClassOf` or data properties such as `ds:hasDimensions`. At the bottom of the image, we can recognize the definitions and the used prefixes with their corresponding URLs linked to the ontologies.

7.2.3 Simple implementation of metaclipR into SUNSET: First developed code

Once a simple provenance generation was achieved, the focus shifted to implementing metaclipR functions within a SUNSET execution script. Subsequently, after a thorough familiarization process with the metaclipR and SUNSET R functions and their use, we cooperatively created a simple code that merges in the same execution both their functionalities. Then, the code is organized into several sections that simultaneously incorporate metaclipR and SUNSET functions, illustrating how they can be merged.

In the first place, the script begins by sourcing several SUNSET modules such as Loading, Calibration, Skill, Saving, and Visualization from the copied repository branch. It then loads the metaclipR library which contains the provenance generation functions based on the METACLIP framework. Each line calls an R script from their specific path and loads 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 execution instructions in the recipe. 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 metaclipR function `metaclipR .BiasCorrection()`, which includes information about the method used for the bias correction -a form of calibration-, the arguments passed, and the references for graphs and datasets.

```
> calibrated_data <- Calibration(recipe, data)
> metadata_calibrated <- metaclipR.BiasCorrection(
                      package = "modules/Calibration/Calibration.R",
                      version = "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 the SUNSET 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, since, as a preliminary approach, no metaclipR functions were utilised to describe this operation.

```
> skill_metrics <- Skill(recipe, calibrated_data)
> probabilities <- Probabilities(recipe, calibrated_data)
```

Finally, the open graphics devices are closed by means of `dev.off()`. 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 provided in the Annexes B.2. Overall, the code executes part of the SUNSET workflow for sample data and simultaneously extracts dataset and calibration metadata by means of the functions of the metaclipR package.

7.2.4 First METACLIP Integration in SUNSET (Preliminary project conclusions)

The script processes data as stated before and generates two outputs: a JSON graph and a visual plot that represent the data provenance and relationships in the METACLIP Interpreter. The

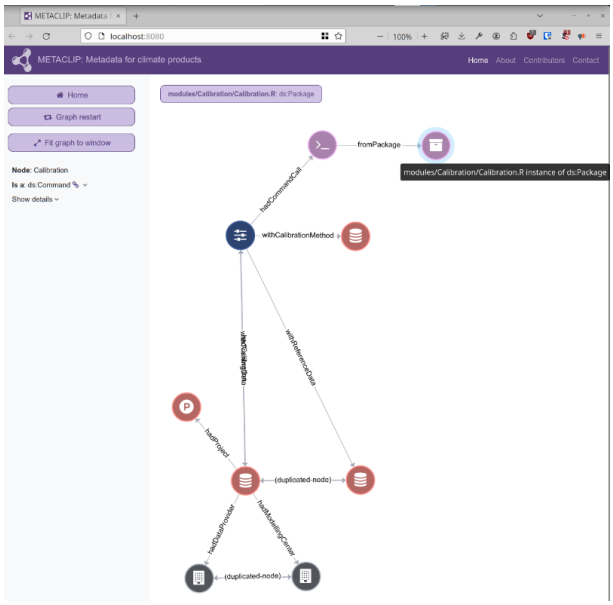

Figure 10: Visual representation of the obtained JSON in the METACLIP Interpreter.

JSON file is shown in the Appendices B.3 and the visual plot in Figure 10. The JSON comprehensively encapsulates the relationships and dependencies among various provenance entities defined based on the METACLIP ontology. Thus, the METACLIP functions properly establish 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, subject-predicate-object. The visual representation of the graph effectively translates the detailed provenance information from the JSON file into a more accessible format.

By means of the preliminary project we 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 for a particular type of 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 preliminary project implied the use of various new development environments: GitHub, GitLab, R programming language, conda environment, etc. This resulted in improved technical skills regarding the new software encountered and learning 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 diverse workflow executions, and a wider understanding of provenance definition and management within the climate scenario.

7.3 Implementation of metaclipR: SUNSET_PROVENANCE

The preliminary project demonstrated that it is possible, to some extent, to implement a provenance framework in SUNSET using the metaclipR package. This section aims to build on the work done in the preliminary project, presenting the first full-provenance approach for SUNSET through the integration of metaclipR. The developed functions combine metaclipR and in-house code to structure provenance generation specifically for SUNSET. First, an overview of the compatibility between metaclipR and SUNSET is provided to illustrate the approach that led to the composition and structure of the code. Second, the developed code is reviewed and its structure explained.

7.3.1 metaclipR compatibility with SUNSET (Second approach)

metaclipR functions are designed to be implemented directly in the main script or in the R console after their corresponding climate4R operation to extract and generate the provenance. This makes it complicated to directly implement them in SUNSET as one would in climate4R, since it would result in overly long and complicated scripts to write. In that sense, the aim is to include this functionality as automated and as user-friendly as possible, extracting and generating provenance information automatically when a module is executed. To achieve this, we aim to create a set of functions that, on the one hand, manage the execution of metaclipR functions accordingly with the SUNSET modules and, on the other hand, manage to extract all provenance from the SUNSET environment -recipe, `s2dv_cube` and configuration files. Consequently, the developed code is hybrid as it embeds metaclipR functions with in-house code.

As a first approach, we selected `metaclipR.Dataset()`, `metaclipR.DatasetSubset()`, `metaclipR.BiasCorrection()`, `metaclipR.Anomaly()`, `metaclipR.ClimateIndex()` and `metaclipR .Interpolation()` as the core functions to generate provenance. These functions were chosen for their relevance to common operations with SUNSET and their ability to comprehensively document the transformations applied to the data from simple information provided as input. This selection establishes a strong foundation for the provenance tracking functions, capturing key steps from data retrieving to analysis.

At the same time, `sunset_prov_verification()` and `sunset_prov_downscale()` were developed in-house based on the structure of the metaclipR package. These functions are specifically designed to handle provenance for SUNSET. Operations regarding verification processes and downscaling methods specific to SUNSET are not covered by the existing metaclipR functions. By creating these additional tools, we ensure that all significant aspects of data manipulation within SUNSET are documented, providing a sufficient and transparent provenance trail. However, as this approach is still preliminary, not all possible configurations and operations of SUNSET are considered, only those essential or simpler to develop, with the aim to afterwards extend the functionalities to the entire framework.

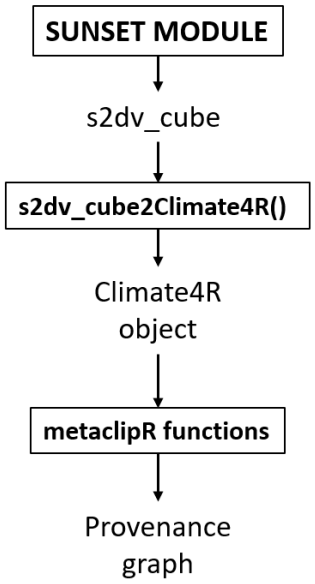

Figure 11: Schema of `s2dv_cube2Climate4R` function within a provenance function execution.

7.3.2 `s2dv_cube2Climate4R()`

As previously mentioned, some metaclipR functions require the output of the corresponding climate4R operation as input. Due to the high computational cost of running climate4R in parallel with SUNSET to obtain the necessary inputs for generating provenance, which would duplicate the amount of data loaded into memory, we used a function to convert `s2dv_cube` objects to climate4R objects, `climate4R2s2dv_cube()`. A climate4R object, as we refer to it, is a list comprising six elements. Firstly, `Variable` contains details about the represented variable like its name, description, and units. Secondly, `Data` holds the actual numerical data in a multidimensional array format along with dimension information. Thirdly, `xyCoords` contains spatial coordinate details such as longitude, latitude, and projection information. Fourthly, `Dates` includes temporal information such as start and end dates, and seasonal subsets. Fifthly, `InitializationDates` lists the initialization dates for different dataset members. Lastly, `Members` lists the members of the dataset. Additionally, the object carries attributes like dataset name, source, and URL, providing metadata about its origin and location. Overall, it represents a subset of a climate dataset with associated variables, spatial, temporal, and metadata information

The function `s2dv_cube2Climate4R`, authored by Eren Duzenil, transforms an `s2dv_cube` object into a list compatible with climate4R, ensuring coordinates "x" and "y" are in ascending order and dimensions organized as `time`, `lat` and `lon`. The function `s2dv_cube2Climate4R` accepts several parameters, including the `s2dv_cube` object itself, names of the dimensions (`time, start date, latitude, longitude, and ensemble`), and a parameter indicating whether to return the data array along with the metadata.

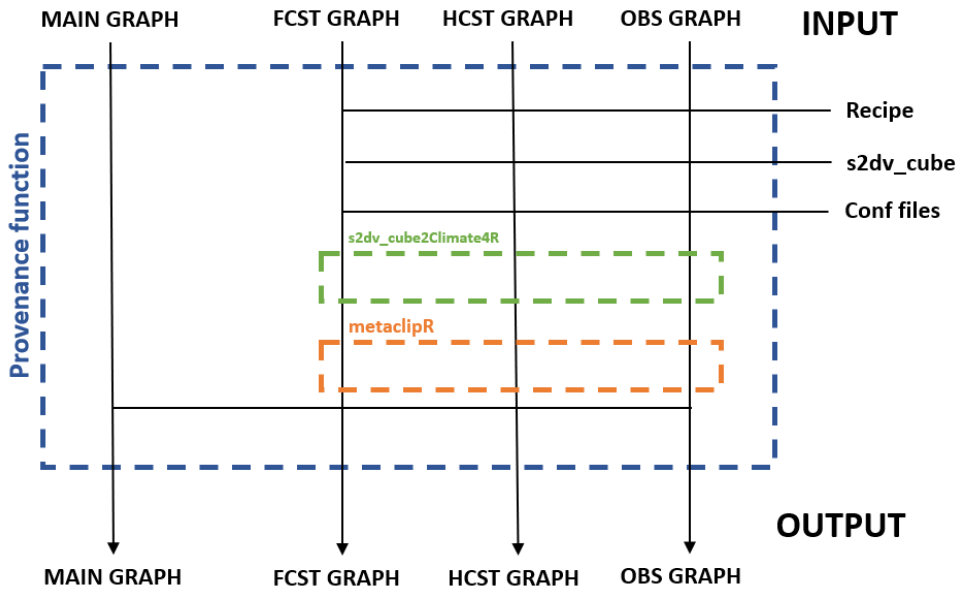

Figure 12: Structure of SUNSET_PROVENANCE functions.

7.3.3 Developed code with metaclipR

The developed set of functions is called SUNSET_PROVENANCE. Each function within SUNSET_PROVENANCE follows a similar structured approach to ensure the accurate generation and integration of provenance graphs. Additionally, each function incorporates a specific metaclipR function to structurally generate provenance according to SUNSET. Initially, the functions process the input recipe and the data passed from the SUNSET module, retrieving the predefined steps, parameters and methods. If necessary, the data is transformed using `s2dv_cube2Climate4R()` to align with the specific requirements of the metaclipR functions. This first step is crucial to ensure that the data is in the correct format and state.

Following this, the functions work with a list containing the main provenance graph and the three subgraphs (forecast, if loaded, hindcast, and observations), which are progressively built and updated. Each metaclipR function independently manipulates and enhances these subgraphs by adding nodes, edges, and labels, capturing the details of the data processing and transformations being applied. Although each dataset is processed by a separate function, some functions dynamically generate vertices that interconnect them. For example, the provenance function `sunset_prov_calibration()` establishes a relation defined as `ds:withReference` between the observational dataset and the calibrated hindcast, linking the observational dataset node with the hindcast calibration node. This approach enables each function to focus on a specific dataset and aspect of the workflow, as well as for setting complex relationships within the graph.

As the functions execute, they generate intermediate `igraph` objects and determine the main node for each dataset, which is crucial for maintaining the continuity and integrity of the provenance tracking. These intermediate graphs are stored and passed along to subsequent functions, ensuring that each step can independently contribute to the overall provenance graph without interference. Lastly, the functions merge the individual provenance graphs into a main graph,

encapsulating all the processing steps and their interdependencies. The output is a list that includes the updated main graph along with the three subgraphs, each annotated with their respective node name. This output structure provides a clear and detailed provenance record. Also, it eases the generation of provenance for each individual dataset.

Note that all functions except `sunset_prov_dataset()`, which is the starting point, take as input the recipe, the data processed from the previous module, and a list containing the main provenance graph, the forecast (if loaded), the hindcast and observational provenance graphs. This approach derives from the way metaclipR defines nodes and labels when generating the `igraph`. Since each metaclipR function generates a list containing the `igraph` object and the name of the last or main node for each dataset, it is necessary to run and save each individual graph (forecast, hindcast, and observational) at every step to ensure each operation can be performed separately in the next function. This way, the developed functions generate two or three provenance graphs independently within an execution and then combine them into the main graph. This is needed to preserve the flexibility in SUNSET, as some workflows may not use all the possible datasets, and the modules can be called in different orders. The structure described is shown in Figure 12.

7.3.4 Example execution

In this section, an example script for executing the provenance functions and certain SUNSET modules is shown. First, the process starts by loading the recipe file. The `prepare_outputs()` function reads the recipe, setting up the necessary parameters to be extracted during the execution.

```
> recipe_file <- "recipes/atomic_recipes/recipe_seasonal_provenance.yml"
> recipe <- prepare_outputs(recipe_file)
```

Afterwards, the Loading.R function from SUNSET's Loading module loads the data specified in the recipe, which includes hindcast and observational data. The `sunset_prov_dataset()` function then generates the initial provenance graph for each of the loaded datasets, capturing data such as type of dataset and data provider. Moreover, this provenance function also adds information regarding the Dataset Subset and the Interpolation taken place.

```
> data <- Loading(recipe)
> graph_dataset <- sunset_prov_dataset(recipe, data)
```

The `Calibration()` function calibrates the data based on the specified method on the recipe. Then, `sunset_prov_calibrtion()` function updates the provenance graph with a Calibration step node, linking it to its corresponding dataset node.

```
> calibrated_data <- Calibration(recipe, data)
> graph_calibration <- sunset_prov_calibration(recipe, calibrated_data,
    graph_dataset)
```

From the Anomalies module, `Anomalies()` function computes the anomalies from the calibrated data, comparing it with the climatology. Next, `sunset_prov_anomalies.R` includes details about the anomalies calculation to the provenance graph, linking it to the calibration nodes.

