H2020-SC5-01-2017

MED-GOLD

Turning climate-related information into added value for traditional **MED**iterranean **G**rape, **OL**ive and **D**urum wheat food systems

Deliverable 4.5

A handy easy-to-use manual for stakeholders and practitioners of the climate service tool.

PART III: the durum wheat/pasta sector

This project has received funding from the European Union's Horizon 2020 research and innovation programme under gran agreement No 776467.

DOCUMENT STATUS SHEET

Deliverable Title	**A handy easy-to-use manual for stakeholders and practitioners of the climate service tool. PART III: the durum wheat/pasta sector**	
Brief Description	A manual for the climate tools developed for the durum wheat sector	
WP number	WP4	Co-design of climate service for durum wheat/pasta
Lead Beneficiary		
Contributors	*HORTA (Valentina Manstretta), JRC (Andrej Ceglar), BARILLA (C. Monotti), NOA (C. Giannakopoulos, M. Gratsea), ENEA (Sandro Calmanti)*	
Creation Date	17/02/2021	
Version Number	1.2	
Version Date	18/05/2021	
Deliverable Due Date	31/05/2021	
Actual Delivery Date	31/05/2021	
Nature of the Deliverable	R	*R - Report* *P - Prototype* *D - Demonstrator* *O - Other*
Dissemination Level/ Audience	PU	*PU - Public* *PP - Restricted to other programme participants, including the Commission services* *RE - Restricted to a group specified by the consortium, including the Commission services* *CO - Confidential, only for members of the consortium, including the Commission services*

Revision History Log

Version	Date	Created / Modified by	Pages	Comments
1.0	17-02-2021	NOA	13	Initial Draft
1.1	18-05-2021	HORTA/JRC/NOA /BARILLA	16	Section 6 and 7 on GRANODURO.NET and clisagri
1.2	26-05-201	ENEA	45	Section 8 on seasonal forecast data processing

All partners involved in the production/implementation of the deliverable should comment and report (if needed) in the above table. The above table should support the decisions made for the specific deliverable in order to include the agreement of all involved parties for the final version of the document.

Finally, after the peer review process, the deliverable should be modified according to the comments and the reflections to the comments should be reported in the above table.

Disclaimer

Table of Contents

List of tables and figures

EXECUTIVE SUMMARY

This user guide is designed to provide guidance to potential users of the pilot climate service tools developed in the frame of the MED-GOLD project for the durum wheat sector. In this document you will find information about:

- Granoduro.net
- Clisagri

The purpose of this manual is to provide all users of the provided services with a reference manual containing step-by-step instructions on how to log-on, use the tools, download and interpret the results.

The aim of the climate service tools is to enable the end-users to adapt their decision making strategies to the climate conditions and climate change, by providing indicators assessing the risks associated with unfavourable climate conditions during sensitive crop growth stages.

1. OBJECTIVES

With this deliverable, the project has contributed to the achievement of the following objectives (DOA, PartB Table1.1):

No.	Objective	Yes
1	To co-design, co-develop, test, and assess the added value of proof-of-concept climate services for olive, grape, and durum wheat	X
2	To refine, validate, and upscale the three pilot services with the wider European and global user communities for olive, grape, and durum wheat	
3	To ensure replicability of MED-GOLD climate services in other crops/climates (e.g., coffee) and to establish links to policy making globally	
4	To implement a comprehensive communication and commercialization plan for MED-GOLD climate services to enhance market uptake	
5	To build better informed and connected end-user communities for the global olive oil, wine, and pasta food systems and related policy making	

2. IMPACT

No.	Expected impact	Yes
1	Providing added-value for the decision-making process addressed by the project, in terms of effectiveness, value creation, optimised opportunities and minimised risk	The user manual is essential for promoting and exploiting the developed climate services of the project, since it provides all the necessary information to the end-users (farmers, agronomists, policy makers, scientists) to guide and encourage them to use the tool and take full advantage of the product.
2	Enhancing the potential for market uptake of climate services demonstrated by addressing the added value	
3	Ensuring the replicability of the methodological frameworks for value added climate services in potential end-user markets	
4	To implement a comprehensive communication and commercialization plan for MED-GOLD climate services to enhance market uptake	
5	To build better informed and connected end-user communities for the global olive oil, wine, and pasta food systems and related policy making	

3. DEFINITIONS

Concepts and terms used in this document and needing a definition are included in the following table:

Concept / Term	Definition
Granoduro.net®	The decision Support System (DSS) developed by HORTA for the durum wheat providers of BARILLA
DELPHI model	DELPHI modelling system is a CNR-BARILLA proprietary software to perform yield and grain protein content forecasting as well as risk of pest and diseases

4. ACRONYMS

Acronyms used in this document and needing a definition are included in the following table:

Acronym	Definition
CLISAGRI	It is a modelling system based on a set of dynamic agro-climatic indicators that bring key information to crop producers during different stages of the crop growth.
ERA5	ECMWF Atmospheric ReAnalysis 5
CU	Crop Units
DSS	Decision Support System
GDD	Growing Degree Days
TSUM1	Total temperature (in degree-days) from emergence until flowering
TSUM2	Total temperature (in degree-days) from flowering to maturity
rh	relative humidity
ws	wind speed
d2m	2 m dew point temperature
t2m	2 m atmospheric temperature
RPSS	Ranked Probability Skill Score

5. REFERENCES

The following documents, although not part of this document, amplify or clarify its contents. Reference documents are those not applicable and referenced within this document. They are referenced in this document in the form [RD.x]:

Ref.	Title	Code	Version	Date
[RD.1]	First Feedback report from users on durum wheat pilot service development	D 4.6	final	2019
[RD.2]	Second Feedback report from users on durum wheat pilot service development	D 4.7	final	2020
[RD.3]	Report on the identified specific needs and opportunities	D 4.1	final	2018
[RD.4]	Design of innovative agro-climatic systems for durum wheat	D 4.2	final	2020
[RD.5]	Evaluation of the pilot service	D 4.3	final	2021
[RD.6]	Clisagri: An R package for agro-climate services. in Climate Services, 20, 100197.	Ceglar, A. et al.	final	2020

6. GRANODURO.NET

6.1. GENERAL INFORMATION

Granoduro.net® is a commercial Decision Support System(DSS) for the sustainable management of the durum wheat crop, which is used by farmers, technicians and agronomists in the Barilla supply chain. It provides advice for crop management based on observed weather data and predictions. In the framework of the MED-GOLD project, a new prototype functionality using seasonal forecasts was added to the system.

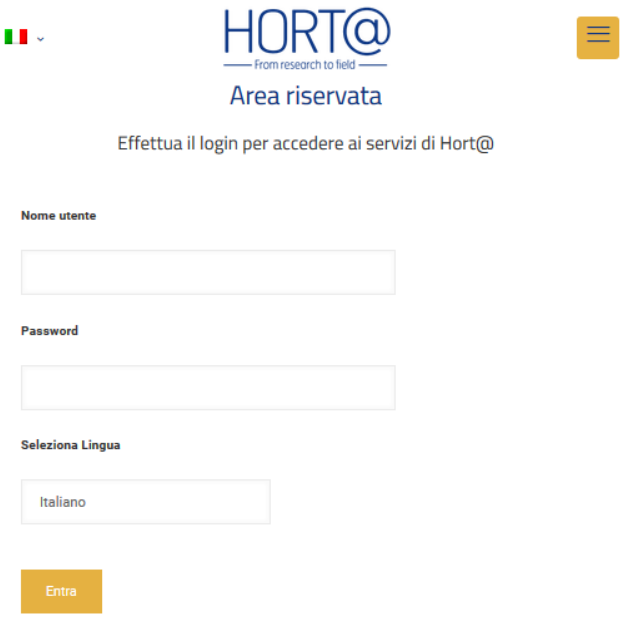
As reported in previous deliverables, during the MED-GOLD project the new prototype functionality was co-developed with users of durum wheat sector [RD.1, RD.2, RD.3, RD.4, RD.5] and presents several tabs: one for the phenological development of durum wheat; one displaying risk indexes for the main wheat diseases, one displaying climatic indexes calculated by the Clisagri R package and finally one displaying the value of the same indexes calculated on the base of historical data. The new prototype functionality based on bias-corrected seasonal forecasts data has been developed. For the calculation of some standardised indicators, also historical ERA5 observation data are used. In the new functionality, models are fed with weather station data for past and present, and then with seasonal forecast data starting from the dates in which they are issued.

6.2 STEP-BY-STEP INSTRUCTIONS

6.2.1 USER ACCESS

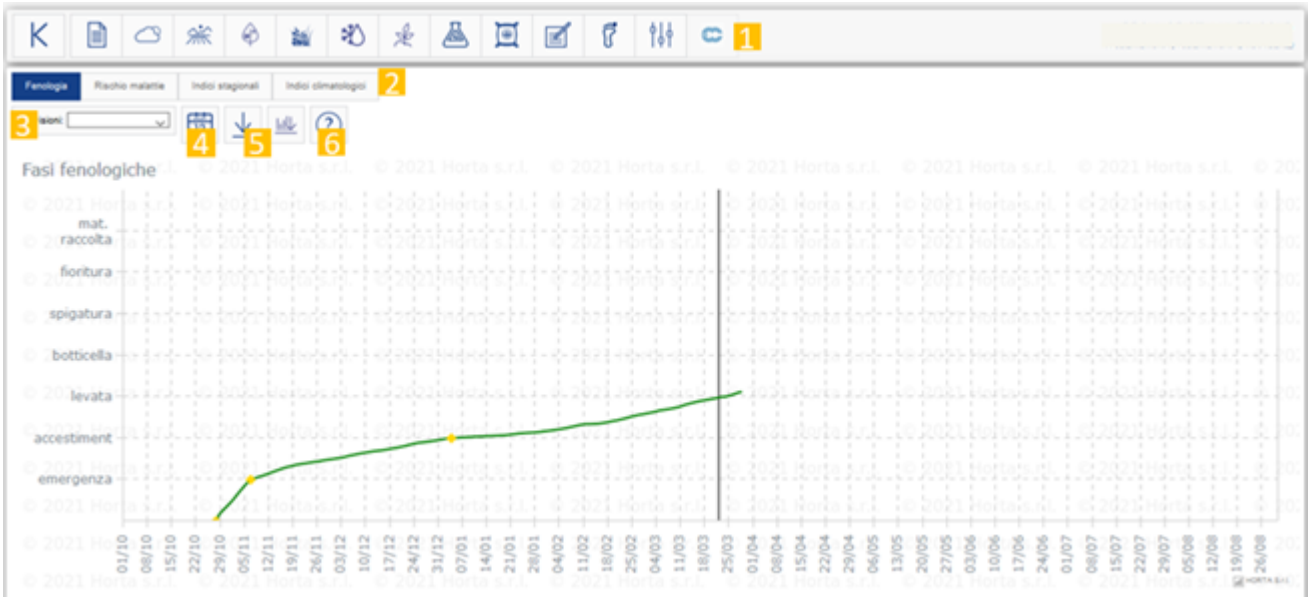
Granoduro.net® can be accessed by registered users through the login page of the Horta website (www.horta-srl.it/area-riservata). Users need to enter username and password (Fig. 6-1).

Figure 6-1: Granoduro.net® login page (https://www.horta-srl.it/area-riservata/)



When accessing the DSS, the user can visualise all his registered CU (Crop Units), and for each one of them can select the icon for the new prototype functionality (Fig. 6-2, No 1).

Figure 6-2: Overview of the new prototype functionality in granoduro.net®. Numbers refer to options explained in the text.



