Package 'xaci'

September 9, 2026

Title Actuarial Climate Index

Version 0.1.0

Description Computes the Actuarial Climate Index (ACI) and its components (temperature, precipitation, drought, wind, sea level) from gridded climate data ('NetCDF') and tide gauge records. Implements the methodology described in Garrido, Milhaud & Olympio (2025) <https://hal.science/hal-04491982v2> for a French/European actuarial climate index, building on the American Academy of Actuaries framework, to any country in the world.

License MIT + file LICENSE

URL https://github.com/XavierMilhaud/xaci

BugReports https://github.com/XavierMilhaud/xaci/issues

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ncdf4 (>= 1.19), dplyr (>= 1.1), zoo (>= 1.8), readr (>= 2.1), ggplot2 (>= 3.4.0), tidyr, sf, rnaturalearth, geodata, terra, units, parallel

Suggests testthat (>= 3.0.0), patchwork (>= 1.1.0), ecmwfr (>= 2.0.0), rnaturalearthdata, gganimate, gifski, knitr, rmarkdown, spelling

RoxygenNote 7.3.3

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Xavier Milhaud [aut, cre, ctb], Dabakh Kane [ctb], Esteban Mauboussin [ctb]

Maintainer Xavier Milhaud <xavier.milhaud.research@gmail.com>

Repository CRAN

Date/Publication 2026-09-09 15:20:02 UTC

Contents

`.spatraster_to_list` *Convert a (small) SpatRaster back to the package's plain-list format*

Description

Once data has been reduced to daily (or coarser) resolution, it is small enough to hand off to the rest of the existing base-R pipeline (`resample_monthly()`, `standardize_metric()`, `.compute_aci_grid()`, etc.) unchanged. This is the bridge between the terra-based loading/reduction steps and that pipeline.

Usage

```
.spatraster_to_list(r)
```

Arguments

`r`	A `terra::SpatRaster`, reasonably small (daily resolution or coarser – NOT intended for hourly data).

Value

A list with elements `data` ([lon x lat x time] array), `lon`, `lat`, `time` – the same structure produced by `load_netcdf()`.

`aci` *Actuarial Climate Index (ACI)*

Description

Computes the full Actuarial Climate Index by combining all five components: temperature (T90 and T10), precipitation, drought, wind, and sea level.

`aggregate_granularity` *Aggregate a monthly data frame to a given temporal granularity*

Description

Aggregate a monthly data frame to a given temporal granularity

Usage

```
aggregate_granularity(df, granularity = "month")
```

Arguments

`df`	A data.frame with "YYYY-MM-01" Date row names (monthly).
`granularity`	One of "month", "season", "semester", "year".

Value

A data.frame with aggregated values and appropriate row names.

`animate_aci_map` *Animate the ACI or a component over time*

Description

Produces a GIF animation (via **gganimate**) showing the evolution of a spatial variable at each time step. Requires **gganimate** and **gifski** (or **png**) to be installed.

Usage

```
animate_aci_map(
  data,
  variable = "ACI",
  country_abbrev = NULL,
  admin_level = NULL,
  crs_metric = 4326,
  var_label = "Standardised anomaly",
  title = "ACI over time",
  palette = "RdBu",
  reverse = TRUE,
  n_breaks = 9,
  borders = TRUE,
  fps = 4,
  width = 800,
  height = 600,
  save_path = NULL
)
```

Arguments

`data`	Grid-cell list or admin `data.frame`.
`variable`	Variable to animate. Default `"ACI"`.
`country_abbrev`	Three-letter ISO code.
`admin_level`	Integer or `NULL`.
`crs_metric`	EPSG code, used for admin choropleth projection only. Default `4326`. Ignored (with a warning if explicitly set to something else) in raster (grid-cell) mode, where the basemap is always reprojected to EPSG:4326 to match the fixed lon/lat raster grid.
`var_label`	Colour-bar label. Default `"Standardised anomaly"`.
`title`	Plot title. Default `"ACI over time"`.
`palette`	RColorBrewer palette. Default `"RdBu"`.
`reverse`	Logical. Default `TRUE`.
`n_breaks`	Integer. Default `9`.
`borders`	Logical. Default `TRUE`.

fps	Frames per second. Default 4.
width	Output width in pixels. Default 800.
height	Output height in pixels. Default 600.
save_path	File path for the output GIF. If NULL (default), the animation is rendered to the viewer but not saved.

Value

A **gganimate** animation object (invisibly).