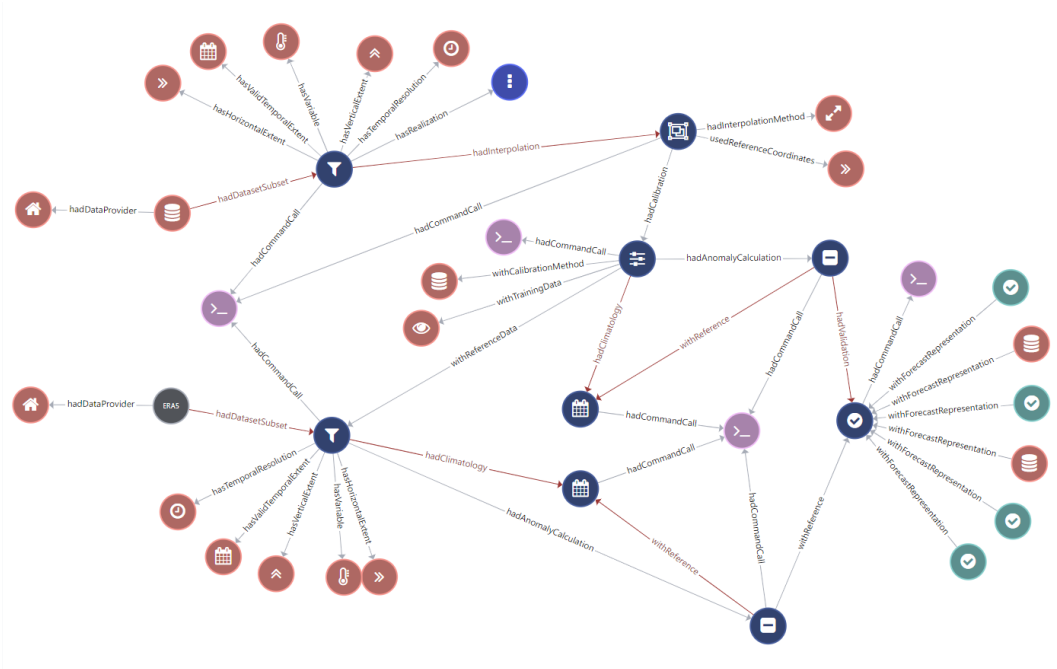

Figure 13: Output interactive provenance graph resulting from the example execution of SUNSET_PROVENANCE functions. Plotted with the METACLIP Interpreter.

```
> anomalies_data <- Anomalies(recipe, calibrated_data)
> graph_anomalies <- sunset_prov_anomalies(recipe, anomalies_data,
    graph_calibration)
```

The `Skill()` function calculates skill metrics to evaluate the performance of the model. It computes all the metrics specified in the Recipe. Later, `sunset_prov_verification()` updates the provenance graph with a verification node, along with nodes for each specific metric.

```
> data <- Loading(recipe)
> graph_dataset <- sunset_prov_dataset(recipe, data)
```

```
> skill_metrics <- Skill(recipe, anomalies_data)
> graph <- sunset_prov_verification(recipe, calibrated_data,
    graph_anomalies)
```

Lastly, by means of the metaclipR function `graph2json` the JSON file is generated, where all provenance information is stored in suitable format in the specified directory.

```
graph2json(graph$main_graph$graph, "/esarchive/scratch/apuiggro/sunset/
    graph.json")
```

The resulting JSON is shown in the Appendices B.3 and the interactive graph plotted in the METACLIP Interpreter in Figure 13. Other output images from different executions of SUNSET_PROVENANCE functions are provided in Figure 14, Figure 15, Figure 16 and Figure 17.

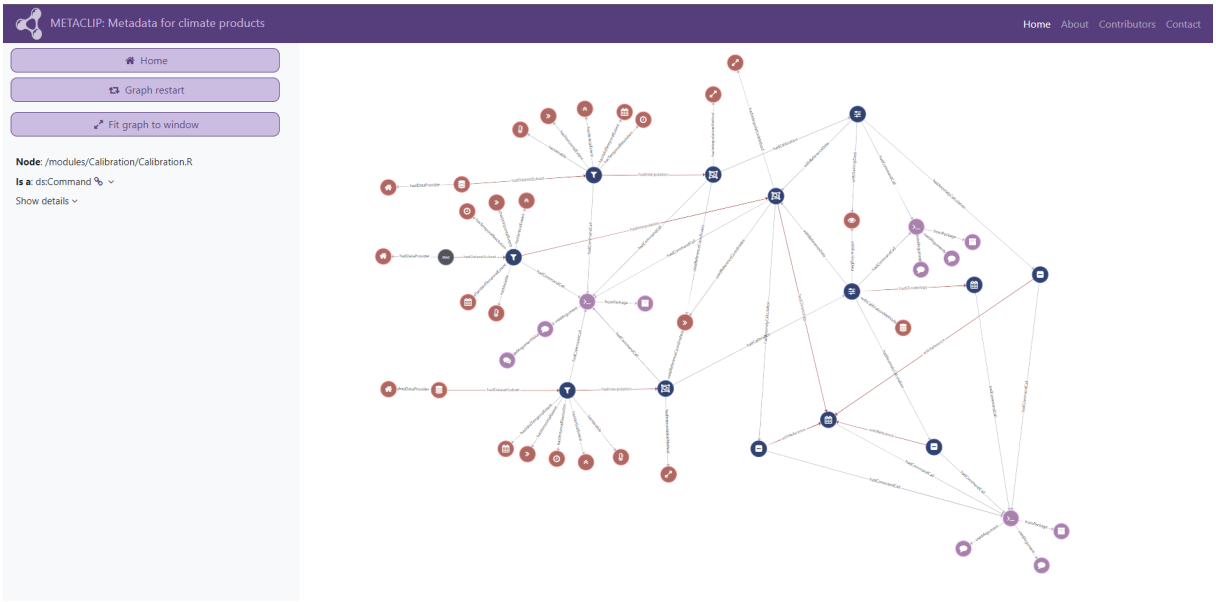

Figure 14: Example graph generated with SUNSET_PROVENANCE functions. The graph describes the loading of Meteo-France-System7 model data for forecast and hindcast, as well as ERA5 observational data. It also represents the computation of calibration and anomalies.

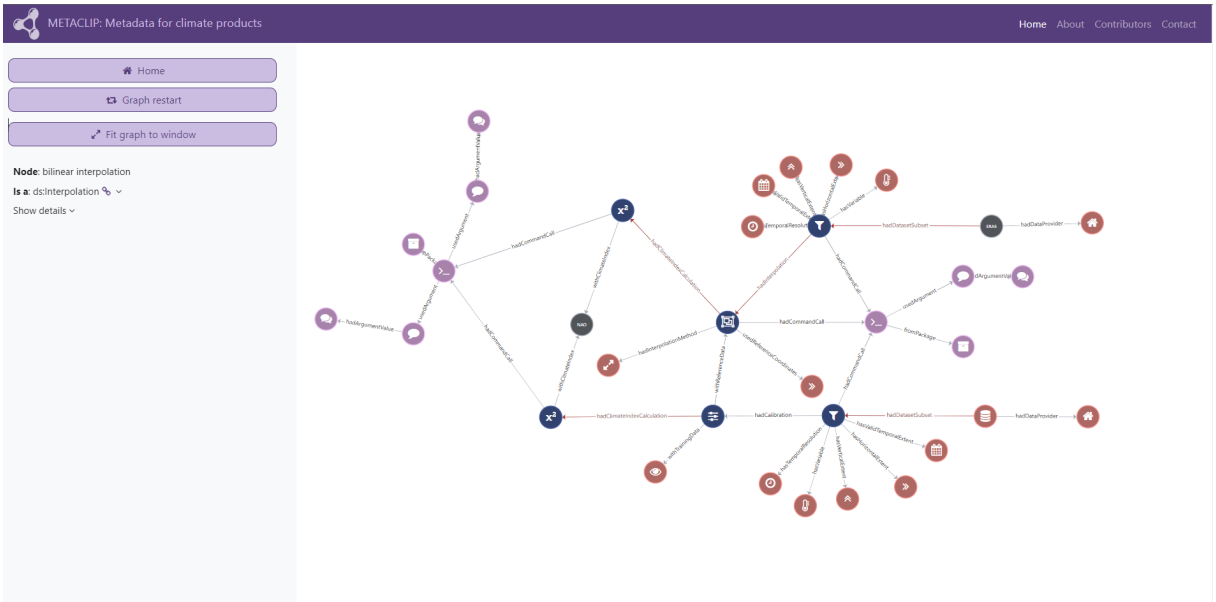

Figure 15: Example graph generated with SUNSET_PROVENANCE functions. The graph describes the loading of hindcast and observational data. It also represents the computation of the NAO index.

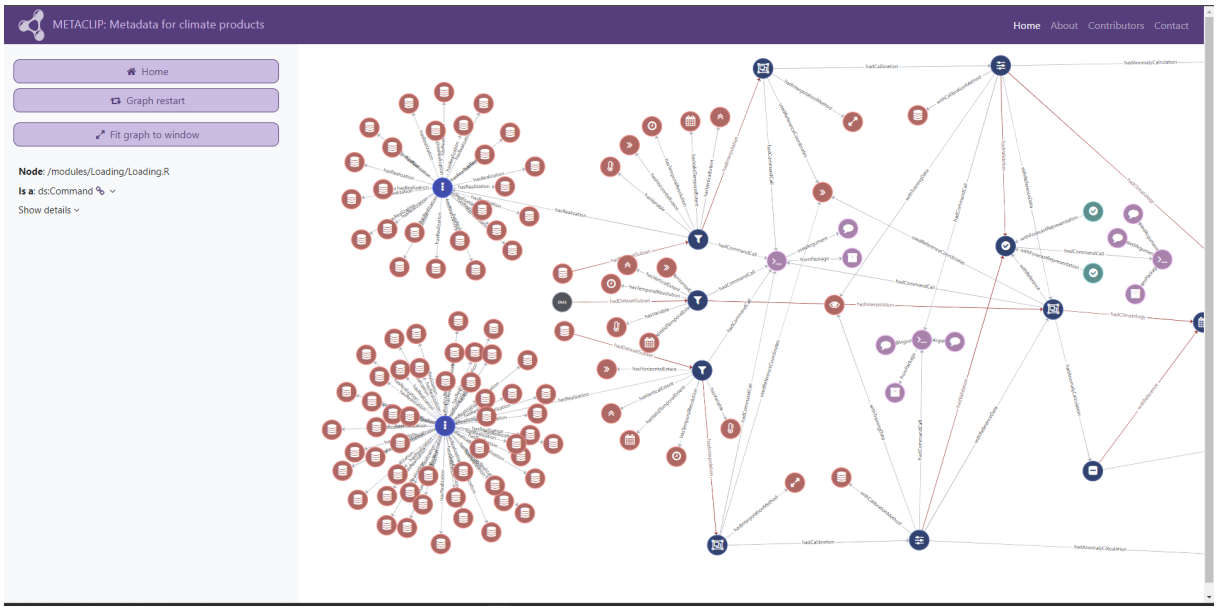

Figure 16: Example graph generated with SUNSET_PROVENANCE functions. The graph identifies the Member realization and the Downscaling step from the anomalies as `cal:CalibrationMethod`. Additionally, it shows the `veri:Verification` node connected to the computed Skill metrics. Overall, it illustrates almost the full capabilities of the SUNSET_PROVENANCE set of functions. The second part of the graph is provided in Figure 17.

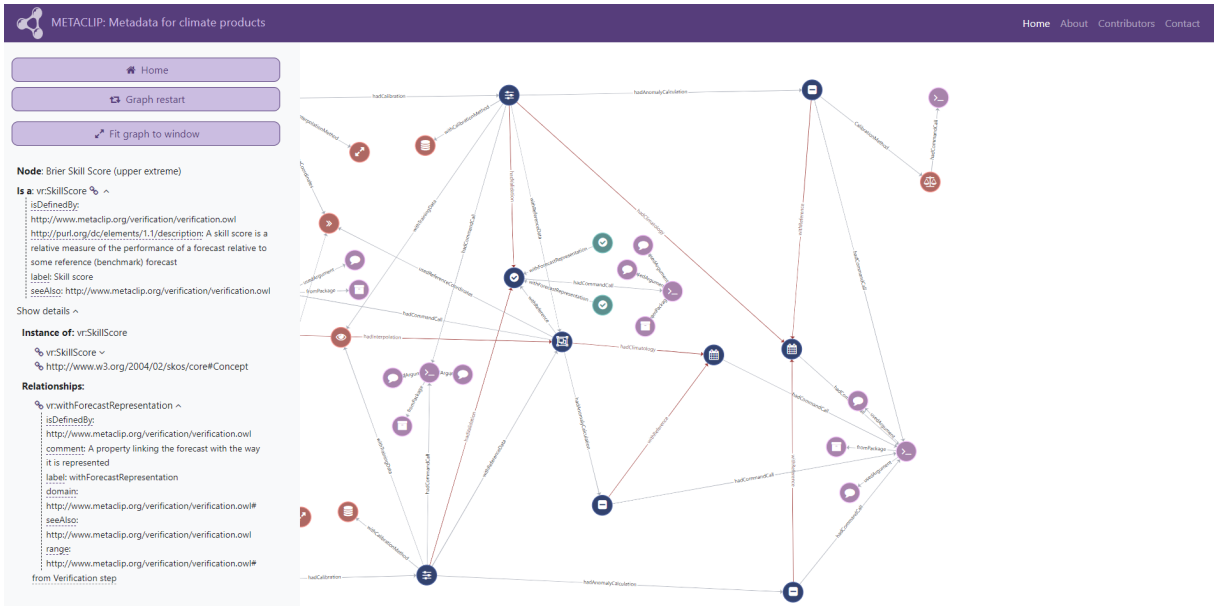

Figure 17: Second part of the graph shown in Figure 16.

7.4 Limitations of metaclipR in SUNSET

Despite the promising and advanced results compared to the preliminary project, the SUNSET_ PROVENANCE functions face several limitations. Firstly, the provenance generation is restricted to what metaclipR can offer, thus we have had to develop additional functions beyond the standard metaclipR functions, making us consider developing an entirely new set. There are issues with tracking and identifying nodes within the `igraph` object and its serialization in JSON format due to metaclipRs node definition, and it is challenging to add annotation properties easily. Moreover, most functions do not account for all SUNSET operations or methods, which limits their applicability. This reliance on the metaclipR package restricts the flexibility needed for a full provenance tracking framework for SUNSET.

