Package: CAWET (via r-universe)

May 29, 2026

Title Calculation of Agricultural Water Needs and Evapotranspiration

Version 1.1.0

Description An R tool for calculating theoretical irrigation water consumption over a territory. CAWET extracts, models and analyzes agro-climatic data to produce maps, graphics and comparative outputs across different scenarios. The workflow includes data extraction from online databases (via RADIS), modeling (primarily using CropWat), and visualization of results through graphics and maps.

License AGPL (>= 3)

URL https://inrae.r-universe.dev/CAWET, https://umr-g-eau.pages-forge.inrae.fr/cawet

BugReports https://forge.inrae.fr/umr-g-eau/cawet/-/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, sf, ggplot2, stringr, data.table, scales, tidyr, lubridate, cli, stats, utils, openxlsx, glue, RADIS, CropWat, slider, tools, terra

Additional_repositories https://inrae.r-universe.dev

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs libabsl-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev make libharfbuzz-dev libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

Repository https://inrae.r-universe.dev

Date/Publication 2026-03-30 20:51:52 UTC

RemoteUrl https://forge.inrae.fr/umr-g-eau/cawet

RemoteRef main

RemoteSha d2e88c0de36099dc5f74aa38856d1682c927e938

Contents

`creer_parcelles_autour_point_centre`
Create plots around a center (for "RADIS noir" option)

Description

This function creates plots around a point, with a given surface area.

Usage

```
creer_parcelles_autour_point_centre(lon, lat, crops, areas, col_crop, col_area)
```

Arguments

lon	numeric longitude in decimal degrees.
lat	numeric latitude in decimal degrees.
crops	character vector crops to create.
areas	numeric vector surface areas of the crops listed in crops in m2 (in the same order as crops).
col_crop	character standardized name of the column with the crop names.
col_area	character standardized name of the column with the surface area.

Value

A sf::sf object containing a plot for each crop, with the corresponding surface area.

---

creer_parcelles_autour_point_repgraph

Create plots around a center, and change the center for each crop (for "RADIS noir" option)

Description

This function creates plots around a point, with a given surface area.

Usage

```
creer_parcelles_autour_point_repgraph(
  lon,
  lat,
  crops,
  areas,
  col_crop,
  col_area
)
```

Arguments

lon	numeric longitude in decimal degrees.
lat	numeric latitude in decimal degrees.
crops	character vector crops to create.
areas	numeric vector surface areas of the crops listed in crops in m2 (in the same order as crops).
col_crop	character standardized name of the column with the crop names.
col_area	character standardized name of the column with the surface area.

Value

A sf::sf object containing a plot for each crop, with the corresponding surface area.

FpCAWET *Construct Full Path for CAWET*

Description

This function constructs a full file path for CAWET by checking if the provided path is already an absolute path. If it is not, it combines it with the specified working directory.

Usage

```
FpCAWET(Working_path, path)
```

Arguments

Working_path	A string representing the working directory path.
path	A string representing the file path to be checked or combined.

Value

A string representing the full file path.

get_Run_Config *Get or create a CAWET run configuration*

Description

Get or create a CAWET run configuration

Usage

```
get_Run_Config(
  path = NULL,
  Working_path = dirname(dirname(path)),
  Scenario = NULL
)

get_Path_Run(path = choose.dir(), Working_path, new = FALSE)
```

Arguments

path	Character string specifying the path to an existing run directory. If NA the function retrieves the most recent run directory. By default, a folder selection dialog will be opened.
Working_path	Character string specifying the working directory.

Scenario	Optional data frame containing scenario data. If provided, it will be saved to the run directory. If NULL, the function will attempt to read the scenario data from the run directory.
new	Logical indicating whether to create a new run directory. If TRUE, a new run directory will be created. If FALSE, the most recent existing run directory will be retrieved.

Details

Behavior depending on the value of Scenario:

- If Scenario is provided (non-NULL):
  - get_Run_Config calls get_Path_Run(..., new = TRUE) which creates a new run directory at file.path(Working_path, "Runs", paste0("Run_", <timestamp>)).
  - The provided Scenario data frame is written into Scenarii.csv inside the new run directory.
  - A Charged_inputs directory is created inside the run directory and its path is returned as Path_Run_Chargementdata.
- If Scenario is NULL:
  - get_Run_Config calls get_Path_Run(..., new = FALSE) which:
    - If path is NA: finds the most recent existing run directory under file.path(Working_path, "Runs") (the last Run_ entry).
    - If path is provided: validates that the directory exists and returns it.
  - The function attempts to read Scenarii.csv from the chosen run directory using read_ordonnanceur() to populate Scenario.
  - The Charged_inputs directory is created if missing and its path returned.