6.2.2 New prototype functionality

In the new prototype functionality, there are different tab options, on the basis of the information the user wants to visualise (Fig. 6-2, No 2). The tabs which can be selected refer to:

- crop phenology, indicating when the main crop stages are forecasted to be,
- the risk for the main diseases of wheat,
- climatic indicators based on hydrological balance, cold and hot stress,
- historical values of climatic indicators.

The user can select the month in which seasonal forecasts have been issued, from the list of all the forecasts available (Fig. 6-2, No 3).

Outputs are then displayed in the area of the main graph. The user can also choose to display the same results in a table, by clicking on the button (Fig. 6-1, No 4), or to download the output (Fig. 6-1, No 5). A Help tab (Fig. 6-1, No 6) is available to display additional information.

6.3 HOW TO DOWNLOAD/INTERPRET THE RESULTS

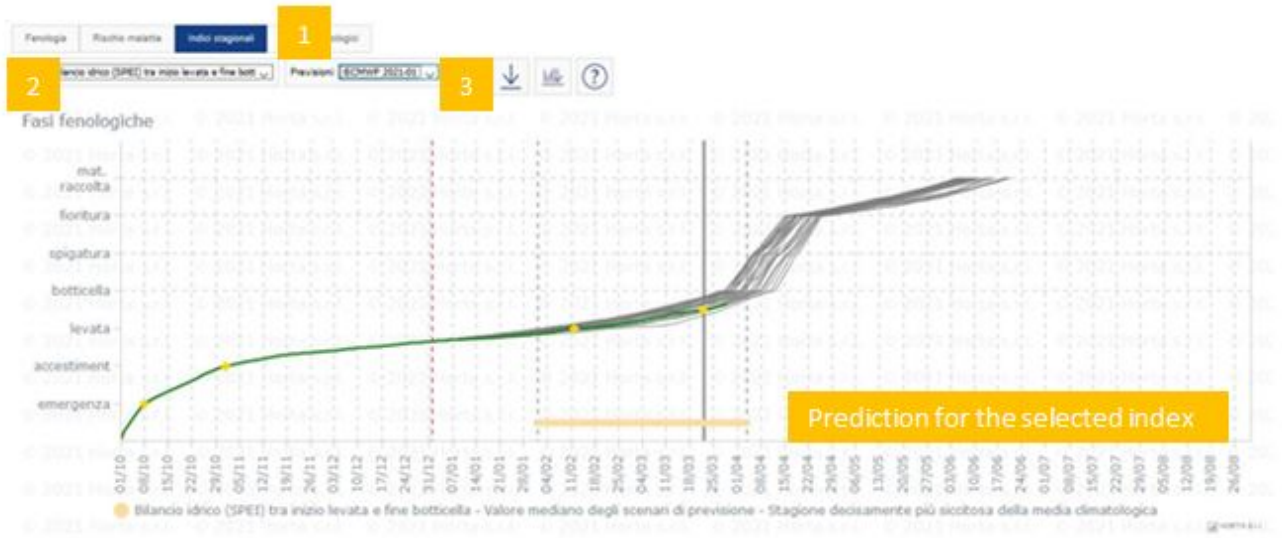
Use Case

You are a farmer growing durum wheat in different fields close to the Italian city of Ravenna. You want to make sure that you apply the correct amount and type of nitrogen fertiliser when the crop has the highest nutrient demand. You know that fertiliser application has a direct impact on grain yield and quality, so you need to know the climate conditions to plan accordingly. Is it going to be specially wet or dry during fertilizers' application?

- On the platform, select one of your crop units
- Choose the seasonal forecast functionality (Fig. 6-2, No 1)
- Select the climatic indicators tab (Fig. 6-3, No 1)
- Choose the indicator on hydrological balance between the beginning of stem elongation and the end of booting, a period with a high nutrient demand (Fig. 6-3, No 2)
- Finally, specify the month in which the forecast is issued (e.g. February) (Fig. 6-3, No 3)

The coloured bar shows you that the selected period could be dryer than normal (Fig. 6-3) and then you can plan the intervention in the field accordingly.

Figure 6-3: Screenshot from the use case in granoduro.net®.



7. CLISAGRI

7.1 GENERAL INFORMATION

Clisagri is based on a set of dynamic agro-climatic indicators that bring key information to crop producers during different stages of the crop growth. A dynamic approach has been implemented to assess the risks associated with climate extremes occurring during sensitive crop phenological stages by integrating dedicated agro-climate risk indicators with a dynamic model to predict crop phenology. Clisagri quantifies the occurrence of different climatic extremes such as drought, excessive wetness, heat stress and cold stress during sensitive crop growth stages. These sensitive phases generally occur in different periods every year as a consequence of inter-annual climate variability and Clisagri offers an effective way to dynamically take this variability into account.

Furthermore, Clisagri is designed to test a complete range of possible crop varieties, and an optimization algorithm based on genetic algorithms has been implemented to select the variety that is 'the most fit' for given climatic conditions [RD.6].

Figure 7-1 represents the types of indicators calculated by the Clisagri package. The indicators can be divided into three groups: hydrological balance indicators, heavy rainfall indicators and temperature stress indicators. The Clisagri package enables the calculation of indicators for fixed time periods or dynamically, where the indicator estimation is coupled to a phenological development model. Figures 7-2 and 7-3 provide a description of implemented agro-climate indicators in Clisagri, as provided in the: https://github.com/ec-jrc/Clisagri

Figure 7-1: Different types of agro-meteorological indicators are needed to fully characterize meteorological conditions during different stages of winter wheat growth. Four groups of indicators are here chosen to characterize hydrological balance, excessive wetness, cold stress and heat stress conditions.

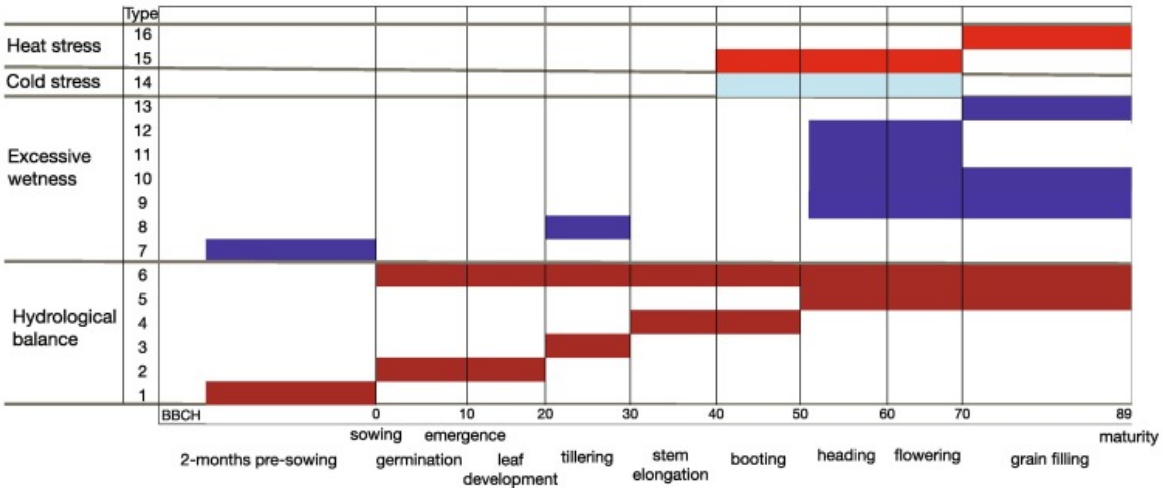


Figure 7-2: Description of hydrological balance indicators (https://github.com/ec-jrc/Clisagri/blob/master/clisagri description#)

```
########## hydrological balance ##############
########## 1 - hydrological balance during pre-sowing peridod (September and October)
########## 2 - hydrological balance around sowing (i.e. from sowing (DVS=0) until emergence (DVS=0.01))
########## 3 - hydrological balance during tillering (from DVS=0.1 to DVS=0.33)
########## 4 - hydrological balance between stem elongation period (DVS=0.34) and heading (DVS=0.89)
########## 5 - hydrological balance between stem flowering period (DVS=0.90) and maturity (DVS=2)
########## 6 - hydrological balance between sowing (DVS=0) and maturity (DVS=2)

# type=1
### hydrological balance during pre-sowing period (September and October)
### no DVS: SPEI for period between September - October
### DVS present: SPEI for period September - October
# type=2
### hydrological balance between sowing (i.e. from germination (BBCH=0) and emergence (BBCH=9))
### no DVS: SPEI for period between October and November
### DVS present: SPEI for period between BBCH 0 (sowing) and BBCH 9 (emergence)
# type=3
### hydrological balance during tillering (from BBCH=20 to BBCH=29)
### no DVS: SPEI for period between December and March
### DVS present: SPEI for period between no tillers (BBCH=20) and end of tillering (BBCH=29)
# type=4
### 4 - hydrological balance between beginning of stem elongation period (BBCH=30) and end of booting (BBCH=49)
### no DVS: SPEI for period between April and May
### DVS present: SPEI for period between beginning of stem elongation period (BBCH=30) and end of booting (BBCH=49)
# type=5
### 5 - hydrological balance between beginning of heading period (BBCH=50) and full maturity (BBCH=89)
### no DVS: SPEI for period between May and June
### DVS present: SPEI for period between beginning of heading period (BBCH=50) and full maturity (BBCH=89)
# type=6
### 6 - hydrological balance between sowing (BBCH=0) and full maturity (BBCH=89)
### no DVS: SPEI for period between November and June
### DVS present: SPEI for period between sowing (BBCH=0) and full maturity (BBCH=89)
```

Figure 7-3: Description of excessive wetness and temperature stress indicators (https://github.com/ec-jrc/Clisagri/blob/master/clisagri description#)

```
######### heavy/excessive rainfall indicators #############
# type=7 – rainfall during pre-sowing period (September and October)
### rainfall cumulate before sowing
### no DVS: rainfall cumulate for period between September and October
### DVS present: rainfall cumulate for period between September and October
# type=8 – number of days with rain above 10 mm during tillering period
### no DVS: number of days with rain above 10 mm between December and March
### DVS present: rnumber of days with rain above 10 mm during tillering (BBCH=20 – 29)
# type=9 – number of days with rain above 40 mm during tillering
### no DVS: number of days with rain above 40 mm between December and March
### DVS present: number of days with rain above 40 mm during tillering (BBCH=20 – 29)
# type=10 – number of days with rain above 5 mm between beginning of heading and full maturity
### no DVS: number of days with rain above 5 mm between April and June
### DVS present: number of days with rain above 5 mm between beginning of heading and full maturity (BBCH=51 – 89)
# type=11 – number of days with rain above 40 mm between beginning of heading and full maturity
### no DVS: number of days with rain above 40 mm between April and June
### DVS present: number of days with rain above 40 mm between beginning of heading and full maturity (BBCH=51 – 89)
# type=12 – maximum number of consecutive days with rain above 5 mm between beginning of heading and end of enf of flowering
### no DVS: maximum number of consecutive days with rain above 5 mm between April and May
### DVS present: maximum number of consecutive days with rain above 5 mm between beginning of heading and end of flowering (BBCH=51 – 69)
# type=13 – maximum number of consecutive days with rain above 5 mm between end of flowering and full maturity
### no DVS: maximum number of consecutive days with rain above 5 mm between May and June
### DVS present: maximum number of consecutive days with rain above 5 mm between end of flowerign and full maturity (DVS=69 – 89)

#######
######### cold stress indicators #############
######### Number of days with minimum daily temperature below 2 deg. C between booting (BBCH=41) and end of flowering (BBCH=69)
# type=14 – Number of days with minimum daily temperature below 2 deg. C between booting and flowering
### no DVS: number of days with minimum daily temperature below 2 deg. C between April and May
### DVS present: number of days with minimum daily temperature below 2 deg. C between booting (BBCH=41) and end of flowering (BBCH=69)

#######
######### heat stress indicators #############
## type 15 – Number of hot days with maximum daily temperature above 28 deg. C between end of stem elongation  and end of flowering
## no DVS: Number of hot days with maximum daily temperature above 28 deg. C between March and May
## DVS present: Number of hot days with maximum daily temperature above 28 deg. C between end of stem elongation (BBCH=39) and end of flowering (BBCH=69)
## type 16 – Number of hot days with maximum daily temperature above 28 deg. C between during grain filling period
## no DVS: Number of hot days with maximum daily temperature above 28 deg. C between May and June
## DVS present: Number of hot days with maximum daily temperature above 28 deg. C during grain filling period (BBCH=69 – 89)
```

7.2 STEP-BY-STEP INSTRUCTIONS

Clisagri can be obtained from https://github.com/ec-jrc/Clisagri/. After downloading the package to a local machine, the functions are ready to use without installation. Below, we present an example use of the package.