Why dontrun

This example is wrapped in dontrun because it depends on the optional **gganimate** and **gifski** packages (declared in Suggests, not guaranteed to be installed on every machine running R CMD check), and because rendering a multi-frame GIF takes well over 5 seconds, making it unsuitable for donttest as well.

Examples

```
## Not run:
lon <- seq(-5, 8, length.out = 10)
lat <- seq(42, 51, length.out = 8)
arr <- array(stats::rnorm(10 * 8 * 5), dim = c(10, 8, 5))
grid <- list(ACI = arr, lon = lon, lat = lat,
             time = seq_len(5))

animate_aci_map(grid, variable = "ACI", country_abbrev = "FRA",
                fps = 2, save_path = file.path(tempdir(), "aci_animation.gif"))

## End(Not run)
```

---

apply_mask *Apply a country mask to a NetCDF data array*

Description

Sets grid cells to NA where the mask value is below threshold.

Usage

```
apply_mask(dataset, mask_path, var_name, threshold = 0.8)
```

Arguments

dataset	List returned by load_netcdf().
mask_path	Path to the mask NetCDF file (variable: country).
var_name	Name of the variable inside dataset.
threshold	Numeric threshold; cells with mask < threshold become NA. Default 0.8.

Value

The dataset list with `data` masked in-place.

`apply_mask_terra` *Apply a country mask to a SpatRaster*

Description

Equivalent of `apply_mask()`: sets cells to `NA` where the mask value is below `threshold`. Processed block-by-block by terra.

Usage

```
apply_mask_terra(r, mask_path, threshold = 0.8, chunk_size = 20000)
```

Arguments

`r`	A `terra::SpatRaster` (e.g. from `load_netcdf_terra()`).
`mask_path`	Path to the mask NetCDF file (variable: `country`).
`threshold`	Numeric threshold. Default `0.8`.
`chunk_size`	Nombre max de couches traitees par bloc quand `nlyr(r)` depasse 65535. Default `20000` (marge confortable sous la limite, ajustable selon la RAM/disque disponibles).

Details

Note sur la limite de 65535 couches : le format interne utilise par `terra::mask()` pour ecrire son resultat sur disque (fichier temporaire) ne supporte pas plus de 65535 couches en sortie. Au-dela (cas frequent en donnees horaires sur plusieurs decennies : ~300k couches pour 35 ans), on ne peut pas masquer l'objet en un seul appel. Cette fonction bascule donc automatiquement sur un traitement par blocs temporels (‘chunk_size‘ couches a la fois), ecrits en GeoTIFF (BigTIFF) puis recombines en une seule source multi-fichiers, sans jamais materialiser l'ensemble en RAM. Le masque etant purement spatial (identique a chaque pas de temps), le decoupage en blocs ne change pas le resultat : un pixel exclu par le masque l'est de la meme facon dans chaque bloc.

Value

The masked `terra::SpatRaster`.

`assign_sealevel_to_admin` *Assign PSMSL tide-gauge stations to administrative units and compute coastal factors*

Description

Performs a spatial join between PSMSL station coordinates and administrative unit polygons, and computes for each unit the fraction of its perimeter that is coastline. The coastline layer is cropped to the country bounding box before intersection to speed up computation. All geometries are projected to `crs_metric` before length calculations to ensure results are in metres.

Usage

```
assign_sealevel_to_admin(country_abbrev, admin_level = 1, crs_metric = 4326)
```

Arguments

`country_abbrev`	ISO-3 country code (e.g. `"FRA"`).
`admin_level`	Integer >= 0. Administrative level fetched via GADM (see `.load_admin_sf()` in `utils.R`): `0` for the national boundary, `1` for regions, `2` for departments, etc. Default `1`.
`crs_metric`	Integer. EPSG code of a metric CRS appropriate for the country, used for accurate length calculations. Default `4326` (WGS84, not recommended for production — prefer a local CRS such as `2154` for France or `27700` for the UK).

Value

A list with two elements:

`station_ids` Named list: keys are administrative unit names, values are integer vectors of PSMSL station IDs within that unit.

`factors` Named numeric vector: keys are administrative unit names, values are the coastal fraction (coastline length / total perimeter) in `[0, 1]`. Zero for landlocked units.

`build_admin_mask` *Build a spatial weight matrix mapping ERA5 grid cells to administrative units*

Description

For each ERA5 grid cell, computes the fraction of its surface area that falls within each administrative unit polygon (using `sf` geometry intersection). All geometries are projected to `crs_metric` before area calculations to ensure results are in metres squared. The result can be computationally expensive for fine grids and large countries — consider caching it with `saveRDS()`.