Note: the default path uses choose.dir(); when creating a new run the run timestamp ensures unique run folder names.

Value

get_Run_Config returns a list with the following elements:

- Working_path: The working directory used for the run.
- Path_Run: The path to the created or selected run directory.
- Scenario: A data frame containing the scenario information used for the run.
- Path_Run_Chargementdata: The path to the Charged_inputs directory inside the run.

get_Path_Run returns a character string with the path to the run directory.

Functions

- get_Run_Config(): This function retrieves or creates a run configuration for CAWET, including the run path, scenario data, and path to charged input data.
- get_Path_Run(): Get or create a run path

get_soil_depth *Get soil depth*

Description

This function retrieves soil depth information for the given parcels with a specific grid (maille_y). It performs a spatial intersection between the parcels and the soil depth data obtained from the specified source, then calculates the total surface area for each soil depth category within each parcel.

Usage

```
get_soil_depth(sf, source = "BDGSF")
```

Arguments

sf	An sf object containing parcels for which soil depth information is to be retrieved. The sf object should have a column named "id_parcel" that uniquely identifies each parcel and a column named "surface_m2" representing the surface area of each parcel.
source	A character string specifying the source of soil depth data. Default is "BDGSF".

Value

A data frame with parcel identifiers and their corresponding soil depth information.

get_soil_textures *Get soil textures*

Description

This function retrieves soil texture information for the given parcels with a specific grid (maille_y). It performs a spatial intersection between the parcels and the soil texture data obtained from the specified source, then calculates the mean values of clay, silt, and sand for each parcel.

Usage

```
get_soil_textures(parcelles_avec_maille_y, Depth)
```

Arguments

parcelles_avec_maille_y	An sf object containing parcels with a specific grid (maille_y) for which soil texture information is to be retrieved. The sf object should have a column named "id_parcel" that uniquely identifies each parcel.
Depth	A data frame containing soil depth information for the parcels, with a column named "id_parcel" that uniquely identifies each parcel and a column named "soil_depth" representing the soil depth category.

Value

A data frame with parcel identifiers and their corresponding mean values of clay, silt, and sand.

---

ggsave *Suppress messages from ggplot2::ggsave*

Description

This function is a wrapper around `ggplot2::ggsave` that suppresses messages. It can be used to avoid cluttering the console with messages when saving plots.

Usage

```
ggsave(...)
```

Arguments

`...`	Arguments passed to `ggplot2::ggsave`.

Value

The return value of `ggplot2::ggsave`, invisibly.

---

grille_safran *SAFRAN climate grid used by CAWET*

Description

`grille_safran` is an `sf` polygon grid derived from Météo-France SAFRAN mesh coordinates. It is used to associate each territory/plot with a SAFRAN climate cell for weather data retrieval.

Usage

```
grille_safran
```

Format

An `sf` object with 8,969 rows and 4 variables:

id_maille Character. SAFRAN grid cell identifier.

lambx Integer. SAFRAN X coordinate in Lambert-93 (hectometers).

lamby Integer. SAFRAN Y coordinate in Lambert-93 (hectometers).

geometry POLYGON geometry (CRS: EPSG 2154, RGF93 / Lambert-93).

Source

Météo-France public dataset "coordonnees-des-mailles_339.csv": https://donneespubliques.meteofrance.fr/client/document/coordonnees-des-mailles_339.csv

Examples

```
data(grille_safran)
sf::st_crs(grille_safran)
```

`prepare_territory_for_existing_plots` *Prepare the shapefile with the polygons for which data will be retrieved.*

Description

This function creates a sf object with polygons associated to identifiers, area, and standardized column names. Polygons can be either agricultural plots associated to a crop and a bigger polygon, or directly the big polygons (in which case, agricultural plots will be retrieved for each year from the RPG database).

