Metadata encoding with `metaclipR`

An extension of the METACLIP Provenance Framework for climate4R

J. Bedia & D. San Martín (Predictia)

2018-05-28

Abstract

The objective of METAdata for CLImate Products (METACLIP) is to encode the metadata that ensures the traceability and reproducibility of any kind of climate product (data files, plots, maps ...). The `metaclipR` package has been developed in order to ensure that all the operations undertaken under the `climate4R` framework for the R computing environment are adequately recorded following the METACLIP scheme. As a result, `metaclipR` interprets the commands and arguments passed to the different `climate4R` packages and maps them onto the semantic framework defined in the METACLIP vocabularies, eventually leading to a RDF representation of data provenance. This vignette illustrates a case-study used in a recent paper by Iturbide *et al.* (submitted), to demonstrate the use of `metaclipR` and how the provenance of any research outcome can be explored through the use of the METACLIP interpreter, a web-based visualization tool.

Contents

1 Introduction

The objective of METAdata for CLImate Products (METACLIP) is to encode the metadata that ensures the traceability and reproducibility of any kind of climate product (data files, plots, maps ...). This requires the deployment of a comprehensive framework in order to track all relevant operations undertaken through complex data workflows. To this aim, METACLIP is based on semantics exploiting web standard Resource Description Framework (RDF), through the design of domain-specific extensions of standard vocabularies (e.g. PROV-O) describing the different aspects involved in climate product generation. By introducing semantics to the metadata description, METACLIP ensures an effective communication of the information to a wide range of users with different levels of expertise.

NOTE: In the following, only the `metaclipR` commands will be syntax-highlighted, while the rest of climate data operations performed with the different `climate4R` packages will be presented without syntax highlighting.

---

2 Package installation

`metaclipR` is currently hosted in METACLIP's Public GitHub repository. Using package `devtools` is highly recommended to ease installation:

```
devtools::install_github("metaclip/metaclipR")
```

3 Preparation

First of all, the package `metaclipR` is loaded. A message will appear informing wether the connection with the remote ontology files has been successfull or not. In positive case, the package will be able to automatically map the metadata information encoded for individuals, thus yielding a much more enriched metadata description. Otherwise, th apackage can still be used, but the usage of individuals for certain classes won't be possible.

```
library(metaclipR)
```

```
## metaclipR version 1.0.1 (2018-05-28) is loaded
```

```
## Succesfully opened the connection with remote ontology files
```

A number of additional packages from the `climate4R` bundle are required to run this example. To obtain a more detailed overview of their functionalities and installation instructions, please refer to the climate4R page, with relevant links to the different wikis and additional resources.

```
library(loadeR)
library(transformeR)
library(visualizeR)
```

Also, the package `climate4R.climdex` will be used to calculate the ETCDDI indices used in this example:

```
library(climate4R.climdex)
```

In addition, other external packages useful for data representation will be used. In particular, the `RColorBrewer` package will be used in the following examples to produce maps with visually attractive color schemes, and to mimic the color palettes used by Iturbide *et al.* (submitted) to present the `climate4R` framework.

```
library(RColorBrewer)
```

Furthermore, for conciseness of a few code chunks, some command calls will be sometimes concatenated using the pipe operator (`%>%`) from package `magrittr` (Bache and Wickham 2014).

```
library(magrittr)
```

3.1 Loading intermediate R objects

Some intermediate steps that may take up to a few minutes to be executed (data loading, bias corrected ensemble etc.) can be directly loaded from a remote location instead of being calculated. This is indicated when relevant in the **DIRECT OBJECT DOWNLOAD** boxes. A specific function (`my_load`) has been prepared to this aim, in order to avoid possible system-dependent remote data loading errors:

```
my_load <- function(file.url, verbose = TRUE) {
    tmpfile <- tempfile()
    download.file(url = file.url, destfile = tmpfile)
    load(tmpfile, verbose = verbose)
}
```

The argument is the full URL to the target R data object (as a character string), that will be indicated when relevant.

4 Example 1 - Calculation of a climate index from E-OBS data

This example is envisaged to showcase the main characteristics of `metaclipR` and how it is applied within the `climate4R` framework. To this aim, the different panels of Figure 2 from Iturbide *et al.* (submitted) are reproduced, and all the provenance information recorded and embedded in the final graphical output (a PNG file), ready to be inspected with the METACLIP visualization tool.

For instance, the following steps are contemplated for a full provenance description of the above-mentioned Fig. 2, including the following steps:

1. Definition of the data sources. In this case, the E-OBS observational database and one of the EURO-CORDEX RCM-GCM couplings (Jacob *et al.* 2014) for the Historical and RCP85 experiments.
2. Data loading and subsetting. The `loadeR` package allows the retrieval of user-defined dmensional slices of the dataset (i.e. dataset subsetting), directly retrieved from the KNMI's OPeNDAP Service and local datasets. Furthermore, the loading function `loadGridData`, performs

data aggregation on-the-fly, and is able to apply filters (via condition and threshold arguments), so different climate indices based on absolute threshold exceedances can be also computed on-the-fly.

3. Climate Index calculation. The original E-OBS dataset has a daily temporal resolution. The indices are computed using the `climate4R.climdex` package (Bedia 2018), a wrapper of the ETCCDI index routines implemented by package `climdex.pcic` (Bronaugh 2015), adapted to the data structures of `climate4R`.
4. Climatology calculation (to compute the mean number of days per year, considering the climatological period 1971-2000)
5. Regridding (to interpolate the rotated RCM grid to the regular 0.25 E-OBS grid)
6. Calculation of the RCM bias for the climate index predictions
7. Creation of maps (climatological maps and a bias map)

4.1 Definition of the Dataset

Usually, the first step for provenance tracking is to describe the primary data source. In METACLIP, this is achieved by the class `Dataset` of the `datasource` vocabulary (it will be referred by indicating the prefix followed by the class name as in `ds:Dataset` hereafter), that extends `prov:Entity`, and splits into 6 different subclasses attending to the nature of the data at hand (`ds:MultiDecadalSimulation`, `ds:Observations`, `ds:Reanalysis`, etc.). Further classes are linked to `ds:Dataset` (via object properties), providing further provenance details such as the `ds:ModellingCenter` producing the data and the `ds:DataProvider` distributing the data id different from the latter (these are defined as subclasses of `prov:Agent`), the `ds:Project` in which the data can be framed (e.g. CMIP5, CORDEX etc.) or experiments (e.g. evaluation, historical, RCP85 etc.), defined as subclasses of `prov:Activity`. Other details are also given that are specific for each subclass of `ds:Dataset` (e.g., the driving `ds:GCM` for a given `ds:RCM` simulation – who extend `prov:SoftwareAgent`–, the URLs serving as entry points for the data etc.).

Because there is a number of entities that are well known and of common use in many climate applications, there is a wide number of "Individuals" defined in the datasource vocabulary that are instances of these classes, and that are enriched with additional annotations providing extra

metadata. The helper function `knownClassIndividuals` can help to identify the specific individuals defined for each particular class from a given METACLIP vocabulary (by default, it wll access the `datasource` vocabulary), as later shown.

```
knownClassIndividuals(classname = "GCM", vocabulary = "datasource")
```

```
##  [1] "GFDL-ESM2M"            "EC-EARTH"
##  [3] "CanESM2"               "MIROC-ESM"
##  [5] "ERA-Interim"           "MPI-ESM-MR"
##  [7] "NORESM1-M"             "CM5A-MR"
##  [9] "CNRM-CM5"              "CSIRO-Mk3.6.0"
## [11] "HADGEM2-ES"            "NCEP-NCAR_Reanalysis1"
## [13] "MPI-ESM-LR"
```