Usage

```
build_admin_mask(
  lon,
  lat,
  country_abbrev,
  admin_level = 1,
  resolution = 0.25,
  crs_metric = 4326,
  cache_dir = .gadm_cache_dir()
)
```

Arguments

lon	Numeric vector of ERA5 longitudes.
lat	Numeric vector of ERA5 latitudes.
country_abbrev	ISO-3 country code (e.g. "FRA").
admin_level	Integer >= 0. Administrative level fetched via GADM: 0 for the national boundary, 1 for regions, 2 for departments, etc. (availability depends on how finely GADM subdivides the given country). Default 1.
resolution	Numeric. Half-width of ERA5 grid cells in degrees. Default 0.25.
crs_metric	Integer. EPSG code of a metric CRS appropriate for the country, used for accurate area calculations. Default 4326 (WGS84, not recommended for production — prefer a local CRS such as 2154 for France or 27700 for the UK).
cache_dir	Directory used to cache the downloaded GADM administrative boundaries. Default: a persistent per-user cache directory (see tools::R_user_dir()).

Value

A list with elements:

weights A named list of length length(lon) * length(lat). Each element is a named numeric vector of (unit -> fractional area weight) pairs.

units Character vector of all administrative unit names.

lon Input longitudes.

lat Input latitudes.

---

calculate_aci *Calculate the Actuarial Climate Index*

Description

This is the main entry point of the package. It instantiates all five climate components, standardises them over the reference period, and combines them into a single ACI time series:

Usage

```
calculate_aci(
  country_abbrev,
  study_period,
  reference_period,
  years = NULL,
  temperature_data_path = NULL,
  precipitation_data_path = NULL,
  wind_u10_data_path = NULL,
  wind_v10_data_path = NULL,
  mask_data_path = NULL,
  sealevel_dir = NULL,
  percentile_high = 90,
  percentile_low = 10,
  granularity = "month",
  area = TRUE,
  factor = 1/5,
  max_dist_km = 500,
  admin_level = NULL,
  crs_metric = 4326,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL,
  computed_components = FALSE,
  engine = c("base", "terra"),
  cores = 1L
)
```

Arguments

`country_abbrev`	Three-letter ISO 3166-1 alpha-3 country code (e.g. `"FRA"`). Used to build default file paths and to download PSMSL tide-gauge data.
`study_period`	Character vector of length 2: `c("YYYY-MM-DD", "YYYY-MM-DD")` defining the study window.
`reference_period`	Character vector of length 2: `c("YYYY-MM-DD", "YYYY-MM-DD")` defining the climatological reference period used for standardisation.
`years`	Integer vector of years covered by the NetCDF files (e.g. `2011:2015`). Used to build default file paths when any `*_data_path` argument is `NULL`. Not needed if all paths are supplied explicitly.
`temperature_data_path`	Path to the hourly 2-m temperature NetCDF (`t2m` variable). If `NULL` (default), built automatically from `country_abbrev` and `years`.
`precipitation_data_path`	Path to the precipitation NetCDF (`tp` variable). If `NULL`, built automatically.

`wind_u10_data_path`
: Path to the u-component wind NetCDF (`u10` variable). If `NULL`, built automatically.

`wind_v10_data_path`
: Path to the v-component wind NetCDF (`v10` variable). If `NULL`, built automatically.

`mask_data_path`
: Path to the country mask NetCDF (`country` variable). If `NULL`, built automatically.

`sealevel_dir`
: Character or `NULL`. Path to a directory of already-downloaded PSMSL `.txt` files. If `NULL` (default), data are downloaded automatically to a sub-directory of `tempdir()`.

`percentile_high`
: Numeric. Upper percentile used for the hot temperature component. Default `90`. The corresponding column in the output will be named `t<percentile_high>` (e.g. `t90`).

`percentile_low`
: Numeric. Lower percentile used for the cold temperature component. Default `10`. The corresponding column in the output will be named `t<percentile_low>` (e.g. `t10`).

`granularity`
: Temporal aggregation level. One of `"month"` (default), `"season"`, `"semester"`, `"year"`. Seasons follow meteorological convention: DJF (Dec-Feb), MAM (Mar-May), JJA (Jun-Aug), SON (Sep-Nov). December is attributed to the following year's winter (e.g. December 2010 -> "2011-DJF").

`area`
: Logical. Only used when `admin_level = NULL`. If `TRUE` (default), returns a national scalar time series (spatial mean over the country). If `FALSE`, returns the full grid-cell-level ACI and all components as arrays `[lon x lat x time]`, suitable for mapping or custom aggregation. Ignored when `admin_level` is not `NULL`.