This section is dedicated to providing a comprehensive examination of the limitations and incompatibilities faced with the metaclipR package when attempting to develop its implementation into SUNSET. Particularly, it delves into the challenges encountered when applying metaclipR outside climate4R, focusing on the inputs metaclipR functions require, its flexibility when generating and manipulating provenance graphs, and the suitability of the ontology definitions for describing provenance in SUNSET. Moreover, in this section we justify why, out of the first proposed options to carry on the project -using the metaclipR functions, developing in-house functions, or a hybrid approach- we decided on the second option, also considering several metaclipR functions for a hybrid approach.

Firstly, even though some functions can be directly implemented without the need of a climate4R object most of them require it. For example, the function `metaclipR.Datasubset()` generates provenance by extracting information directly from the climate4R object to define classes such as `ds:VerticalExtent` or `ds:TemporalInstant`. Consequently, we face the issue of having to convert data from `s2dv_cube` to climate4R object every time we want to generate provenance for each individual SUNSET module. That is, we would have to convert the data at least twice per module in order to properly extract the provenance from the object and run the SUNSET workflow. Additionally, even with the conversion function `climate4R2s2dv_cube()`, we encounter errors when defining grids, data subset grids, variables, members, and other elements due to the differing structures of the objects.

Moreover, metaclipR functions heavily rely on the UDG (User Data Gateway) repository, especially when defining the dataset and dataset subset. That is, they are developed such that when it detects a known dataset all provenance metadata is automatically added to the graph. When using an "unknown" dataset some functions do not recognize the definitions in the ontology or in the UDG repository and either do not execute or generate less provenance than expected.

Secondly, the metaclipR package is specifically developed to extract provenance information from a climate4R workflow execution. Thus, most of the functions are thought to be implemented after specific execution of climate4R, focusing exclusively on the way the packages are developed and on the transformations they execute. In that sense, metaclipR provides a reduced flexibility when implemented outside climate4R, resulting in less provenance generation. For example, SUNSET can load in a single module execution forecast, hindcast and observational data, but metaclipR only computes provenance for one dataset at the time. This limitation complicates the establishment of relationships among the transformed datasets. In addition, the metaclipR functions define the nodes with a function that generates a random string called

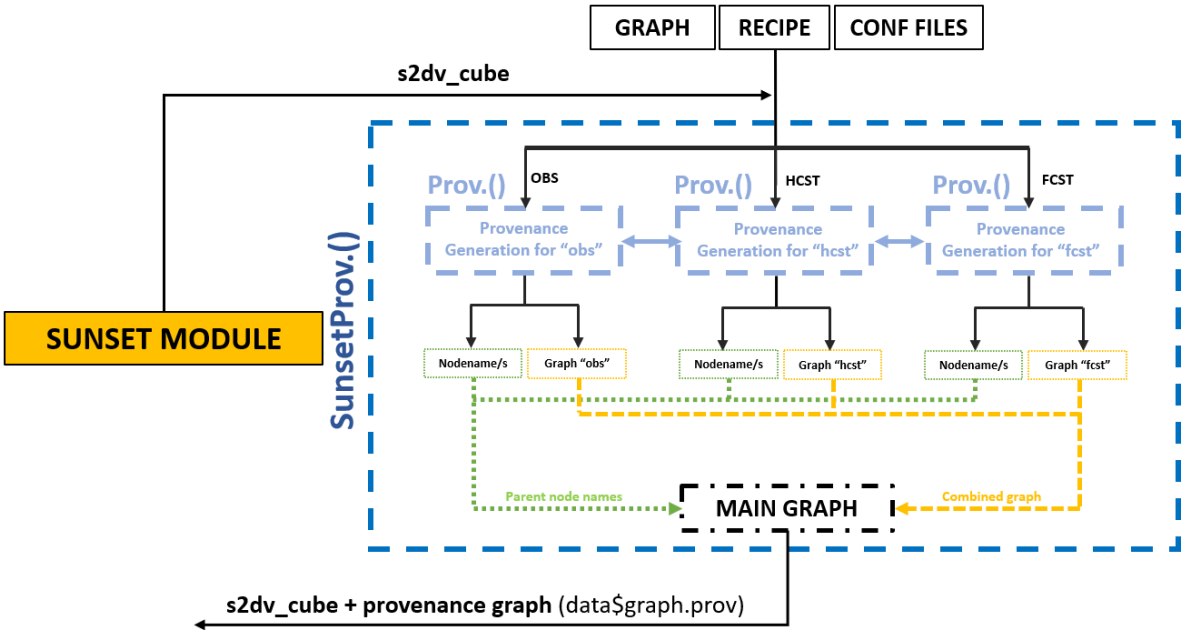

Figure 18: Structure of `SunsetProv()` functions.

`randomName()`. This makes the traceability, distinction, and connection between nodes complex to recognize and manipulate in the `igraph` object.

Due to the specificity of metaclipR, provenance generation is limited to the transformations and operations computed by climate4R functions. Thus, functions that generate provenance for certain transformations, such as `metaclipR.BiasCorrection()`, `metaclipR.Regridding()`, or `metaclipR.AnomalyCalculation()`, are limited to describing the methods established by climate4R. In other words, some transformation methods used by SUNSET cannot be established in the provenance graph using metaclipR. Additionally, metaclipR functions do not always allow the addition of annotation properties, such as comments, titles, or URLs, making it difficult to establish specific metadata of SUNSET.

Finally, in order to fully describe the provenance of a climate product from SUNSET we would have to expand the ontology defined by METACLIP and add more elements specific to certain data transformations. For example, `ds:DecadalHindcast` does not exist in the datasource ontology and it would be required to properly define the provenance of a hindcast dataset with a decadal temporal extent. Another example relates to SUNSET's Downscaling module. There is no function of the metaclipR package, nor a Class in the ontology, that describes downscaling. Further discussion regarding the ontology expansion can be found in the Section 7.6.

Based on the issues just described, we concluded we can not reach full provenance for SUNSET using only metaclipR. Therefore, we decided to develop in-house functions and consider a hybrid approach, adding auxiliary metaclipR functions where needed. This approach aims to enhance the flexibility and completeness of the provenance tracking framework for SUNSET, ultimately ensuring a robust and comprehensive provenance solution.

7.5 Creation of fully in-house functions to generate provenance in SUNSET: SUNSET_PROV

After encountering several issues with the metaclipR package, we decided to develop in-house functions to describe provenance for SUNSET. We identified that, metaclipR lacked functions to describe certain transformations executed in SUNSET, such as downscaling or unit transformation. Some functions were so specific to climate4R that they required modifications to function properly. Additionally, the way a provenance graph is defined in metaclipR makes it difficult to trace and identify the node names of all the represented steps, which complicates the provenance definition and management of complex workflows. Therefore, we decided to create our own functions to gain specificity in the provenance definitions and have a set of functions that could easily integrate within SUNSET, extracting information directly from the `s2dv_cube` when necessary. These in-house functions are a superset of the metaclipR functions which remain compliant with the METACLIP ontology defined in Section 6.2.5, ensuring compatibility and standardization with its elements. Furthermore, this section explains the structure and functionalities of these functions, as well as the collaborative approach with the METACLIP developers to expand the METACLIP ontology.

The new set of functions is called SUNSET_PROV and allows for a full provenance implementation in a workflow execution for atomic recipes[4]. Almost fully independent from the metaclipR package, SUNSET_PROV accounts for more specific and rigorous provenance generation within the SUNSET framework. This is achieved by introducing more complex relations in the graph generation and annotation properties such as `dc:title`, `dc:description`, `dc:referenceURL`, and others. Among several advantages, the ability to add these relations and properties allows for more complete and precise provenance to be included in the graph, surpassing the capabilities of metaclipR.

This means, in other words, that we can generate more specific provenance within METACLIP's framework, using its four ontologies while additionally adding more information. Moreover, SUNSET_PROV functions do not require a conversion function like `s2dv_cube2Climate4R` since they work directly with the `s2dv_cube` object. Moreover, SUNSET_PROV can create an `igraph` object, include it inside the SUNSET modules output as an element of the list called `graph.prov`, and upload it every time it is executed. The main provenance functions, with prefix "SunsetProv" work modularly, just like the SUNSET functions, which allow them to integrate seamlessly within the modules to generate provenance automatically. Additionally, a configuration file was created to store all information regarding calibration methods, interpolation methods, and metrics. The provenance function can extract valuable definitions and elements from this file and introduce them into the node using annotation properties [109].

7.5.1 Developed code

All functions from the SUNSET_PROV set of functions can be classified into three types. First, the `Prov()` functions are designed to replace metaclipR functions, with their primary purpose being to generate provenance graphs more specific to SUNSET. The differences with metalcipR

[4] Atomic recipes are recipes utilised for defining atomic executions. An atomic execution refers to the execution of a single, indivisible unit of work within a workflow or computational process. In the case of SUNSET, an atomic execution refers to a workflow in which a single climate model for a single climate variable is evaluated.

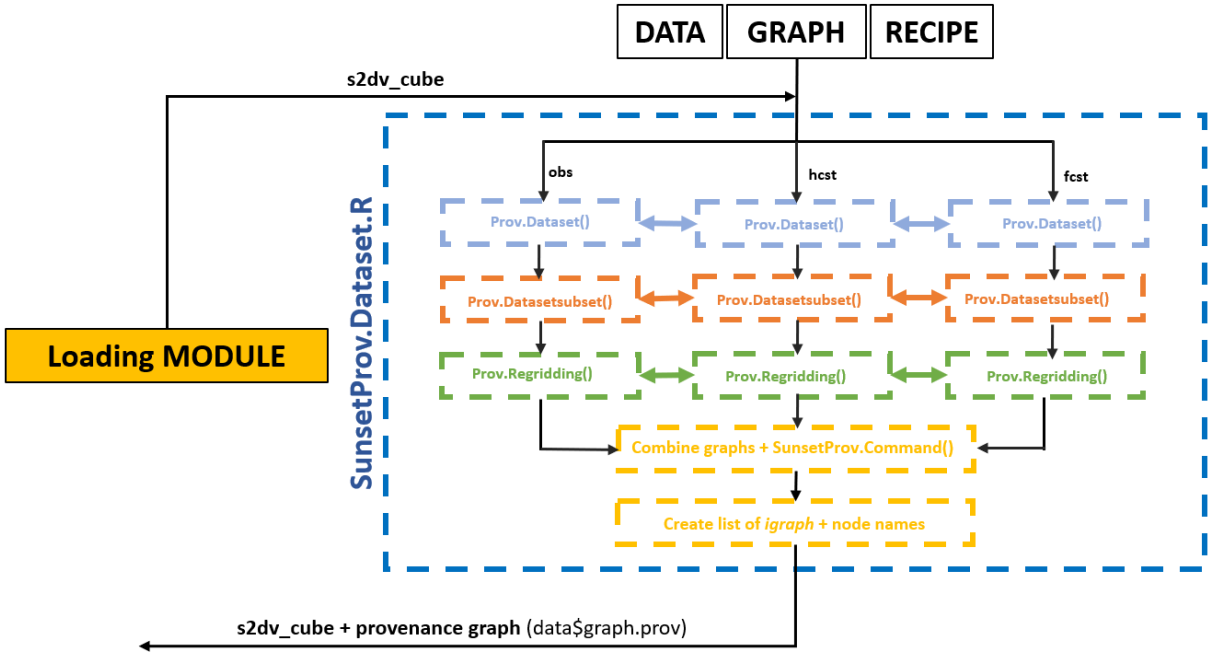

Figure 19: Structure of `SunsetProv.Dataset()`.

functions lies in the specificity of the provenance record, the simplicity of the node definition and their automation when extracting and defining provenance elements. Most `Prov()` functions, such as `Prov.Calibration()`, take as input the recipe, the processed data as a list of `s2dv_cube objects`, the prior provenance graph -containing the main graph and the last node names-, and a `type` parameter which is required to specify the type of dataset, either forecast ("fcst"), hindcast ("hcst") and observations ("obs"). Even if those are the predominant inputs, some `Prov()` functions use additional parameters to gain specificity on a particular operation. Then, the provenance information is extracted from `s2dv_cube)`, the recipe, and the configuration files; then generated by means of the igraph package.

Second, the `SunsetProv()` functions were developed to structure the `Prov()` functions for each specific SUNSET module and the operation or transformation taking place within it. That is, most `SunsetProv()` functions incorporate `Prov()` functions, structuring them based on the dataset, the relationships among elements, and the transformations applied. In other words, these functions execute at least twice the adequate `Prov()` function for each dataset, generating several graphs that will later be combined [5]. Additionally, these functions add the corresponding command call provenance and set the output list, which includes the main graph and node names.

Lastly, additional functions were developed as tools to support provenance generation and conversion. Notably, some of these tools include adapted metaclipR functionalities to complement

[5] Note that the multiple execution of the `Prov()` functions at each `SunsetProv()` function accounts for the three different types of datasets. Contrarily, in the prior section (7.4) we argumented that to the multiple executions for each dataset of the metaclipR functions, we had to add the number of times the conversion function `s2dv_cube2Climate4R()` was executed. This resulted in augmenting the number of functions executed to double

the SUNSET_PROV set of functions. A brief explanation of each function can be found in Table 3, for the `Prov()` functions, in Table 4, for the `SunsetProv()` functions, and in Table 5, for the auxiliary functions.

7.5.2 Creation of unit tests for continuous integration

To ease continuous integration in SUNSET and in the developed code, we created five unit tests. Unit tests are automated tests that check and validate the performance and functionalities of individual components of the code, in this case, by analyzing the outputs of each separate execution of a SUNSET_PROV function. Unit tests are crucial to ensure continuous integration, as they ensure each part of the developed code works correctly [110]. In software development, continuous integration refers to the practice where developers merge their code changes into a shared repository frequently. Each separate integration can afterwards be verified by automated unit tests. This approach allows for dissecting and addressing code-related issues or bugs early in the development stage, improving the quality and reducing the time taken to complete new features. To develop the unit tests we utilised the R package testthat, which is already used within SUNSET for similar purposes. This package is specifically developed for unit testing in R code, providing functions that compare the actual outputs with the expected results [111] [112][113].