Therefore, `metaclipR` can achieve the annotation any data source (either "known" or new) following the METACLIP schema. However, in order to ease the application within `climate4R`, there is a wide number of "pre-defined" datasets widely used, including all the datasets available through the PUBLIC role in the Santander MetGroup User Data Gateway (UDG), described in detail by Iturbide *et al.* (submitted), and also by Cofiño *et al.* 2018. For these "well-known" data sources, the full metadata description is automatically achieved. In order to ascertain which datasets can be "automatically" annotated, the helper function `showUDGDatasources` can be used. It prints the contents of an internal look-up table from which metadata will be directly read. As an example, in the following subsection, the E-OBS datasource is recorded. Further CORDEX RCM data that will be used later will be annotated in a similar way.

```
head(showUDGDatasources())
```

```
##                     name        type
## 1                  JRA55  reanalysis
## 2 ECMWF_ERA-Interim-ESD  reanalysis
## 3           WATCH_WFDEI observation
## 4        PIK_Obs-EWEMBI observation
## 5 E-OBS_v14_0.50regular observation
```

```
## 6 E-OBS_v14_0.44rotated observation
##                                                                      url
## 1           http://meteo.unican.es/tds5/dodsC/jra55/JRA55_Dataset.ncml
## 2             http://meteo.unican.es/tds5/dodsC/interim/interim20.ncml
## 3             http://meteo.unican.es/tds5/dodsC/wfdei/wfdei_daily.ncml
## 4 http://meteo.unican.es/tds5/dodsC/interim/daily/interim20_daily.ncml
## 5    http://meteo.unican.es/tds5/dodsC/eobs/e-obs_v14_0.50regular.ncml
## 6    http://meteo.unican.es/tds5/dodsC/eobs/e-obs_v14_0.44rotated.ncml
##   ModellingCenter Project Experiment  GCM  RCM  Run SoftwareVersion
## 1             JMA    <NA>       <NA> <NA> <NA> <NA>            <NA>
## 2           ECMWF    <NA>       <NA> <NA> <NA> <NA>            <NA>
## 3            <NA>   WATCH       <NA> <NA> <NA> <NA>            <NA>
## 4             PIK  ISIMIP       <NA> <NA> <NA> <NA>            <NA>
## 5            KNMI     ECA       <NA> <NA> <NA> <NA>            <NA>
## 6            KNMI     ECA       <NA> <NA> <NA> <NA>            <NA>
##   SimulationDomain      DatasetSubclass
## 1           Global           Reanalysis
## 2           Global           Reanalysis
## 3             <NA> ObservationalDataset
## 4             <NA> ObservationalDataset
## 5             <NA> ObservationalDataset
## 6             <NA> ObservationalDataset
```

A schematic overview of the vocabulary structure is provided in the following Figure. For illustration purposes, the individuals defined for some of the classes are also displayed.

4.2 Working with the observations (E-OBS)

E-OBS is a daily gridded observational dataset of reference in Europe (Haylock *et al.* 2008). It is available through a public OPeNDAP repository maintained by the KNMI. In this example, the 0.25-degree regular grid of maximum temperature will be used. E-OBS is defined as an individual

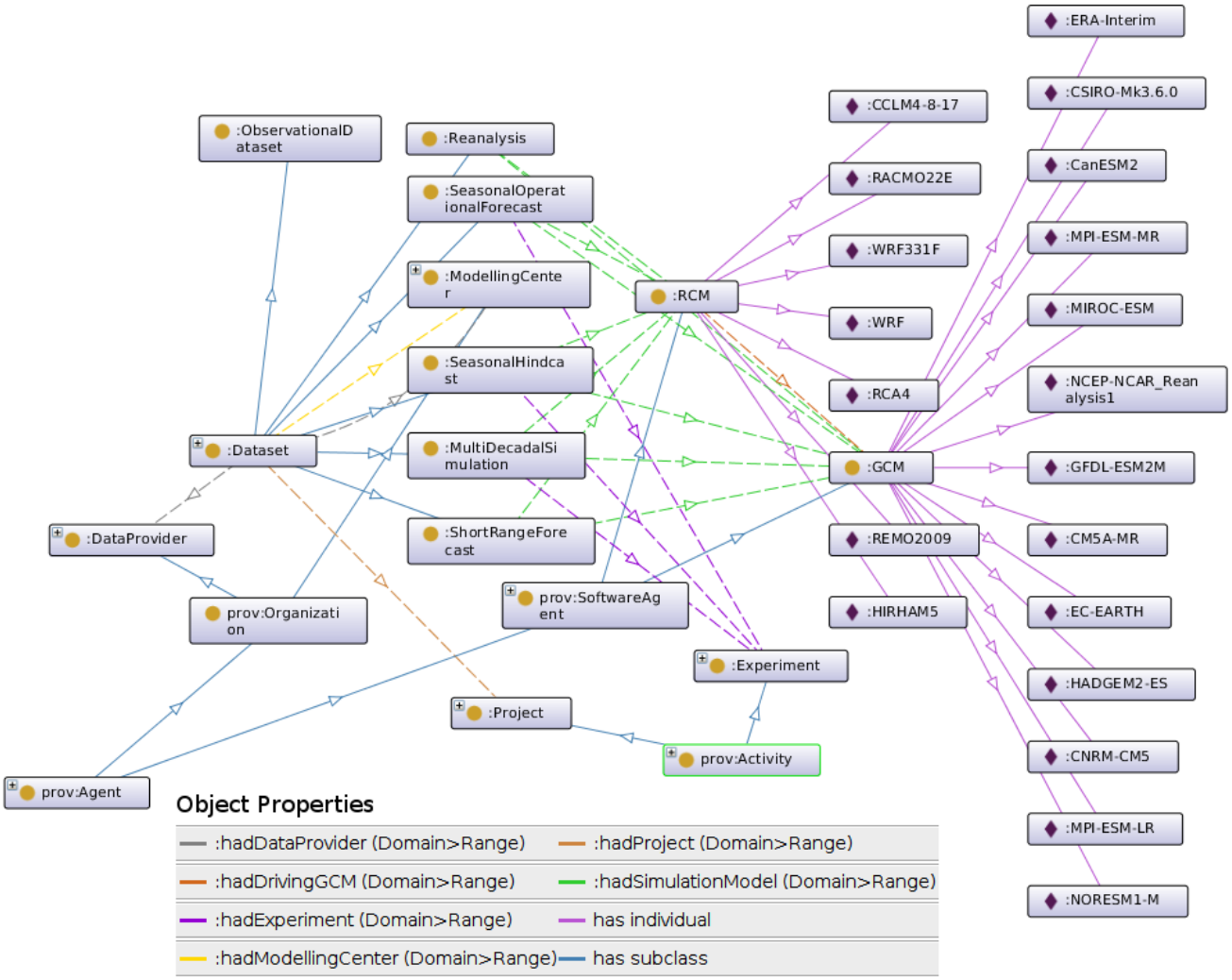


Figure 1: Schematic overview of the main classes (yellow circles), object properties (arrows) and individuals (purple diamonds) of the datasource ontology used to describe a climate dataset. To avoid a congested graph, only the individuals defined for ds:GCM and ds:RCM are shown (there are other individuals describing :ModellingCenter, :Project and :DataProvider classes). The PROV-O classes extended by the datasource ontology are indicated by the :prov prefix

by METACLIP's datasource ontology, being also a dataset available through the UDG. Therefore, it can be located in the reference table of known data sources (it is located in the 7th row):

```
showUDGDatasources()[7, ]
```

```
##                     name        type
## 7 E-OBS_v14_0.25regular observation
##                                                                  url
## 7 http://meteo.unican.es/tds5/dodsC/eobs/e-obs_v14_0.25regular.ncml
##   ModellingCenter Project Experiment  GCM  RCM  Run SoftwareVersion
## 7            KNMI     ECA       <NA> <NA> <NA> <NA>            <NA>
##   SimulationDomain      DatasetSubclass
## 7             <NA> ObservationalDataset
```

Therefore, the dataset name is *E-OBS_v14_0.25regular*. Using this name as input value for the `Dataset.name` argument in `metaclipR.Dataset` function will therefore indicate that the dataset is known, and all the required metadata will be automatically annotated by the function. However, in this example we won't use the UDG E-OBS version, but a more recent one available from the KNMI server. The usage of known datasets from the UDG is later illustrated in Examples 2 and 3.

