# Package: RADIS (via r-universe)

May 15, 2026

**Type** Package

**Title** RADIS - Spatial Weather and Environmental datas Extraction Package

**Version** 0.2.1

**Description** The RADIS package aims to facilitate the extraction of environmental data from various cartographic or spatial sources or APIs. It allows for the retrieval of this information using specific coordinates or spatial grids. This package also formats the extracted data in a way that makes it easily exploitable for further analysis.

**License** AGPL (>= 3)

**URL** https://https://forge.inrae.fr/umr-g-eau/radis, https://umr-g-eau.pages-forge.inrae.fr/radis/, https://inrae.r-universe.dev/RADIS

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

**Depends** R (>= 2.10)

**Imports** dataverse, dplyr, happign, httr, httr2, leaflet, logger, magrittr, mapview, methods, purrr, rappdirs, raster, readr, rlang, sf, stars, stats, terra, tibble, tidyr, units, utils

**Suggests** colorspace, ggplot2, knitr, rmarkdown, spelling, testthat (>= 3.0.0), tmap, withr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Language** en-US

**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** 2026-01-11 12:29:09 UTC

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

**RemoteRef** main

**RemoteSha** 5611e58cb023e57192f0dd2fd18b7d062578cf79

# Contents



---

`convert_list_to_tibble` *Convert list provided by the APIs into a tibble*

---

## Description

Convert list provided by the APIs into a tibble

## Usage

```
convert_list_to_tibble(l)
```

## Arguments

| | |
|---|---|
| l | a list provided by the API (See query_api) |

### Details

This function is used internally by all the retrieving data functions for converting data after the call to query_api.

### Value

A tibble::tibble with one row by record and one column by field.

---

`get_data_from_dataverse`

*Download and Cache a File from Dataverse*

---

### Description

Downloads a file from a Dataverse dataset using its DOI and caches it locally. If the file already exists in the cache directory and `force_download` is `FALSE`, the cached file is used. Otherwise, the file is downloaded from the Dataverse server.

### Usage

```
get_data_from_dataverse(
  doi,
  server,
  key = "",
  filename,
  cache_dir = tempdir(),
  force_download = FALSE,
  progress = FALSE
)
```

### Arguments

| | |
|---|---|
| `doi` | character The DOI of the Dataverse dataset. |
| `server` | character The base URL of the Dataverse server. |
| `key` | character The API token from dataverse website. See here for more information about token usage on dataverse platforms. It is required for some datasets (e.g. Info&Sols). By default, it is an empty string. |
| `filename` | character The name of the file to download from the dataset. |
| `cache_dir` | character The directory where the file should be cached. By default, it uses the system's cache directory (See radis_cache_dir). |
| `force_download` | logical If `TRUE`, forces re-download of the file even if it exists in the cache. |
| `progress` | logical If `TRUE`, shows download progress. Default is `FALSE`. |

### Value

character The local file path to the downloaded (or cached) file.

get_drias_daily *Retrieve DRIAS daily weather data from API*

## Description

Generic and method-specific functions to retrieve daily DRIAS data from the DRIAS/Explore2 climate database. The data are extracted from the RADIS API hosted at https://api.g-eau.fr.

## Usage

```
get_drias_daily(x = NULL, ...)

## S3 method for class '`NULL`'
get_drias_daily(x, LAMBX__greater, ...)

## S3 method for class 'numeric'
get_drias_daily(
  x,
  LAMBX__less,
  LAMBY__greater,
  LAMBY__less,
  LAMBX__greater = x,
  GCM,
  RCM,
  BC,
  RCP,
  ETP = "",
  DATE__greater = NULL,
  DATE__less = NULL,
  fields = NULL,
  base_url = Sys.getenv("RADIS_API_BASE_URL", "https://api.g-eau.fr"),
  ...
)

## S3 method for class 'bbox'
get_drias_daily(x, ...)

## S3 method for class 'sf'
get_drias_daily(x, overlay_mode = "extent", ...)
```

## Arguments

| | |
|---|---|
| x | Input object. Can be NULL, an sf object, or a bounding box (bbox). |
| ... | Additional arguments passed to methods, query_api, and spatial_combination. |
| LAMBX__greater | integer Minimum X coordinate in meters in meters (EPSG:27572). |
| LAMBX__less | integer Maximum X coordinate in meters (EPSG:27572). |

| | |
|---|---|
| LAMBY__greater | integer Minimum Y coordinate in meters (EPSG:27572). |
| LAMBY__less | integer Maximum Y coordinate in meters (EPSG:27572). |
| GCM | character Global Climate Model (GCM) |
| RCM | character Regional Climate Model (RCM) |
| BC | character Bias Correction method ("ADAMONT" or "CDFt") |
| RCP | character Representative Concentration Pathway (RCP) scenario ("rcp26", "rcp45", or "rcp85") |
| ETP | character Method used to compute potential evapotranspiration ("FAO" or "Hg0175"). Required when requesting "evspsblpot" field. |
| DATE__greater | character or Date Minimum date (YYYY-MM-DD). If NULL, no date filter is applied. |
| DATE__less | character or Date Maximum date (YYYY-MM-DD). If NULL, no date filter is applied. |
| fields | character vector Variables to extract, See below, the description of the returned values for available variables |
| base_url | Base URL for the API (default: "https://api.g-eau.fr"). |
| overlay_mode | Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:<br>• "within": selects only data strictly within the mask polygon(s).<br>• "intersecting": includes data that partially or fully intersect the polygon(s).<br>• "extent": includes all data within the bounding box of the mask polygon(s).<br>• "intersect_geometry": performs a full geometric intersection, returning clipped geometries.<br>• "aggregate": aggregates data by polygon (e.g., using zonal statistics). |

## Details

**Data Source:**

This function retrieves DRIAS/Explore2 climate projection data for France via the RADIS API (https://api.g-eau.fr/__docs__).