Each unit test is designed to either test specific SUNSET_PROV functions or particular cases for a dataset loading or certain transformations. The aim was to create unit tests as diverse as possible to ensure all the features of the developed package are checked. The unit tests verify the structure of the generated provenance graph at each step of the workflow, including node names, node positions, node classes, and other aspects. They also confirm that the structure of the generated JSON file is correct. This approach helps in verifying the functionality and reliability of the functions across various scenarios. Each test is structured as follows:

- The **first unit test** is a simple atomic execution of SUNSET, running the Calibration, Anomalies, Skill and Visualization modules. It loads seasonal hindcast and observational data from ECMWF (European Centre for Medium-Range Weather Forecasts) and ERA5 respectively. The resulting provenance graph is shown in Figure 20. The recipe (F.1.1), script (F.1.2) and resulting JSON are provided (F.1.3) in the Appendices F.1.
- The **second unit test** is a simple atomic execution of SUNSET that computes all four Niño indices. The script runs the Loading, Anomalies, Indices, and Skill modules. It loads seasonal hindcast and observational data from ECMWF and ERA5 respectively. The resulting provenance graph is shown in Figure 21. The recipe (F.2.1), script (F.2.2) and resulting JSON are provided (F.2.3) in the Appendices F.2.
- The **third unit test** is a simple atomic execution of SUNSET, including the Downscaling modules as the main difference with the rest. It also runs the Loading, Anomalies, and Skill modules. It loads seasonal hindcast and observational data from ECMWF and ERA5 respectively. The resulting provenance graph is shown in Figure 22. The recipe (F.3.1), script (F.3.2) and resulting JSON are provided (F.3.3) in the Appendices F.3.
- The **fourth unit test** is a simple atomic execution of SUNSET, running the Loading, Calibration, Anomalies, Skill and Visualization modules. It loads seasonal forecast, hindcast

`Prov()` functions	Description
Prov.Anomalies.R [E.1.1]	Generates provenance for SUNSET Anomalies module execution. It adds two different nodes that represent transformations: `ds:Climatology` and `ds:Anomaly`, along with their corresponding relations.
Prov.Calibration.R [E.1.2]	Generates provenance for a SUNSET's Calibration module execution. At its core, it includes a `cal:Calibration` node in the graph, linked to the data to be calibrated by the property `cal:hadCalibration` and to the reference data by `cal:withReferenceData`. Other elements are included in the graph such as the Horizontal Extent. Relations are established for each type of dataset.
Prov.Dataset.R [E.1.3]	Prov.Dataset.R starts off the graph by means of the igraph function `my_empty_graph()` and creates the first dataset nodes. Further, it accounts for the origin of the dataset, defining the `ds:ModellingCenter`, `ds:DataProvider` and `ds:Project` nodes. It extracts all provenance information from the recipe and SUNSET configuration files.
Prov.DatasetSubset.R [E.1.4]	One of the most complex `Prov()` functions, it constructs `ds:DatasetSubset` as a central node. Then other nodes that define variables, horizontal spatial extent, vertical spatial extent, realizations, and temporal period are added and properly linked to the main dataset subset node. Primarily, it provides provenance for the selected dataset subset.
Prov.Regridding.R [E.1.5]	Prov.Regridding.R is developed to be executed after Prov.DatasetSubset.R and generates provenance for the regridding operation that takes place in the SUNSET Loading module. It retrieves the method and the target grid from the recipe, establishing the corresponding provenance nodes and relationships in the graph.
Prov.Skill [E.1.6]	This function retrieves all metrics computed in the Skill module from the recipe and generates their nodes, linking them to the verification main node. It takes an additional parameter, `Indices`, which accounts for cases where indices are computed before the Skill module. Moreover, it retrieves data from a configuration file to add a `dc:description` to each metric node.
Prov.Visualization.R [E.1.7]	Prov.Visualization.R generates nodes and vertices to describe graph generation in the Visualization module, commonly for each of the Skill module outputs. Moreover, the Visualization module can itself generate the final provenance JSON file format serializing the `igraph` object and embedding it into the output images.

Table 3: "Prov" functions

`SunsetProv()` functions	Description
SunsetProv.Anomalies.R [E.1.8]	Function to integrate inside SUNSET's Anomalies module to generate full provenance for anomaly calculation. It combines all the graphs generated by Prov.Anomalies.R, adds a command call node and sets the outputs.
SunsetProv.Calibration.R [E.1.9]	It was developed to integrate within the Calibration module and generate full provenance regarding a calibration step. It unites all graphs generated by Prov.Calibration.R and generates the command call node. Also, this function sets the outputs as a list containing the graph and node names.
SunsetProv.Dataset.R [E.1.11]	This function is designed to integrate within SUNSET's Loading module and executes Prov.Dataset.R, Prov.DatasetSubset.R and Prov.Regridding.R for each of the loaded datasets. Then, combines the two or three graphs, adds the command call provenance and sets the output list.
SunsetProv.Downscaling.R [E.1.12]	SunsetProv.Downscaling.R does not include any `Prov()` function inside. Consequently, it generates provenance directly using the igraph package functions. Since only hindcast data can be downscaled currently, this function generates provenance by exclusively creating one node to describe the downscaling operation.
SunsetProv.Indices.R [E.1.13]	This function does not incorporate any `Prov()` function, as all provenance is generated directly using igraph functions. SunsetProv.Indices.R is embedded within the Indices module, retrieving indices computed from the recipe and the `s2dv_cube`. It generates a provenance node for each index, providing documentation of the indices calculations.
SunsetProv.Skill.R [E.1.14]	It was developed to integrate within the Skill module and generate full provenance regarding a verification step. It unites all graphs generated by Prov.Skill.R and generates the command call node. Also, this function sets the outputs as a list containing the graph and node names.
SunsetProv.Visualization.R [E.1.15]	SunsetProv.Visualization.R is designed to generate provenance of the output files, currently for skill metrics and ensemble forecasts plots. First, it uploads the provenance graph according to the specified visualization options in the recipe. Second, it converts the `igraph` object to JSON. Finally, the function embeds the provenance information in JSON format in the image file.

Table 4: "SunsetProv functions.

Auxiliary functions	Description
getSpatialExtent.R	This function generates provenance information of the spatial coordinates of a climate data object in `s2dv_cube` format. Then it returns a graph containing a single SpatialExtent node.
graph2json.R	This metaclipR function is used to convert (serialize) an `igraph` object to JSON format.
keep_latest_node.R	From a list of nodenames of an `igraph` object, the function returns the last node created. It is used to link the last computed node with the ensemble forecast node in SunsetProv.Visualization.R.
igraph-helpers.R	From the metaclipR package, igraph-helpers.R contains a set of functions used as tools to manipulate and extract information from an `igraph` object seamlessly. These tools simplify tasks and help analyze data more efficiently.
metaclip-helpers.R	From the metaclipR package, metaclip-helpers.R contains a set of functions used as tools to generate provenance graphs. Built upon several igraph functions, these tools add additional features to ease tasks and help generate complex graphs.
embedJSON.R	This function embbeds the JSON file into images. We used it to add the provenance information to the SUNSET graphical outputs, each one with its corresponding provenance graph.

Table 5: Auxiliary functions.

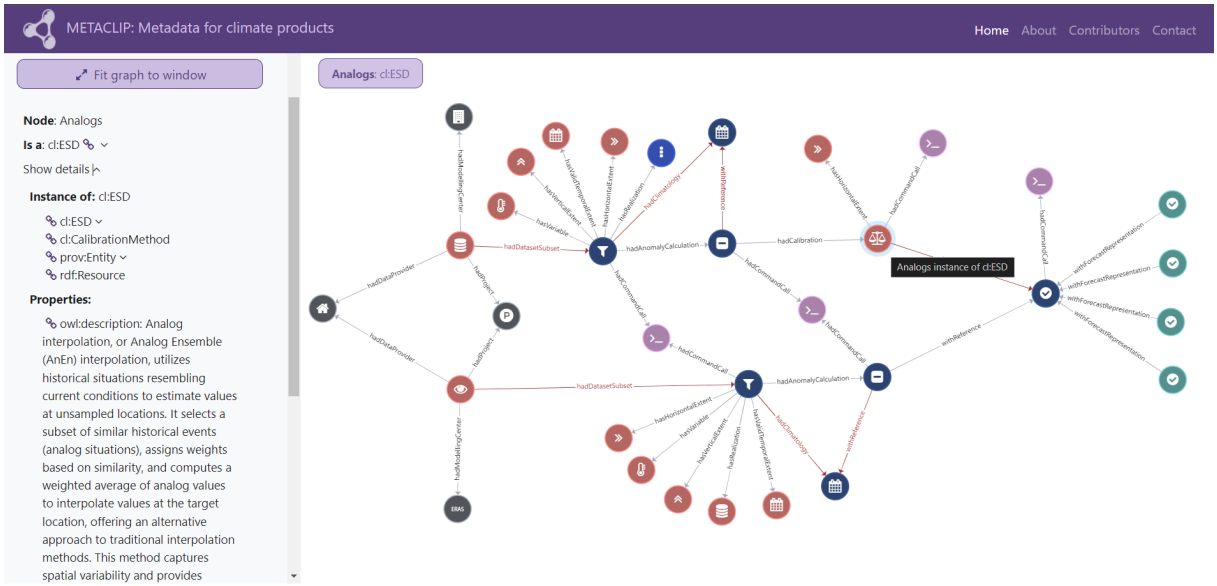

Figure 20: First unit test. Simple atomic execution of SUNSET, running the Calibration, Anomalies, Skill, and Visualization modules, loading seasonal hindcast and observational data from ECMWF (European Centre for Medium-Range Weather Forecasts) and ERA5 respectively. The graph is generated with the METACLIP Interpreter [114].

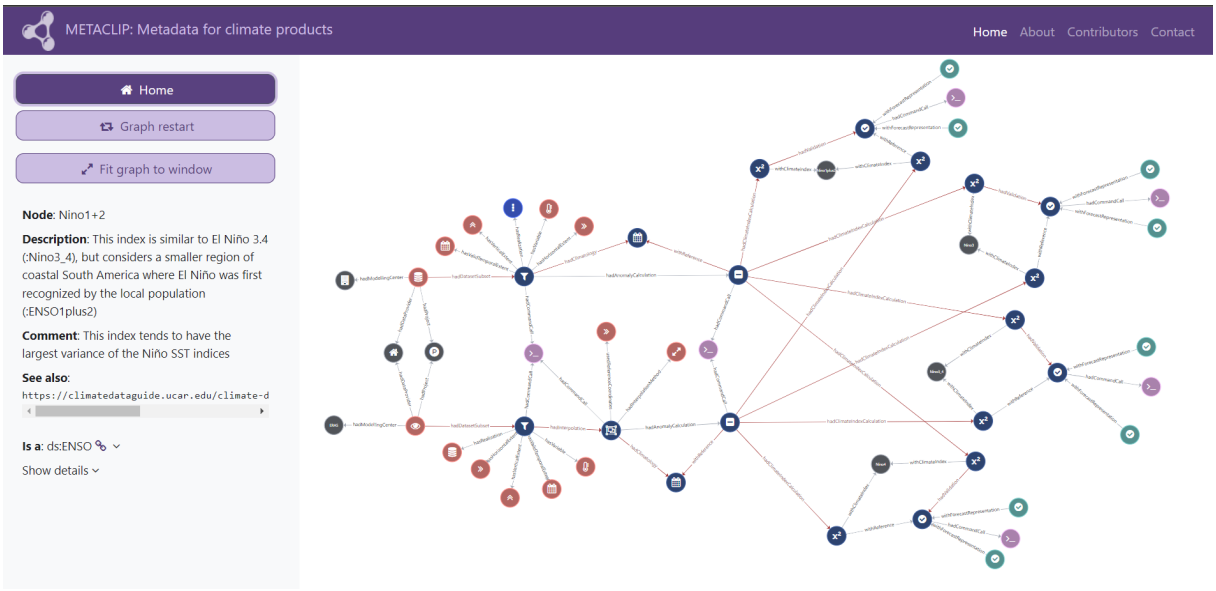

Figure 21: Second unit test. Simple atomic execution of SUNSET that computes all four El Niño indices, loading seasonal hindcast and observational data from ECMWF and ERA5 respectively. The graph is generated with the METACLIP Interpreter [114].

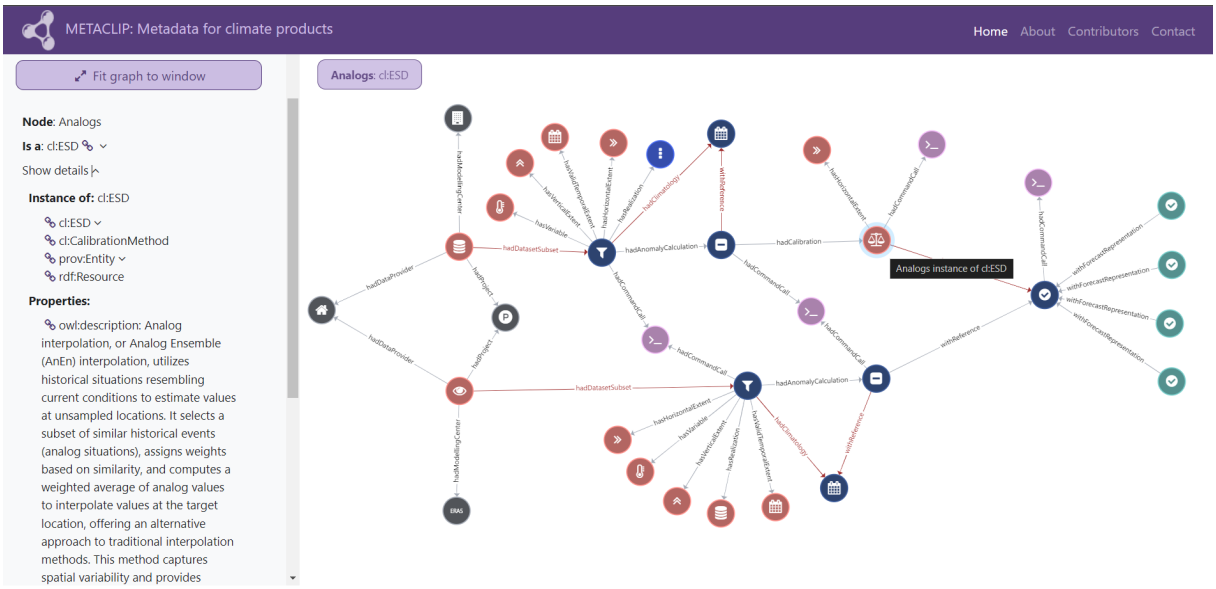

Figure 22: Third unit test. Simple atomic execution of SUNSET that computes a downscaling operation, loading seasonal hindcast and observational data from ECMWF and ERA5 respectively. The graph is generated with the METACLIP Interpreter [114].