Similarly, KNMI is also a known institution, for which a specific individual exists:

```
"KNMI" %in% knownClassIndividuals("ModellingCenter")
```

```
## [1] TRUE
```

When the data comes from the UDG data provider (see `knownClassIndividuals("DataProvider")`), the URL pointing to the data is automatically recorded. However, in this case we are using an alternative data provider (KNMI), and its URL can be optionally included. This URL will be internally encoded in the provenance info as a data property:

```
eobs.url <- "http://opendap.knmi.nl/knmi/thredds/dodsC/e-obs_0.25regular/tx_0.25deg_reg_v17.0.nc
metadata.EOBS <- metaclipR.Dataset(Dataset.name = "E-OBS_v17_0.25regular",
                                   DataProvider = "KNMI",
                                   DataProvider.URL = eobs.url,
```

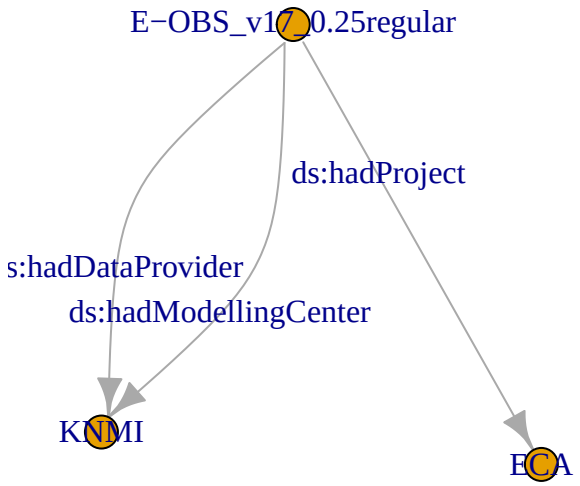


Figure 2: An RDF graph showing the Data source description. Entities (nodes) are linked by properties (arrows) following the so called "triples" of the form Subject–Predicate–Object. Thus, the RDF graph shown in formed by a set of three triples

```
Dataset.subclass = "ObservationalDataset",
Project = "ECA",
ModellingCenter = "KNMI")
```

To give an idea of the operation just undertaken, the function has started an RDF graph with the dataset information, roughly displayed in the following figure. There is of course much more information inside the graph (relevant URLs, class belonging and other annotations), but this requires the specific METACLIP visualization tool to be conveniently displayed in an interactive way. Also note that in this particular example, the data provider and the modelling center correspond to the same individual (KNMI). This is not necessarily so in other cases, and often the data produced by modelling centers is distributed by other providers (e.g., the UDG, ESGF etc.). There is also an associated Project (the European Climate Assessment -ECA-, see e.g. Klein-Tank *et al.* 2002), responsible for the generation of the data.

```
plot(metadata.EOBS$graph)
```

Once a primary data source (in this case the EOBS dataset) has been described, subsequent operations will take place leading to different transformations. The next step in the data workflow is subsetting, by which a particular spatiotemporal slice of data for a specific variable is extracted from the database for working. All the transformations experienced by the data throughout the data workflow are incoded in METACLIP with the *Step* class of the *datasource* vocabulary. The

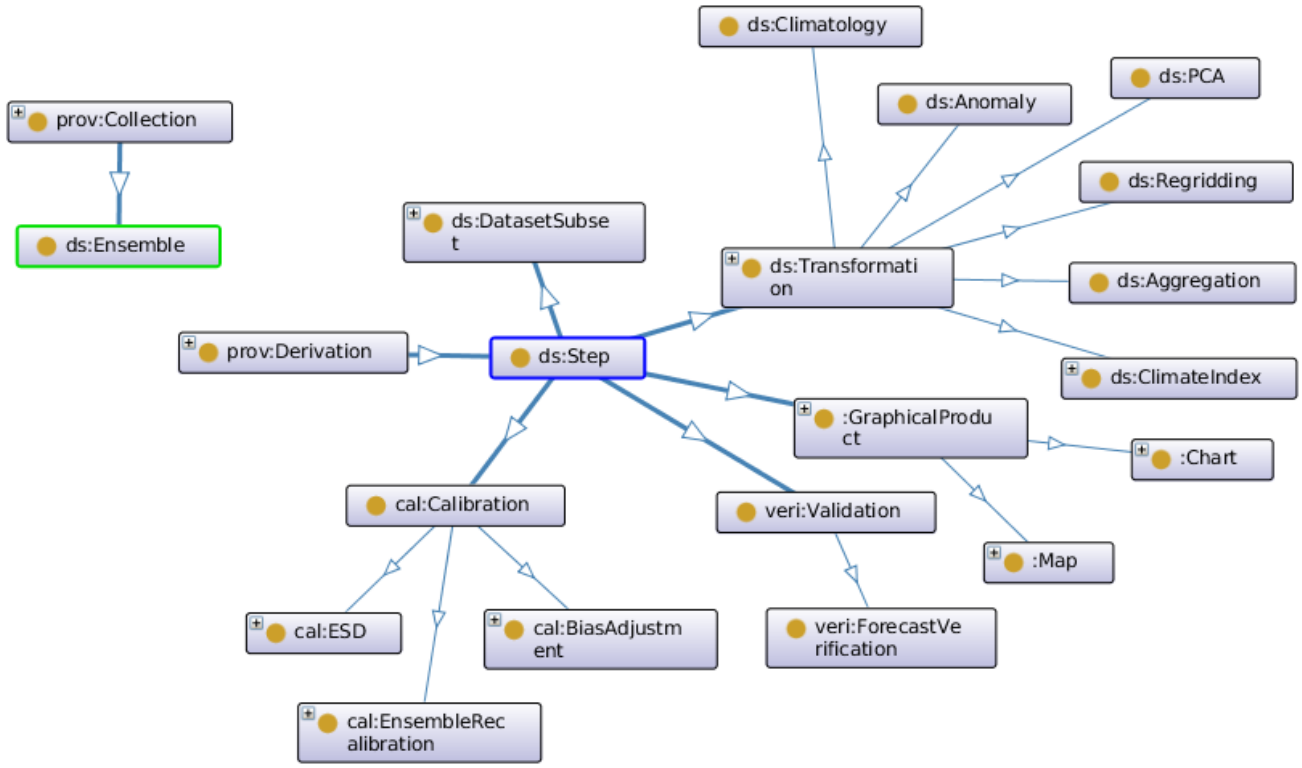


Figure 3: Schematic overview of the main classes (yellow circles, blue arrows represent subclasses) of the different METACLIP ontologies used to describe data transformations. The vocabulary defining each class is indicated by their prefixes

Step class extends the *Derivation* class from the PROV-O vocabulary defined as "a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity". *Steps* are therefore important in the METACLIP Framework.

In the following, different data transformations are sequentially described, from subsetting to climate index computation, climatology calculation and final map creation. Further transformations are addressed in Examples 2 and 3 (anomaly calculation, bias correction . . .).

4.3 Remote Data loading (subsetting)

As shown in *Iturbide et al.*, the function `loadGridData` from package `loadeR` was used to access the specific data slice used in the study. Note that `loadGridData` performs several steps in one single command call, depending on the different arguments used. Thus, it is possible to undertake dimensional subsetting + index.calculation + aggregation on-the-fly when using this data loading function. For this reason, a specific `metaclipR` function has been designed to account for this

characteristic. This allows a more accurate description of the different transformations experienced by the original data following the METACLIP schema when using this important `climate4R` data loading feature.