**Reference:**

Marson, Paola; Corre, Lola; Soubeyroux, Jean-Michel; Sauquet, Éric, 2024, "Rapport de synthèse sur les projections climatiques régionalisées", doi:10.57745/PUR7ML, Recherche Data Gouv, V1

More information about DRIAS climate projections is available at: https://www.drias-climat.fr

**Getting Available Scenario Combinations:**

**Before using this function**, you need to determine which combinations of GCM, RCM, BC, RCP, and ETP parameters are available. Use the get_drias_scenario() function to query available combinations

The returned data frame shows valid combinations of parameters that can be used with this function.

**Parameter Options:**

- **GCM**: Global Climate Model - varies by scenario (e.g., "CNRM-CERFACS-CNRM-CM5", "ICHEC-EC-EARTH", "MOHC-HadGEM2-ES")
- **RCM**: Regional Climate Model - varies by GCM (e.g., "CNRM-ALADIN63", "SMHI-RCA4", "CLMcom-CCLM4-8-17")
- **BC**: Bias Correction method - "ADAMONT" or "CDFt" (note: CDFt has limited variable availability)
- **RCP**: Representative Concentration Pathway - "rcp26", "rcp45", or "rcp85" for future projections, or use historical data
- **ETP**: Evapotranspiration method - "FAO" or "Hg0175" (required only when requesting "evspsblpot" variable)

**Coordinate System:**

Coordinates must be provided in Lambert II extended projection (EPSG:27572):

- **LAMBX**: X coordinate (Easting) in meters
- **LAMBY**: Y coordinate (Northing) in meters

If you have coordinates in a different CRS, use `sf::st_transform()` to convert to EPSG:27572 before querying, or pass an `sf` object directly (conversion is handled automatically)

**Available Fields:**

Here are the available fields:

- "evspsblpot": Potential evapotranspiration (mm/d)
- "huss": Specific humidity at 2m (kg/kg)
- "prsn": Snowfall (mm/d)
- "prtot": Total precipitation (mm/d)
- "rlds": Downward longwave radiation (W/m²)
- "rsds": Downward shortwave radiation (W/m²)
- "sfcWind": Surface wind speed (m/s)
- "tas": Air temperature at 2m (°C)
- "tasmax": Maximum air temperature at 2m (°C)
- "tasmin": Minimum air temperature at 2m (°C)

## Value

A [`stars`][stars::st_as_stars][stars::st_as_stars()] object containing the DRIAS daily weather data. If `overlay_mode` is specified (`sf` method), the output will be masked by the input spatial object.

## See Also

`get_drias_scenario()` to query available scenario combinations

## Examples

```
# First, check available scenarios
scenarios <- get_drias_scenario()
scenarios <- scenarios[sample(nrow(scenarios), 1), ]
scenarios

# Then retrieve climate data for a specific scenario
climate <- get_drias_daily(
  LAMBX__greater = 275000, LAMBX__less = 304000,
  LAMBY__greater = 1810000, LAMBY__less = 1840000,
  DATE__greater = "2000-01-01", DATE__less = "2000-01-31",
  GCM = scenarios$GCM, RCM = scenarios$RCM,
  BC = scenarios$BC, RCP = scenarios$RCP,
  fields = scenarios$variable, ETP = scenarios$ETP
)

# Using an sf object (coordinates automatically transformed)
sf <- sf::read_sf(
  system.file("extdata/study_area/test.shp", package = "RADIS")
)
climate_sf <- get_drias_daily(
  sf,
  DATE__greater = "2000-01-01", DATE__less = "2000-01-31",
  GCM = scenarios$GCM, RCM = scenarios$RCM,
  BC = scenarios$BC, RCP = scenarios$RCP,
  fields = scenarios$variable, ETP = scenarios$ETP
)
```

---

# get_drias_scenario *Query available DRIAS climate scenarios*

---

## Description

Retrieve the list of available DRIAS climate scenario combinations from the RADIS API. This function is useful for discovering which combinations of Global Climate Models (GCM), Regional Climate Models (RCM), Bias Correction methods (BC), scenarios (RCP), variables, and evapotranspiration methods (ETP) are available before calling `get_drias_daily()`.

## Usage

```
get_drias_scenario(
  GCM = "",
  RCM = "",
  BC = "",
  RCP = "",
  variable = "",
  ETP = "",
  base_url = Sys.getenv("RADIS_API_BASE_URL", "https://api.g-eau.fr")
)
```

## Arguments

| | |
|---|---|
| GCM | character Global Climate Model name. If empty (default), all available GCMs are returned. |
| RCM | character Regional Climate Model name. If empty (default), all available RCMs are returned. |
| BC | character Bias Correction method ("ADAMONT" or "CDFt"). If empty (default), all available methods are returned. |
| RCP | character RCP name ("rcp26", "rcp45", or "rcp85"). If empty (default), all available RCPs are returned. |
| variable | character Climate variable name. If empty (default), all available variables are returned. See Details for available variables. |
| ETP | character Evapotranspiration calculation method ("FAO" or "Hg0175"). If empty (default), all available methods are returned. |
| base_url | character Base URL of the RADIS API (default: "https://api.g-eau.fr"). |

## Details

### Data Source:

This function queries the `/drias_scenario` endpoint of the RADIS API at https://api.g-eau.fr/__docs__ to retrieve available DRIAS/Explore2 climate scenario combinations.