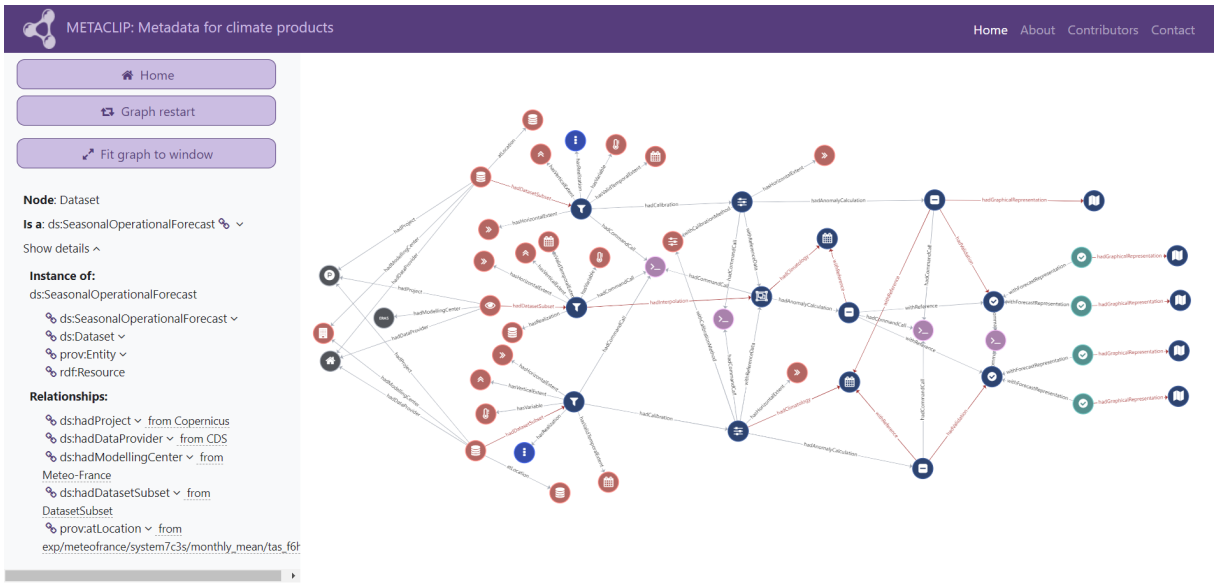

Figure 23: Fourth unit test. Simple atomic execution of SUNSET, loading seasonal forecast, hindcast and observational data from Meteo France System 7 and ERA5 respectively. The graph is generated with the METACLIP Interpreter [114].

and observational data from Meteo France System 7 and ERA5 respectively. The resulting provenance graph is shown in Figure 23. The recipe (F.4.1), script (F.4.2) and resulting JSON are provided (F.4.3) in the Appendices F.4.

- Laslty, the **fifth unit test** is a simple atomic execution of SUNSET, running the Loading, Calibration, Anomalies, Skill and Visualization modules. It loads decadal forecast, hindcast and observational data from EC-Earth3 and ERA5 respectively. The resulting provenance graph is shown in Figure 24. The recipe (F.1.1), script (F.1.2) and resulting JSON are provided (F.1.3) in the Appendices F.1.

No example execution is explained due to the simplicity of atomic execution with provenance integration. The developed provenance functions seamlessly integrate within the R code, allowing for provenance generation simply by adding an element to the recipe defined as `Provenance: yes`. The executions follow the same structure as they would without provenance generation. The only module that requires a specific input is the Visualization module, which requires the provenance graph as input. More details about each particular execution can be found in the Appendices E.2 and on the SUNSET GitLab repository [44].

Although executions with and without provenance generation follow a similar structure, there is a slight increase in time primarily attributed to the overhead introduced by capturing and generating provenance data. To measure this overhead, we considered the 'user' time (CPU time spent executing the user's code), 'system' time (CPU time spent executing system calls on behalf of the program), and 'elapsed time' (total wall-clock time from start to finish). The measurements are presented in Table 6 and Table 7. The results underscore that the additional computational load introduced by the provenance implementation is minor, indicating a low to moderate impact on performance.

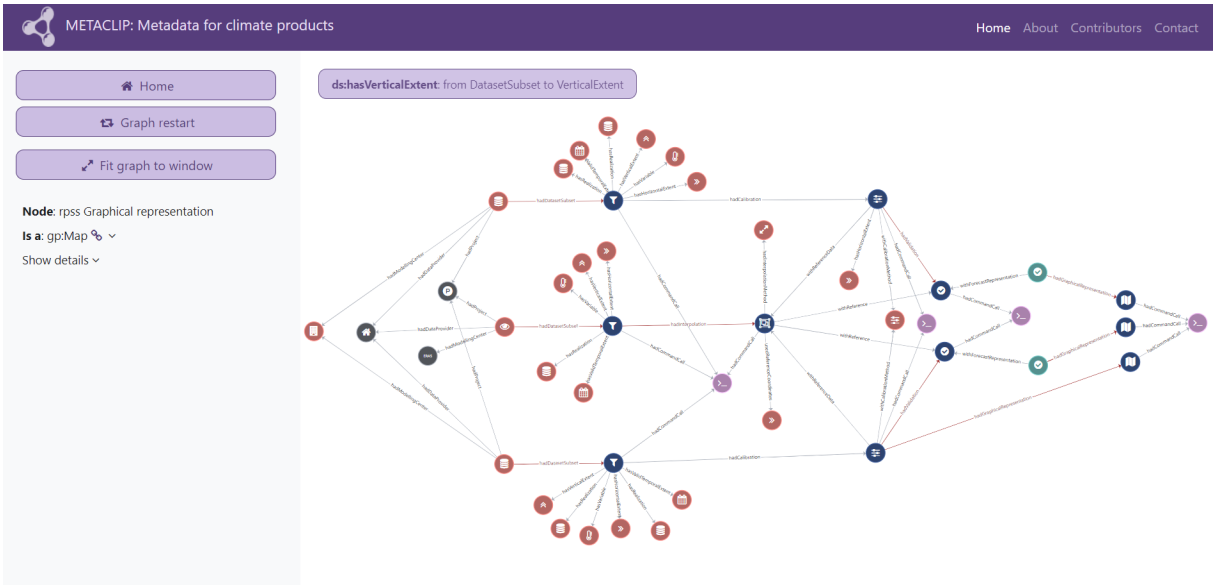

Figure 24: Fith unit test. Simple atomic execution of SUNSET, loading decadal forecast, hindcast and observational data from EC-Earth3 and ERA5 respectively. The graph is generated with the METACLIP Interpreter [114].

	user	system	elapsed
Without provenance	42.107	0.208	48.226
With provenance	48.005	0.425	59.810

Table 6: Execution times of Unit test 1, comparing the execution with provenance and without provenance generation

	user	system	elapsed
Without provenance	265.445	29.135	100.971
With provenance	266.392	32.186	117.152

Table 7: Execution times of Unit test 4, comparing the execution with provenance and without provenance generation

7.6 METACLIP ontology extension

As mentioned in the Theoretical Foundations section (Section 6), an ontology plays a crucial role when defining provenance elements, as information can be inferred by simply locating them within the ontology framework. Therefore, by classifying a data element or property within METACLIP, one can extract valuable information and provenance details not only from its definition but also through its complex relationships with other elements. This interconnected structure sustains and gives meaning to the data provenance generated. For the METACLIP framework, this is achieved with the interpreter, which illustrates the exact definition and hierarchy of each element within the ontology. However, to effectively implement an ontology framework, its definitions and relations must be accurately defined to associate with the elements from which we want to generate provenance.

In the scenario of implementing the METACLIP framework into SUNSET, we faced several challenges related to the ontology, as it does not encompass all the necessary terms and relationships required in this context of climate verification workflows. For example, METACLIP currently lacks a class to describe unit conversion, which is essential to properly define the operations undertaken in SUNSET's Units module. To address this issue, we had two options: either adapt the provenance generation code to fit the existing ontology, stretching the definitions beyond their range, or attempt to expand the current ontology with new terms and relations. The ontology extension approach involves adding new classes, individuals, and object properties to ensure all aspects of SUNSET -datasets, transformations, methods, etc- are accurately represented. Notably, most of the issue could be resolved using annotation properties, as they allow the addition of valuable information to the provenance definitions. However, other SUNSET elements necessarily required a specific term in the ontology to be adequately described.

To accomplish the desired expansion of the ontologies, we contacted METACLIP developers via GitHub and made a pull request to the vocabularies repository. The first objective was to include a new class in the Calibration ontology that described Downscaling transformations. We defined the new elements as a `cal:Downscaling` class, subclass of `ds:Calibration`, with the following `dc:description`: "This process involves the utilization of mathematical algorithms or numerical simulations to extrapolate localized climate phenomena and detailed spatial patterns from broader-scale atmospheric or oceanic information, thus facilitating more precise analyses and projections at regional or local scales.".

First, we made a fork of the METACLIP vocabularies repository in GitHub [115], where the changes were to be manually included. Then, we identified the integration point within the OWL file, "Calibration.owl", where the new definition should be added. Afterwards, based on OWL, RDF, and PROV standards the new class was defined, including its IRI, 4096 http://www.metaclip.org/calibration/calibration.owl#Downscaling, and the rest of properties `rd:SubClassOf`, `dc:description`, `dc:title` and `rdfs:seeAlso` with their corresponding objects. Note that when the class is defined as a subclass of `ds:Step`, the rest of subclasses that precede it are automatically associated with `cal:Downscaling`.

The implemented code within the OWL file is shown in Listing 15. In it, we can identify how relations and properties are defined in different ontologies such as Dublin Core[84], "dc",or RDFS[71], "rdfs", each of which defines a particular aspect of the class. Moreover, we can identify how a Class is defined according to OWL, `owl:Class`. The `rdf:about` attribute

specifies the unique IRI, `rdf:subClassOf` indicates that `cal:Downscaling` is a subclass of `cal:Calibration`, `dc:description` provides a detailed explanation of what the class entails, `dc:title` assigns a human-readable title, and `rdf:seeAlso` adds an external link for additional information.

```
<!-- http://www.metaclip.org/calibration/calibration.owl#Downscaling -->

<owl:Class rdf:about="http://www.metaclip.org/calibration/calibration.owl#
   Downscaling">
<rdfs:subClassOf rdf:resource="http://www.metaclip.org/calibration/
   calibration.owl#Calibration"/>
<dc:description>A downscaling operation refers to the computational
   procedure applied to transform coarser-resolution climate data into
   finer-resolution representations, typically achieved through statistical
    or dynamical methodologies. This process involves the utilization of
   mathematical algorithms or numerical simulations to extrapolate
   localized climate phenomena and detailed spatial patterns from broader-
   scale atmospheric or oceanic information, thus facilitating more precise
    analyses and projections at regional or local scales.</dc:description>
     <dc:title>Downscaling</dc:title>
     <rdfs:seeAlso>https://en.wikipedia.org/wiki/Downscaling</rdfs:seeAlso>
</owl:Class>
```

Listing 16: RDF in TURTLE syntax

Overall, the main objective of the ontology expansion was not only to generate more specific provenance for SUNSET but also to establish a precedent and a methodology for adding new terms in the future. This approach ensures that new elements can be integrated into the ontology, enhancing the completeness of provenance generation. Moreover, this process involved a collaborative effort with the METACLIP developers, highlighting the importance of community collaboration in software development.

	Development of the project	Execution of the project	Risks and limitations
Environment	Environmental Impact	Environmental Impact	Environmental risks and limitations
Economics	Cost	Viability analysis	Economic risks and limitations
Society	Social Impact	Social Impact	Social risks and limitations

Table 8: Sustainability matrix defined by the UPC [116].

8 Sustainability Report

In this section, we analyze the sustainability impact of this project across several areas including environmental, economic, and social impacts. Sustainability reports are increasingly important in both scientific developments and business contexts, as they promote the responsible management of resources and the minimization of negative environmental and social effects. Moreover, this type of analysis also encourages economic viability and social well-being. Through a detailed examination of these aspects, this section aims to provide an overview of the project's contributions to sustainable development and its relation with the sustainable development goals of the 2030 Agenda.

8.1 Sustainability matrix

In the Table 8, we can find what we refer to as the sustainability matrix, a comprehensive tool utilised to assess the sustainability impact of a project in various fields established by the UPC (Universitat Politècnicna de Catalunya)[116]. It serves as a structured framework for organizing and analyzing the environmental, economic, and social aspects of the work done. For this project, we employ the matrix to structure the sustainability impact of the provenance implementation in SUNSET, considering all aspects involved in its approach and execution.

8.1.1 Environmental impact

The environmental impact of the project, primarily focusing on software development and computational executions — including data loading, climate data processing, and provenance generation — is relatively minimal. This is attributed to the project's need for additional computational resources, which, while in use, have a limited overall impact on the environmental footprint. Notably, deliberate efforts were made to mitigate the ecological impact by avoiding excessive resource consumption. This approach involved minimizing reliance on clusters and HPC (High Performance Computing) structures, such as the MN5 (MareNostrum 5). Instead, code development, testing, and executions were primarily conducted locally, utilising BSC workstations (Local PCs). Moreover, during executions, a limited amount of data was utilised to reduce the computational cost, and the underlying energy consumption. Overall, by leveraging local resources and minimizing energy consumption, the project aimed to reduce its environmental footprint without compromising on performance or efficiency.

To measure the environmental impact of the Bachelor's Thesis, we considered the CO2 emissions generated by the use of the personal laptop and the workstation (as they were used simultaneously). We calculated the CO2 emissions over a span of 615 hours and considered the power consumption of a laptop to be 200 watts and 100 watts for the workstation. While the

usual laptop consumption typically varies from 30 to 70 watts, we opted for a higher value of 200 watts, taking into account the power adapter's capacity and the potential maximum power draw. Similarly, the workstation's power consumption was estimated at 100 watts . Furthermore, we consider the CO2 emissions to be an average of 0.5 kg of CO2 per kW per hour [117] and the average CO2 absorption of a tree 25 kg CO2 per tree per year [118]. The results are shown below.