In this example, the function `loadGridData` performs subsetting according to the arguments specified. Climate index calculation and aggregation will be performed afterwrads using other command calls. The data collocation parameters used for subsetting are indicated by the different specific arguments (`var`, `lonLim`, `latLim`, `season` and `years`, in this case).

```
lon <- c(-10, 20)
lat <- c(35, 46)

tasmax <- loadGridData(dataset = eobs.url,
                       var = "tx",
                       season = 1:12,
                       years = 1971:2000,
                       lonLim = lon,
                       latLim = lat)
```

DIRECT OBJECT DOWNLOAD

The output of this function can be directly loaded from:

```
my_load("http://www.metaclip.predictia.es/notebook_data/EOBSv17_tasmax.Rdata")
```

Note that no arguments indicating temporal aggregation (`time`, `aggr.m` etc.) are being used. This means that we are loading the data in its native temporal resolution, that in this case is daily. Next, a call to the corresponding `metaclipR` function is done to record this subsetting step in the data workflow:

```
metadata.EOBS <- metaclipR.loadeR(package = "loadeR",
                                  version = "1.4.0",
                                  graph = metadata.EOBS,
```

```
                    output = "tasmax",
                    fun = "loadGridData",
                    arg.list = list(dataset = eobs.url,
                                    var = "tx",
                                    season = 1:12,
                                    years = 1971:2000,
                                    lonLim = lon,
                                    latLim = lat))
```

4.4 Climate index calculation

The function `climdexGrid` is the workhorse for the calculation of all the ETCCDI core indices, indicated by the `index.code` argument. Additional specific arguments from the `climdex.pcic` package routines can be passed to this function (these are detailed in the help menu of `climdexGrid`). Here, we apply the default configuration of the SU index (Summer Days, i.e., the number of days per year recording a maximum temperature above 25ºC).

```
SU <- climdexGrid(index.code = "SU", tx = tasmax)
```

A specific `metaclipR` function (`metaclipR.etccdi`) has been designed for the specific characteristics of the ETCDDI indices:

DIRECT OBJECT DOWNLOAD

The output of this function can be directly loaded from:

```
my_load("http://www.metaclip.predictia.es/notebook_data/EOBSv17_SU_YY.Rdata")
```

```
metadata.EOBS.SU <- metaclipR.etccdi(graph = metadata.EOBS,
                                     output = "SU",
                                     arg.list = list(index.code = "SU"))
```

Note that the temporal resolution (as well as other relevant metadata) is updated after climate

index calculation. In this case, the original daily maximum temperature (daily resolution) has been aggregated to an annual dataset after the index calculation. As a result, we preserve the original metadata description of the temperature, and create a new object with the climate index definition (the former will be later used for bias-correcting the future RCM projections of temperature, see Example 3.)

4.5 Climatology calculation

The climatological mean field is next calculated:

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

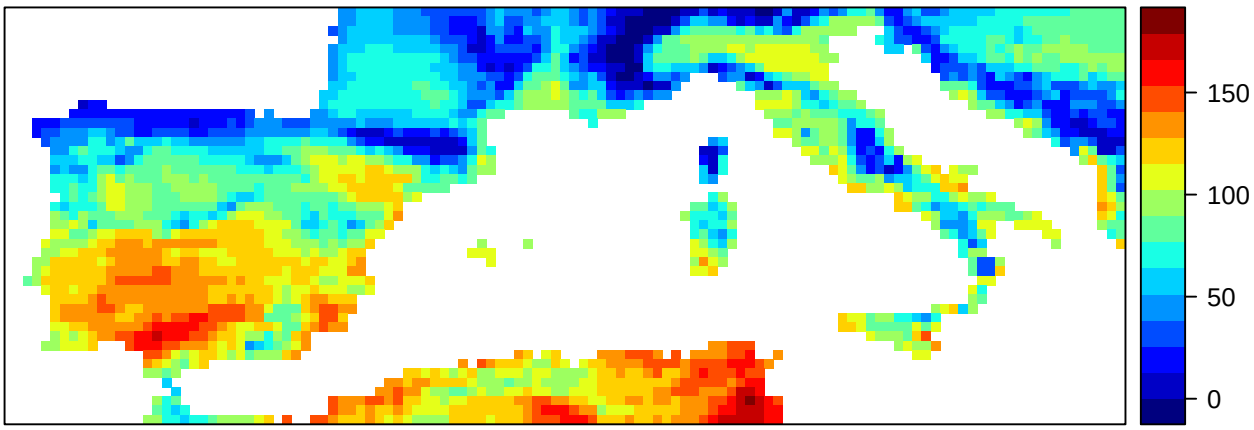
And the corresponding step's metadata is annotated. Note that by default, the function will asume that the `"mean"` cell method is being used (i.e., the climatological mean.)

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

4.6 Climatology map

The default behaviour of `spatialPlot` will produce a basic map without text and with a basic "rainbow"-type color palette.

```
spatialPlot(SU.clim)
```



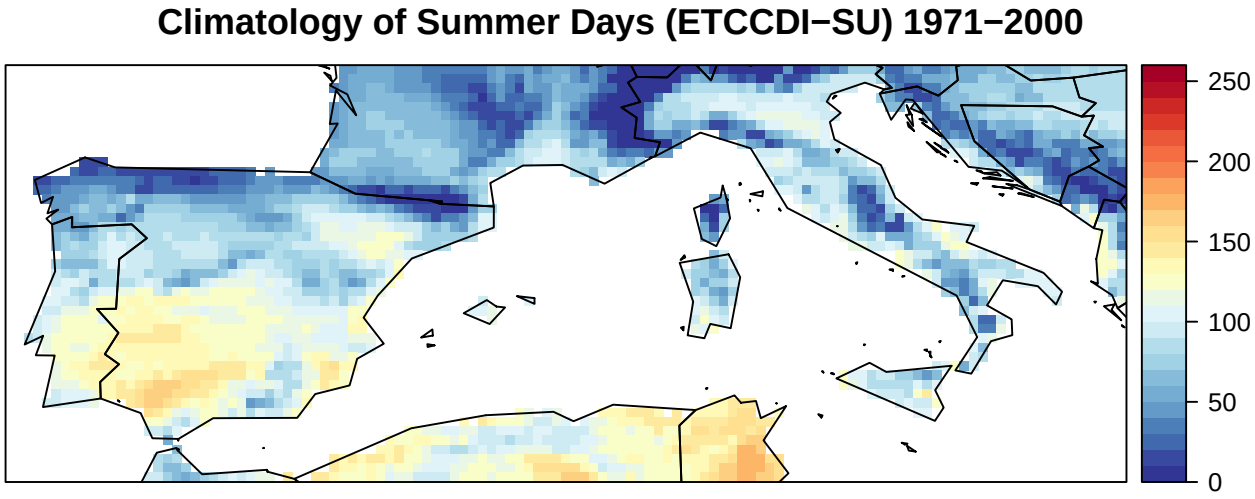


Figure 4: Mean number of summer days (SU index from ETCCDI) for the 30-year climatological period 1971-2000. The figure reproduces Figure 2a from Iturbide et al.

In order to reproduce the figure 2a in Iturbide *et al.* (submitted), we add further customization options, indicating the same color palette and lines delimiting political boundaries:

```
SU.colors <- colorRampPalette(colors = rev(brewer.pal(11, "RdYlBu")))
main <- "Climatology of Summer Days (ETCCDI-SU) 1971-2000"
spatialPlot(SU.clim, col.regions = SU.colors(61),
            at = seq(0,260,10),
            backdrop.theme = "countries",
            main = main)
```

The metadata is next updated with the step generating the graphical output. The function `metaclipR.SpatialPlot` has been specifically designed to describe the provenance of graphical products generated with this function:

```
metadata.EOBS.SUplot <- metaclipR.SpatialPlot(graph = metadata.EOBS.SU,
                                              input.grid = SU.clim,
                                              arg.list = list(grid = SU.clim,
                                                              col.regions = SU.colors(61),
                                                              at = seq(0, 260, 10),
                                                              backdrop.theme = "countries",
                                                              main = main))
```

4.7 Final Fig creation and metadata embedding

Finally, both the file containing the map and the embedded metadata can be produced using the helper function `embedFig`, that undertakes all operations (metadata encoding + graphical product generation + metadata compression + metadata embedding) in a single function call:

```
embedFig(plot.fun = "spatialPlot",
         arg.list = list(grid = SU.clim,
                         col.regions = SU.colors(61),
                         at = seq(0,260,10),
                         backdrop.theme = "countries",
                         main = main),
         full.metadata = metadata.EOBS.SUplot$graph,
         format = "png",
         filename = "EOBS_SU_climatology.png",
         width = 950, height = 800, res = 150)
```

The final output figure is available for metadata inspection and download in the METACLIP Interpreter page http://www.metaclip.org/interpreter

5 Example 2: Construction of a Euro-CORDEX RCM ensemble from the User Data Gateway and bias calculation

This example follows with the Case Study 2 in Iturbide *et al.*, and shows how `metaclipR` is used to describe the provenance of a bias map from an ensemble of RCMs from EURO-CORDEX. For consistency with the previous example, the European domain used in the first case is maintained (unlike the mentioned paper, that makes a focus on the Iberian Peninsula). In particular, simulation data from the Historical experiment for the reference period 1971-2000 are loaded, considering 4 different RCM-GCM couplings. The SU index is calculated upon this data, and its bias *w.r.t.* E-OBS (described in the previous example) is computed. This operation requires a regridding step by which the rotated RCM projection is interpolated onto the regular E-OBS grid. Finally, a bias plot is created, considering as reference the index climatology previously calculated in Example 1.