`factor`
: Numeric in `[0, 1]`. Weight of the sea-level component at the national level, representing the fraction of coastal area. Default `1/5`. At the administrative unit level, this argument is ignored: the coastal fraction is computed automatically per unit by `assign_sealevel_to_admin()`. At grid-cell level, α is set to 1 for cells within `max_dist_km` of a tide-gauge station, and 0 otherwise.

`max_dist_km`
: Numeric. Maximum distance (km) from a tide-gauge station for a grid cell to receive a sea-level signal. Cells beyond this threshold receive `NA` for sea level and $\alpha = 0$. Only used when `area = FALSE` and `admin_level = NULL`. Default `500`.

`admin_level`
: Integer or `NULL`. If `NULL` (default), returns a single national index. Otherwise computes the ACI per administrative unit at the given level (1 = regions, 2 = departments, etc.). The sea-level component is automatically excluded ($\alpha = 0$) for units without coastal tide-gauge stations.

`crs_metric`
: Integer. EPSG code of a metric CRS appropriate for the country, used for accurate area and length calculations in `build_admin_mask()` and `assign_sealevel_to_admin()`. Only used when `admin_level` is not `NULL`. Default `4326` (WGS84, not recommended for production — prefer a local CRS such as `2154` for France or `27700` for the UK).

save — Logical. If TRUE, saves the grid-cell-level object to save_dir before aggregation. Default FALSE.

save_dir — Character. Directory for the cached .rds file. Created if it does not exist. Default NULL, which resolves to a sub-directory of tempdir().

load_dir — Character. Directory from which to reload previously saved .rds component files when computed_components = TRUE. Default NULL, which resolves to a sub-directory of tempdir().

computed_components — Logical. If TRUE, rds files storing results of the computations of ACI components at grid cell level are reused. Default FALSE.

engine — Character. "base" (default) uses the original ncdf4-based loading for all components. "terra" switches temperature_component(), wind_component(), drought_component(), and precipitation_component() to their memory-safe terra-based equivalents (see .resolve_component_functions()), recommended for long, high-resolution historical series (40+ years hourly) that would otherwise saturate RAM. sealevel_component() is unaffected by this argument in either case.

cores — Positive integer, default 1. Only used when engine = "terra", and only by temperature_component_terra() (passed through to calculate_percentiles_terra()'s own cores – see its performance note: the rolling-window percentile step has no built-in parallelism in terra itself, so this is where parallelizing across spatial tiles helps most). Silently ignored for engine = "base" and for the other three components, which don't have a cores argument.

Details

$$ACI = \frac{T_{high} - T_{low} + P + D + \alpha \cdot SL + W}{5 + \alpha}$$

where α is the sea-level erosion factor (coastal fraction, default 1/5). For administrative units without coastal stations, $\alpha = 0$ and the denominator becomes 5. At grid-cell level, $\alpha \in \{0, 1\}$ depending on whether the cell lies within max_dist_km of a tide-gauge station.

Value

National scalar (area = TRUE**,** admin_level = NULL**)** A data.frame with columns t<percentile_high>, t<percentile_low>, precipitation, drought, wind, sealevel, and ACI, indexed by dates at the chosen granularity.

Grid-cell (area = FALSE**,** admin_level = NULL**)** A named list with one array [lon x lat x nt] per component (ACI, t<percentile_high>, t<percentile_low>, precipitation, drought, wind, sealevel), plus lon, lat, and time (character vector of period labels at the chosen granularity).

Administrative (admin_level **integer)** A data.frame indexed by dates at the chosen granularity, with one <component>_<unit> column per administrative unit FOR EACH component (ACI, t<percentile_high>, t<percentile_low>, precipitation, drought, wind, sealevel) – e.g. ACI_<unit>, t90_<unit>, precipitation_<unit>, etc. This mirrors the grid-cell mode's per-component structure, and lets plot_aci_map() plot any individual component at admin level, not just ACI itself.

Examples

The four examples below all require real ERA5 NetCDF files (temperature, precipitation, wind, country mask), obtained beforehand via download_era5/download_mask and a Copernicus CDS API key (see cds_set_key). They cannot be run automatically by R CMD check and are therefore wrapped in dontrun: (1) national ACI at annual granularity; (2) grid-cell ACI (full spatial output); (3) ACI with custom percentiles (T95/T5); and (4) ACI by department (admin level 2), using Lambert-93 for France.