Usage

```
prepare_territory_for_existing_plots(
  Working_path,
  Scenario,
  COL = list(poly = "id_poly", plot = "id_parcel", crop = "code_cultu", area =
    "surface_m2"),
  col_map,
  external_plots
)
```

Arguments

`Working_path`	character working directory.
`Scenario`	vector contains information regarding the scenario of interest.
`COL`	list standardized column names for polygon, plot, crop, and area identifiers.
`col_map`	data.frame correspondence table between standardized column names (`col_res`) and original data column names (`col_user`).
`external_plots`	logical If TRUE, agricultural plots are provided by the user.

Value

A sf::sf object containing polygons for which data must be retrieved.

`prepare_territory_for_non_existing_plots` *Prepare the shapefile with the polygons for which data will be retrieved in the case agricultural plots are fictional.*

Description

This function creates a sf object with polygons associated to identifiers, area, and standardized column names. Polygons are agricultural plots associated to a crop and a bigger polygon

Usage

```
prepare_territory_for_non_existing_plots(
  Working_path,
  shp_link,
  COL = list(poly = "id_poly", plot = "id_parcel", crop = "code_cultu", area =
    "surface_m2"),
  col_map
)
```

Arguments

`Working_path`	character working directory.
`shp_link`	character name of the original data file.
`COL`	list standardized column names for polygon, plot, crop, and area identifiers.
`col_map`	data.frame correspondence table between standardized column names (`col_res`) and original data column names (`col_user`).

Value

A sf::sf object containing a polygons for which data must be retrieved.

`process_soil_characteristic` *Computes the value of a soil characteristic for all relevant soil horizons, for each row of a data.frame.*

Description

This function returns a numeric vector with a soil characteristic averaged or summed for all soil horizons, depending on the soil depth.

Usage

```
process_soil_characteristic(df, variable, FUN)
```

Arguments

df	data.frame result of RADIS::get_soil_texture or RADIS::get_soil_awc joined with the result of RADIS::get_soil_depth (depth in [m]).
variable	character one of the column returned by get_soil_texture
FUN	function the function to use to aggregate the soil characteristic (mean or sum)

Value

A numeric vector containing averaged texture characteristic.

---

read_CAWET_data *Read CAWET Data File*

Description

Read CAWET Data File

Usage

```
read_CAWET_data(Working_path, file)

read_ordonnanceur(chemin_ordonnanceur)
```

Arguments

Working_path	The working directory path where the CAWET data file is located.
file	The name of the CAWET data file to read.
chemin_ordonnanceur	The full path to the Ordonnanceur file.

Value

A data frame containing the contents of the CAWET data file. `read_ordonnanceur` reads an Ordonnanceur file (Excel or CSV) and returns its contents as a data frame with all columns converted to character.

Functions

- `read_CAWET_data()`: Reads a CAWET data file (CSV or Excel) from the specified working path.

  This function reads a CAWET data file (CSV or Excel) from the specified working path.
- `read_ordonnanceur()`: Read Ordonnanceur File (Excel or CSV)

run *Run the complete CAWET workflow*

Description

This function orchestrates the entire CAWET workflow, including verification of the scheduler, extraction of relevant information, modeling/simulation of crops, and generation of result illustrations.

Usage

```
run(
  chemin_ordonnanceur = file.choose(),
  Scenario = read_ordonnanceur(chemin_ordonnanceur),
  Working_path = dirname(dirname(chemin_ordonnanceur)),
  cfgRun = get_Run_Config(Working_path = Working_path, Scenario = Scenario)
)
```

Arguments

chemin_ordonnanceur	(Optional) Character string specifying the path to the scheduler (ordonnanceur) file. Uses an interactive file chooser (file.choose()) by default.
Scenario	(Optional) data.frame produced by read_ordonnanceur(). If provided it is reused and passed to get_Run_Config() to avoid re-reading the ordonnanceur. By default the function calls read_ordonnanceur().
Working_path	(Optional) Character string specifying the working directory path.
cfgRun	(Optional) Running configuration created by get_Run_Config().

Value

Returns invisibly the running configuration (See get_Run_Config()).

Run_Cropwat_CAWET *Run Cropwat CAWET simulation for a given crop, irrigation method and sequence*

Description

This function runs the Cropwat CAWET simulation for a specified crop, irrigation method, and sequence. It takes into account various parameters such as irrigation settings, crop characteristics, meteorological data, and simulation settings to generate detailed output results.