This example showcases the inclusion of the *member* dimension in the `climate4R` data structures to work with ensembles, inheriting part of the Unidata'a Common Data Model features and corresponding metadata. Ensmebles are represented in METACLIP by the class *ds:Ensemble*, that extends the superclass *prov:Collection* from the PROV-O ontology. Thus, in METACLIP, "ensembles" are collections of entities that can be datasets, subsets of datasets or other previously transformed climate data.

The inclusion of the *member* dimension poses several advantages from the user point of view. For instance, the code is significantly simplified when dealing with multi-members (either when working with stochastic predictions or multi-model ensembles) since most of the **climate4R** operations (e.g. index calculation, regridding, validation etc.) are implemented to deal with grids containing the member dimesion, and therefore the necessary looping over several members is done behind the scenes. Furthermore, the use of members is also beneficial from the computational point of view, since most relevant functions have the option to parallelize across members through the optional argument `parallel` (and other parallelization configurations via argument `n.cores`), thus providing ease of use and computational efficiency.

Note that the provenance of the E-OBS reference (stored in the R object `metadata.EOBS.SU` from the previous example), is re-used in this example, and joined with the metadata representation of the RCM processing and bias calculation.

```
models <- UDG.datasets()
ensemble.h <- head(models$name[grepl("EUROCORDEX44.*hist", models$name)], 4)
ensemble.h
```

```
## [1] "EUROCORDEX44_ICHEC-EC-EARTH_r12i1p1_RCA4_v1_historical"
## [2] "EUROCORDEX44_CERFACS-CNRM-CM5_r1i1p1_RCA4_v1_historical"
## [3] "EUROCORDEX44_ICHEC-EC-EARTH_r1i1p1_RACMO22E_v1_historical"
## [4] "EUROCORDEX44_ICHEC-EC-EARTH_r3i1p1_HIRHAM5_v1_historical"
```

5.1 Dataset provenance definition

Four different *ds:Dataset* class objects are next defined. As long as all data come from the UDG, the function will annotate all the metadata from these datasets automatically, and therefore only 2 arguments are needed to fully describe the datasets provenance: their name (`Dataset.name`) and the data provider (`DataProvider`). Note that the data provider is also a known data provider described by its own individual class in METACLIP.

```
hist.list <- lapply(ensemble.h, function(ds) {
    metaclipR.Dataset(Dataset.name = ds, DataProvider = "UDG")
})
```

5.2 Loading and manipulating the EUROCORDEX simulations from the UDG

For conciseness, in this step, data loading and regridding are directly performed by concatenating both command calls (`loadGridData` and `interpGrid`) using the pipe operator (`%>%`) from package `magrittr`. In the regridding step, the rotated RCM coordinates (see the previous example), are interpolated onto the regular grid of 0.25 degree resolution of E-OBS (contained in the obhject `ref.grid`):

```
ref.grid <- getGrid(tasmax)
TXh.list <- lapply(ensemble.h, function(x) {
    loadGridData(dataset = x,
                 var = "tasmax",
                 season = 1:12,
                 lonLim = lon,
                 latLim = lat,
                 years = 1971:2000) %>% interpGrid(new.coordinates = ref.grid,
                                                   method = "nearest")})
```

DIRECT OBJECT DOWNLOAD

The output of this function can be directly loaded from (~94 Mb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/histlist.rda")
```

The following loop will define the dataset subsets for each dataset previously defined.

```
hist.list <- lapply(1:length(hist.list), function(x) {
    metaclipR.loadeR(package = "loadeR", version = "1.4.0",
                     output = TXh.list[[x]],
                     graph = hist.list[[x]],
                     fun = "loadGridData",
                     arg.list = list(dataset = ensemble.h[[x]],
                                     var = "tasmax",
                                     season = 1:12,
                                     lonLim = lon,
                                     latLim = lat,
                                     years = 1971:2000))
})
```

And next the regridding operation performed on each model is recorded

```
hist.list <- lapply(1:length(hist.list), function(x) {
    metaclipR.Regridding(package = "transformeR", version = "1.3.3",
                         graph = hist.list[[x]],
                         fun = "interpGrid",
                         arg.list = list(method = "nearest",
                                         new.coordinates = ref.grid))})
```

Once the 4 GCM-RCM couplings have been identified, and the subset for the target season, time slice and spatial domain are defined, the ensemble can be constructed.

5.3 Ensemble construction

Ensemble construction can be achieved using the function `bindGrid.member` from `transformeR`:

```
ensemble <- bindGrid.member(TXh.list)
```

DIRECT OBJECT DOWNLOAD

The output of this function can be directly loaded from (~133Mb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/ensemble.rda")
```

The function takes care of all data collocation aspects (spatial and temporal checks) to ensure the consistency among the different ensemble members, and if valid, joins the different models along a new `member` dimension yielding a new data array with all the necessary metadata:

```
getShape(ensemble)
```

```
## member   time    lat    lon
##      4  10958     34     57
```

```
ens.meta <- metaclipR.Ensemble(graph.list = hist.list)
```

5.4 Index calculation

```
SU.ens <- climdexGrid(tx = ensemble, index.code = "SU")
```

```
## Loading objects:
##   SU.ens
```

DIRECT OBJECT DOWNLOAD

The output of this function can be directly loaded from (~140Kb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/SU.ens.rda")
```

```
ens.meta <- metaclipR.etccdi(graph = ens.meta,
                             output = "SU.ens",
```

```
                                    arg.list = list(index.code = "SU"))
```

5.5 Bias calculation

First, the climatology of each RCM is calculated, so for each GCM-RCM coupling, we get the mean number of summer days for the historical period (1971-2000):

```
ens.clim <- climatology(SU.ens)
```

The anomaly step is recorded in the provenance description:

```
ens.meta <- metaclipR.Climatology(graph = ens.meta,
                                  arg.list = list(clim.fun = list(FUN = "mean", na.rm = TRUE),
                                                  by.member = TRUE))
```

Secondly, the bias is directly calculated by substracting the E-OBS climatology from the RCM climatology just computed. This can be done with function `scaleGrid` (formerly called `localScaling`, currently deprecated but still functional).

```
bias <- scaleGrid(ens.clim, base = SU)
```

This step is recorded as a validation step using `metaclipR.Validation`, considering "Bias" as the quality aspect addressed. At this point, the provenance information from the E-OBS SU climatology (Example 1) and from the RCM Ensemble (this example) are joined within a single RDF graph:

```
ens.meta <- metaclipR.Validation(package = "transformeR", version = "1.3.3",
                                 PredictionGraph = ens.meta,
                                 ReferenceGraph = metadata.EOBS.SU,
                                 fun = "scaleGrid",
                                 QualityAspect = "Bias",
                                 arg.list = list(grid = "ens.clim",
                                                 base = "SU",
                                                 clim.fun = list(FUN = "mean",
                                                                 na.rm = TRUE),
                                                 by.member = TRUE))
```

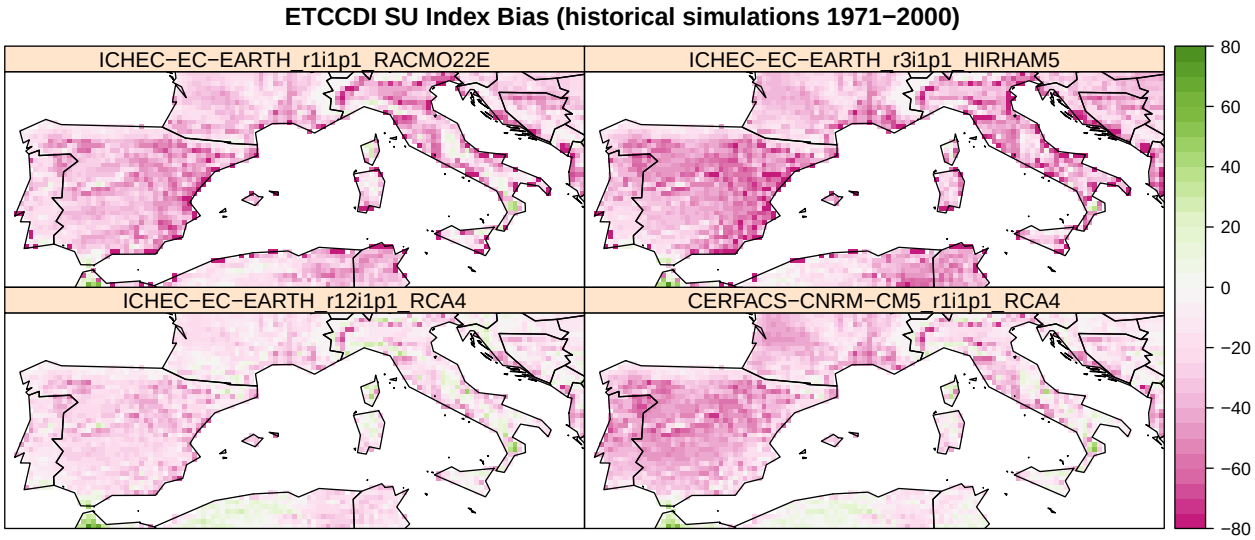


Figure 5: Bias of the EUROCORDEX RCMs (historical simulation) w.r.t. the E-OBS reference for the period 1971-2000

5.6 Bias map generation and final metadata embedding

Finally, the bias map is generated using the the function `spatialPlot` from package `visualizeR` (Frías *et al.* 2018), from the bias object:

```
bias.colors <- colorRampPalette(brewer.pal(n = 9, "PiYG"))
main = "ETCCDI SU Index Bias (historical simulations 1971-2000)"
panel.lab <- gsub("^EUROCORDEX44_", "", ensemble.h) %>% gsub(pattern = "_v1_historical$",
                                                             replacement = "")
spatialPlot(bias, col.regions = bias.colors(59),
            backdrop.theme = "countries",
            set.min = -80, set.max = 80,
            at = seq(-80,80,5),
            names.attr = panel.lab,
            main = main)
```

The metadata corresponding to the graphical product is also recorded:

```
ens.meta <- metaclipR.SpatialPlot(graph = ens.meta, input.grid = bias,
                                  arg.list = list(col.regions = bias.colors(59),
                                                  backdrop.theme = "countries",
```

```
                                    set.min = -80, set.max = 80,
                                    at = seq(-80,80,5),
                                    main = main,
                                    names.attr = panel.lab))
```

Finally, the final figure file is generated with `embedFig`, as in the previous example:

```
embedFig(plot.fun = "spatialPlot", full.metadata = ens.meta$graph,
         format = "png", filename = "ensembleBias.png",
         width = 1200, height = 850, res = 150,
         arg.list = list(grid = bias,
                         col.regions = bias.colors(59),
                         set.min = -80, set.max = 80,
                         at = seq(-80,80,5),
                         backdrop.theme = "countries",
                         main = main))
```

The final output figure is available in the example gallery of the METACLIP Interpreter http://www.metaclip.org/interpreter.

6 Example 3: Bias correction of the RCM ensemble from Example 2

Climate projections generally exhibit systematic deviations from observations, as illustrated in bias maps generated in the previous section. Removing those biases is typically the first step towards useful/actionable climate information and thus a prerequisite for subsequent use, e.g. in impact studies and/or for climate services. In this example, a typical bias correction exercise is illustrated. To this aim, future projections (2041-2070) from the RCP85 are loaded, and a bias correction method widely applied in impact studies (non-parametric quantile mapping) is applied to the maximum temperature grids prior to SU index calculation. Finally, projected delta maps are constructed upon the bias-corrected data. A full provenance description building on the METACLIP framework is achieved throughout this example.

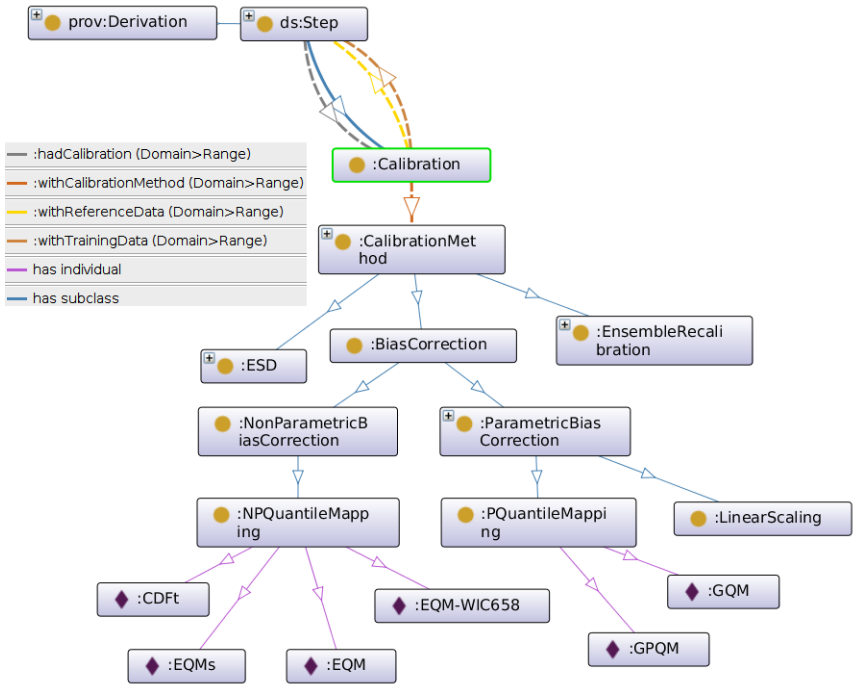


Figure 6: Schematic overview of the "calibration" vocabulary with its main classes (yellow circles), object properties (arrows) and individuals (purple diamonds). For illustration, some individual instances of the BiasCorrection superclass are depicted, including the EQMs method (pertaining to the non-parametric quantile mapping subclass), that is used in this example.

The *calibration* vocabulary contains a semantic description of the different downscaling and bias correction methodologies, extending the *ds:Step* class previously defined in the *datasource* vocabulary to describe data transformations.

6.1 Loading the RCP85 projections (+ regridding)

First of all, the origin of the data is defined. As previously illustrated, we first identify the datasets to be loaded.

```
models <- UDG.datasets()

(ensemble.rcp85 <- head(models$name[grepl("EUROCORDEX44.*rcp85", models$name)], 4))
```

```
## [1] "EUROCORDEX44_ICHEC-EC-EARTH_r12i1p1_RCA4_v1_rcp85"

## [2] "EUROCORDEX44_CERFACS-CNRM-CM5_r1i1p1_RCA4_v1_rcp85"

## [3] "EUROCORDEX44_ICHEC-EC-EARTH_r1i1p1_RACMO22E_v1_rcp85"

## [4] "EUROCORDEX44_ICHEC-EC-EARTH_r3i1p1_HIRHAM5_v1_rcp85"
```

Because the data are being retrieved from the UDG, the data access layer of the climate4R framework, all the metadata from this datasets is conveniently annotated behind the scenes by the function metaclipR.Dataset. This step initializes a new RDF graph containing the provenance information of the RCP85 projections.

```
rcp85.list <- lapply(ensemble.rcp85, function(ds) {
    metaclipR.Dataset(Dataset.name = ds, DataProvider = "UDG")
})
```

For conciseness, in this step, data loading and regridding are directly performed by concatenating both command calls (loadGridData and interpGrid) using the pipe operator (%>%) from package magrittr (Bache and Wickham 2014).