Formulas

$$\text{Energy (kWh)} = \text{Power (kW)} \times \text{Time (hours)} \tag{1}$$
$$\text{CO}_2 \text{ Emissions (kg)} = \text{Energy (kWh)} \times \text{CO2(kg CO}_2\text{/kWh)} \tag{2}$$
$$\text{Carbon Absorption per Tree per Year} = 25 \text{ kg CO}_2 \tag{3}$$

Laptop

$$\text{Energy (kWh)} = 0.2 \text{ kW} \times 615 \text{ hours} \tag{4}$$
$$\text{Energy (kWh)} = 123 \text{ kWh} \tag{5}$$
$$\text{CO}_2 \text{ Emissions (kg)} = 123 \text{ kWh} \times 0.5 \text{ kg CO}_2\text{/kWh} \tag{6}$$
$$\text{CO}_2 \text{ Emissions (kg)} = 61.5 \text{ kg CO}_2 \tag{7}$$
$$\text{Number of Trees in a year} \approx \frac{61.5 \text{ kg CO}_2}{25 \text{ kg CO}_2\text{/tree}} \tag{8}$$
$$\text{Number of Trees in a year} \approx 2.46 \text{ trees/year} \tag{9}$$

Workstation

$$\text{Energy (kWh)} = 0.1 \text{ kW} \times 615 \text{ hours} \tag{10}$$
$$\text{Energy (kWh)} = 61.5 \text{ kWh} \tag{11}$$
$$\text{CO}_2 \text{ Emissions (kg)} = 61.5 \text{ kWh} \times 0.5 \text{ kg CO}_2\text{/kWh} \tag{12}$$
$$\text{CO}_2 \text{ Emissions (kg)} = 30.75 \text{ kg CO}_2 \tag{13}$$
$$\text{Number of Trees in a year} \approx \frac{30.75 \text{ kg CO}_2}{25 \text{ kg CO}_2\text{/tree}} \tag{14}$$
$$\text{Number of Trees in a year} \approx 1.23 \text{ trees/year} \tag{15}$$

Total

$$\text{Total Energy (kWh)} = 123 \text{ kWh} + 61.5 \text{ kWh} \tag{16}$$
$$\text{Total Energy (kWh)} = 184.5 \text{ kWh} \tag{17}$$
$$\text{Total CO}_2 \text{ Emissions (kg)} = 61.5 \text{ kg CO}_2 + 30.75 \text{ kg CO}_2 \tag{18}$$
$$\text{Total CO}_2 \text{ Emissions (kg)} = 92.25 \text{ kg CO}_2 \tag{19}$$
$$\text{Total Number of Trees in a year} \approx \frac{92.25 \text{ kg CO}_2}{25 \text{ kg CO}_2\text{/tree}} \tag{20}$$
$$\text{Total Number of Trees in a year} \approx 3.69 \text{ trees/year}$$

The results reveal that the consumption when utilising the laptop and the workstation simultaneously results in considerable emissions. The environmental impact assessment reflects that the combined emissions from the laptop and the workstation are approximately 92.25 kg CO2. To compare with, these emissions are equivalent to a carbon captured by approximately 4.32 mature trees.

In summary, the analysis underlines the importance of considering energy consumption and carbon emissions in a software development environment, highlighting the need for sustainable practices. Additionally, quantifying the emissions provides valuable insights and informs future decision-making processes to promote sustainability.

8.1.2 Economic impact

This section aims to expose the economic impacts of the project, delving into its economic approximate cost and its direct impact on benefit maximization. Moreover, we briefly analyze possible economic limitations related to the project, as well as its long term viability and maintenance.

On one hand, the project's cost, excluding additional expenses such as transport, food, and taxes incurred during the total hours worked, primarily consisted of the personal hours invested in research and development. Assuming the average salary of an ETSEB (Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona) student is €11/hour and a total of 615 hours worked, the total monetary cost amounts to approximately €6,765. It's worth noting that this figure could vary significantly if other factors are considered, such as taxes paid by the business (BSC), individual taxes, contract clauses, etc. On average, this represents the cost of developing a provenance integration within a software by an undergraduate student from the ETSEB.

On the other hand, the implementation of adequate data provenance within a workflow or a data processing framework can lead to a potential increase in economic benefits. Empowering the trustworthiness and reliability of data through a well-recognized provenance framework can result in companies that acquire it to increase their demand. The increased demand can translate into higher revenue and improved market positioning. Additionally, it may foster innovation and efficiency within the organization, contributing to long-term economic viability.

Lastly, another economic aspect to consider is the future maintenance of the provenance implementation within SUNSET. As stated in Section 6.1.4, SUNSET is a tool that is still in the early development stages, which will require constant maintenance of the provenance functions to adapt to its changes and developments. Therefore, to preserve the provenance framework within SUNSET, resources will have to be allocated to its maintenance, increasing the project's long-term economic impact.

8.1.3 Social impact

The social impact of the Bachelor's Thesis includes social long term effects and broader implications. In the current global context, climate change stands as one of the most pressing challenges, affecting nature and communities worldwide. Even if the project does not directly contribute to climate change effects mitigation, its promotion of responsible and effective climate data management considerably contributes to improving in the current climate crisis scenario.

Enhancing data provenance within climate forecast verification and post-processing frameworks allows for reproducibility of experiments, accelerates the evolution of research, and provides a stronger foundation for the obtained results. This enables for more accurate predictions and assessments of climate trends, which are crucial for identifying adaptation and mitigation strategies. At the same time, the trustworthiness of data prevents data trafficking and other unethical practices.Then, in the long term, this provenance implementation contributes to improving scientific research and advancement to tackle the challenge of climate change, as well as the decision-making strategies derived from climate predictions. Moreover, the use of SUNSET ranges from agriculture to the sustainable energy industry, actively contributing to green and sustainable practices. These industries use SUNSET to generate reliable forecasts and make strategic decisions based on them.

Beyond the earth science scenario, the project highlights the importance of trust and reliability across web resources, enabling users and companies to access well-defined pieces of information. By establishing clear provenance data, the implementation enhances transparency and accountability in digital ecosystems. Overall, the data reliability proposed by this project enhances user experiences and strengthens the foundations of digital infrastructure, promoting innovation and collaboration. Similarly, the collaborative approach -mostly based on the collaboration with METACLIP developers- demonstrates the importance of cooperation within the software development area, allowing for faster improvements. Moreover, this collaborative ethos ensures the final product meets diverse user and developer needs more effectively.

8.2 Ethical implications

This project responds to the need for reliable and effective climate data management. This need arises from the growing area of climate data modeling and the subsequent demand to ensure data accuracy, reproducibility, and transparency. Considering the ethical implications, data provenance within climate data processing is substantial for policy decision-making and contributes to the development of more sustainable and trustworthy practices.

The project follows the professional code of ethics applied in data science and software development, which ensures open dealings, responsibility, and careful handling of data at all stages. Additionally, it complies with the ethical code of the UPC [119], which governs research ethics at the Universitat Politècnica de Catalunya. These commitments are articulated in the thesis to guarantee adherence to ethical standards, thus consolidating the integrity of the project.

8.3 Relationship with the Sustainable Development Goals

The 2030 Agenda for Sustainable Development [120] was adopted by the United Nations General Assembly in 2015, marking a historic commitment by all member states to achieve a better and more sustainable future. It introduces 17 Sustainable Development Goals (SDGs) [121], each targeting different but interconnected area of developmement, from eradicating poverty and hunger to promoting peace, partnerships, and addressing social and ecological challenges.

This bachelor's degree project aligns closely with several of these SDGs. However, it particularly aligns with the 13th Sustainable Development Goal, which is defined as "Climate Action".

Figure 25: Sustainable Development Goals (SDGs) [121].

SDG 13 aims to take urgent action to combat climate change and its impacts by regulating emissions and promoting developments in renewable energy.

9 Future work and developments

In this section, we briefly delve into possible future developments of the SUNSET_PROV framework and, more generally, of provenance integration. One of the primary objectives in the near future is to further expand the METACLIP ontology to include more terms and classes that can be suitable to describe SUNSET transformations, such as unit conversion or probability operations. By doing so, the SUNSET provenance framework can significantly grow and describe data more precisely.

Another significant development is to expand the capabilities of the SUNSET_PROV set of functions to handle multi-variable and multi-model workflows. Currently, the provenance framework is designed for atomic workflows. To accommodate more complex executions within SUNSET, we plan to develop a more flexible code that would allow for the generation of complete provenance.

Furthermore, we plan improve the testing coverage by incorporating checks that verify whether the defined classes and elements belong to the ontologies. This involves connecting the provenance functions directly with the ontologies, checking definitions and relations directly from the OWL files of the remote repository. This would ensure that the provenance framework stays updated with METACLIP developments and changes. Lastly, within the software area, we plan to integrate a coverage tool to check the extent to which the unit tests cover the in-house code. This way, we can verify the correct functionalities of the test scripts and improve them to cover a higher percentage of the developed code.

Beyond SUNSET, there is potential to implement similar provenance frameworks, utilising other ontologies or even generating our own, across other workflows or software within the department. The growing interest in data provenance makes it a very valuable feature to integrate in data management and processing programs. Additionally, there exists an opportunity to implement provenance in Autosubmit, a workflow manager developed in the BSC [122]. Additionally, as SUNSET already supports running workflows with the Autosubmit workflow manager, and since Autosubmit adopts the RO-Crate Workflow Run Profile standard for provenance, there is now a possibility to combine RO-Crate with METACLIP, to have an implementation that reports the general workflow provenance, with METACLIP complementing it with the details of the post-processing. [123].

10 Conclusions

As initially stated, this project aimed to incorporate METACLIP's provenance framework into SUNSET. The initial discussions on the best approach for achieving provenance integration led to the development of in-house functions known as the SUNSET_PROV set of functions. This approach required a thorough analysis of the incompatibilities encountered with the metaclipR package. Along with the preliminary project work, the discussion provided a deeper understanding of the structure of provenance functions of a workflow similar to SUNSET, as well as a deeper comprehension of the languages used to define provenance elements. Overall, these early stages, which focused on research and short code developments, established the foundation of the provenance integration in SUNSET.

The SUNSET_PROV set of functions ultimately provided a close-to-complete provenance implementation within SUNSET. These functions embedded extensive and detailed data provenance into the workflow outputs as needed, including information of every transformation undertaken, directories, descriptions and reference URLs. Specifically, they achieve a full provenance description for atomic executions, ensuring that every step of the process is meticulously documented and traceable.

Provenance files generated by the SUNSET_PROV functions are fully-readable by the METACLIP Interpreter, enhancing user interactivity and accessibility. This capability allowed for immediate feedback and validation of the provenance data, ensuring that all transformations and processes were accurately recorded and easily accessible. By facilitating this level of interaction and transparency, the integration significantly improves the usability and reliability of the provenance information, making it a valuable asset for both developers and users. Moreover, we were able run the interpreter locally.

On top of the technical aspect of this Bachelor's degree thesis, we would also like to highlight the collaboration with the creators of METACLIP. By engaging with an external group and adhering to open-source best practices, the project exemplifies effective software engineering practices. The contributions made to the project not only enhance SUNSET but also provide valuable feedback and improvements to METACLIP. This aspect of the work underscores the importance of community engagement and iterative development in producing robust and adaptable software solutions. Such practices are essential for good software engineering and reflect a commitment to advancing both the SUNSET framework and the broader open-source community.

In conclusion, the integration of METACLIP's provenance framework into SUNSET through the development of SUNSET_PROV successfully achieved the proposed objectives. This project not only combines two software systems but also sets a precedent for future projects seeking to incorporate detailed data provenance or extend the current functions. The skills and knowledge acquired throughout the stay at the BSC, including the importance of meticulous documentation, collaboration, and adherence to open-source principles, are paramount in the field of software engineering. Overall, this thesis represents a meaningful contribution to the ongoing evolution of provenance frameworks and their application in complex workflows.

References

[1] World Wide Web Consortium. *PROV: The Provenance Ontology*. Accessed June 2024. 2011. URL: https://www.w3.org/2011/prov/wiki/Main_Page.

[2] Earth System Documentation (ES-DOC). *ES-DOC Search Portal*. Accessed: 2024-06-11. 2024. URL: https://search.es-doc.org/.

[3] Earth System Documentation (ES-DOC). *ES-DOC Homepage*. Accessed: 2024-06-11. 2024. URL: https://es-doc.org/.

[4] J. Bedia et al. *The METACLIP semantic provenance framework for climate products*. Vol. 119. Elsevier, 2019, pp. 445–457. DOI: 10.1016/j.envsoft.2019.07.005. URL: https://doi.org/10.1016/j.envsoft.2019.07.005.

[5] V. Agudetse et al. *SUNSET: the SUbseasoNal to decadal climate forecast post-processiNg and asSEssmenT suite*. CERISE-101082139. Reading, UK, Jan. 2024.

[6] NOAA National Centers for Environmental Information. *Weather vs. Climate*. Accessed March 2024. 2018. URL: https://www.ncei.noaa.gov/news/weather-vs-climate.

[7] National Geographic Society. *Weather or Climate: What's the Difference?* Accessed February 2024. 2023. URL: https://education.nationalgeographic.org/resource/weather-or-climate-whats-difference/.

[8] Wharton School of the University of Pennsylvania. *FAQ*. Accessed Feburary 2024. 2012. URL: https://web.archive.org/web/20120711003404/http://qbox.wharton.upenn.edu/documents/mktg/research/FAQ.pdf.

[9] United Nations Institute for Training and Research (UNITAR). *Predicting and Projecting*. Accessed March 2024. 2024. URL: https://unitar.org/sites/default/files/media/publication/doc/guide_predicting_and_projecting.pdf.

[10] King's Centre for Visualization in Science. *Climate Model Hindcasting*. Accessed March 2024. 2024. URL: https://applets.kcvs.ca/ClimateModelHindcasting/.

[11] K. McGuffie and A. Henderson-Sellers. *The Climate Modelling Primer*. 4th. Accessed March 2024. Wiley-Blackwell, 2014. URL: https://bcs.wiley.com/he-bcs/Books?action=index&itemId=111994337X&bcsId=8794.

[12] B. Rose. *The climate system and climate models*. Accessed March 2024. 2024. URL: https://brian-rose.github.io/ClimateLaboratoryBook/courseware/climate-system-models.html.