Figure 7-4 represents the base functions implemented in the Clisagri. The current version supports the calculation of 16 indicators with either static or dynamic time periods (defined by crop development stage). Additionally, the graphical tool allows mapping intensity of climate events that could affect growth and yield of pre-identified crop varieties, as given by the combination of TSUM1 and TSUM2 (thermal requirements for vegetative and reproductive periods, respectively) for the standard approach or combination of TSUMs (characterizing each sub-phase) in the multi-phase phenological model.

Figure 7-4: List of Clisagri core available functions.

Function	Description
clis.agri	Calculates a set of agro-climate indicators characterizing weather conditions during sensitive stages of durum wheat growth
phenology	Function calculating dynamic phenological development stages of wheat based on provided weather data and wheat variety related parameters characterizing thermal requirements for vegetative and reproductive growing periods
phenology.parameters	Description of phenological parameters used by phenology function
hazard.map	Displays 2D plot of occurrence of event based on provided range of TSUM1 and TSUM2
clim.plot	Displays time series plot of intensity of unfavourable events based on different combinations of TSUM1 and TSUM2
phenology.breeder	Function calculating dynamic phenological development stages of wheat based on provided weather data and wheat variety related parameters characterizing thermal requirements for 6 different sub-phases
breeder.calc	Fitness function providing fitness score for genetic optimization algorithm
breeder.plot	Displays phenological development stages of optimal variety, selected by genetic algorithm, and indicators that were used to minimize the impact on crop growth

7.2.1 Use Cases

The phenological development of durum wheat can be simulated by using the phenology function of Clisagri, which requires daily meteorological data (daily minimum and maximum temperatures), showing dates and a parameter matrix. The parameter description for the phenological model can be invoked by calling the function phenology.parameters(). These parameters should reflect the wheat variety under consideration, since they are used to simulate daily development rate, vernalization and photoperiod sensitivity. The default set of parameters represents a typical Mediterranean durum wheat variety grown in Italy. The most relevant parameters are TSUM1 and TSUM2 representing, respectively, the thermal requirement to reach flowering and physiological maturity. The function phenology can be used to simulate DVS for a unique combination of TSUM1 and TSUM2, as provided in the parameters matrix.

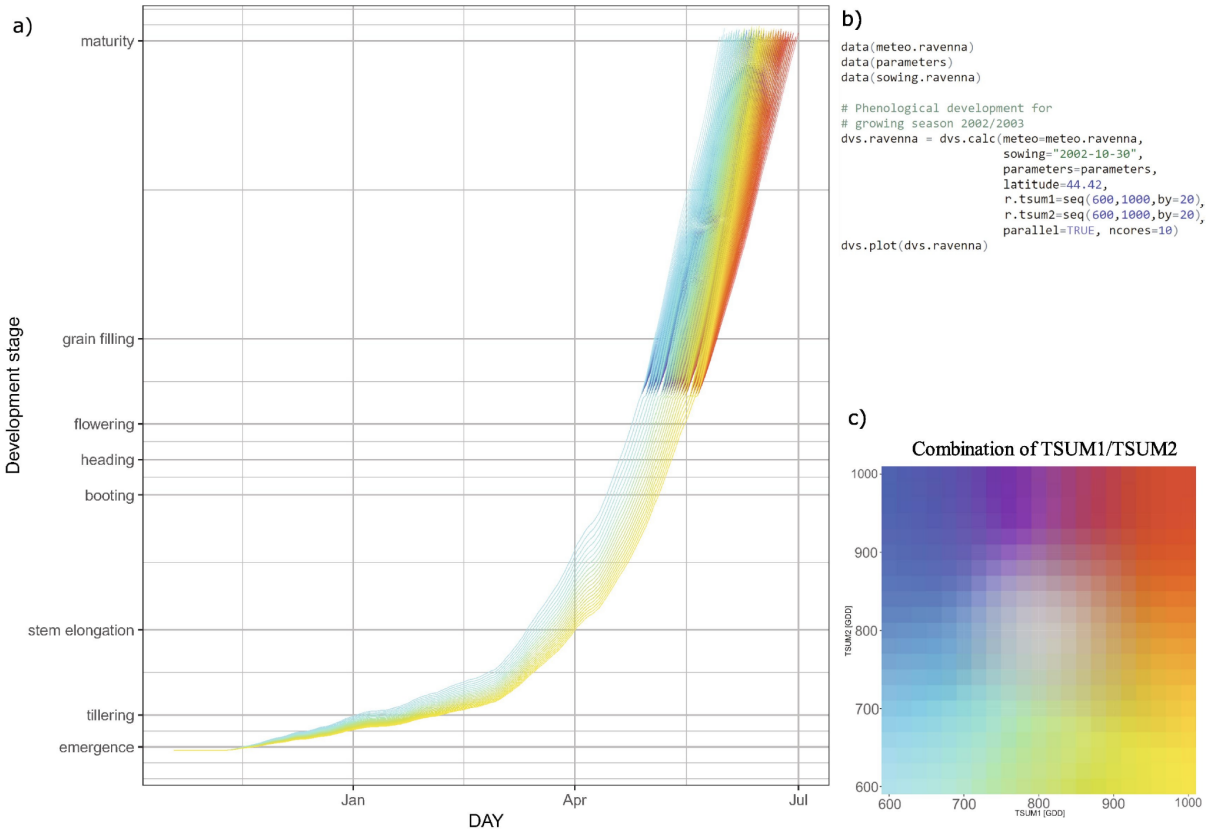
In order to get information for the phenological development for the growing season 2002/2003 in Ravenna, one should type the following R code:

data(meteo.ravenna)
data(parameters)
data(sowing.ravenna)

dvs.ravenna = dvs.calc(meteo=meteo.ravenna, sowing="2002-10-30", parameters=parameters, latitude=44.42, r.tsum1=seq(600,1000,by=20), r.tsum2=seq(600,1000,by=20), parallel=TRUE, ncores=10)
dvs.plot(dvs.ravenna)

Figure 7-5 shows the simulated development stage in Ravenna (assuming the sowing date being at the end of October 2002), using different combinations of TSUM1 and TSUM2 (represented by different colors in Fig. 7-5c). Assuming the base temperature of 0 °C, the range of TSUM1 and TSUM2 between 600 and 1000 growing degree days (GDD) is typical for Mediterranean durum wheat varieties grown in Italy. Fig. 7-5a demonstrates that the timing of different development stages can vary considerably depending on the selected variety. For example, flowering occurrences can range between mid-April and the beginning of May, depending on TSUM1. Different combinations of TSUM1/TSUM2 give many more possibilities on the development after flowering occurs (i.e. for each TSUM1 there is an entire range of TSUM2 values defining the thermal length of the reproductive period). The varieties with the longest thermal requirement would have reached maturity until the end of June in 2003.

Figure 7-5: Simulated durum wheat development stage for the crop growing season 2002/3 in Ravenna (a) based on different combinations of thermal requirements for vegetative (TSUM1) and reproductive (TSUM2) growing periods. To distinguish between different combinations, the color of each DVS time series in (a) is related to scale in (c), with each color representing a unique combination of TSUM1/TSUM2. The R code to produce the DVS time series graph is shown in (b).

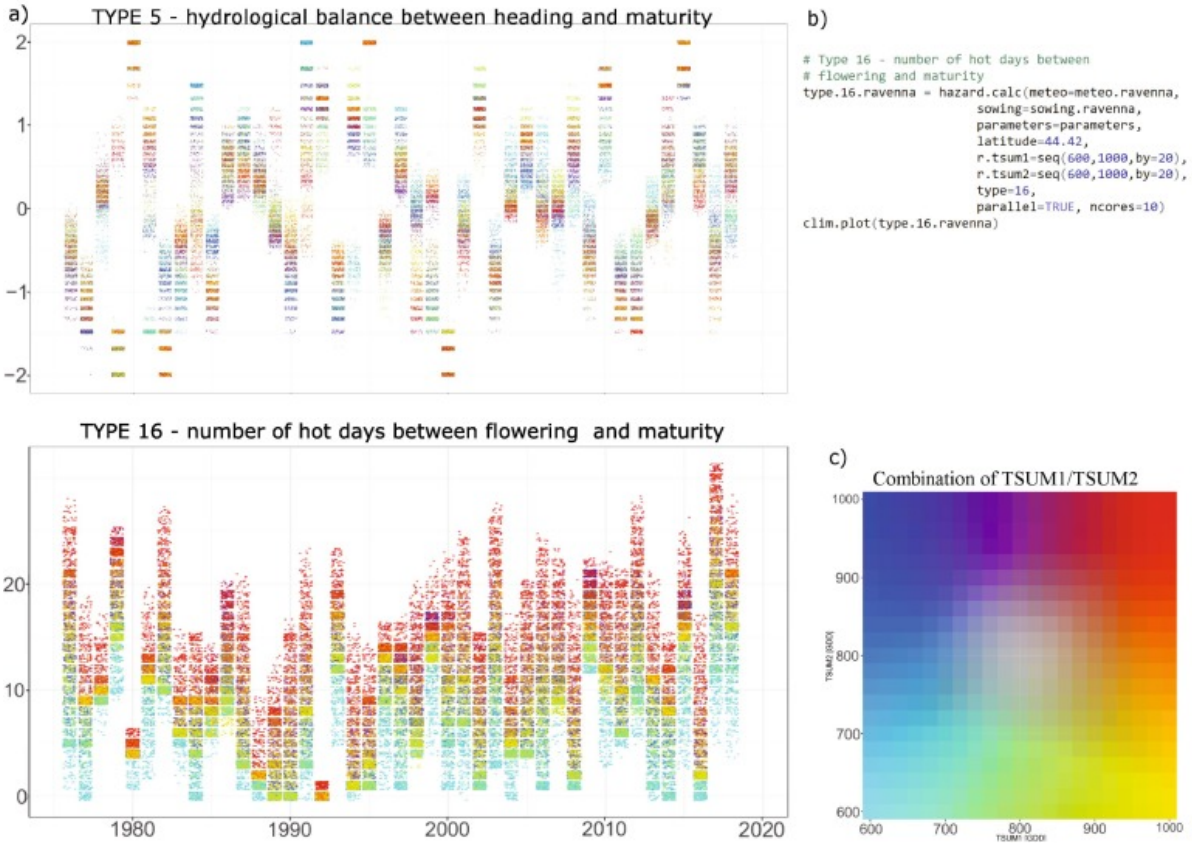


To further illustrate the climate effects during the different growing stages, we calculate the hydrological balance and the number of hot days between the beginning of flowering and maturity (indicators 5 and 16,

respectively). Choosing different combinations of TSUM1/TSUM2 clearly results in a wide range of drought and heat stress conditions that crops could experience during and after the flowering (Fig. 7-6). The R code used to produce the number of hot days (also shown in Figure 7-6b) is:

type.16.ravenna = hazard.calc(meteo=meteo.ravenna, sowing=sowing.ravenna, parameters=parameters, latitude=44.42, r.tsum1=seq(600,1000,by=20), r.tsum2=seq(600,1000,by=20), type=16, parallel=TRUE, ncores=10)
clim.plot(type.16.ravenna)

Figure 7-6: Simulated hydrological balance between heading and maturity (a) and number of hot days between flowering and maturity, based on different combinations of TSUM1/TSUM2 (indicated by different colors). To distinguish between different combinations, the color of each simulated indicator is related to scale in (c), with each color representing a unique combination of TSUM1/TSUM2. The R code to produce the indicator of type 16 (number of hot days) is shown in (b).