Examples

```
## Not run:
result <- calculate_aci(
  temperature_data_path   = "t2m_1960-2020.nc",
  precipitation_data_path = "tp_1960-2020.nc",
  wind_u10_data_path      = "u10_1960-2020.nc",
  wind_v10_data_path      = "v10_1960-2020.nc",
  country_abbrev          = "FRA",
  mask_data_path          = "mask_FRA.nc",
  study_period            = c("1980-01-01", "2020-12-31"),
  reference_period        = c("1961-01-01", "1990-12-31"),
  granularity             = "year",
  area                    = TRUE,
  factor                  = 1/5
)

grid <- calculate_aci(
  temperature_data_path   = "t2m_1960-2020.nc",
  precipitation_data_path = "tp_1960-2020.nc",
  wind_u10_data_path      = "u10_1960-2020.nc",
  wind_v10_data_path      = "v10_1960-2020.nc",
  country_abbrev          = "FRA",
  mask_data_path          = "mask_FRA.nc",
  study_period            = c("1980-01-01", "2020-12-31"),
  reference_period        = c("1961-01-01", "1990-12-31"),
  granularity             = "year",
  area                    = FALSE,
  max_dist_km             = 500
)

result_custom <- calculate_aci(
  temperature_data_path   = "t2m_1960-2020.nc",
  precipitation_data_path = "tp_1960-2020.nc",
  wind_u10_data_path      = "u10_1960-2020.nc",
  wind_v10_data_path      = "v10_1960-2020.nc",
  country_abbrev          = "FRA",
  mask_data_path          = "mask_FRA.nc",
  study_period            = c("1980-01-01", "2020-12-31"),
  reference_period        = c("1961-01-01", "1990-12-31"),
  granularity             = "year",
  percentile_high         = 95,
  percentile_low          = 5
)
```

```
result_dept <- calculate_aci(
  temperature_data_path   = "t2m_1960-2020.nc",
  precipitation_data_path = "tp_1960-2020.nc",
  wind_u10_data_path      = "u10_1960-2020.nc",
  wind_v10_data_path      = "v10_1960-2020.nc",
  country_abbrev          = "FRA",
  mask_data_path          = "mask_FRA.nc",
  study_period            = c("1980-01-01", "2020-12-31"),
  reference_period        = c("1961-01-01", "1990-12-31"),
  granularity             = "year",
  area                    = FALSE,
  admin_level             = 2,
  crs_metric              = 2154
)

## End(Not run)
```

`calculate_halfday_component` *Calculate the half-day (day or night) temperature component*

Description

Calculate the half-day (day or night) temperature component

Usage

```
calculate_halfday_component(
  dataset,
  reference_period,
  part_of_day,
  extremum,
  percentile,
  above_thresholds
)
```

Arguments

`dataset`	Full sub-daily `t2m` dataset (list).
`reference_period`	Character vector `c("YYYY-MM-DD", "YYYY-MM-DD")`.
`part_of_day`	`"day"` or `"night"`.
`extremum`	`"min"` or `"max"`.
`percentile`	Numeric percentile (e.g. 90 or 10).
`above_thresholds`	Logical. `TRUE` counts days above the threshold (hot extremes); `FALSE` counts days below (cold extremes).

Value

A list with `data` [lon × lat × months] and monthly `time`.

`calculate_halfday_component_terra` *Calculate the half-day (day or night) temperature component (terra version)*

Description

Calculate the half-day (day or night) temperature component (terra version)

Usage

```
calculate_halfday_component_terra(
  r,
  reference_period,
  part_of_day,
  extremum,
  percentile,
  above_thresholds,
  mask_path = NULL,
  threshold = 0.8,
  cores = 1L
)
```

Arguments

`r`	A `terra::SpatRaster`, hourly resolution, temperature already in the desired unit (e.g. Celsius), with `terra::time()` set. **Non masque** : le masquage se fait ici, une fois les donnees reduites (voir note ci-dessous).
`reference_period`	Character vector `c("YYYY-MM-DD", "YYYY-MM-DD")`.
`part_of_day`	`"day"` or `"night"`.
`extremum`	`"min"` or `"max"`.
`percentile`	Numeric percentile (e.g. 90 or 10).
`above_thresholds`	Logical. `TRUE` counts days above the threshold (hot extremes); `FALSE` counts days below (cold extremes).
`mask_path`	Path to the mask NetCDF file, or `NULL` (no masking).
`threshold`	Numeric threshold for the mask. Default `0.8`.
`cores`	Passed to `calculate_percentiles_terra()`'s own `cores` (see its performance note) – the rolling-window percentile step is by far the most expensive part of this function. Default `1` (sequential, identical to previous behaviour).