### Reference:

Marson, Paola; Corre, Lola; Soubeyroux, Jean-Michel; Sauquet, Éric, 2024, "Rapport de synthèse sur les projections climatiques régionalisées", doi:10.57745/PUR7ML, Recherche Data Gouv, V1

More information: https://www.drias-climat.fr

### Available Variables:

The following climate variables are available:

- "evspsblpot": Potential evapotranspiration (mm/d)
- "huss": Specific humidity at 2m (kg/kg)
- "prsn": Snowfall (mm/d)
- "prtot": Total precipitation (mm/d)
- "rlds": Downward longwave radiation (W/m²)
- "rsds": Downward shortwave radiation (W/m²)
- "sfcWind": Surface wind speed (m/s)
- "tas": Air temperature at 2m (°C)
- "tasmax": Maximum air temperature at 2m (°C)
- "tasmin": Minimum air temperature at 2m (°C)

### Usage Tips:

- Call without arguments to see all available combinations
- Filter by specific parameters to narrow down options
- Use the results to select valid parameter combinations for `get_drias_daily()`

## Value

A data.frame containing available scenario combinations with columns: GCM, RCM, BC, RCP, variable, and ETP.

## See Also

get_drias_daily() to retrieve actual climate data

## Examples

```
# Get all available scenarios
all_scenarios <- get_drias_scenario()

# Get only ADAMONT bias-corrected scenarios
adamont_scenarios <- get_drias_scenario(BC = "ADAMONT")

# Get scenarios for a specific GCM and RCP
specific_scenarios <- get_drias_scenario(
  GCM = "CNRM-CERFACS-CNRM-CM5",
  RCP = "rcp45"
)

# Get available precipitation scenarios
precip_scenarios <- get_drias_scenario(variable = "prtot")
```

---

# get_fyre_climate *Retrieve FYRE Climate data from API*

---

## Description

Generic and method-specific functions to retrieve daily FYRE Climate data via the G-EAU API, based on spatial and temporal filters.

This dataset has a spatial resolution equivalent to the SAFRAN reanalysis (See get_fyre_climate), and provides an ensemble of 25 members from 1871 to 2012 for temperature and precipitation.

Supports input as sf objects, bounding boxes, or explicit coordinate and date ranges.

## Usage

```
get_fyre_climate(x = NULL, ...)

## S3 method for class 'sf'
get_fyre_climate(x, api_format = "ncdf", overlay_mode = "extent", ...)

## S3 method for class 'bbox'
get_fyre_climate(x, ...)
```

```
## S3 method for class '`NULL`'
get_fyre_climate(x = NULL, LAMBX__greater, ...)

## S3 method for class 'numeric'
get_fyre_climate(
  x,
  LAMBX__less,
  LAMBY__greater,
  LAMBY__less,
  DATE__greater = NULL,
  DATE__less = NULL,
  member,
  fields = NULL,
  base_url = Sys.getenv("RADIS_API_BASE_URL", "https://api.g-eau.fr"),
  api_format = "ncdf",
  overlay_mode = "extent",
  LAMBX__greater = x,
  ...
)
```

## Arguments

| | |
|---|---|
| x | Input object. Can be NULL, an sf object, or a bounding box (bbox). |
| ... | Additional arguments passed to methods, query_api, and spatial_combination. |
| api_format | Data format either "csv" or "ncdf" |
| overlay_mode | Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:<br>• "within": selects only data strictly within the mask polygon(s).<br>• "intersecting": includes data that partially or fully intersect the polygon(s).<br>• "extent": includes all data within the bounding box of the mask polygon(s).<br>• "intersect_geometry": performs a full geometric intersection, returning clipped geometries.<br>• "aggregate": aggregates data by polygon (e.g., using zonal statistics). |
| LAMBX__greater | integer Minimum X coordinate in meters in meters (EPSG:27572). |
| LAMBX__less | integer Maximum X coordinate in meters (EPSG:27572). |
| LAMBY__greater | integer Minimum Y coordinate in meters (EPSG:27572). |
| LAMBY__less | integer Maximum Y coordinate in meters (EPSG:27572). |
| DATE__greater | character or Date Minimum date (YYYY-MM-DD). If NULL, no date filter is applied. |
| DATE__less | character or Date Maximum date (YYYY-MM-DD). If NULL, no date filter is applied. |
| member | integer Ensemble member number (1 to 25) |

| | |
|---|---|
| `fields` | character vector Variables to extract, See below, the description of the returned values for available variables |
| `base_url` | Base URL for the API (default: "https://api.g-eau.fr"). |

## Details

Excerpt from the notice available on https://doi.org/10.5281/zenodo.4005573: **FYRE Climate** (French hydrometeorological REanalysis Climate dataset) is a 25-member ensemble of 142-year high-resolution reanalysis of precipitation and temperature over France, from 1 January 1871 to 29 December 2012. FYRE Climate results from the assimilation of historical daily station observations of temperature and precipitation into the SCOPE Climate reconstructions (Caillouet et al., 2019) through a Kalman filter ensemble approach (Devers et al., 2020). FYRE Climate provides an ensemble of 25 equally-plausible spatially-coherent gridded bivariate time series. Data are available at a daily time step on a 8 km grid over France. Values cover grid cells located only within metropolitan France national borders (including Corsica). The FYRE Climate dataset is fully described by Devers et al. (2021).