Usage

```
Run_Cropwat_CAWET(
  Irrigation_param,
  mod_irr,
  Sequence,
  carac_date_irr,
  Link_RPG_mod,
  Crop,
  Crops_param,
  Meteo_maille,
  liste_not_parametredcrops
)
```

Arguments

Irrigation_param	A data frame containing irrigation parameters.
mod_irr	A string representing the irrigation method.
Sequence	A data frame containing sequence information.
carac_date_irr	A logical value indicating whether to use specific irrigation dates
Link_RPG_mod	A data frame linking RPG crop names to Cropwat crop names.
Crop	A string representing the crop name.
Crops_param	A data frame containing crop parameters.
Meteo_maille	A data frame containing meteorological data for the grid cell
liste_not_parametredcrops	A vector of strings representing crops that are not parameterized.

Value

None. The function saves the detailed output results to a CSV file.

---

run_script1 *Preparation of input data for CAWET model using RADIS data*

Description

This script loads and prepares input data for the CAWET model using RADIS data. It reads scenario information from an Excel file, processes spatial data for plots, and prepares meteorological and soil data for each scenario and year. It saves the prepared data in a structured directory for further analysis.

Usage

```
run_script1(cfgRun)
```

Arguments

cfgRun	(Optional) Running configuration created by get_Run_Config().

Details

Input files and data sources

- chemin_ordonnanceur: scheduler file read with read_ordonnanceur() (scenario-level configuration used throughout the workflow).
- **Important convention**: every Scenario$XXX used in this function corresponds to the values found in the XXX column of the scheduler file (one value per scenario row).
- Working_path/Model_param/odr_to_rpg.csv: mapping table used to harmonize crop codes between ODR and RPG nomenclatures.
- Météo-France grid coordinates file (downloaded from https://donneespubliques.meteofrance.fr/client/docum used to build the climate network mesh.
- Scenario$Wanted_output: scenario-specific file read with read_CAWET_data() to control optional map/graph exports.
- Plot/territory inputs (depending on Scenario$Plots_RADIS_RPG):
  - Existing plots ("Yes"/"External_plot_map") through prepare_territory_for_existing_plots().
  - Synthetic territory ("Non_existant_plots") through prepare_territory_for_non_existing_plots().
- Optional soil input files:
  - Scenario$Soil_information_link when Scenario$Sol_fixed_RADIS == "No".
  - Optional supplementary AWC raster directory from Scenario$Sol_carte_sup_RU_link_to_doss (files matching RU_*.tif).
- Optional climate inputs:
  - Scenario$Climat_link_fixe for forced/unique climate mode.
  - Scenario$Climat_link_Drias parameter table for DRIAS extraction.

Main processing steps

1. Read scheduler, initialize run directories with get_Run_Config(), and validate unique scenario names.
2. Build the SAFRAN grid geometry and scenario-specific working folders.
3. Load and save wanted outputs table (Wanted_outputs.csv).
4. Build plot geometries per scenario and year:
   - Download RPG polygons (RADIS::get_rpg_data()) when requested.
   - Harmonize crop codes using odr_to_rpg.csv.
   - Keep/rename standard columns (id_poly, id_parcel, code_cultu, surface_m2).
5. Spatially link plots to climate meshes and sub-meshes:
   - Create network/subnetwork intersections.
   - Export link table between plots and meshes.
   - Optionally generate cartographic diagnostic figures.
6. Enrich plots with soil attributes (depth, AWC, texture), depending on scenario mode:

- External fixed soil table.
- RADIS BDGSF / Info&Sols retrieval.
- Optional AWC replacement from supplementary raster maps.

7. Aggregate yearly plot information by crop/polygon/mesh/sub-mesh and write spatial outputs.
8. Retrieve climate time series (Unique/Forced, SAFRAN, or DRIAS) and write per-scenario/per-mesh files.
9. Build and save linear yearly tables (surface, depth, AWC, clay, silt, sand) per crop, polygon and sub-mesh.
10. For synthetic territories, export additional illustration and map files.