Value

Same structure as `calculate_halfday_component()`: `list(data, time, lon, lat)`.

`calculate_maximum_precipitation_over_window`
Calculate maximum precipitation over a rolling window

Description

Computes the rolling `window_size`-day sum of precipitation, then takes the monthly maximum.

Usage

```
calculate_maximum_precipitation_over_window(
  dataset,
  var_name = "tp",
  window_size = 5L
)
```

Arguments

`dataset`	List returned by `load_component()` for `tp`. May be sub-daily (e.g. hourly ERA5 data); it is always resampled to daily totals first.
`var_name`	Variable name in `dataset`. Default `"tp"`, inherited from ERA5 (variable name in the NetCDF).
`window_size`	Rolling window in DAYS. Default `5`.

Value

A list with `data` [lon x lat x months] and `time`.

`calculate_maximum_precipitation_over_window_terra`
Calculate maximum precipitation over a rolling window (terra version)

Description

Calculate maximum precipitation over a rolling window (terra version)

Usage

```
calculate_maximum_precipitation_over_window_terra(
  r,
  var_name = "tp",
  window_size = 5L,
  mask_path = NULL,
  threshold = 0.8
)
```

Arguments

r	A terra::SpatRaster, hourly (or sub-daily) resolution precipitation, with terra::time() set. **Non masque** : le masquage se fait ici, une fois les donnees reduites a la resolution journaliere (voir note ci-dessous).
var_name	Variable name in dataset. Default "tp", inherited from ERA5 (variable name in the NetCDF).
window_size	Rolling window in DAYS. Default 5.
mask_path	Path to the mask NetCDF file, or NULL (no masking).
threshold	Numeric threshold for the mask. Default 0.8.

Value

Same structure as calculate_maximum_precipitation_over_window().

---

calculate_percentiles *Compute temperature percentile thresholds for each day of year*

Description

Uses a rolling window over the reference period followed by a group by day-of-year percentile.

Usage

```
calculate_percentiles(dataset, n, reference_period, part_of_day)
```

Arguments

dataset	Full sub-daily t2m dataset (list).
n	Percentile (e.g. 90 or 10).
reference_period	Character vector c("YYYY-MM-DD", "YYYY-MM-DD").
part_of_day	"day" or "night".

Value

A named numeric vector of length 366 (day-of-year 1–366).

`calculate_percentiles_terra` *Compute temperature percentile thresholds for each day of year (terra version)*

Description

Equivalent of `calculate_percentiles()`. **More experimental than the other functions in this file**: it relies on `terra::roll()` for the rolling-window quantile, whose exact argument names have changed across terra versions – check `?terra::roll` against your installed version (`packageVersion("terra")`) and validate on a small subset before running on the full 40-year series.

Usage

```
calculate_percentiles_terra(
  r,
  n,
  reference_period,
  part_of_day,
  filename = "",
  cores = 1L
)
```

Arguments

`r`	A `terra::SpatRaster`, hourly resolution, with `terra::time()` set (replaces the `dataset` argument of the base-R version).
`n`	Percentile (e.g. 90 or 10).
`reference_period`	Character vector `c("YYYY-MM-DD", "YYYY-MM-DD")`.
`part_of_day`	`"day"` or `"night"`.
`filename`	Optional output path for the final thresholds.
`cores`	How many spatial tiles to process IN PARALLEL (a ceiling, further capped by `.safe_cores_terra()` based on RAM available and a single tile's size). Default `1` (sequential – tiles are still used for memory safety even at `cores = 1`, just processed one after another instead of concurrently; see performance note above). **Note:** the NUMBER of tiles is decided independently of `cores`, purely to keep a single `terra::roll()` call's memory footprint bounded (see `.calculate_percentiles_terra_tiled()`) – `cores` only controls how many of those (already memory-safe) tiles run at once.

Details

Performance AND memory note: the rolling-window quantile (`terra::roll()`) is by far the most expensive step of the whole `*_terra` pipeline – it invokes a custom R callback (`stats::quantile()`) at every timestep of `reference_period` (filtered to day/night hours), for every cell. **Crucially, a**

single `terra::roll()` **call over a whole country's grid can crash R from memory pressure alone, with NO parallelism involved** – observed empirically even with `cores = 1` on a 16GB machine, for a France-wide grid with 13 years of reference data filtered to daytime hours (`window_size = 80`). Because of this, the raster is now ALWAYS split into small spatial tiles sized to a conservative, fixed memory target (see `target_tile_gb` in `.calculate_percentiles_terra_tiled()`), regardless of `cores` – `cores` only controls how many of those already-memory-safe tiles run concurrently (default 1: one at a time). Unlike `terra::tapp()`, `terra::roll()` has no built-in `cores` argument in the terra version this package was tested against (1.7.65) – there is no GPU path either (neither terra/GDAL nor base R ship a GPU rolling-quantile primitive).