[13] World Meteorological Organization. *Essential Climate Variables*. Accessed March 2024. 2024. URL: https://gcos.wmo.int/en/essential-climate-variables/table.

[14] CREAF. *Copernicus Climate Change Service (C3S): Production of Essential Climate Variable Datasets based on Earth Observations - Albedo, FAPAR & LAI*. Accessed March 2024. 2016. URL: https://www.creaf.cat/copernicus-climate-change-service-c3s-production-essential%20-climate-variable-datasets-based-earth-observations-albedo-fapar-lai.

[15] UCAR. *Ocean|Climate Data Guide*. Accessed March 2024. 2023. URL: https://climatedataguide.ucar.edu/variables/ocean.

[16] W. M. Washington and C. L. Parkinson. *An Introduction to Three-Dimensional Climate Modeling*. University Science Books, 2005. URL: https://books.google.es/books?hl=es&lr=&id=6RQ3dnjE8lgC&oi=fnd&pg=PR11&dq=numerical+models+and+

predictions+climate&ots=idW_ZrjW8r&sig=6xCREcaQaloUN6yqu30C3mG4XLg#v=onepage&q=numerical%20models%20and%20predictions%20climate&f=false.

[17] World Meteorological Organization. *Forecast Verification - Methods and FAQ*. Accessed April 2024. 2009. URL: https://www.cawcr.gov.au/projects/verification/verif_web_page.html.

[18] S2S4E. *Climate Services Factsheets Guide*. Accessed March 2024. 2020. URL: https://s2s4e.eu/climate-services/case-studies/factsheets_guide.

[19] C. Buontempo et al. *Climate Service Development*. Accessed April 2024. 2014. URL: https://www.sciencedirect.com/science/article/pii/S2212096314000321?via%3Dihub.

[20] A. H. Murphy. *What is a good forecast? An essay on the nature of goodness in weather forecasting*. Vol. 8. 2. 1993, pp. 281–293. DOI: \doi{10.1175/1520-0434(1993)008<0281:WIAGFA>2.0.CO;2}.

[21] World Meteorological Organization. *Forecast Verification - Methods and FAQ*. Accessed April 2024. 2024. URL: https://www.cawcr.gov.au/projects/verification/verif_web_page.html.

[22] The Climate Data Factory. *Data processing*. Accessed March 2024. 2021. URL: https://help.theclimatedatafactory.com/en/article/data-processing-8ei0ku/.

[23] National Center for Atmospheric Research Staff (Eds). *The Climate Data Guide: Climate Data Processing Software*. Accessed April 2024. 2017. URL: https://climatedataguide.ucar.edu/climate-tools/climate-data-processing-software.

[24] R. Sluiter. *Interpolation Methods for Climate Data*. Accessed March 2024. 2009. URL: https://www.researchgate.net/publication/242783501_Interpolation_methods_for_climate_data.

[25] William Fleetwood Sheppard. *Interpolation*. Ed. by Hugh Chisholm. 11th ed. Vol. 14. 1911, pp. 706–710.

[26] H. Dobesch, P. Dumolard, and I. Dyras, eds. *Spatial Interpolation for Climate Data: The Use of GIS in Climatology and Meteorology*. ISTE Ltd., 2007.

[27] O. E. Tveito et al. *The Use of Geographic Information Systems in Climatology and Meteorology*. Springer, 2006.

[28] Copernicus Climate Change Service. *Bias Correction*. Accessed: 2024-06-17. 2021. URL: https://climate.copernicus.eu/sites/default/files/2021-01/infosheet7.pdf.

[29] Uwe Ehret et al. "HESS Opinions: Should we apply bias correction to global and regional climate model data?" In: *Journal of Climate* 28.17 (2015), pp. 6407–6417. DOI: 10.1175/JCLI-D-14-00754.1. URL: https://journals.ametsoc.org/view/journals/clim/28/17/jcli-d-14-00754.1.xml.

[30] Douglas Maraun. "Bias correction, quantile mapping, and downscaling: Revisiting the inflation issue". In: *Journal of Climate* 26.6 (2013), pp. 2137–2143. DOI: 10.1175/JCLI-D-12-00821.1. URL: https://journals.ametsoc.org/view/journals/clim/26/6/jcli-d-12-00821.1.xml.

[31] Climate Prediction Center. *Climate Glossary*. Accessed March 2024. 2006. URL: https://web.archive.org/web/20061006000000/http://www.cpc.ncep.noaa.gov/products/outreach/glossary.shtml.

[32] O. Pita-Díaz and D. Ortega-Gaucin. *Analysis of Anomalies and Trends of Climate Change Indices in Zacatecas, Mexico*. Vol. 8. 4. 2020, p. 55. DOI: 10.3390/cli8040055.

[33] J. Tang. *Statistical downscaling and dynamical downscaling of regional climate in China: Present climate evaluations and future climate projections*. Wiley Online Library, 2016. DOI: 10.1002/2015JD023977. URL: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1002/2015JD023977.

[34] Geophysical Fluid Dynamics Laboratory. *Climate Model Downscaling*. Accessed March 2024. URL: https://www.gfdl.noaa.gov/climate-model-downscaling/.

[35] National Center for Atmospheric Research. *Climate Indices*. Accessed March 2024. URL: https://climatedataguide.ucar.edu/climate-data/indices.

[36] National Center for Atmospheric Research. *Nino SST Indices (Nino 1+2, 3, 3.4, 4; ONI and TNI)*. Accessed March 2024. URL: https://climatedataguide.ucar.edu/climate-data/nino-sst-indices-nino-12-3-34-4-oni-and-tni.

[37] Copernicus Climate Data Store. *Resampling and Aggregating Data*. Accessed March 2024. URL: https://cds.climate.copernicus.eu/toolbox/doc/how-to/14_how_to_resample_and_aggregate/14_how_to_resample_and_aggregate.html.

[38] Met Office. *The Science Behind the Probability of Precipitation*. Accessed March 2024. 2018. URL: https://web.archive.org/web/20181016041811/https://www.metoffice.gov.uk/news/in-depth/science-behind-probability-of-precipitation.

[39] American Meteorological Society. *Weather Analysis and Forecasting*. Accessed March 2024. 2021. URL: https://www.ametsoc.org/index.cfm/ams/about-ams/ams-statements/statements-of-the-ams-in-force/weather-analysis-and-forecasting2/.

[40] T.M. Hamill and J. Juras. *Measuring Forecast Skill: Is it Real Skill or is it the Varying Climatology?* Vol. 132. 2006, pp. 2905–2923. DOI: 10.1002/qj.4120. URL: https://rmets.onlinelibrary.wiley.com/doi/full/10.1002/qj.4120.

[41] EUMeTrain. *Skill Scores*. Accessed April 2024. URL: https://resources.eumetrain.org/data/4/451/english/msg/ver_cont_var/uos5/uos5_ko1.htm#:~:text=The%20skill%20score%20in%20this,improvement%20over%20the%20standard%20forecast.

[42] D.S. Wilks. *Statistical Methods in the Atmospheric Sciences*. 3rd. Accessed March 2024. Elsevier, 2011. URL: http://store.elsevier.com/Statistical-Methods-in-the-Atmospheric-Sciences/Daniel-Wilks/isbn-9780123850225/.

[43] U.S. Government. *Climate Data and Prediction Analysis*. Accessed March 2024. URL: https://corpora.tika.apache.org/base/docs/govdocs1/190/190605.pdf.

[44] BSC. *SUNSET: Framework for Climate Data Post-Processing*. Accessed February 2024. URL: https://earth.bsc.es/gitlab/es/sunset/-/wikis/home.

[45] startR Contributors. *startR: Easy Start for Large Distributed Data Sets in R*. Accessed February 2024. URL: https://cran.r-project.org/web/packages/startR/index.html.

[46] CSTools Contributors. *CSTools: Climate Forecasting Tools*. Accessed February 2024. URL: https://cran.r-project.org/web/packages/CSTools/index.html.

[47] CSTools Contributors. *s2dv_cube: A Data Structure in CSTools*. Accessed February 2024. URL: https://search.r-project.org/CRAN/refmans/CSTools/html/s2dv_cube.html.

[48] Unidata. *CDO: Climate Data Operators*. Accessed March 2024. 2012. URL: https://www.unidata.ucar.edu/software/netcdf/workshops/2012/third_party/CDO.html.

[49] Gareth James et al. *An Introduction to Statistical Learning*. Available at: https://www.statlearning.com/. Springer, 2013.

[50] FH Joanneum. *Cross-Validation Explained*. 2005–2006.

[51] CSDownscale Authors. *CSDownscale: A Tool for Climate Downscaling*. Accessed March 2024. n.d. URL: https://cran.r-project.org/package=CSDownscale.
[52] GitLab. *GitLab*. 2024. URL: https://gitlab.com.
[53] K. Werder, B. Ramesh, and R. Zhang (Sophia). *Establishing Data Provenance for Responsible Artificial Intelligence Systems*. Vol. 13. 2. 2022, 22:1–22:23. DOI: 10.1145/3503488.
[54] M. S. Mayernik et al. *Data Conservancy Provenance, Context, and Lineage Services: Key Components for Data Preservation and Curation*. Vol. 12. 2013, pp. 158–171. DOI: 10.2481/dsj.12-039.
[55] S. D. Viglas. *Data Provenance and Trust*. Vol. 12. 2013, GRDI58–GRDI64. DOI: 10.2481/dsj.GRDI-010.
[56] World Wide Web Consortium (W3C). *Resource Description Framework (RDF)*. Accessed April 2024. URL: https://www.w3.org/RDF/.
[57] World Wide Web Consortium (W3C). *OWL 2 Web Ontology Language Document Overview*. Accessed April 2024. URL: https://www.w3.org/TR/owl2-overview/.
[58] World Wide Web Consortium (W3C). *PROV-DM: The PROV Data Model*. Accessed: June 2024. URL: https://www.w3.org/TR/prov-dm/.
[59] Infomesh. *Introduction to the Semantic Web*. Accessed April 2024. 2001. URL: http://infomesh.net/2001/swintro/.
[60] Ontotext. *What is the Semantic Web*. Accessed April 2024. URL: https://www.ontotext.com/knowledgehub/fundamentals/what-is-the-semantic-web/.
[61] World Wide Web Consortium. *W3C Standards*. Accessed April 2024. URL: https://www.w3.org/standards/.
[62] Oxford Semantic Technologies. *What is an IRI? What does IRI mean?* Accessed April 2024. URL: https://www.oxfordsemantic.tech/faqs/what-is-an-iri-what-does-iri-mean.
[63] Tim Berners-Lee, James Hendler, and Ora Lassila. *The Semantic Web*. Vol. 284. 5. 2001, pp. 34–43.
[64] Jeffrey T. Pollock. *Semantic Web For Dummies*. For Dummies, 2009. ISBN: 978-0-470-39679-7.
[65] John Smith. *Understanding Ontologies in Knowledge Management*. Vol. 55. 2023, pp. 101–115. DOI: 10.1001/jwsem.2023.55.101.
[66] Jane Doe. *Ontologies and the Semantic Web*. Academic Press, 2024. ISBN: 978-1-234-56789-0.
[67] W3C OWL Working Group. *OWL2 web Ontology Language document overview*. Accessed April 2024. 2012. URL: https://www.w3.org/TR/owl2-overview.
[68] Pascal Hitzler. *A Review of the Semantic Web Field*. Vol. 64. 2. 2021, pp. 76–83. DOI: 10.1145/3397512.
[69] Thomas B. Passin. *Explorer's Guide to the Semantic Web*. Manning Publications, 2004. ISBN: 978-1-932394-20-7.
[70] W3C. *RDF Syntax Namespace*. Accessed June 2024. URL: http://www.w3.org/1999/02/22-rdf-syntax-ns#.
[71] W3C RDF Working Group. *RDF Schema 1.1*. Accessed April 2024. 2014. URL: https://www.w3.org/TR/rdf-schema/.
[72] W3C. *XSD: XML Schema Definition*. Accessed June 2024. URL: http://www.w3.org/2001/XMLSchema#.

[73] Gobierno Vasco. *Introducción al RDF y su uso en el Gobierno Vasco*. Gobierno Vasco. Accessed April 2024. URL: https://www.euskadi.eus/contenidos/informacion/opendata_rdf_euskadi/es_info/adjuntos/RDF.pdf.

[74] World Wide Web Consortium (W3C). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Accessed April 2024. URL: https://www.w3.org/TR/rdf-concepts/.

[75] W3Schools. *JSON Syntax*. Accessed April 2024. 2024. URL: https://www.w3schools.com/js/js%5C_json%5C_syntax.asp.

[76] W3C XML Core Working Group. *W3C XML Schema*. Accessed April 2024. May 2001. URL: https://www.w3.org/XML/Schema.

[77] W3Schools. *XML Tutorial*. Accessed April 2024. 2024. URL: https://www.w3schools.com/xml/xml_whatis.asp.

[78] J. Bedia et al. *The METACLIP semantic provenance framework for climate products*. Vol. 119. 10.1016/j.envsoft.2019.07.005. Elsevier, 2019, pp. 445–457. URL: https://doi.org/10.1016/j.envsoft.2019.07.005.

[79] METACLIP Project. *METACLIP GitHub Repository*. Accessed April 2024. 2024. URL: https://github.com/metaclip.

[80] W3C Provenance Working Group. *PROV-Overview: An Overview of the PROV Family of Documents*. Accessed April 2024. 2013. URL: https://www.w3.org/TR/prov-overview/.

[81] W3C Provenance Working Group. *PROV-PRIMER: An Introduction to the PROV Family of Documents*. Accessed April 2024. 2013. URL: https://www.w3.org/TR/prov-primer/.

[82] W3C Provenance Working Group. *PROV-DM: The PROV Data Model*. Accessed April 2024. 2013. URL: https://www.w3.org/TR/prov-dm/.

[83] W3C Provenance Working Group. *PROV-O: The PROV Ontology*. Accessed April 2024. 2013. URL: https://www.w3.org/TR/prov-o/.

[84] Dublin Core Metadata Initiative. *Dublin Core Metadata Element Set, Version 1.1: Reference Description*. Accessed April 2024. 2012. URL: http://dublincore.org/documents/dces/.