**References to cite:**

Devers, A., Vidal, J.-P., Lauvernet, C., & Vannier, O. (2020). FYRE Climate: Precipitation [Data set]. In Climate of the Past (v1.0.0, Vol. 17, Numéro 5, p. 1857-1879). Zenodo. https://doi.org/10.5281/zenodo.4005573

Devers, A., Vidal, J.-P., Lauvernet, C., & Vannier, O. (2020). FYRE Climate: Temperature [Data set]. In Climate of the Past (v1.0.0, Vol. 17, Numéro 5, p. 1857-1879). Zenodo. https://doi.org/10.5281/zenodo.4006472

- `get_fyre_climate()`: Generic function to dispatch methods based on the class of `x`.
- `get_fyre_climate.sf()`: Accepts an `sf` object, transforms it to EPSG:27572, extracts its bounding box, and dispatches to the `bbox` method.
- `get_fyre_climate.bbox()`: Accepts a bounding box and dispatches to the default method with extracted coordinates.
- `get_fyre_climate.default()`: Accepts explicit coordinate and date range arguments, constructs an API query, and returns the result.

## Value

Depending on `api_format`, a tibble::tibble or a stars object.

The tibble::tibble has the following columns depending on `fields` parameter:

- `x`: Easting coordinate (meters) of the cell centroid in the Lambert II étendu projection (EPSG:27572)
- `y`: Northing coordinate (meters) of the cell centroid in the Lambert II étendu projection (EPSG:27572)
- `date`: Date (YYYYMMDD)
- `ptot`: Total precipitation (daily cumulative 06-06 UTC) (mm and tenths)
- `tas`: Temperature (daily average) (°C and hundredths)

The `[stars][stars::st_as_stars][stars::st_as_stars()]` object has dimensions `x`, `y`, and `time`, with the following variables depending on `fields` parameter: `ptot` and `tas`.

## Examples

```
# Extracting data from FYRE Climate using an sf object
sf <- sf::read_sf(
  system.file("extdata/study_area/test.shp", package = "RADIS")
)
get_fyre_climate(sf, member = 1, api_format = "csv")

# Comparing precipitation time series from the 25 members at Paris during winter 1909-1910
climate_members <- lapply(1:25, function(m) {
  get_fyre_climate(
    LAMBX__greater = 602510, LAMBX__less = 602510,
    LAMBY__greater = 2427300, LAMBY__less = 2427300,
    DATE__greater = "1909-10-01", DATE__less = "1910-01-31",
    member = m,
    fields = "ptot",
    api_format = "csv"
  ) %>% dplyr::mutate(member = m)
}) %>% dplyr::bind_rows()

library(ggplot2)
ggplot(climate_members, aes(x = as.Date(date), y = ptot, color = as.factor(member))) +
 geom_point() +
 labs(
  title = "FYRE Climate Temperature Time Series at Paris (1909-1910)",
  x = "Date",
 y = "Precipitation (mm)",
 color = "Member"
) +
theme_minimal()
```

---

# get_rpg_data *Get Crop Data from RPG and/or RPG completed.*

---

## Description

This function retrieves crop data from the Registre Parcellaire Graphique (RPG) via IGN Web Feature Service and / or from the completed RPG via the G-EAU API. Completed RPG data are produced by INRAE US-ODR.

## Usage

```
get_rpg_data(sf, year, id, crs = sf::st_crs(sf), source = c("IGN", "ODR"))

get_rpg_data_from_ign(sf, year, crs)

get_rpg_data_from_odr(sf, year, crs)
```

## Arguments

| | |
|---|---|
| sf | An sf object representing the study area. |
| year | integer specifying the year for which crop data is required. Must be between 2010 and 2022 if the source is only "IGN", between 2018 and 2023 if the source is only "ODR" and between 2018 and 2022 is the sources are both "IGN" and "ODR" |
| id | character specifying the column name in sf that contains the unique identifier for spatial features. |
| crs | Coordinate Reference System of the returned object |
| source | character vector sources of the RPG data (can be "IGN" and/or "ODR") |

## Value

An sf object containing crop data joined with the input spatial object. The returned object includes the following columns:

- id_parcel: The parcel identifier (ODR parcels are prefixed with "0DR").
- surf_parc: The declared surface area of the parcel.
- code_cultu: The crop code.
- source: The source of the data ("IGN" or "ODR").
- geometry: The geometry of the parcel.

## Examples

```
library(sf)
# Example spatial object
sf <- read_sf(
  system.file("extdata/study_area/test.shp", package = "RADIS")
)
# Retrieve crop data for the year 2020
crop_data <- get_rpg_data(sf, year = 2020, id = "id")
```

---

# get_sim2_daily *Retrieve SIM2 daily weather data from API*

---

## Description

Generic and method-specific functions to retrieve daily SIM2 data (obtained with the Safran - Isba model) via the G-EAU API based on spatial and temporal filters. These data are produced by Météo-France and located at https://www.data.gouv.fr/fr/datasets/6569b27598256cc583c917a7/. They are updated daily, have a spatial resolution of about 8km, and are available from 1958. Supports input as sf objects, bounding boxes, or explicit coordinate and date ranges.