```
TXrcp85.list <- lapply(ensemble.rcp85, function(x) {
    loadGridData(dataset = x,
                 var = "tasmax",
                 season = 1:12,
                 lonLim = lon,
                 latLim = lat,
                 years = 2041:2070) %>% interpGrid(new.coordinates = ref.grid,
                                                   method = "nearest")})
```

DIRECT OBJECT DOWNLOAD

The output of this function can be directly loaded from (~92Mb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/rcp85list.rda")
```

The subsetting step is recorded using metaclipR.loadGridData:

```
rcp85.list <- lapply(1:length(rcp85.list), function(x) {
    metaclipR.loadeR(package = "loadeR", version = "1.4.0",
                     output = TXrcp85.list[[x]],
```

```
                    graph = rcp85.list[[x]],
                    fun = "loadGridData",
                    arg.list = list(dataset = ensemble.rcp85[[x]],
                                    var = "tasmax",
                                    season = 1:12,
                                    lonLim = lon,
                                    latLim = lat,
                                    years = 2041:2070))
})
```

And next the regridding step is recorded:

```
rcp85.list <- lapply(1:length(rcp85.list), function(x) {
    metaclipR.Regridding(package = "transformeR", version = "1.3.3",
                         graph = rcp85.list[[x]],
                         fun = "interpGrid",
                         arg.list = list(method = "nearest",
                                         new.coordinates = ref.grid))})
```

6.2 Bias correction

The function `biasCorrection` from the `climate4R` package `downscaleR` (Bedia et al. 2017) is used to undertake bias correction. Here, a standard (non-parametric) empirical quantile mapping is used (EQMs). This EQMs implementation is based on adjusting 99 percentiles and linearly interpolating inside this range every two consecutive percentiles; outside this range a constant extrapolation (using the correction obtained for the 1st or 99th percentile) is applied. In order to explicitly model the seasonal cycle, the variant EQMs considers a 31-day moving window centred on every calendar day to calibrate the method. The method (and some of its variants) is described in e.g. in Gutiérrez *et al.* 2018 as part of a comprehensive intercomparison experiment. EQM (and its variantes) is implemented in `downscaleR` (Bedia *et al.* 2017) using the `biasCorrection` function, with the options `method = "eqm"` and `extrapolation = "constant"` (including `precipitation = TRUE` and `pr.threshold = 1` for precipitation, which does not apply in this example). For EQMs

(here used), the extra argument `window = c(30,1)` was included.

```
library(downscaleR)
```

This is a computationally demanding task, due to the large size of the ensemble (a regular 25 km grid for the entire Euro-Mediterranean domain, and 30 years of daily data). As a result, instead of directly applying the `biasCorrection` function to the whole ensemble in a single call (may cause memory depletion in some computers), we split the task by models and generate a list of corrected ensemble members, that can be aggregated into a single ensemble structure later. Thus, in the provenance description, the bias correction step will be represented prior to ensemble construction.

```
TXrcp85.eqm.list <- lapply(1:length(TXrcp85.list), function(i) {
     biasCorrection(y = tasmax, x = TXh.list[[i]], newdata = TXrcp85.list[[i]],
                            precipitation = FALSE, method = "eqm", window = c(30,1))
})
```

DIRECT OBJECT DOWNLOAD

The output of the above code chunk can be directly loaded from (~167Mb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/rcp85_eqm_list.rda")
```

```
rcp85.list <- lapply(1:length(rcp85.list), function(i) {
    metaclipR.BiasCorrection(package = "downscaleR", version = "3.0.0",
                             BC.method = "EQMs",
                             fun = "biasCorrection",
                             arg.list = list(method = "eqm",
                                             window = c(30, 1),
                                             extrapolation = "constant",
                                             precipitation = FALSE),
                             TrainingGraph = hist.list[[i]],
                             ReferenceGraph = metadata.EOBS,
```

```
                                graph = rcp85.list[[i]])
})
```

6.3 Ensemble construction

```
##             used   (Mb) gc trigger   (Mb)   max used    (Mb)
## Ncells   2660898  142.2    5684620  303.6    5684620   303.6
## Vcells 389946926 2975.1 1289670476 9839.5 1405562044 10723.6
## Loading objects:
##   ens.rcp85.eqm
```

After correction, the ensemble is built. We use the utility function `bindGrid.member` from `transformeR` to this aim:

```
ens.rcp85.eqm <- bindGrid.member(TXrcp85.eqm.list)
```

DIRECT OBJECT DOWNLOAD

The output of the above code chunk can be directly loaded from (~172Mb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/ensemble_rcp85_eqm.rda")
```

And the corresponding ensemble construction is recorded in the provenance record:

```
metadata.rcp85 <- metaclipR.Ensemble(output = ens.rcp85.eqm, graph.list = rcp85.list)
```

6.4 Climate Index Calculation

As in the previous example, the `climate4R.climdex` package is used to calculate the SU index:

```
## Loading objects:
##   su.rcp85.eqm
su.rcp85.eqm <- climdexGrid(tx = ens.rcp85.eqm, index.code = "SU")
```

DIRECT OBJECT DOWNLOAD

The output of the above code chunk can be directly loaded from (~285Kb):

```
my_load("http://www.metaclip.predictia.es/notebook_data/SU_rcp85_eqm.rda")
```

```
metadata.rcp85 <- metaclipR.etccdi(graph = metadata.rcp85, output = "su.rcp85.eqm",
                                   arg.list = list(index.code = "SU"))
```

6.5 Climatology

The future climatology is next computed:

```
su.rcp85.clim <- climatology(su.rcp85.eqm, clim.fun = list(FUN = "mean", na.rm = TRUE),
                             by.member = TRUE)

metadata.rcp85 <- metaclipR.Climatology(graph = metadata.rcp85,
                                        arg.list = list(clim.fun = list(FUN = "mean", na.rm = TR
                                                        by.member = TRUE))
```

6.6 Anomaly calculation

The delta change is computed as an anomaly considering the climatology of the future index projections, taking the baseline observations as the reference climatology:

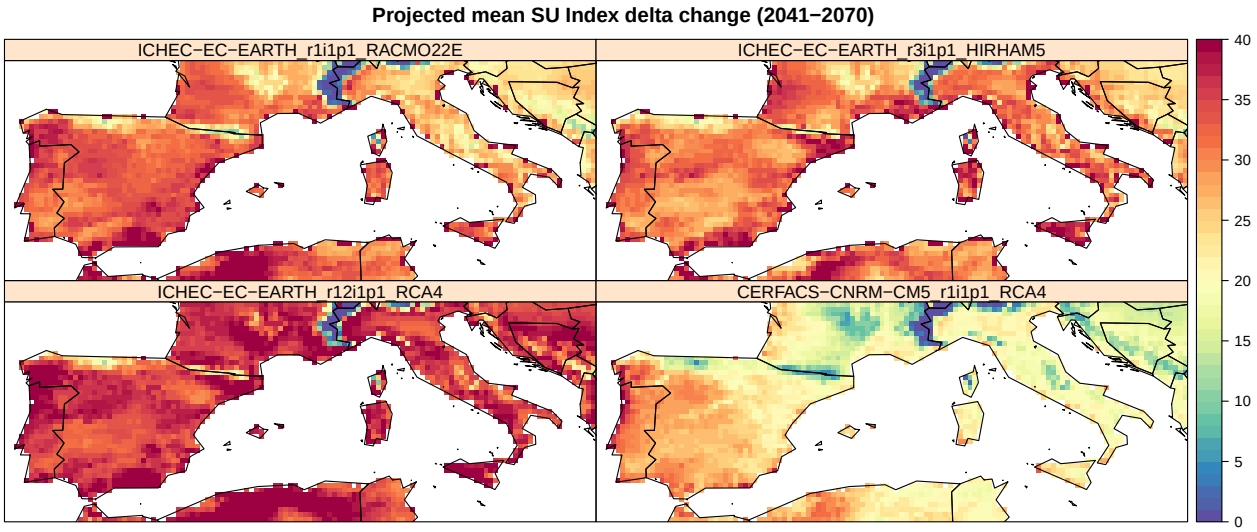
```
SU.rcp85.anomaly <- scaleGrid(grid = su.rcp85.clim, base = SU.clim)

metadata.rcp85 <- metaclipR.AnomalyCalculation(graph = metadata.rcp85,
                                               package = "transformeR",
                                               version = "1.3.3",
                                               fun = "scaleGrid",
                                               arg.list = list(clim.fun = list(FUN = mean,
                                                                               na.rm = TRUE),
                                                               base = "SU.clim",
                                                               time.frame = "none",
```

```
                                                    by.member = TRUE,
                                                    spatial.frame = "gridbox"),
                                          referenceGraph = metadata.EOBS.SU)
```