Further information on Clisagri with additional use cases can be found in [RD.6].

8. SEASONAL CLIMATE PREDICTION PROCESSING

8.1 GENERAL INFORMATION

The design of the seasonal climate prediction data processing workflow has been described in [RD.4] and is structured over four components:

- the data retrieval and storage;
- the data pre-processing, which prepares the raw data for the required computation
- a computation layer which includes the bias correction of seasonal forecast data and the computation of bioclimatic indicators;
- a data export layer where data are made available to the users or to other components of the MED-GOLD ICT Platform.

In this section a complete step-by-step procedure is described to retrieve data from the Copernicus Data Store and run a processing workflow that produces:

- bias-corrected data seasonal forecast data interpolated on the ERA5 reference grid;
- bioclimatic indicators based on the package **`clisagri`** developed by JRC.

The processing workflow is described in Figure 8-1 and the software package which is necessary to execute the workflow is available on the MED-GOLD repository (https://start.med-gold.eu/source/durum/). In the followings, a brief description of each component of the workflow is provided.

8.2 STEP-BY-STEP INSTRUCTIONS

`MEDGOLD_UPDATE-SEASONAL_cdsapi.py`

This PYTHON script provides an interface with the climate data store to retrieve the multi-model seasonal forecast data which are necessary for the computation of bioclimatic indicators and that can be used in other downstream applications such as GRANODURO.NET®. In particular the following models (*origin)* and corresponding data streams are considered:

```
#Choose model and stream
origins  =  ['ecmwf' ,'meteo_france','dwd','cmcc']
systems  =  ['5'    ,'6'                ,'2'   ,'3'   ]
```

For each model datastream, daily data for the following variables are retrieved:

```
vars = [
         'maximum_2m_temperature_in_the_last_24_hours',
         'minimum_2m_temperature_in_the_last_24_hours',
       'total_precipitation',
       'surface_solar_radiation_downwards',
       '10m_u_component_of_wind',
       '10m_v_component_of_wind',
       '2m_dewpoint_temperature',
       '2m_temperature',
       'mean_sea_level_pressure',
       ]
```

Provided a root data path as follows

```
dpath  =  '$HOME/DATA/SEASONAL'
```

model data are stored locally in GRIB format in a separate folder for each year

```
exp_dir = dpath + '/' + origin + '/' + 'GRIB' + '/' + year
```

MEDGOLD_UPDATE-ERA5_cdsapi.py

This PYTHON script provides an interface with the climate data store to retrieve observational data for the available reanalysis on the CDS. In particular, hourly data are retrieved for the selected reanalysis (for example ERA5 or ERA5-Land), for the following variables:

```
vars = ['total_precipitation',
       '2m_temperature',
       '2m_dewpoint_temperature',
       'surface_solar_radiation_downwards',
       '10m_u_component_of_wind',
       '10m_v_component_of_wind','surface_pressure']
```

Provided a root data path as follows

```
dpath  =  '$HOME/DATA/REANALYSIS/<OBSNAME>'
```

model data are stored locally in NETCDF format in a separate folder for each year

```
exp_dir = dpath + '/' + year
```

grib2netcdf.cdo

This application converts the seasonal forecast data from the initial GRIB format to NetCDF and performs some simple additional computation which include the derivation of wind speed (*ws*) from the wind components (*u10, v10*) retrieved from the CDS, and the derivation of relative humidity (*rh*) using dew point temperature (*d2m*) and 2-meter atmospheric temperature (*t2m*). Also, this application derives daily cumulated values of rainfall and solar radiation which are normally available as total cumulated values from the beginning of the forecast until any future lead time.

Provided a root data path as follows

```
ARCHIVELOCAL  =  '$HOME/DATA/SEASONAL'
```

the output data in NetCDF format are stored in separate folders for each model.

```
ARCHIVENCDF=${ARCHIVELOCAL}/${MODELNAME}/NCDF
```

hourly2daily.cdo

This application works on the ERA5 reanalysis data, which are retrieved from the CDS as hourly records. The application computes the daily statistics and derives the values of wind speed (*wss*) and relative humidity (*rh*) using the same formulas adopted for the seasonal forecast data. The current version of the R package `CSTools` requires that the observational data are split into different files for each month of the year. Therefore, the application **`hourly2daily.cdo`** performs also the splitting of data into monthly files.

Provided a root data path as follows

```
ARCHIVELOCAL  =  '$HOME/DATA/REANALYSIS/<OBSNAME>/'
```

the output data in NetCDF format are split into monthly files and stored in separated folders for year in the dataset:

```
ARCHIVESPLIT=${ARCHIVELOCAL}/MONTHLY/${year}
```

`MEDGOLD_biascorrection.R`

This application prepares the bias-corrected data and makes them available for the end-users and as input for the computation of bioclimatic indices. A key resource for the processing of climate data is the R package CSTools (https://cran.r project.org/web/packages/CSTools/index.html) which provides all the required functionalities to a) find the data in the local file system; b) import data into the current workspace as standardized multi-dimensional arrays; c) perform the bias-correction as described in [RD.4] d) perform the skill assessment of the forecasts and d) prepare the output files in NetCDF format.

The application performs the following key operations.

1. Load the required data into the local workspace.
2. Sub-domain selection and interpolation.
3. Applies *CST_Calibration* and the *CST_QuantileMapping* algorithms as appropriate (see [RD.4] for more details)
4. Export bias-corrected data. Two different formats, according to the needs of different types of users, are available:
    a. NetCDF export files containing gridded data for the entire sub-domain selected in step 2.
    b. .csv export files containing time series for a set of selected grid cells

BATCH MODE

The script is prepared to be executed in batch mode with input from the command line as follows:

```
Rscript biascorrection.R ARG_TYPE ARG_VAR ARG_YREN ARG_MONSTS ARG_DOMAIN ARG_EXPNAME ARG_OBSNAME
```

It is assumed that the list of arguments is either void or provides a full list of arguments and arguments must be provided in the correct order:

`ARG_TYPE`: either `GRID` or `AREA` in order to process selected grid points or an entire area
`ARG_VAR`: name of the variable to be processed (e.g. `tp, tmax2m, tmin2m, t2m, rh, ssrd, wss`)

ARG_YREN: last year to be processed the for bias correction
ARG_MONSTS: starting date for the seasonal forecast to be processed
ARG_DOMAIN: name of the subdomain to biascorrect (see naming of subdomains in the script)
ARG_EXPNAME: folder name for the model forecast
ARG_OBSNAME: folder name for the reference observations

If the ARG_TYPE is set to GRID an input file with the coordinates of the grid cells to be processed must be provided. In the example available on the project repository, the name of the input file is

```
MDG-GDN_ITALY_ERA5_r1440x720_v1.1.csv
```

And an example of the corresponding content is provided in Appendix A.

OUTPUT

Provided a root data path as follows

```
dpath    =  '$HOME/DATA/SEASONAL/'
```

the output data are stored in a different folder structure, depending on the type of processing selected with ARG_TYPE.

- For the AREA processing, NetCDF files are stored in separated folders for year in the dataset:

```
out_basepath_exp <- paste0('dpath/',exp_name,'/BC/NCDF/',year)
```

- For the GRID processing comma separated value (csv) files are stored in separated folders for each cell ID and for each year in the dataset:

```
out_basepath_exp <- paste0('dpath/',exp_name,'/BC/CSV/',<cellID>,’/’year)
```

MEDGOLD_clisagri.R

This application computes several bioclimatic indicators which are relevant for the durum wheat sector as discussed in the deliverable 4.7 (RD.2). It performs the following operations:

1. Load required data
2. Sub-domain selection if required
3. Merging of the observational data with the ensemble forecasts to allow the computation of bio-climatic indices. See DEL4.2 for more details
4. The computation of the bioclimatic indices.
5. Computation of the Ranked Probability Skill Score (RPSS)
6. Write output files:

The application uses the package Clisagri published by JRC and produced the following output files:

a. `<MODELNAME>_<DOMAINNAME>_c3s_<INDEXNAME>_<BCTYPE>_stdt_<MONTH>_<YEAR>_PROBABILITY.nc`: **probability** associated to the considered terciles for all the available past years;

b. `<MODELNAME>_<DOMAINNAME>_c3s_<INDEXNAME>_<BCTYPE>_stdt_<FORECASTMONTH>_<FORECASTYEAR>_TERCILE.nc`: **most likely tercile** according to the distribution of the ensemble forecasts and therefore to the probability associated to the considered terciles;

c. `<OBSNAME>_<DOMAINNAME>_<INDEXNAME>_<HARVESTYEAR>_PERCENTILE.nc`: **percentile** associated to the value of the bio-climatic index computed with the observational data;

d. `<MODELNAME>_<DOMAINNAME>_c3s_<INDEXNAME>_<BCTYPE>_stdt_<FORECASTMONTH>_<FORECASTYEAR>_RPSS.nc`: **RPSS** values using <OBSNAME> as a reference observation.

In particular, provided a root data path as follows

```
dpath    =   '$HOME/DATA/SEASONAL/'
```