## Usage

```
get_sim2_daily(x = NULL, ...)

## S3 method for class 'sf'
get_sim2_daily(x, api_format = "ncdf", overlay_mode = "extent", ...)

## S3 method for class 'bbox'
get_sim2_daily(x, ...)

## S3 method for class '`NULL`'
get_sim2_daily(x = NULL, LAMBX__greater, ...)

## S3 method for class 'numeric'
get_sim2_daily(
  x,
  LAMBX__less,
  LAMBY__greater,
  LAMBY__less,
  DATE__greater = NULL,
  DATE__less = NULL,
  fields = NULL,
  base_url = Sys.getenv("RADIS_API_BASE_URL", "https://api.g-eau.fr"),
  api_format = "ncdf",
  overlay_mode = "extent",
  LAMBX__greater = x,
  ...
)
```

## Arguments

| | |
|---|---|
| x | Input object. Can be NULL, an sf object, or a bounding box (bbox). |
| ... | Additional arguments passed to methods, query_api, and spatial_combination. |
| api_format | Data format either "csv" or "ncdf" |
| overlay_mode | Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:<br>• "within": selects only data strictly within the mask polygon(s).<br>• "intersecting": includes data that partially or fully intersect the polygon(s).<br>• "extent": includes all data within the bounding box of the mask polygon(s).<br>• "intersect_geometry": performs a full geometric intersection, returning clipped geometries.<br>• "aggregate": aggregates data by polygon (e.g., using zonal statistics). |
| LAMBX__greater | integer Minimum X coordinate in meters in meters (EPSG:27572). |
| LAMBX__less | integer Maximum X coordinate in meters (EPSG:27572). |
| LAMBY__greater | integer Minimum Y coordinate in meters (EPSG:27572). |

| | |
|---|---|
| LAMBY__less | integer Maximum Y coordinate in meters (EPSG:27572). |
| DATE__greater | character or Date Minimum date (YYYY-MM-DD). If NULL, no date filter is applied. |
| DATE__less | character or Date Maximum date (YYYY-MM-DD). If NULL, no date filter is applied. |
| fields | character vector Variables to extract, See below, the description of the returned values for available variables |
| base_url | Base URL for the API (default: "https://api.g-eau.fr"). |

## Details

- `get_sim2_daily()`: Generic function to dispatch methods based on the class of `x`.
- `get_sim2_daily.sf()`: Accepts an `sf` object, transforms it to EPSG:27572, extracts its bounding box, and dispatches to the `bbox` method.
- `get_sim2_daily.bbox()`: Accepts a bounding box and dispatches to the default method with extracted coordinates.
- `get_sim2_daily.default()`: Accepts explicit coordinate and date range arguments, constructs an API query, and returns the result as a tibble.

## Value

A tibble with possibly the following columns depending on `fields` parameter:

- `LAMBX`: Easting coordinate of cell centroid (meters) in the Lambert II étendu projection (EPSG:27572)
- `LAMBY`: Northing coordinate of cell centroid (meters) in the Lambert II étendu projection (EPSG:27572)
- `DATE`: Date (YYYYMMDD)
- `PRENEI_Q`: Solid precipitation (daily cumulative 06-06 UTC) (mm and tenths)
- `PRELIQ_Q`: Liquid precipitation (daily cumulative 06-06 UTC) (mm and tenths)
- `PE_Q`: Effective rain (daily cumulative) (mm and tenths)
- `T_Q`: Temperature (daily average) (°C and tenths)
- `TINF_H_Q`: Minimum temperature of the 24 hourly temperatures (°C and tenths)
- `TSUP_H_Q`: Maximum temperature of the 24 hourly temperatures (°C and tenths)
- `FF_Q`: Wind (daily average) (m/s and tenths)
- `DLI_Q`: Atmospheric radiation (daily cumulative) (J/cm2)
- `SSI_Q`: Visible radiation (daily cumulative) (J/cm2)
- `EVAP_Q`: Actual evapotranspiration (daily cumulative 06-06 UTC) (mm and tenths)
- `ETP_Q`: Potential evapotranspiration (Penman-Monteith formula) (mm and tenths)
- `Q_Q`: Specific humidity (daily average) (g/kg)
- `HU_Q`: Relative humidity (daily average) (%)
- `SWI_Q`: Soil moisture index (daily average 06-06 UTC) (%)
- `DRAINC_Q`: Drainage (daily cumulative 06-06 UTC) (mm and tenths)

- RUNC_Q: Runoff (daily cumulative 06-06 UTC) (mm and tenths)
- WG_RACINE_Q: Liquid water content in the root layer at 06 UTC (m3/m3)
- WGI_RACINE_Q: Frozen water content in the root layer at 06 UTC (m3/m3)
- RESR_NEIGE_Q: Snow water equivalent (daily average 06-06 UTC) (mm and tenths)
- RESR_NEIGE6_Q: Snow water equivalent at 06 UTC (mm and tenths)
- HTEURNEIGE_Q: Snow depth (daily average 06-06 UTC) (m)
- HTEURNEIGE6_Q: Snow depth at 06 UTC (m)
- HTEURNEIGEX_Q: Maximum snow depth during the day (m)
- SNOW_FRAC_Q: Fraction of the grid cell covered by snow (daily average 06-06 UTC) (%)
- ECOULEMENT_Q: Flow at the base of the snowpack (mm and tenths)