6.7 Delta map

As in the previous examples, the final map is created, indicating a number of customization parameters to the plot (title, color palette etc.).

```
# Color palette
delta.colors <- brewer.pal(11, "Spectral") %>% rev() %>% colorRampPalette()
panel.lab <- gsub("^EUROCORDEX44_", "", ensemble.rcp85) %>% gsub(pattern = "_v1_rcp85$",
                                                                 replacement = "")
main <- "Projected mean SU Index delta change (2041-2070)"
spatialPlot(grid = SU.rcp85.anomaly,
            set.min = 0,
            set.max = 40,
            at = seq(0,40,1),
            col.regions = delta.colors(61),
            backdrop.theme = "countries",
            main = main,
            names.attr = panel.lab)
```



The metadata of the final product is encoded using `metaclipR.SpatialPlot`:

```
metadata.rcp85 <- metaclipR.SpatialPlot(graph = metadata.rcp85,
                                        input.grid = su.rcp85.clim,
                                        arg.list = list(set.min = 0,
                                                        set.max = 40,
                                                        at = seq(0,40,1),
                                                        col.regions = delta.colors(61),
                                                        backdrop.theme = "countries",
                                                        main = main,
                                                        names.attr = panel.lab))
```

And the final figure file is created in png format, including all the metadata description (compressed) as an embedded file metadata:

```
embedFig(plot.fun = "spatialPlot",
         full.metadata = metadata.rcp85$graph,
         format = "png",
         width = 1200, height = 800, res = 150,
         filename = "deltaSUmap_2041-70.png",
         arg.list = list(grid = SU.rcp85.anomaly,
                         set.min = 0,
```

```
                set.max = 40,
                at = seq(0,40,1),
                col.regions = delta.colors(61),
                backdrop.theme = "countries",
                main = "Projected mean SU Index delta change (2041-2070)",
                names.attr = panel.lab))
```

As in the previous cases, the final output figure is available in the exmplae gallery of the METACLIP Interpreter page http://www.metaclip.org/interpreter.

7 References

- Bache S.M. and Wickham H. (2014). magrittr: A Forward-Pipe Operator for R. R package version 1.5. https://CRAN.R-project.org/package=magrittr
- Bedia J., Gutiérrez J.M., Herrera S., Iturbide M. and Manzanas, R. (2017). downscaleR: An R package for bias correction and statistical downscaling. R package version 3.0.0. https://github.com/SantanderMetGroup/downscaleR/wiki
- Bedia J. (2018). climate4R.climdex: Climate Change Index calculation for climate4R data. R package version 0.1.3. http://meteo.unican.es/climate4R
- Bronaugh, D. (for the Pacific Climate Impacts Consortium), 2015. climdex.pcic: PCIC Implementation of Climdex Routines. R package version 1.1-6. https://CRAN.R-project.org/package=climdex.pcic
- Cofiño, A.S., Bedia, J., Iturbide, M., Vega, M., Herrera, S., Fernández, J., Frías, M.D., Manzanas, R., Gutiérrez, J.M., 2018. The ECOMS User Data Gateway: Towards seasonal forecast data provision and research reproducibility in the era of Climate Services. Climate Services 9, 33–43. https://doi.org/10.1016/j.cliser.2017.07.001
- Frías, M.D., Iturbide, M., Manzanas, R., Bedia, J., Fernández, J., Herrera, S., Cofiño, A.S., Gutiérrez, J.M., 2018. An R package to visualize and communicate uncertainty in seasonal climate prediction. Environmental Modelling & Software 99, 101–110. https://doi.org/10.

1016/j.envsoft.2017.09.008

- Gutiérrez, J.M. *et al.* (2018). An intercomparison of a large ensemble of statistical downscaling methods over Europe: Results from the VALUE perfect predictor cross-validation experiment. International Journal of Climatology. https://doi.org/10.1002/joc.5462
- Haylock, M.R., Hofstra, N., Klein Tank, A.M.G., Klok, E.J., Jones, P.D., New, M., 2008. A European daily high-resolution gridded data set of surface temperature and precipitation for 1950–2006. Journal of Geophysical Research 113. https://doi.org/10.1029/2008JD010201
- Iturbide, M., Bedia, J., Herrera, S., Bano-Medina, J., Fernández, J., Frías, M.D., Manzanas, R., San Martin, D., Cimadevilla, E., Cofiño, A., Gutiérrez, J., (*submitted*). climate4R: An R-based framework for Climate Data Access, Postprocessing and Bias Correction. Submitted to Environmental Modelling and Software.
- Jacob, D. *et al.* (2014). EURO-CORDEX: new high-resolution climate change projections for European impact research. Reg Environ Change 14, 563–578. https://doi.org/10.1007/s10113-013-0499-2
- Klein Tank, A.M.G. *et al.* (2002). Daily dataset of 20th-century surface air temperature and precipitation series for the European Climate Assessment. Int. J. Climatol. 22, 1441–1453. https://doi.org/10.1002/joc.773

8 Session info

```
print(sessionInfo(package = c("loadeR", "transformeR", "visualizeR", "metaclipR")))
```

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.4 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
```

```
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
##  [3] LC_TIME=es_ES.UTF-8           LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=es_ES.UTF-8       LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=es_ES.UTF-8          LC_NAME=es_ES.UTF-8
##  [9] LC_ADDRESS=es_ES.UTF-8        LC_TELEPHONE=es_ES.UTF-8
## [11] LC_MEASUREMENT=es_ES.UTF-8    LC_IDENTIFICATION=es_ES.UTF-8
##
## attached base packages:
## character(0)
##
## other attached packages:
## [1] loadeR_1.4.0      transformeR_1.3.3 visualizeR_1.2.0  metaclipR_1.0.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.16            lattice_0.20-35
##  [3] png_0.1-7               glmnet_2.0-16
##  [5] rprojroot_1.3-2         digest_0.6.12
##  [7] foreach_1.4.4           SpecsVerification_0.5-2
##  [9] plyr_1.8.4              backports_1.1.2
## [11] CircStats_0.2-4         stats4_3.4.4
## [13] evaluate_0.10.1         spam_2.1-3
## [15] utils_3.4.4             data.table_1.10.4-3
## [17] raster_2.6-7            Rgraphviz_2.22.0
## [19] Matrix_1.2-14           PCICt_0.5-4.1
## [21] rmarkdown_1.9           stringr_1.3.0
## [23] RcppEigen_0.3.3.4.0     igraph_1.2.1
## [25] RCurl_1.95-4.10         munsell_0.4.3
## [27] proxy_0.4-19            compiler_3.4.4
```

```
## [29] gridGraphics_0.3-0      pkgconfig_2.0.1
## [31] stats_3.4.4             BiocGenerics_0.24.0
## [33] htmltools_0.3.6         evd_2.3-2
## [35] gridExtra_2.3           downscaleR_3.0.0
## [37] dtw_1.18-1              codetools_0.2-15
## [39] grDevices_3.4.4         mapplots_1.5
## [41] deepnet_0.2             MASS_7.3-50
## [43] bitops_1.0-6            grid_3.4.4
## [45] gtable_0.2.0            magrittr_1.5
## [47] datasets_3.4.4          scales_0.5.0
## [49] graph_1.56.0            stringi_1.1.7
## [51] sm_2.2-5.4              sp_1.2-7
## [53] latticeExtra_0.6-28     akima_0.6-2
## [55] padr_0.4.0              graphics_3.4.4
## [57] climdex.pcic_1.1-6      vioplot_0.2
## [59] boot_1.3-20             base_3.4.4
## [61] loadeR.java_1.1.1       RColorBrewer_1.1-2
## [63] iterators_1.0.9         tools_3.4.4
## [65] climate4R.climdex_0.1.3 maps_3.3.0
## [67] fields_9.6              abind_1.4-5
## [69] parallel_3.4.4          yaml_2.1.18
## [71] colorspace_1.3-2        verification_1.42
## [73] caTools_1.17.1          dotCall64_0.9-5.2
## [75] rJava_0.9-8             knitr_1.20
## [77] methods_3.4.4
```