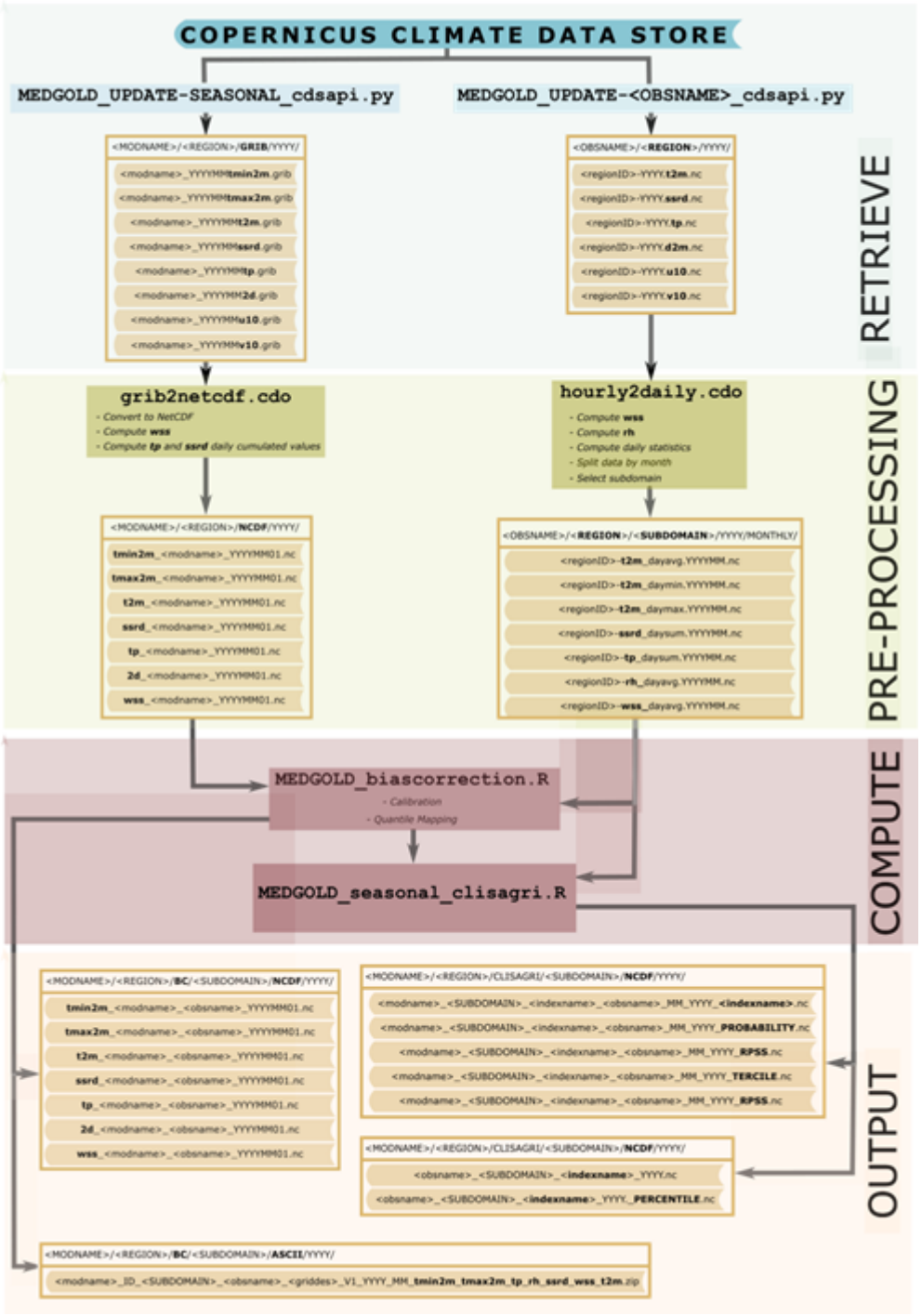
NetCDF files are stored in separated folders for year in the dataset:

```
out_basepath_exp <- paste0('dpath/',exp_name,'/CLISAGRI/',year)
```

Note that the `year` included in the file path is referred to harvest date, which can be different from the year of the corresponding starting date. For example, the folder *2021* contains the indicators computed on the basis of the forecast started in November 2020 and the starting date of the forecast is indicated in the name of the file start.

In order to provide a complete reference to the modelling approach, the full listing of the script `MEDGOLD_clisagri.R` is provided in APPENDIX B.

Figure 8-1: MED-GOLD seasonal forecast data processing workflow.


COPERNICUS CLIMATE DATA STORE
MEDGOLD_UPDATE-SEASONAL_cdsapi.py
MEDGOLD_UPDATE-<OBSNAME>_cdsapi.py
RETRIEVE
grib2netcdf.cdo
- Convert to NetCDF
- Compute wss
- Compute tp and ssrd daily cumulated values
hourly2daily.cdo
- Compute wss
- Compute rh
- Compute daily statistics
- Split data by month
- Select subdomain
PRE-PROCESSING
MEDGOLD_biascorrection.R
- Calibration
- Quantile Mapping
MEDGOLD_seasonal_clisagri.R
COMPUTE
OUTPUT

8.2.1 Use case

As a sample use case for the tools described in section 8.2 an **experimental operational setup to produce bias-corrected data and bio-climatic indices** is reported.

The workflow described in Figure 8-1 can be executed by installing the *crontab* described in Table 8-1 below. The shell scripts used in the sample crontab are reported in APPENDIX C-F.

The *crontab* script reporte in Table 8-1 has been successfully to prepare seasonal forecast data for: the modelling system DELPHI, for the DSS GRANODURO.NET, and for the MED-GOLD Dashboard described in DEL 1.8 with minimum required human intervention to check the successful execution of all required steps in the workflow. This system will be used at least until the end of the project to produce the relevant data streams and enable prototype services for the forthcoming participatory workshops

Table 8-1: A sample crontab for the execution of the workflow reported in Fig. 8-1

#Min	#Hour	#DayOfMonth	#Month	#Dayofweek
00	02	*	0	<CRONDIR>/MEDGOLD_RETRIEVE-UPDATE_CRON.sh
00	04	*	*	<CRONDIR>/GRIB2NCDF_MEDGOLD_CRON.sh
17	20	*	*	<CRONDIR>/BIASCORRECTION_ON.sh
01	11	*	1	<CRONDIR>/CLISAGRI.sh

9. CONTACT AND PENDING ISSUES

For the granoduro.net® new prototype functionality, inquiries and comments should be sent to v.manstretta@horta-srl.com.

For the Clisagri functionality, inquiries and comments should be sent to andrej.ceglar@ec.europa.eu.

For the climate data processing, inquiries and comments should be sent to sandro.calmanti@enea.it and alessandro.dellaquila@enea.it

ANNEX A.

Sample content input data file for the bias correction of single grid cells. Each row contains: the ID of the grid cell, the latitude and longitude of the center of the grid cell, the latitude and longitude of the four corners of the grid cell

```
X.ID,lat,lon,latNW,lonNW,latNE,lonNE,latSE,lonSE,latSW,lonSW
000001,47.875,5.75,47.75,5.625,47.75,5.875,48,5.875,48,5.625
000002,47.625,5.75,47.5,5.625,47.5,5.875,47.75,5.875,47.75,5.625
000003,47.375,5.75,47.25,5.625,47.25,5.875,47.5,5.875,47.5,5.625
000004,47.125,5.75,47,5.625,47,5.875,47.25,5.875,47.25,5.625
000005,46.875,5.75,46.75,5.625,46.75,5.875,47,5.875,47,5.625
000006,46.625,5.75,46.5,5.625,46.5,5.875,46.75,5.875,46.75,5.625
000007,46.375,5.75,46.25,5.625,46.25,5.875,46.5,5.875,46.5,5.625
000008,46.125,5.75,46,5.625,46,5.875,46.25,5.875,46.25,5.625
000009,45.875,5.75,45.75,5.625,45.75,5.875,46,5.875,46,5.625
```

ANNEX B.