## Examples

```
# Retrieve data in CSV format

# Retrieve daily SIM2 climate data within specified bounding box
climate <- get_sim2_daily(
  LAMBX__greater = 275000,
  LAMBX__less = 304000,
  LAMBY__greater = 1810000,
  LAMBY__less = 1840000,
  DATE__greater = "2023-01-01",
  DATE__less = "2023-01-31",
  fields = c("PRENEI_Q", "PRELIQ_Q", "T_Q", "ETP_Q"),
  api_format = "csv"
)
head(climate)

# Using an sf object
sf <- sf::read_sf(
  system.file(
    "extdata/study_area/sub_basins/sub-basins.geojson",
    package = "RADIS"
  )
)
get_sim2_daily(
  sf,
  DATE__greater = "2023-01-01",
  DATE__less = "2023-01-31",
  fields = c("PRENEI_Q", "PRELIQ_Q", "T_Q", "ETP_Q"),
  api_format = "csv",
)

# Retrieve data in NetCDF format and mask with an sf object
nc_data <- get_sim2_daily(
  sf,
  DATE__greater = "2023-01-01",
  DATE__less = "2023-01-08",
  fields = c("PRENEI_Q", "PRELIQ_Q", "T_Q", "ETP_Q"),
```

```
  api_format = "ncdf"
)
print(nc_data)

# Plot the grid of the temperature variable for the first date
library(ggplot2)
ggplot() +
  stars::geom_stars(data = nc_data["T_Q"]) +
  facet_wrap(~time, nrow = 2) +
  scale_fill_distiller(
    palette = "RdBu",
    direction = -1, # reverse so blue = low, red = high
    name = "Temp (°C)",
    limits = c(0, 15)
  )

# Retrieve data in NetCDF and aggregate spatially using overlay_mode

nc_data_agg <- get_sim2_daily(
  sf,
  DATE__greater = "2023-01-01",
  DATE__less = "2023-01-08",
  fields = c("PRENEI_Q", "PRELIQ_Q", "T_Q", "ETP_Q"),
  api_format = "ncdf",
  overlay_mode = "aggregate"
)
print(nc_data_agg)

# Plot the temperature variable for the first date
ggplot() +
  stars::geom_stars(data = nc_data_agg["T_Q"]) +
  facet_wrap(~time, nrow = 2) +
  scale_fill_distiller(
    palette = "RdBu",
    direction = -1, # reverse so blue = low, red = high
    name = "Temp (°C)",
    limits = c(0, 15)
  )
```

---

`get_soil_awc` *Get Soil Available Water Capacity (AWC)*

---

## Description

Retrieves soil available water capacity (AWC) spatial data for a given study area. The function supports different data sources, with "BDGSF" as the default.

## Usage

```
get_soil_awc(
```

```
  sf,
  source = "BDGSF",
  overlay_mode = "aggregate",
  key = Sys.getenv("API_TOKEN_DATAVERSE"),
  cache_dir = radis_cache_dir("soil_awc", source),
  force_download = FALSE,
  ...
)

get_soil_awc_from_bdgsf(sf, cache_dir, force_download, ...)

get_soil_awc_from_infosols(
  sf,
  with_coarse_elements = TRUE,
  key,
  cache_dir,
  force_download,
  base_url = Sys.getenv("RADIS_API_BASE_URL", "https://api.g-eau.fr"),
  ...
)
```

## Arguments

| | |
|---|---|
| `sf` | An `sf` object representing the study area. |
| `source` | character The source of the soil AWC data. Can be either "BDGSF" (default, described here) or "Info&Sols" (described here). |
| `overlay_mode` | Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:<br>• `"within"`: selects only data strictly within the mask polygon(s).<br>• `"intersecting"`: includes data that partially or fully intersect the polygon(s).<br>• `"extent"`: includes all data within the bounding box of the mask polygon(s).<br>• `"intersect_geometry"`: performs a full geometric intersection, returning clipped geometries.<br>• `"aggregate"`: aggregates data by polygon (e.g., using zonal statistics). |
| `key` | character The API token from dataverse website. See here for more information about token usage on dataverse platforms. It is required for some datasets (e.g. Info&Sols). By default, it is an empty string. |
| `cache_dir` | character The directory where the file should be cached. By default, it uses the system's cache directory (See radis_cache_dir). |
| `force_download` | logical If `TRUE`, forces re-download of the file even if it exists in the cache. |
| `...` | Further arguments passed to spatial_combination and get_data_from_dataverse |
| `with_coarse_elements` | logical Only for "Info&Sols" source. Whether to include coarse elements in the soil AWC data. If `TRUE`, the function retrieves the AWC for the entire soil profile |

(0-200 cm) and its standard deviation. If FALSE, it retrieves the AWC for specific depth intervals (0-5 cm, 5-15 cm, 15-30 cm, 30-60 cm, 60-100 cm, and 100-200 cm).

base_url Base URL for the API (default: "http://api.g-eau.fr").

## Details

By default, the API token is retrieved from the environment variable API_TOKEN_DATAVERSE. So you need to set this variable with your Dataverse API token before running the function or pass it directly as an argument.

BDGSF datasets don't require an API token for access, so we don't need to pass the key argument to the download function.

## Value

An sf object containing soil AWC data intersected with the study area.

## Examples

```
# Using the example study area
sf <- sf::read_sf(
  system.file("extdata/study_area/test.shp", package = "RADIS")
)
# BDGSF source
awc_bdgsf <- get_soil_awc(sf = sf, source = "BDGSF")
awc_bdgsf
plot(awc_bdgsf[, c("soil_awc_max", "soil_awc_min")])

# Info & Sols source (90 m raster)
awc_infosols <- get_soil_awc(sf = sf, source = "Info&Sols")
awc_infosols
plot(awc_infosols[, 1:2])
```

---

# get_soil_depth *Get Soil Depth for a Study Area*

---

## Description

Retrieves soil depth data for a given study area as an sf object. The function can support multiple data sources, with " Base de Données Géographique des Sols de France (BDGSF)" as the default source. The function will:

- Download and cache soil depth data if not already available locally.
- Call spatial_combination to process the data for the study area according to the parameter overlay_mode

## Usage

```
get_soil_depth(
  sf,
  source = "BDGSF",
  overlay_mode = "aggregate",
  cache_dir = radis_cache_dir("soil_depth/BDGSF"),
  force_download = FALSE,
  ...
)

get_soil_depth_from_bdgsf(sf, cache_dir, force_download, ...)
```