Value

A `terra::SpatRaster` with 366 layers (day-of-year 1-366).

`calculate_period_wind_exceedance_frequency`
Calculate monthly wind exceedance frequency

Description

Calculate monthly wind exceedance frequency

Usage

```
calculate_period_wind_exceedance_frequency(
  u10_dataset,
  v10_dataset,
  reference_period
)
```

Arguments

`u10_dataset`	List for `u10`.
`v10_dataset`	List for `v10`.
`reference_period`	Character vector `c("start", "end")`.

Value

A list with `data` [lon x lat x periods] and `time`.

calculate_period_wind_exceedance_frequency_terra *Calculate monthly wind exceedance frequency (terra version)*

Description

Drop-in, memory-safe replacement for `calculate_period_wind_exceedance_frequency()`. Only the initial `wind_power_terra()` call differs from the base-R version; the threshold, crossing, and monthly-aggregation steps reuse the exact same helpers, since they already operate on small, daily-resolution data.

Usage

```
calculate_period_wind_exceedance_frequency_terra(
  u10_r,
  v10_r,
  reference_period,
  mask_path = NULL,
  threshold = 0.8
)
```

Arguments

`u10_r`, `v10_r`	`terra::SpatRaster`, hourly resolution, non masque.
`reference_period`	Character vector `c("start", "end")`.
`mask_path`	Path to the mask NetCDF file, or `NULL` (no masking).
`threshold`	Numeric threshold for the mask. Default `0.8`.

Value

Same as `calculate_period_wind_exceedance_frequency()`.

calculate_rolling_sum *Calculate the rolling sum of a climate variable*

Description

Computes a rolling (sliding-window) sum along the time dimension of a 3-D climate array.

Usage

```
calculate_rolling_sum(dataset, var_name, window_size)
```

Arguments

dataset	List returned by load_component().
var_name	Variable name (for documentation; the array in dataset$data is used directly).
window_size	Integer number of time steps for the rolling window.

Value

A list with the same structure as dataset but with data replaced by the rolling sum (leading incomplete windows are NA).

---

cds_set_key *Set your CDS Personal Access Token*

Description

Wrapper around wf_set_key for the new CDS API (post-October 2024). The token is stored securely in the system keyring and never needs to appear in your scripts again.

Usage

```
cds_set_key(token)
```

Arguments

token	Character. Your CDS Personal Access Token (UUID format).

Details

To obtain your token:

1. Create a free account at https://cds.climate.copernicus.eu.
2. Go to *your profile* → *Personal Access Token*.
3. Copy the token (a UUID string) and pass it to this function once.

Value

Invisibly NULL. The token is saved in the system keyring.

Examples

```
## Not run:
cds_set_key("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")

## End(Not run)
```

component *Base Climate Component Helpers*

Description

Low-level helpers shared by all ACI component functions.

component_terra *Terra-based Loading and Hourly-to-Daily Reduction*

Description

Memory-safe alternatives to `load_netcdf()`/`apply_mask()` and to the hourly-resolution steps of the temperature, precipitation and wind pipelines (`temp_extremum()`, `calculate_percentiles()`, the daily resampling inside `wind_power()`).

Why this file exists: for a whole country at hourly resolution over 40+ years, `ncdf4::ncvar_get()` materialises the full `[lon x lat x time]` array in RAM in one shot (tens of GB). `terra` reads NetCDF lazily (via GDAL) and processes computations block-by-block, writing results to disk instead of accumulating them in memory.

Only the hourly-scale steps are ported here. Once data has been reduced to DAILY resolution (via `resample_daily_terra()`, `temp_extremum_terra()`, ...), the resulting object is small enough (~40 years x 365 days) to convert back to a plain array with `.spatraster_to_list()` and hand off, unchanged, to the rest of the existing (already tested) pipeline – `resample_monthly()`, `standardize_metric()`, `.compute_aci_grid()`, etc.

days_above_wind_thresholds
Calculate days with wind power above the 90th-percentile threshold

Description

Calculate days with wind power above the 90th-percentile threshold

Usage

```
days_above_wind_thresholds(u10_dataset, v10_dataset, reference_period)
```

Arguments

`u10_dataset`	List for `u10`.
`v10_dataset`	List for `v10`.
`reference_period`	Character vector `c("start", "end")`.