Code for the script `MEDGOLD_clisagri.R`.

```
################################################################################
#
#     MED-GOLD Agro Clim
#
#    v1.0 02/07/2019 S. Calmanti - Initial code based on CSTools
#    v1.1 04/07/2019 S. Calmanti - Write NetCDF output with the sys5 standards
#       v2.1 12/03/2020 S. Calmanti and M. De Felice - Parallelization of agro index
computation
#    v2.2 31/03/2020 S. Calmanti Format output for ICT platform
#    v2.3 11/02/2021 S. Calmanti Switch to clisagri
#
################################################################################
#
# INFO - Batch mode submission
# R CMD BATCH <filename>.R  > <filename>.Rout 2>&1 &
#
#   --vanilla   Combine   --no-save,   --no-restore,   --no-site-file,   --no-init-file   and
--no-environ
# --slave Make R run as quietly as possible
#

#Clean the workspace
rm(list=ls())

library(abind)
library(multiApply)
library(ncdf4)
library(easyNCDF)
library(CSTools)
library(zeallot)
library(lubridate)
library(ggplot2)
library(dplyr)
library(s2dverification)
library(easyVerification)
library(parallel)
library(tidyverse)
#source('./Ragroclim.R')
source('./clisagri.r')
source('./f_check_nans.r')
source('./f_ind2prob.r')
source('./f_prob2terc.r')
source('./f_percentile.r')
source('./f_agronames.r')


############################
# Set the type of processing
############################

EXP_COMPUTE <- TRUE
ntypes <- 16

#########################
# Set resources
```

```
##########################

ncpus_read <- 8  #Max cpus for the CST_Load function
ncpus_proc <- 51   #Max cpus for the computation of agro.index

##########################
# Set your local path here
##########################

# This is the climate model baseline path.
exp_basepath <- '/fas_specs/a/sandro/DATA/SEASONAL/ECMWF/NOBC/NCDF'
#
# This is the baseline path for the reference observational data
obs_basepath <- '/fas_c/UTENTI/sandro/DATI/REANALYSIS/ERA5/MONTHLY'
#
# This is the baseline path for the bias corrected data
# Basepath for the clisagri indices
out_basepath_agro <- '/fas_specs/a/sandro/DATA/SEASONAL/ECMWF/CLISAGRI/HRES'

plt_basepath <- '/home/sandro/R/PLOT/MEDGOLD/AGRO/'


##########################
#       Set domain
##########################

idomain <- 3

gridres <- 'r1440x720'
#gridres <- 'r360x180'

##########################
#       Set dates
##########################

harvest_yrst  <- 1994
harvest_yren  <- 2021
harvest_years <- harvest_yrst:harvest_yren

#Set dates for the observations that cover the entire time interval
#over which the clisagri indicators are computed (September-July)
obs_dayst   <- '01'
obs_monst   <- '09'
maxlead_ac <- 365-31-31  #September-June


#Set dates
exp_dayst <- '01'
exp_monsts <- c('02','11','12','01','10','03','04','05')
exp_monsts <- c('09','11','02','04')
maxlead <- 210

biascorrection_tag <- 'not'

#####################################
# Set names an attributes of variables
#####################################

expname <- 'ecmwf'
expname_out <- 'ECMWF'
```

```
obsname <- 'ERA5'
obsname_out <- 'ERA5'

#####################################
#         Start processing
#####################################

#  imonst <- 1
for ( imonst in seq(1:length(exp_monsts))) {

  t0 <- Sys.time()

  exp_monst <- exp_monsts[imonst]
  if (as.numeric(exp_monst) < 6 ) {
    exp_yrst <- harvest_yrst
    exp_yren <- harvest_yren
    obs_yrst <- harvest_yrst - 1
    obs_yren <- harvest_yren - 1
  } else {
    exp_yrst <- harvest_yrst - 1
    exp_yren <- harvest_yren - 1
    obs_yrst <- harvest_yrst - 1
    obs_yren <- harvest_yren - 1
  }

  exp_sdates <- paste0(as.character(seq(exp_yrst, exp_yren)), exp_monst, exp_dayst)
  obs_sdates <- paste0(as.character(seq(obs_yrst, obs_yren)), obs_monst, obs_dayst)

  #####################################
  #         Load data
  #####################################

  for (ivar in seq(1, length(variable))) {
    #Assign var name based on variable[[ivar]]$exp_var_name
                          print(paste0("Processing:       ",variable[[ivar]]$exp_var_name,
variable[[ivar]]$exp_var_unit,"Month: ",exp_monst))
    start_time <- Sys.time()

    if (EXP_COMPUTE) {
      exp <- list(list(
        name = expname,
                                                                    path          =
file.path(exp_basepath,'$YEAR$/$VAR_NAME$_$EXP_NAME$_ERA5_$START_DATE$_ITALY.nc'),
        #path = file.path(exp_basepath,'$YEAR$/$VAR_NAME$_$EXP_NAME$_$START_DATE$.nc'),
        nc_var_name = variable[[ivar]]$exp_var_name,
        var_min = variable[[ivar]]$exp_var_min
      ))
    }

    obs <- list(list(
      name = obsname,
                                                                path            =
file.path(paste0(obs_basepath,'/$YEAR$/ERA5-EU-',variable[[ivar]]$obs_var_name,'_',variabl
e[[ivar]]$obs_var_suffix,'.$YEAR$$MONTH$.nc')),
      nc_var_name = variable[[ivar]]$obs_var_name,
      var_min = variable[[ivar]]$obs_var_min
    ))

    if (EXP_COMPUTE) {
      c(expimport_dat, obs_dat) %<-% CST_Load(
```

```
        var = variable[[ivar]]$exp_fil_name,
        exp = exp,
        obs = obs,
        sdates = exp_sdates,
        storefreq = 'daily',
        output = 'lonlat',
        leadtimemax = maxlead,
        latmin = domain[[idomain]]$latmin,
        latmax = domain[[idomain]]$latmax,
        lonmin = domain[[idomain]]$lonmin,
        lonmax = domain[[idomain]]$lonmax,
        grid = gridres,
        nprocs = ncpus_read,
        path_glob_permissive = TRUE)

      attr(expimport_dat, 'class') <- 's2dv_cube'
      attr(obs_dat, 'class') <- 's2dv_cube'
      exp_nsdates <- dim(expimport_dat$data)['sdate']
      exp_ftime   <- dim(expimport_dat$data)['ftime']
      exp_nmem    <- dim(expimport_dat$data)['member']
      exp_nlon    <- dim(expimport_dat$data)['lon']
      exp_nlat    <- dim(expimport_dat$data)['lat']

      # ###################
      # # Remove NaNs
      # ###################
      if (variable[[ivar]]$exp_var_name=='tp') {
        expimport_dat$data[which(is.na(expimport_dat$data))] <- 0
      } else {
                                                                  expimport_dat$data        <-
remove.NANs.int(expimport_dat$data,direction='ftime',variable[[ivar]]$exp_var_name)
                                                                  expimport_dat$data        <-
remove.NANs.int(expimport_dat$data,direction='lonlat',variable[[ivar]]$exp_var_name)
      }

        #Note: both exp and obs have the same name as exp: note that the naminf of obs
wouldn't work for temperature
      assign(paste0('expimport_', variable[[ivar]]$exp_var_name, sep=""),expimport_dat)
    }

    obs_dat <- CST_Load(
      var = variable[[ivar]]$exp_fil_name,
      exp = NULL,
      obs = obs,
      sdates = obs_sdates,
      storefreq = 'daily',
      output = 'lonlat',
      leadtimemax = maxlead_ac,
      latmin = domain[[idomain]]$latmin,
      latmax = domain[[idomain]]$latmax,
      lonmin = domain[[idomain]]$lonmin,
      lonmax = domain[[idomain]]$lonmax,
      grid = gridres,     #Same grid as exp
      nprocs = ncpus_read,
      path_glob_permissive = TRUE)

    attr(obs_dat, 'class') <- 's2dv_cube'
    obs_nsdates <- dim(obs_dat$data)['sdate']
    obs_ftime   <- dim(obs_dat$data)['ftime']
    obs_nmem    <- dim(obs_dat$data)['member']
```

```
    obs_nlon    <- dim(obs_dat$data)['lon']
    obs_nlat    <- dim(obs_dat$data)['lat']

    if (variable[[ivar]]$exp_var_name=='tp') {
      obs_dat$data[which(is.na(obs_dat$data))] <- 0
    } else {
                                                                  obs_dat$data           <-
remove.NANs.int(obs_dat$data,direction='ftime',variable[[ivar]]$exp_var_name)
                                                                  obs_dat$data           <-
remove.NANs.int(obs_dat$data,direction='lonlat',variable[[ivar]]$exp_var_name)
    }

    assign(paste0('obs_', variable[[ivar]]$exp_var_name,sep=""),obs_dat)


    ############################
    # Prepare integrated forecast
    ############################

    #
    # Prepare integrated forecast by merging observation forecast and climatology
    # as input for agro.
    #

    exp_dat_ac_data <- rep(0,1*exp_nmem*obs_nsdates*obs_ftime*obs_nlat*obs_nlon)
    dim(exp_dat_ac_data)<-c(1,exp_nmem,obs_nsdates,obs_ftime,obs_nlat,obs_nlon)

    #Define number of day from start of index computation to start of forecast
    exp_nmonst <- as.numeric(exp_monst)
    obs_nmonst <- as.numeric(obs_monst)
    if ( exp_nmonst >= obs_nmonst) {
                start_of_forecast   <-   sum(days_in_month(seq(obs_nmonst,exp_nmonst)))   -
days_in_month(exp_nmonst) + 1
    } else {
      start_of_forecast <- sum(days_in_month(seq(obs_nmonst,12))) +
        sum(days_in_month(seq(1,exp_nmonst))) - days_in_month(exp_nmonst) + 1
    }

    #Define number of days from end of forecast to end of index computation
    end_of_forecast <- start_of_forecast + maxlead - 1
    if (end_of_forecast > obs_ftime ) {end_of_forecast <- obs_ftime}
    length_of_forecast <- length(seq(start_of_forecast:end_of_forecast))

    #Fill exp_dat_ac_data with the required data

    for ( imem in seq(1,exp_nmem)) {
                                    exp_dat_ac_data[1,imem,,1:start_of_forecast-1,,]      <-
obs_dat$data[1,1,,1:start_of_forecast-1,,]
                          exp_dat_ac_data[1,imem,,start_of_forecast:end_of_forecast,,]    <-
expimport_dat$data[1,imem,,1:length_of_forecast,,]
      if (end_of_forecast < obs_ftime) {
        #
        #Resample over obs_nsdates
        #
                               exp_dat_ac_data[1,imem,,(end_of_forecast+1):obs_ftime,,]   <-
obs_dat$data[1,1,sample(obs_nsdates),(end_of_forecast+1):obs_ftime,,]
      }
    }

    #
```

```
    # Redefine exp_dat with integrated data
    #
    exp_dat <- obs_dat   #Use the last available obs to re-init
    exp_dat$data <- exp_dat_ac_data
    #rm(exp_dat_ac_data)

    ##########################################
    #   Assign exp_dat to the relevant variable
    ##########################################
    assign(paste0('exp_', variable[[ivar]]$exp_var_name,sep=""),exp_dat)

  } #End load dates (loop over variables)


  print(paste0('Start date:',exp_monst,"Data Read completed after: ",Sys.time() - t0))

  t0c=Sys.time()

  ############################
  #   Clear memory space
  ############################

  #  Define lats lons and dates for later use
  lats  <- obs_dat$lat
  lons  <- obs_dat$lon
  dates <- obs_dat$Dates$start

  #rm(obs_dat)
  #rm(exp_dat)

  ################################
  #   Start computation of indices
  ################################


  #Init dataframes to store the indices
                                                  agro_obs_map                          <-
array(rep(0,ntypes,1,(obs_nsdates+1),1,obs_nlat,obs_nlon),dim=c(ntypes,1,(obs_nsdates+1),1
,obs_nlat,obs_nlon))
                                                  agro_exp_map                          <-
array(rep(0,ntypes,exp_nmem,(obs_nsdates+1),1,obs_nlat,obs_nlon),dim=c(ntypes,exp_nmem,(ob
s_nsdates+1),1,obs_nlat,obs_nlon))

  #Init data for monitoring
  ngrids <- obs_nlat*obs_nlon
  ncompleted <- 0
  pcompleted <- 0.
  percentile_completed <- 0
  percentile_steps <- 1.
  ilat <- 1
  display_monitor <- FALSE
  t1 <- Sys.time()
  t2 <- Sys.time()


  #! Create a cluster of ncpus_procs
                                                  #!                                (see
https://www.rdocumentation.org/packages/parallel/versions/3.6.2/topics/makeCluster)
  #! Reminder: stopCluster at the end of the process
  cl <- makeCluster(ncpus_proc)
```

```
  #! Each worker in the cluster is independent
  #! clustrCall makes sure that each worker links to the function
  clusterCall(cl, function() { source('./clisagri.r') })
  #! Export all required objects to the workspace of the workers
  clusterExport(cl, varlist = c("EXP_COMPUTE",
                                "lats", "lons",
                                "exp_nmem", "dates", "variable",
                                "exp_nsdates", "obs_ftime", "ntypes",
                                "assign.type", "obs_nsdates",
                                "exp_mn2t24", "exp_mx2t24", "exp_tp",
                                "obs_mn2t24", "obs_mx2t24", "obs_tp",
                                "agro_exp_map",
                                "agro_obs_map",
                                                    "ngrids","ncompleted",  "pcompleted",
"percentile_completed",
                                              "percentile_steps", "display_monitor", "t1", "t2",
"exp_monst", "obs_monst" ),
                envir = environment())
#Re-init time
  t1 <- Sys.time()
  t2 <- Sys.time()

  for ( ilat in seq(1:exp_nlat))   {

    clusterExport(cl, varlist = c("ilat"))

    #Monitor progress


    if (display_monitor) {
      t2 <- Sys.time()
      cat(paste0('\n Start date:',exp_monst,
                 ' - Progress:',sprintf('%2.0f',pcompleted),'%'),
            ' - ',difftime(t2,t1,unit='mins'),'min. for the last ',percentile_steps,'% -
Tot. elapsed:', difftime(Sys.time(),t0c,unit='mins'))
      t1 <- t2
      display_monitor <- FALSE
    }
    ncompleted  <- ncompleted+obs_nlon
    pcompleted <- 100.*ncompleted/ngrids
    if (pcompleted > percentile_steps*(1.+percentile_completed)) {
      display_monitor <- TRUE
      percentile_completed <- percentile_completed + 1
    }