## Arguments

| | |
|---|---|
| sf | An sf object representing the study area. |
| source | character The data source to use for soil depth. Currently supports "BDGSF" (described here). |
| overlay_mode | Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:<br>• "within": selects only data strictly within the mask polygon(s).<br>• "intersecting": includes data that partially or fully intersect the polygon(s).<br>• "extent": includes all data within the bounding box of the mask polygon(s).<br>• "intersect_geometry": performs a full geometric intersection, returning clipped geometries.<br>• "aggregate": aggregates data by polygon (e.g., using zonal statistics). |
| cache_dir | character The directory where the file should be cached. By default, it uses the system's cache directory (See radis_cache_dir). |
| force_download | logical If TRUE, forces re-download of the file even if it exists in the cache. |
| ... | Further arguments passed to spatial_combination and get_data_from_dataverse |

## Value

An spatial object containing soil depth information for the study area, including columns for minimum and maximum soil depth, and area in square meters. The format of the returned object depends on the process operated by spatial_combination.

## Examples

```
# Using the default study area
sf <- sf::read_sf(
  system.file("extdata/study_area/test.shp", package = "RADIS")
)
plot(get_soil_depth(sf = sf, force_download = TRUE))
```

get_soil_texture *Retrieve soil texture information for spatial features*

## Description

These functions retrieve soil texture information for a given spatial feature (sf) object from a specified data source.

## Usage

```
get_soil_texture(
  sf,
  source = "infosols",
  overlay_mode = "aggregate",
  ...,
  cache_dir = radis_cache_dir("soil_textures", source)
)

get_soil_texture_from_radisapi(
  sf,
  source,
  cache_dir = radis_cache_dir("soil_textures", source),
  force_download,
  base_url = Sys.getenv("RADIS_API_BASE_URL", "https://api.g-eau.fr"),
  ...
)

get_soil_texture_from_wcs(
  sf,
  vars = c("sand", "silt", "clay"),
  depths = c("0-5cm", "5-15cm", "15-30cm", "30-60cm", "60-100cm", "100-200cm"),
  stat = "mean",
  cache_dir = radis_cache_dir("soil_textures/soilgrids"),
  force_download = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| sf | An sf object representing the study area. |
| source | A character string specifying the data source to use. Available option are:<br>• "infosols": Roman Dobarco, Mercedes; Bourennane, Hocine; Arrouays, Dominique; Saby, Nicolas; Cousin, Isabelle; Manuel, Martin P., 2022, "Propriétés de granulométrie (argile, limons, sables) et d'éléments grossiers pour la France métropolitaine au pas de 90 m", Recherche Data Gouv, V1, DOI : https://doi.org/10.57745/N4E4NE. |

- "HWSD": Batjes (2016). Geoderma. FAO & IIASA (2023). Harmonized World Soil Database Version 2.0. DOI: 10.4060/cc3823en, DOI : https://doi.org/10.1016/j.geoderma.
- "soilgrids": Poggio, L., de Sousa, L. M., Batjes, N. H., Heuvelink, G. B. M., Kempen, B., Ribeiro, E., and Rossiter, D.: SoilGrids 2.0: producing soil information for the globe with quantified spatial uncertainty, SOIL, 7, 217–240, 2021. DOI : https://doi.org/10.5194/soil-7-217-2021

`overlay_mode` Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:

- "within": selects only data strictly within the mask polygon(s).
- "intersecting": includes data that partially or fully intersect the polygon(s).
- "extent": includes all data within the bounding box of the mask polygon(s).
- "intersect_geometry": performs a full geometric intersection, returning clipped geometries.
- "aggregate": aggregates data by polygon (e.g., using zonal statistics).

`...` Additional arguments passed to the underlying soil texture retrieval functions, and to spatial_combination.

`cache_dir` character The directory where the file should be cached. By default, it uses the system's cache directory (See radis_cache_dir).

`force_download` A `logical` flag; if `TRUE`, forces re-download even if the file already exists. Default is `FALSE`.

`base_url` A `character` scalar with the base URL of the infos&sol API. Default is `"http://api.g-eau.fr"`.

`vars` Character vector of variables to retrieve (e.g. c("sand","silt", "clay")).

`depths` Character vector of depth intervals (e.g. c("0-5cm", "5-15cm",...)).

`stat` Character; statistic to retrieve (default "mean", options include "Q0.5","Q0.05","Q0.95").

## Details

`get_soil_texture()` acts as a wrapper that not only calls the appropriate source-specific retrieval function (depending on the `source` argument) — such as `"HWSD"`, `"Info&Sols"`, or `"Soilgrids"` — but also performs a spatial combination of the retrieved soil data with the input geometries, according to the method specified by the `overlay_mode` parameter.

Disclaimer: The server providing the Regridded Harmonized World Soil Database (HWSD) is not reliable as of the date of this release (2025-11-05). Use at your own risk.

## Value

An object containing soil texture information corresponding to the input `sf` features.

A `SpatRaster` object containing the requested soil texture layer clipped to the extent of `sf`.

A `SpatRaster` stack containing requested variables.

## Examples

```
library(sf)
# Using an sf object
sf <- sf::read_sf(
  system.file("extdata/study_area/test.shp", package = "RADIS")
)
# Retrieve soil texture from infos&sols
sf_texture <- get_soil_texture(sf, source = "infosols")

# Retrieve soil texture from HWSD
sf_texture_hwsd <- get_soil_texture(sf, source = "hwsd")

# Retrieve soil texture from SoilGrids
sf_texture_soilgrids <- get_soil_texture(sf, source = "soilgrids")
```

---

# get_spatial_format *Get the spatial format of a geospatial object*

---

## Description

Get the spatial format of a geospatial object

## Usage