Outputs created

For each scenario, files are written under `cfgRun$Path_Run_Chargementdata/<index>_<Scenario>/`:

- `Wanted_outputs.csv`
- `Networkplots_<Year>.shp`
- `Link_plots_mailles_<Year>.csv`
- `Plots_all_RADIS_information_<Year>.shp`
- `Crops_all_RADIS_information.shp` (all years combined)
- `Meteo/Meteo.csv` (unique/forced climate mode) or `Meteo/Meteo_maille<id>.csv` (network mode)
- Optional maps in `Meteo/`:
  - `Plots_in_climaticalnetwork_<Year>.png`
  - `Plots_in_climaticalnetwork_<Year>_withsubnetwork.png`
  - `Illustration_plots_maille.png` (synthetic territory mode)
- `Formodification/Territoire_repartition_cultures_<Year>.csv`
- `Formodification/Soil_average_<Year>.csv`
- Linear tables:
  - `Surf_table_<Scenario>.csv`
  - `Prof_table_<Scenario>.csv`
  - `AWC_table_<Scenario>.csv`
  - `Arg_mean_table_<Scenario>.csv`
  - `Lim_mean_table_<Scenario>.csv`
  - `Sab_mean_table_<Scenario>.csv`
- Additional file for synthetic territory:
  - `Plots_location_notcenteredforillustration.shp`

Value

A list containing:

- `Run_path`: The path to the run directory.
- `Scenario`: A data frame containing scenario data.
- `Path_Run_Chargementdata`: The path to the charged input data directory.

run_script2 *Run Script 2 - Simulations for crop water management modeling*

Description

This function runs the simulation (script 2) for crop water management modeling. It performs the following tasks:

- Sets up a portable R library for package management.
- Loads required R packages.
- Reads scenario configurations and input data. Processes simulations for different crops, polygons, and irrigation methods.
- Saves detailed output results and generates verification graphs.
- Handles missing crop parameter warnings.

Usage

```
run_script2(cfgRun = get_Run_Config(Working_path = Working_path))
```

Arguments

cfgRun	(Optional) Running configuration created by get_Run_Config().

Value

None. The function saves output files to the specified directories.

run_script3_1 *Plotting simulation results for a CAWET scenario*

Description

This script generates graphs based on the results of a CAWET scenario simulation. It reads the simulation results and creates various visualizations such as pie charts for crop distribution, bar charts for annual and monthly irrigation data, and comparison graphs with validation data if provided. It saves the generated graphs in a specified directory for further analysis.

Usage

```
run_script3_1(cfgRun = get_Run_Config(Working_path = Working_path))
```

Arguments

cfgRun	(Optional) Running configuration created by get_Run_Config().

Value

This function does not return a value but saves graphs to the specified directory.

`run_script3_2` *Scenario Comparison Script*

Description

This function runs the script 3.2 for comparing scenarios based on the last run or a chosen run path. It generates comparison plots and CSV files for area, irrigation, and ETc in the Results_comparison folder.

Usage

```
run_script3_2(cfgRun = get_Run_Config(Working_path = Working_path))
```

Arguments

`cfgRun`	(Optional) Running configuration created by `get_Run_Config()`.

Value

This function does not return a value but generates comparison plots and CSV files in the Results_comparison folder.

`run_script3_3` *Climatic indicators calculation and graphs generation*

Description

This function calculates climatic indicators from meteorological data and generates corresponding graphs.

Usage

```
run_script3_3(cfgRun = get_Run_Config(Working_path = Working_path))
```

Arguments

`cfgRun`	(Optional) Running configuration created by `get_Run_Config()`.

Value

None. The function saves the results and graphs in the specified directories.

setup_CAWET *Setup CAWET instance in a specified directory*

Description

This function copies all necessary files from the CAWET package's extdata directory to a user-specified destination directory.

Usage

```
setup_CAWET(destdir)
```

Arguments

destdir	Character string specifying the destination directory where the CAWET instance should be set up.

Value

None. The function performs file copy operations.

Examples

```
destdir <- file.path(tempdir(), "CAWET")
setup_CAWET(destdir)
list.files(destdir)
```

st_read *Suppress messages from sf::st_read*

Description

This function is a wrapper around `sf::st_read` that suppresses messages. It can be used to avoid cluttering the console with messages when reading spatial data.

Usage

```
st_read(...)
```

Arguments

...	Arguments passed to `sf::st_read`.

Value

The return value of `sf::st_read`, invisibly.

`Verif_ordonnanceur_CAWET` *Ordonnanceur verification function for CAWET*

Description

This function verifies the content of the ordonnanceur (scenario file) for CAWET. It checks for the existence and correctness of various parameters and files specified in the ordonnanceur.

Usage

```
Verif_ordonnanceur_CAWET(cfgRun)
```

Arguments

`cfgRun`	(Optional) Running configuration created by `get_Run_Config()`.

Value

None. The function stops execution and raises an error if any verification fails.

Index