    Lagroclim <- parLapply(cl, 1:obs_nlon, function(ilon) {

      glat <- lats[ilat]
      glon <- lons[ilon]

      agro_exp <- data.frame()
      if (EXP_COMPUTE) {
        for (imem in seq(1:exp_nmem)) {
          #Get DAY from the saved dates: assume Dates are the same for all variables
          meteo_exp <- data.frame(DAY=dates)
          for (ivar in seq(1:length(variable))) {
            exp_toproc <- get(paste0("exp_", variable[[ivar]]$exp_var_name))
                                                                    exp_toagro      <-
t(exp_toproc$data[1,imem,,,which.min(abs(glat-lats)),which.min(abs(glon-lons))])
```

```
            dim(exp_toagro) <- obs_nsdates*obs_ftime
            meteo_exp <- cbind.data.frame(meteo_exp,exp_toagro)
            names(meteo_exp)[ivar+1] <- variable[[ivar]]$agro_var_name
          }
          #Add average temperature
                                                                    meteo_exp       <-
cbind.data.frame(meteo_exp,TEMPERATURE_AVG=0.5*(meteo_exp$TEMPERATURE_MAX                +
meteo_exp$TEMPERATURE_MIN))
          #Set precipitation NA to zero as a correction to some bug in CST_Load
          meteo_exp$PRECIPITATION[which(is.na(meteo_exp$PRECIPITATION))] <- 0
          meteo_exp$PRECIPITATION   <- 1000.*meteo_exp$PRECIPITATION
          meteo_exp$TEMPERATURE_AVG <- meteo_exp$TEMPERATURE_AVG - 273.15
          meteo_exp$TEMPERATURE_MIN <- meteo_exp$TEMPERATURE_MIN - 273.15
          meteo_exp$TEMPERATURE_MAX <- meteo_exp$TEMPERATURE_MAX - 273.15
          agro <- clisagri(meteo_exp,glat,types=c(1:ntypes))

          agro <- cbind.data.frame(member=imem,agro)
          agro_exp <- rbind.data.frame(agro_exp,agro)
        }
      }


      #Get DAY from the saved dates: assume Dates are the same for all variables
      meteo_obs <- data.frame(DAY=dates)
      imem <- 1
      for (ivar in seq(1:length(variable))) {
        obs_toproc <- get(paste0("obs_", variable[[ivar]]$exp_var_name))
                                                                  obs_toagro        <-
t(obs_toproc$data[1,imem,,,which.min(abs(glat-lats)),which.min(abs(glon-lons))])
        dim(obs_toagro) <- obs_nsdates*obs_ftime
        meteo_obs <- cbind.data.frame(meteo_obs,obs_toagro)
        names(meteo_obs)[ivar+1] <- variable[[ivar]]$agro_var_name
      }
      #Add average temperature
                                                                  meteo_obs         <-
cbind.data.frame(meteo_obs,TEMPERATURE_AVG=0.5*(meteo_obs$TEMPERATURE_MAX                +
meteo_obs$TEMPERATURE_MIN))
      #Set precipitation NA to zero as a correction to some bug in CST_Load
      meteo_obs$PRECIPITATION[which(is.na(meteo_obs$PRECIPITATION))] <- 0
      meteo_obs$PRECIPITATION <- 1000.*meteo_obs$PRECIPITATION
      meteo_obs$TEMPERATURE_AVG <- meteo_obs$TEMPERATURE_AVG - 273.15
      meteo_obs$TEMPERATURE_MIN <- meteo_obs$TEMPERATURE_MIN - 273.15
      meteo_obs$TEMPERATURE_MAX <- meteo_obs$TEMPERATURE_MAX - 273.15

      agro_obs <- clisagri(meteo_obs,glat,types=c(1:ntypes))

      # #
      # #Format agro_obs and agro_exp back to CSTools
      # #
      #
        #    #Remove first year which is not computed for all indices depending on the
starting date (TOCHECK)
      #agro_exp <- dplyr::filter(agro_exp,year>min(year))
      #agro_obs <- dplyr::filter(agro_obs,year>min(year))
      #
      return(list(exp=dplyr::select(agro_exp,value,type),
                  obs=dplyr::select(agro_obs,value,type),
                  ilon=ilon,lat=ilat))
    }  )
```

```
    agro <- lapply(Lagroclim, function(agro) {
    agro_exp_data <- agro$exp$value
    agro_obs_data <- agro$obs$value
    lons=agro$ilon
    lats=agro$ilat

    #   #Now using 10 different agro indices)
    dim(agro_exp_data) <- c(ntypes,(obs_nsdates+1),exp_nmem)
    dim(agro_obs_data) <- c(ntypes,(obs_nsdates+1),1)

    # Reorder dimensions to match the CSTools format
    agro_obs_data <- aperm(agro_obs_data,c(1,3,2))
    agro_exp_data <- aperm(agro_exp_data,c(1,3,2))
    return(list(exp=agro_exp_data,obs=agro_obs_data,lons=lons,lats=lats))
    })

    lexp <- unlist(lapply(agro, function(a) {return(a$exp)}))
    lobs <- unlist(lapply(agro, function(a) {return(a$obs)}))

    dim(lexp) <- c(ntypes,exp_nmem,(obs_nsdates+1),1,obs_nlon)
    dim(lobs) <- c(ntypes,1,(obs_nsdates+1),1,obs_nlon)

    agro_exp_map[,,,,ilat,] <- lexp
    agro_obs_map[,,,,ilat,] <- lobs

  }

  stopCluster(cl)

  ##################################
  #   Save job image
  ##################################

  #2. Set dimension
  names(dim(agro_exp_map)) <- c("dataset","member","sdate","ftime","lat","lon")
  names(dim(agro_obs_map)) <- c("dataset","member","sdate","ftime","lat","lon")


  ##################################
  #   Start computation of skills
  ##################################

  # Remove NaNs
    agro_exp_map <- remove.NANs.int(agro_exp_map,direction='lonlat','agro_exp_map')
    agro_exp_map <- remove.NANs.int(agro_exp_map,direction='sdate','agro_exp_map')
    agro_obs_map <- remove.NANs.int(agro_obs_map,direction='lonlat','agro_obs_map')
    agro_obs_map <- remove.NANs.int(agro_obs_map,direction='sdate','agro_obs_map')

  #3.1 Select indicator type
  scorename <- 'FairRpss'

                                                  month_name                          <-
c('January','February','March','April','May','June','July','August','Septmeber','October',
'November','December')

  for (itype in seq(1,ntypes) ) {

    #Select one index type
    agroexp <- list(data=agro_exp_map[itype,,,,,,drop=FALSE],lat=lats,lon=lons)
    agroobs <- list(data=agro_obs_map[itype,,,,,,drop=FALSE],lat=lats,lon=lons)
```

```
#########################################################
#Compute probability of terciles for the selected index
#########################################################

    agroexp_prob <- array(numeric(),dim=c(3,(obs_nsdates+1),obs_nlat,obs_nlon))
    agroexp_terc <- array(numeric(),dim=c((obs_nsdates+1),obs_nlat,obs_nlon))
    agroobs_perc <- array(numeric(),dim=c((obs_nsdates+1),obs_nlat,obs_nlon))

    #Compute probabilities of terciles for each index
    for (i in c(1:obs_nlon)) {
      for (j in c(1:obs_nlat)) {
        agroexp_prob[,,j,i]<-ind2prob(input=agroexp$data[1,,,1,j,i])
        agroexp_terc[,j,i]<-prob2terc(input=agroexp_prob[,,j,i])
         agroobs_perc[,j,i]<-percentile(agroobs$data[1,1,,1,j,i]) #Per la funzione, partire
da ind2prob e calcolare il quantile
      }
    }

    agroexp <- aperm(agro_exp_map[itype,,,,,],c(4,3,2,1))
    agroobs <- aperm(agro_obs_map[itype,,,,,],c(3,2,1))
    #attr(agroexp, 'class') <- 's2dv_cube'
    #attr(agroobs, 'class') <- 's2dv_cube'

    #4. Send to skill computation
    ## The issue with NAN's must be fixed
    agroexp[which(is.na(agroexp))] <- 0
    agroobs[which(is.na(agroobs))] <- 0
    #agroexp_BS <- CST_Calibration(exp=agroexp,obs=agroobs)

    #Compute fRPSS in cross validation
    rpss <- array(NA,dim=dim(agroobs))
    for (y in 1:(obs_nsdates+1)) {
                                                                          r             <-
veriApply('FairRpss',fcst=agroexp[,,-y,],obs=agroobs[,,-y],ensdim=4,tdim=3,prob=c(1/3,2/3)
)
      rpss[,,y] <- r$skillscore


#r_bc<-veriApply('FairRpss',fcst=agroexp_bc[,,-y,],obs=agroobs[,,-y],ensdim=4,tdim=3,prob=
c(1/3,2/3))
      #rpss_bc[,,y]<-r_bc$skillscore
    }

#########################################################
#  Save output
#########################################################

    #Reshape arrays
    rpss <- aperm(rpss,c(3,2,1))
    rpss[which(!is.finite(rpss))] <- NA
    #Compute average RPSS
    rpss_mean <- apply(rpss,2:3,mean,na.rm=TRUE)

    #rpss_bc<-aperm(rpss_bc,c(3,2,1))
    agroexp <-aperm(agroexp,c(4,3,2,1))
    agroobs <-aperm(agroobs,c(3,2,1))

#######################################
# Define all dimensions
```

```
########################################

    indexname <- index_attr[[itype]]$short_name
    varname <- index_attr[[itype]]$short_name

    make.attr <- function(attr) {
      makeattr <- list(attr)
      names(makeattr) <- attr$short_name
      return(makeattr)
    }

    #Define attributes of dimensions
    attr(lons, 'variables') <- list(lon = list(units = 'degrees_east'))
    names(dim(lons)) <- 'lon'

    attr(lats, 'variables') <- list(lat = list(units = 'degrees_north'))
    names(dim(lats)) <- 'lat'

    bin <- c(1/3,2/3,1); dim(bin)<- length(bin)
    attr(bin, 'variables') <- list(bin = list(units = '(tercile)'))
    names(dim(bin)) <- 'bin'

    ensemble_member<-seq(1,exp_nmem,by=1)
    dim(ensemble_member) <- length(ensemble_member)
    attr(ensemble_member, 'variables') <- list(ensemble_member=list(units='number'))
    names(dim(ensemble_member)) <- 'ensemble_member'


########################################
# Save a different file for each year
########################################


    for (i in 1:length(harvest_years)) {

      harvest_year <- harvest_years[i]
      obs_year <- harvest_year
      if (as.numeric(exp_monst) < 6 ) {
        exp_year <- harvest_year
      } else {
        exp_year <- harvest_year - 1
      }
      dim(exp_year) <- 1
      attr(exp_year, 'variables') <- list(sdate=list(units = 'year'))
      names(dim(exp_year)) <- 'year'

      #Create dir for index file: separate folder for each index
      dirname <- paste0(out_basepath_agro,'/',harvest_year,'/')
      if ( !dir.exists(dirname)) {dir.create(dirname)}

      ################################
      # 1. Save ensemble agro indices
      ################################
      agro_toncdf <- agroexp[,i,,]
      attr(agro_toncdf, 'variables') <- make.attr(index_attr[[itype]])
      names(dim(agro_toncdf))<-c('member','lat','lon')
      parname <- varname

fname<-paste0(dirname,expname_out,'_',domain[[idomain]]$fname,'_c3s_',varname,'_',biascorr
ection_tag,'_stdt_',exp_monst,'_',exp_year,'_',parname,'.nc')
```

```
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats,ensemble_member),fname)

      ##############################
      # 2. Save ERA5 agro indices
      ##############################
      # Check correct year
      agro_toncdf <- agroobs[i,,]
      attr(agro_toncdf, 'variables') <- make.attr(index_attr[[itype]])
      names(dim(agro_toncdf))<-c('lat','lon')
      parname <- varname

fname<-paste0(dirname,obsname_out,'_',domain[[idomain]]$fname,'_',varname,'_',obs_year,'.n
c')
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats),fname)

      ###################################
      # 3. Save probability of terciles
      ###################################
      agro_toncdf <- agroexp_prob[,i,,]
      attr(agro_toncdf, 'variables') <- make.attr(index_attr[[itype]])
      names(dim(agro_toncdf))<-c('bin','lat','lon')
      parname <- 'PROBABILITY'

fname<-paste0(dirname,expname_out,'_',domain[[idomain]]$fname,'_c3s_',varname,'_',biascorr
ection_tag,'_stdt_',exp_monst,'_',exp_year,'_',parname,'.nc')
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats,bin),fname)

      ##############################
      # 4. Save most likely tercile
      ##############################
      agro_toncdf <- agroexp_terc[i,,]
                                          attr(agro_toncdf,       'variables')       <-
list(terc=list(short_name='Tercile',long_name='Most likely tercile',units = 'index'))
      names(dim(agro_toncdf))<-c('lat','lon')
      parname <- 'TERCILE'

fname<-paste0(dirname,expname_out,'_',domain[[idomain]]$fname,'_c3s_',varname,'_',biascorr
ection_tag,'_stdt_',exp_monst,'_',exp_year,'_',parname,'.nc')
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats),fname)

      ##########################
      # 5. Save ERA5 percentile
      ##########################
      agro_toncdf <- agroobs_perc[i,,]
                                          attr(agro_toncdf,       'variables')       <-
list(perc=list(short_name='Percentile',long_name='Observation     percentile',units     =
'index'))
      names(dim(agro_toncdf))<-c('lat','lon')
      parname <- 'PERCENTILE'

fname<-paste0(dirname,obsname_out,'_',domain[[idomain]]$fname,'_',varname,'_',obs_year,'_'
,parname,'.nc')
```

```
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats),fname)

      ######################
      # 6. Save yearly RPSS
      ######################
      agro_toncdf <- rpss[i,,]
       attr(agro_toncdf, 'variables') <- list(rpss=list(short_name='RPSS',long_name='Ranked
Probability Skill Score',units = 'index'))
      names(dim(agro_toncdf))<-c('lat','lon')
      parname <- 'RPSS'

fname<-paste0(dirname,expname_out,'_',domain[[idomain]]$fname,'_c3s_',varname,'_',biascorr
ection_tag,'_stdt_',exp_monst,'_',exp_year,'_',parname,'.nc')
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats),fname)

      ###################
      # 7. Save mean RPSS
      ###################
      agro_toncdf <- rpss_mean
       attr(agro_toncdf, 'variables') <- list(rpss=list(short_name='RPSS',long_name='Ranked
Probability Skill Score',units = 'index'))
      names(dim(agro_toncdf))<-c('lat','lon')
      parname <- 'RPSSmean'

fname<-paste0(dirname,expname_out,'_',domain[[idomain]]$fname,'_c3s_',varname,'_',biascorr
ection_tag,'_stdt_',exp_monst,'_',parname,'.nc')
      print(paste0('Writing file: ',fname))
      #
      ArrayToNc(list(agro_toncdf,lons,lats),fname)

    } #End cycle

  } #End cycle over index type


} #End cycle over starting dates

print(Sys.time() - t0)
```