```
get_spatial_format(s)
```

## Arguments

| | |
|---|---|
| s | A stars object or any object convertible to stars. |

## Value

A character string indicating the spatial format: "raster" or "geometry".

---

# query_api *Query API*

---

## Description

Query API

## Usage

```
query_api(
  url,
  ...,
  output_format = "file",
  file_ext,
  cache_dir = radis_cache_dir(basename(url)),
  force_download = FALSE,
  user_agent = getOption("RADIS.user_agent", "RADIS R package")
)
```

## Arguments

| | |
|---|---|
| `url` | API base URL |
| `...` | Query parameters in format `field="value"`. Query parameters can also be embbeded in a list |
| `output_format` | API data format, either:<br>• `"file"` Save the data to the disk whatever its format<br>• `"csv"` returns a data.frame from a CSV output<br>• `"json"` returns a list from a json output<br>• `"tibble_json"` returns a tibble::tibble from a json output<br>• `"binary"` returns the API output without any conversion (binary format)<br>• `"string"` returns the API output without any conversion (string format) |
| `file_ext` | File extension for the saved file if `output_format="file"`, ignored otherwise |
| `cache_dir` | character The directory where the file should be cached. By default, it uses the system's cache directory (See radis_cache_dir). |
| `force_download` | logical If `TRUE`, forces re-download of the file even if it exists in the cache. |
| `user_agent` | User agent to use for HTTP requests. Defaults to "RADIS R package". Can be defined globally with the option `"RADIS.user_agent"` |

## Value

If `output_format="file"`, it returns the path of the temporary files recorded from API output, otherwise an object depending on the requested format.

---

# `radis_cache_dir` *Get radis cache directory*

---

## Description

`radis_cache_dir` returns the path to a radis cache directory, creating the directory if it does not already exist.

## Usage

```
radis_cache_dir(
  ...,
 base_dir = file.path(Sys.getenv("RADIS_CACHE_DIR", rappdirs::user_cache_dir("radis")),
    "data-raw"),
  create_dir = TRUE
)

clean_radis_cache(cache_dir = radis_cache_dir(""))
```

## Arguments

| | |
|---|---|
| ... | path inside the radis cache directory |
| base_dir | Base directory for the radis cache. By default, it uses the RADIS_CACHE_DIR environment variable if set, otherwise it uses the user cache directory for the "radis" application as determined by the rappdirs::user_cache_dir() function, with an additional "data-raw" subdirectory. |
| create_dir | Logical indicating whether to create the directory if it does not exist. Default is TRUE. |
| cache_dir | Path to the radis cache directory to be cleaned. |

## Details

clean_radis_cache deletes all files and subdirectories inside the specified radis cache directory (all the cache by default).

## Value

Path to the radis cache directory or a subdirectory inside it

---

# spatial_combination *Combine a spatial dataset with mask polygons*

---

## Description

Combine a spatial dataset with mask polygons

## Usage

```
spatial_combination(
  data,
  mask,
  overlay_mode,
  output_format = "default",
  crs = sf::st_crs(mask),
  ...
)
```

### Arguments

| | |
|---|---|
| data | Spatial dataset to combine with mask polygons |
| mask | An sf or a terra::SpatVector object representing the polygons contained in the study area. |
| overlay_mode | Character. Specifies how the spatial mask is applied to the input dataset. Must be one of:<br>• "within": selects only data strictly within the mask polygon(s).<br>• "intersecting": includes data that partially or fully intersect the polygon(s).<br>• "extent": includes all data within the bounding box of the mask polygon(s).<br>• "intersect_geometry": performs a full geometric intersection, returning clipped geometries.<br>• "aggregate": aggregates data by polygon (e.g., using zonal statistics). |
| output_format | character defining the object format or returned value (See details) |
| crs | Coordinate Reference System of the returned object |
| ... | Ignored. Only included for compatibility with other methods/functions. |

### Details

output_format should be one of:

- "default": returns a sf, SpatRaster or a stars object depending if the output is respectively a polygon layer, a raster, or a time series.
- Other possible values are "SpatVector", "SpatRaster", "stars", "sf", or "raster"

### Value

An object of the type defined by output_format (See details) containing the data transformed according to overlay_mode.

If overlay_mode is "intersecting", the return value is a list containing one item by mask polygon with each item containing the rows of data intersecting the concerned mask polygon.

---

## view_data_map *Visualize Spatial Data on an Interactive Map*

---

### Description

Displays an sf object on an interactive map using the mapview package, with optional overlay of a second spatial layer.

### Usage

```
view_data_map(sf, rpg_sf = NULL, cols_to_show, basemaps = "OpenStreetMap")
```

## Arguments

| | |
|---|---|
| sf | An sf object containing the main spatial data to visualize. |
| rpg_sf | Optional. An sf object representing an additional spatial layer to overlay (e.g., RPG polygons). Default is NULL. |
| cols_to_show | Character vector of column names in sf to display in popups and labels. |
| basemaps | Character vector specifying basemap(s) to use. Default is "OpenStreetMap". |

## Details

Only the columns specified in cols_to_show (plus the geometry column) are retained for display. If rpg_sf is provided, it is added as a separate overlay group ("RPG") with a distinct style. Layer controls allow toggling the visibility of the main and overlay layers.

## Value

A mapview object displaying the selected spatial data and optional overlay, with interactive layer controls.

## Examples

```
## Not run:
  view_data_map(my_sf, rpg_sf = my_rpg, cols_to_show = c("name", "value"))
  view_data_map(my_sf, cols_to_show = c("id", "type"), basemaps = c("CartoDB.Positron"))

## End(Not run)
```

# Index