[85] *Dublin Core Metadata Element Set, Version 1.1*. Accessed June 2024. URL: http://purl.org/dc/elements/1.1/.

[86] *GeoSPARQL - A Geographic Query Language for RDF Data*. Accessed June 2024. URL: http://www.opengis.net/ont/geosparql.

[87] *RDF Schema 1.1*. Accessed June 2024. URL: http://www.w3.org/2000/01/rdf-schema.

[88] *SKOS Simple Knowledge Organization System Reference*. Accessed June 2024. URL: http://www.w3.org/2004/02/skos/core.

[89] *Dublin Core Metadata Terms*. Accessed June 2024. URL: http://purl.org/dc/terms/.

[90] *XML Schema Definition Language (XSD) 1.1*. Accessed June 2024. URL: http://www.w3.org/2001/XMLSchema.

[91] *Metaclip Datasource*. Accessed June 2024. URL: http://metaclip.org/datasource.owl.

[92] *Metaclip Calibration*. Accessed June 2024. URL: http://metaclip.org/calibration.owl.

[93] *Metaclip Verification*. Accessed June 2024. URL: http://metaclip.org/verification.owl.

[94] *Metaclip Graphical Output*. Accessed June 2024. URL: http://metaclip.org/graphical_output.owl.

[95] METACLIP. *metaclipR: An R package for METACLIP*. Accessed April 2024. 2023. URL: https://github.com/metaclip/metaclipR/blob/master/README.md.

[96] Maialen Iturbide et al. *The R-based climate4R open framework for reproducible climate data access and post-processing*. Vol. 111. Elsevier, 2019, pp. 42–54. DOI: 10.1016/j.envsoft.2018.09.009. URL: https://www.sciencedirect.com/science/article/abs/pii/S1364815218303049.

[97] Santander MetGroup. *transformeR: A package for transforming climate data*. Accessed April 2024. 2023. URL: https://github.com/SantanderMetGroup/transformeR.

[98] Santander MetGroup. *visualizeR: A package for visualizing climate data*.

[99] Santander MetGroup. *loadeR: A package for loading climate data*. Accessed April 2024. 2023. URL: https://github.com/SantanderMetGroup/loadeR.

[100] Santander MetGroup. *downscaleR: A package for downscaling climate data*. Accessed April 2024. 2023. URL: https://github.com/SantanderMetGroup/downscaleR.

[101] Santander MetGroup. *Climate4R: An R-based framework for climate data analysis*. Accessed May 2024. 2023. URL: https://github.com/SantanderMetGroup/climate4R.

[102] Gabor Csardi and Tamas Nepusz. *igraph: Network Analysis and Visualization*. Accessed June 2024. 2024. URL: https://cran.r-project.org/web/packages/igraph/index.html.

[103] The Apache Software Foundation. *Apache Jena*. URL: https://jena.apache.org.

[104] Mike Bostock. *D3.js-Data-Driven Documents*. Accessed April 2024. 2024. URL: https://d3js.org.

[105] Ignacio Llorente, Alejandro García, and María Gómara. *Practical Guide to Scientific Data Management*. Accessed June 2024. 2019. URL: https://digital.csic.es/handle/10261/213797.

[106] Stanford Center for Biomedical Informatics Research. *Protégé*. Accessed November 2023. 2020. URL: https://protege.stanford.edu/.

[107] Jeremy Hickson et al. *yaml: Methods to Convert R Data to YAML and Back*. Accessed Feburary 2024. 2024. URL: https://CRAN.R-project.org/package=yaml.

[108] Barry Jones et al. *redland: 'Redland' RDF Library Bindings in R*. Accessed Feburary 2024. 2024. URL: https://CRAN.R-project.org/package=redland.

[109] Critical Path Institute (CPI). *Open Biological and Biomedical Ontologies (OBO) Organized Knowledge (OBOOK)*. Accessed June 2024. 2024. URL: https://oboacademy.github.io/obook/.

[110] Martin Fowler. *Continuous Integration*. Accessed June 2024. 2006. URL: https://martinfowler.com/articles/continuousIntegration.html#MakeTheBuildSelf-testing.

[111] IBM. *Continuous Integration*. Accessed June 2024. 2024. URL: https://www.ibm.com/topics/continuous-integration.

[112] GitLab. *Benefits of Continuous Integration*. Accessed June 2024. 2024. URL: https://about.gitlab.com/topics/ci-cd/benefits-continuous-integration/.

[113] Atlassian. *Continuous Integration*. Accessed May 2024. 2024. URL: https://www.atlassian.com/continuous-delivery/continuous-integration.

[114] *METACLIP Semantic Provenance Framework*. Accessed May 2024. URL: http://www.metaclip.org/.

[115] *METACLIP Vocabularies on GitHub*. Accessed May 2024. URL: https://github.com/metaclip/vocabularies.

[116] Universitat Politècnica de Catalunya. *Guia per incorporar l'anàlisi de sostenibilitat i implicacions ètiques en els TFE*. Accessed June 2024. 2023. URL: https://govern.upc.edu/ca/consell-de-govern/consell-de-govern/sessio-07-2023-del-consell-de-govern/comissio-docencia-i-politica-academica-pendent-celebracio/informacio-de-la-guia-per-incorporar-lanalisi-de-sostenibilitat-i-implicacions-etiques-en-els-tfe/informacio-de-la-guia-per-incorporar-lanalisi-de-sostenibilitat-i-implicacions-etiques-en-els-tfe/@@display-file/visiblefile/Guia%20Informe%20Sostenibilitat%20TFG_TFM_versi%C3%B3%20UPC.pdf.

[117] RenSMART. *KWH to CO2 Calculator*. Accessed June 2024. n.d. URL: https://www.rensmart.com/Calculators/KWH-to-CO2.

[118] Ecotree. *How much CO2 does a tree absorb?* Accessed May 2024. 2024. URL: https://ecotree.green/en/how-much-co2-does-a-tree-absorb.

[119] *UPC Code of Ethics*. Accessed May 2024. 2023. URL: https://comite-etica.upc.edu/ca/documentacio/guia-i-codis/codi-etic-upc/codi-etic-ce-upc-eng.pdf.

[120] United Nations. *Transforming our world: the 2030 Agenda for Sustainable Development*. Accessed May 2024. 2015. URL: https://sdgs.un.org/publications/transforming-our-world-2030-agenda-sustainable-development-17981.

[121] United Nations. *Sustainable Development Goals*. Accessed May 2024. 2024. URL: https://sdgs.un.org/goals.

[122] Domingo Manubens-Gil et al. "Seamless management of ensemble climate prediction experiments on HPC platforms". In: (2016), pp. 895–900. DOI: 10.1109/HPCSim.2016.7568429.

[123] Simone Leo. "Recording provenance of workflow runs with RO-Crate". In: (2023). arXiv: 2312.07852v1 [cs.CL]. URL: https://arxiv.org/abs/2312.07852v1.

[124] Stephan Hussmann, Thorsten Ringbeck, and Bianca Hagebeuker. *A Performance Review of 3D TOF Vision Systems in Comparison to Stereo Vision Systems*. Ed. by Asim Bhatti. Rijeka: IntechOpen, 2008. Chap. 7. DOI: 10.5772/5898. URL: https://doi.org/10.5772/5898.

[125] Ulla Wandinger. *Introduction to Lidar*. Ed. by Claus Weitkamp. New York, NY: Springer New York, 2005, pp. 1–18. ISBN: 978-0-387-25101-1. DOI: 10.1007/0-387-25101-4_1. URL: https://doi.org/10.1007/0-387-25101-4_1.

[126] J. Scott Armstrong, Kesten C. Green, and Andreas Graefe. *Answers to Frequently Asked Questions*. Accessed March 2024. 2010. URL: https://unitar.org/sites/default/files/media/publication/doc/guide_predicting_and_projecting.pdf.

[127] Kesten C. Greene and J. Scott Armstrong. *The Ombudsman: Value of Expertise for Forecasting Decisions in Conflicts*. Accessed March 2024. 2007, pp. 1–12. URL: https://unitar.org/sites/default/files/media/publication/doc/guide_predicting_and_projecting.pdf.

[128] K. E. Trenberth. *Numerical Models for Climate Prediction*. Accessed April 2024. 2024. URL: https://www.sciencedirect.com/science/article/abs/pii/B9780123848666000131.

[129] D. J. Stensrud. *Numerical Modeling and Prediction of Weather and Climate*. Cambridge University Press, 2007. URL: https://books.google.es/books?hl=es&lr=&id=Z3jV9QSQJnQC&oi=fnd&pg=PP1&dq=Numerical+Modeling+and+Prediction+meteorologuy&ots=HsatE2HCIX&sig=fWdbwMW9KOe-I7lIanK57M-3yuI#v=

onepage&q=Numerical%20Modeling%20and%20Prediction%20meteorologuy&f=false.

[130] *Data Provenance*. Accessed March 2024. 2024. URL: https://www.nnlm.gov/guides/data-glossary/data-provenance.

[131] Ontotext. *Introduction to Semantic Web*. Accessed April 2024. 2024. URL: https://graphdb.ontotext.com/documentation/10.1/introduction-to-semantic-web.html.

[132] Wikipedia. *Internationalized Resource Identifier*. Accessed April 2024. URL: https://en.wikipedia.org/wiki/Internationalized_Resource_Identifier.

[133] Graph.build. *Semantic Graphs*. Accessed April 2024. 2024. URL: https://graph.build/resources/semantic-graphs.

[134] *Lecture Notes in Computer Science*. LNCS. Springer, 2006. URL: https://link.springer.com/book/10.1007/11890850.

[135] Paolo Missier. *13th USENIX Conference on Theory and Practice of Provenance (TaPP 2013)*. Accessed April 2024. URL: https://www.usenix.org/conference/tapp13/technical-sessions/presentation/missier.

[136] Wikipedia. *PROV (Provenance)*. Accessed April 2024. 2024. URL: https://en.wikipedia.org/wiki/PROV_(Provenance).

[137] J. Bedia and D. San Martín (Predictia). *Metadata encoding with metaclipR*. Accessed April 2024. 2018. URL: file:///mnt/data/2018_metaclip_EMS.pdf.

[138] *Climate Forecast Analysis Hands-on Tutorial (R tools)*. Accessed April 2024. 2023. URL: file:///mnt/data/Climate%20Forecast%20Analysis%20hands-on%20turorial%28R%20tools%29.pdf.

[139] InsideLearn. *Introduction to YAML: A Hands-On Course*. Accessed Feburary 2024. 2024. URL: https://insidelearn.com/introduction-to-yaml-a-hands-on-course.

[140] AWS. *Continuous Integration*. Accessed May 2024. 2024. URL: https://aws.amazon.com/es/devops/continuous-integration/.

[141] TMB. *Memoria sostenibilidad 2021. Informe de gestión 2021 del grupo consolidado de Transports Metropolitans de Barcelona*. Accessed May 2024. 2021. URL: https://www.tmb.cat/documents/20182/111197/Memoria+de+sostenibilitat+de+TMB+2021_es/bd6856dc-45e3-410d-823a-8fa9000d4d6a.

[142] NHS Forest. *How many trees per hectare?* Accessed May 2024. 2024. URL: https://nhsforest.org/how-many-trees-can-be-planted-hectare/.

[143] FIRA. *Benchmarking Carbon Footprint of Furniture Products*. Accessed May 2024. 2024. URL: http://www.healthyworkstations.com/resources/Environment/FIRA.CarbonFootprint.pdf.

[144] P. Berg, H. Feldmann, and H.-J. Panitz. "Bias correction of high resolution regional climate model data". In: *Journal of Hydrology* 448-449 (2012), pp. 80–92. ISSN: 0022-1694. DOI: https://doi.org/10.1016/j.jhydrol.2012.04.026. URL: https://www.sciencedirect.com/science/article/pii/S0022169412003010.

[145] *PROV-DM: The PROV Data Model*. Accessed June 2024. URL: http://www.w3.org/ns/prov.

Appendices

A SUNSET

A.1 Recipe example

B Preliminary project

B.1 RDF generation code

B.2 metaclipR implementation code

B.3 JSON file

C SUNSET_PROVENANCE

C.1 SUNSET_PROVENANCE functions

C.1.1 combination_graph_dataset.R

C.1.2 SkillScore-helpers.R

C.1.3 sunset_prov_anomalies.R

C.1.4 sunset_prov_calibration.R

C.1.5 sunset_prov_dataset.R

C.1.6 sunset_prov_downscale.R

C.1.7 sunset_prov_indices.R

C.1.8 sunset_prov_verification.R

D SUNSET_PROVENANCE execution example

D.1 Generated JSON (example script)

D.2 test_provenance.R

E SUNSET_PROV

E.1 SUNSET_PROV functions

E.1.1 Prov.Anomalies.R

E.1.2 Prov.Calibration.R

E.1.3 Prov.Dataset.R

E.1.4 Prov.DatasetSubset.R

E.1.5 Prov.Regridding.R

E.1.6 Prov.Skill.R

E.1.7 Prov.Visualization.R

E.1.8 SunsetProv.Anomalies.R

E.1.9 SunsetProv.Calibration.R

E.1.10 SunsetProv.Command.R

E.1.11 SunsetProv.Dataset.R

E.1.12 SunsetProv.Downscaling.R

E.1.13 SunsetProv.Indices.R

E.1.14 SunsetProv.Skills.R

E.1.15 SunsetProv.Visualization.R

E.1.16 keep_latest_node.R

E.2 SUNSET_PROV execution example

F Unit tests

F.1 Unit test 1

F.1.1 Recipe Unit test 1

F.1.2 Script Unit test 1

F.1.3 JSON Unit test 1

F.2 Unit test 2

F.2.1 Recipe Unit test 2

F.2.2 Script Unit test 2

F.2.3 JSON Unit test 2

F.3 Unit test 3

F.3.1 Recipe Unit test 3

F.3.2 Script Unit test 3

F.3.3 JSON Unit test 3

F.4 Unit test 4

F.4.1 Recipe Unit test 4

F.4.2 Script Unit test 4

F.4.3 JSON Unit test 4

F.5 Unit test 5

F.5.1 Recipe Unit test 5

F.5.2 Script Unit test 5

F.5.3 JSON Unit test 5