END OF DOCUMENT

ANNEX C.

MEDGOLD_RETRIEVE-UPDATE_CRON.sh

```
#!/usr/bin/sh
######################################################################
#
# Control the periodic access to
WDIR="/home/CRON/"
PYSCRIPTS="MEDGOLD_RETRIEVE-SEASONAL-UPDATE.py MEDGOLD_RETRIEVE-ERA5-UPDATE_cdsapi.py"

cd ${WDIR}

#Perform check befor resubmitting

for PYSCRIPT in ${PYSCRIPTS};
do

 DATASET=$(echo $PYSCRIPT | cut -f 2 -d -)

 PYRUN=`ps a -u sandro | grep ${PYSCRIPT} | grep python`
 PYID=`ps a -u sandro | grep ${PYSCRIPT} | grep python | cut -d ' ' -f 1`
 PYLOG=RETRIEVE-UPDATE-${DATASET}.log
 DFLOG=`diff ./${PYLOG} ./${PYLOG}_last`



 if [ "${PYRUN}" == '' ]; then
  mail -s "START update of ${DATASET} archive" sandro.calmanti@enea.it < ${PYLOG}
  python -u ${PYSCRIPT} > ${PYLOG} 2>&1  &
 else
  if [ "${DFLOG}" == '' ]; then
   mail -s "IDLE update of ${DATASET} archive since last check - kill process ${PYID} and
resubmit" sandro.calmanti@enea.it < ${PYLOG}
   kill -9 ${PYID}
   python -u ${PYSCRIPT} > ${PYLOG} 2>&1  &
  else
   mail -s "OK seasonal ${DATASET} update is running - check logfile"
sandro.calmanti@enea.it < ${PYLOG}
  fi
 fi

#Backup last logfile to check idle processes
cp ${PYLOG} ${PYLOG}_last

done
```

ANNEX D.

GRIB2NCDF_MEDGOLD_CRON.sh

```
#!/usr/bin/sh
#`
WDIR="/home/CRON"
CDOSCRIPT=grib2netcdf_medgold.sh

cd ${WDIR}

#Perform check befor resubmitting
MASTERLOG=GRIB2NCDF_MEDGOLD_MASTER.log
CDOLOG=grib2netcdf_medgold_ECMWF.log
CDORUN=`ps a -u sandro | grep ${CDOSCRIPT}| grep "/usr/bin/sh"`
CDOID=`ps a -u sandro | grep ${CDOSCRIPT} | grep "/usr/bin/sh" | cut -d ' ' -f 1`

echo Start GRIB2NETCDF master

if [ "${CDORUN}" == '' ]; then
  echo "GRIB2NCDF not running - SUBMITTING"
  mail -s "GRIB2NCDF not running - SUBMITTING" sandro.calmanti@enea.it  < /dev/null
  ./${CDOSCRIPT} > ${MASTERLOG} 2>&1
else
  if [ "${DFLOG}" == '' ]; then
   echo "IDLE GRIB2NCDF of multimodel data since last check - kill process ${CDOID} and
resubmit"
   DFLOG=`diff ./${MASTERLOG} ./${MASTERLOG}_last`
   mail -s "IDLE GRIB2NCDF of multimodel data since last check - kill process ${CDOID} and
resubmit" sandro.calmanti@enea.it < ${CDOLOG}
   kill -9 ${CDOID}
   ./${CDOSCRIPT} > ${MASTERLOG} 2>&1
  else
   echo "OK GRIB2NCDF of seasonal multimodel retrieval is running - check logfile"
   mail -s "OK GRIB2NCDF of seasonal multimodel retrieval is running - check logfile"
sandro.calmanti@enea.it < ${CDOLOG}
  fi
fi

#Backup last logfile to check idle processes
cp ${MASTERLOG} ${MASTERLOG}_last
```

ANNEX E.

BIASCORRECTION_ON.sh

```
#!/usr/bin/sh
#
#####
# VERSIONS
# 1.0 Scan local archive
# 1.1 reformat data for CSTools
set +e
now=`date +%Y%m%d-%H%M%S`
YEAR_TODAY=`date +%Y`
MONTH_TODAY=`date +%m`
START_TIME=$SECONDS

#Set working directory
WDIR="/home/CRON"
RSCRIPT=${WDIR}/biascorrection.R
RLOG=${WDIR}/biascorrection_${now}.log

#Init log files
STDOUT=${WDIR}/biascorrection_master.log
echo BIASCORRECTION log of ${now} > ${STDOUT}
echo YEAR_TODAY: ${YEAR_TODAY} >> ${STDOUT}

ARCHIVELOCAL='/fas_specs/a/sandro/DATA/SEASONAL'

# The reference observational data is bassed to the biascorrection routin as a parameter
OBSNAME="ERA5"

# The subdomain over which bias correction is performed, is passed to the biascorrection
routine as a parameter
DOMAIN="ITALY"
DOMAINS="ITALY MAROC"

MODELS=`ls ${ARCHIVELOCAL}`
MODELS="ECMWF"

VARNAMES="tmax2m tmin2m ssrd t2m rh wss tp"
MONTHS="1 2 3 4 5 6 7 8 9 10 11 12"

#############################################################
#
#   START PROCESSING
#
#############################################################

cd ${WDIR}

#
#Cycle over DOMAIN
#
for DOMAIN in ${DOMAINS}
do
#
#Cycle over MODELS
#
```

```
for MODEL in ${MODELS}
do
  ARCHIVEGRIB=${ARCHIVELOCAL}/${MODEL}/GRIB
  ARCHIVENCDF=${ARCHIVELOCAL}/${MODEL}/NCDF
  ARCHIVEBC=${ARCHIVELOCAL}/${MODEL}/BC/NCDF
  YEARS=`ls ${ARCHIVENCDF}`
#
#Cycle over over all possible starting dates
#
  for MONTH in ${MONTHS}
  do
#This is because an initial non-automatized retrieval was done using small letter
#Can be removed when the folder ECMF is removed
    if [ ${MODEL} == "ECMF" ];         then MODELNAME=ecmwf; fi
    if [ ${MODEL} == "ECMWF" ];        then MODELNAME=ecmwf; fi
    if [ ${MODEL} == "DWD" ];          then MODELNAME=dwd; fi
    if [ ${MODEL} == "METEOFRANCE" ];  then MODELNAME=meteo_france; fi
    if [ ${MODEL} == "CMCC" ];         then MODELNAME=cmcc; fi
#
#Cycle over all variables required for MEDGOLD
#
    for VARNAME in ${VARNAMES}
    do
#
# Check matching NCDF and BC
#
      echo >> ${STDOUT}
      echo "CHECKING  -  Modelname: ${MODELNAME}  -  Var name: ${VARNAME} - Start date:
`printf "%02d" "$MONTH"`" >> ${STDOUT}
      BC_COMPLETED=0
      LAST_NCDF=-9999
      for YEAR in ${YEARS}
      do
        WDIRGRIB=${ARCHIVEGRIB}/${YEAR}
        WDIRNCDF=${ARCHIVENCDF}/${YEAR}
        WDIRBC=${ARCHIVEBC}/${YEAR}
        FILEIN=${WDIRNCDF}/${VARNAME}_${MODELNAME}_${YEAR}`printf "%02d" "$MONTH"`01.nc
        FILEOUT=${WDIRBC}/${VARNAME}_${MODELNAME}_${OBSNAME}_${YEAR}`printf "%02d"
"$MONTH"`01_${DOMAIN}.nc
      #     echo DEBUG
      #     echo DEBUG ${YEAR}
        if [ -f ${FILEIN} ]; then
           LAST_NCDF=${YEAR}
      #     echo DEBUG VARNAME: ${VARNAME}  -  LAST_NCDF: ${LAST_NCDF}  -  BC_COMPLETED:
${BC_COMPLETED}  - FILE: ${VARNAME}_${MODELNAME}_${OBSNAME}_${YEAR}`printf "%02d"
"$MONTH"`01_${DOMAIN}.nc
           if [ ! -f ${FILEOUT} ]; then
  #IF The ONLY missing year is the current year con't touch BC_COMPLETED
             if [ ${YEAR} -eq ${YEAR_TODAY} -a ${MONTH} -gt ${MONTH_TODAY} -a
${BC_COMPLETED} -eq 0 ]; then
               echo "Bias corrected ${VARNAME} - starting date ${MONTH} available before
year ${YEAR_TODAY}" >> ${STDOUT}
             else
      #        echo DEBUG ${FILEOUT} not found - setting BC_COMPLETED=1
               BC_COMPLETED=1
             fi
           fi
        else
           echo "Input for ${VARNAME} - starting date ${MONTH} available available until
${LAST_NCDF}" >> ${STDOUT}
```

```
        fi
      done

      if [ ${BC_COMPLETED} -eq 1 ]; then
         echo "!!! Launching bias correction required for ${VARNAME} - starting date
${MONTH} - last year available ${LAST_NCDF}" >> ${STDOUT}
         echo "Rscript ${RSCRIPT} AREA ${VARNAME} ${LAST_NCDF} `printf "%02d" "$MONTH"`
${DOMAIN} ${MODEL} ${OBSNAME}"
         #Send job to a dedicated cpu set
         cpuset.sh waves
         #Launch bias correction
         Rscript ${RSCRIPT} AREA ${VARNAME} ${LAST_NCDF} `printf "%02d" "$MONTH"`
${DOMAIN} ${MODEL} ${OBSNAME} > ${RLOG} 2>&1
      else
         echo "Bias correction COMPLETE  for ${VARNAME} - starting date ${MONTH} - last
year available ${LAST_NCDF}" >> ${STDOUT}
      fi

    done #Cycle over VARNAMES
  done #Cycle over MONTHS
done #Cycle over MODELS
done #Cycle over DOMAINS
exit
```

ANNEX F.

CLISAGRI.sh

```
#!/usr/bin/sh
#
#####
# VERSIONS
# 1.0 Scan local archive
# 1.1 reformat data for CSTools
set +e
now=`date +%Y%m%d-%H%M%S`
YEAR_TODAY=`date +%Y`
MONTH_TODAY=`date +%m`
START_TIME=$SECONDS

#Set working directory
WDIR="/fas_d/b/sandro/R/WORK/MEDGOLD"
RSCRIPT=${WDIR}/MEDGOLD_clisagri_seasonal_v2.3.R
RLOG=${WDIR}/clisagri_${now}.log

############################################################
#
#   START PROCESSING
#
############################################################

cd ${WDIR}

cpuset.sh waves
#Launch bias correction
Rscript ${RSCRIPT} > ${RLOG} 2>&1

exit
```