Value

A list with binary `data` [lon x lat x days] and `time`.

`download` *Download ERA5 and Country Mask Data*

Description

Functions to download ERA5 climate variables and country mask from the Copernicus Climate Data Store (CDS) via the **ecmwfr** package (v2.0+, new CDS API — Personal Access Token only, no UID required).

`download_era5` *Download an ERA5 variable from the Copernicus CDS*

Description

Downloads hourly ERA5 single-level data for one variable and one or several years, for a given geographical bounding box. Files are downloaded year by year and then optionally merged into a single NetCDF using the **ncdf4** package.

Usage

```
download_era5(
  variable = c("t2m", "tp", "u10", "v10"),
  years,
  area = c(51.5, -5.5, 41, 10),
  country_abbrev = NULL,
  dest_dir = NULL,
  merge = TRUE,
  overwrite = FALSE
)
```

Arguments

`variable`	Character. Short name of the ERA5 variable: one of `"t2m"`, `"tp"`, `"u10"`, `"v10"`.
`years`	Integer vector. Years to download (e.g. `1961:1990`).
`area`	Numeric vector of length 4: `c(north, west, south, east)` in decimal degrees. Default is metropolitan France: `c(51.5, -5.5, 41.0, 10.0)`.
`country_abbrev`	Character. Three-letter ISO 3166-1 alpha-3 country code (e.g. `"FRA"`). Used to build the default output path under `tempdir()` when `dest_dir` is not supplied. Ignored when `dest_dir` is supplied explicitly. One of `country_abbrev` or `dest_dir` must be provided.

dest_dir	Character. Directory where files will be saved. If NULL (default), resolves to a sub-directory of tempdir() built from country_abbrev. Created if it does not exist. Pass your own path for a location that persists across sessions.
merge	Logical. If TRUE (default) and more than one year is requested, the individual yearly files are merged into a single file named <variable>_<first_year>_<last_year>.nc using nc_open / ncvar_get and a time-concatenation loop (no external tool required).
overwrite	Logical. If FALSE (default) skip years whose output file already exists.

Details

The CDS Personal Access Token must be stored beforehand with cds_set_key.

Value

Invisibly, the path to the final NetCDF file (merged if merge = TRUE, otherwise the directory containing yearly files).

Examples

```
## Not run:
# Store your token once
cds_set_key("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")

# Automatic path: <tempdir()>/xaci_era5/FRA/tp_1961_1990.nc
download_era5(
  variable       = "tp",
  years          = 1961:1990,
  area           = c(51.5, -5.5, 41.0, 10.0),
  country_abbrev = "FRA"
)

# Or explicit path:
download_era5(
  variable = "tp",
  years    = 1961:1990,
  area     = c(51.5, -5.5, 41.0, 10.0),
  dest_dir = "my/custom/path"
)

## End(Not run)
```

---

download_era5_all *Download all ERA5 variables required by the xaci package*

Description

Convenience wrapper that calls download_era5 four times (one per variable: t2m, tp, u10, v10).

Usage

```
download_era5_all(
  years,
  area = c(51.5, -5.5, 41, 10),
  country_abbrev = NULL,
  dest_dir = NULL,
  merge = TRUE,
  overwrite = FALSE
)
```

Arguments

years	Integer vector of years to download.
area	Bounding box c(north, west, south, east). Default: metropolitan France.
country_abbrev	Character. ISO-3 country code (e.g. "FRA"). Used to build the default output path. One of country_abbrev or dest_dir must be provided.
dest_dir	Character. Output directory. If NULL (default), resolves to a sub-directory of tempdir(). Pass your own path for a location that persists across sessions.
merge	Logical. Merge yearly files. Default TRUE.
overwrite	Logical. Overwrite existing files. Default FALSE.

Value

Invisibly, a named character vector of output paths (one per variable).

Examples

```
## Not run:
cds_set_key("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")

# Produces (by default): <tempdir()>/xaci_era5/FRA/t2m_1961_1990.nc, tp_..., u10_..., v10_...
download_era5_all(
  years          = 1961:1990,
  area           = c(51.5, -5.5, 41.0, 10.0),
  country_abbrev = "FRA"
)

## End(Not run)
```

---

download_mask *Download the country mask from the Copernicus CDS*

Description

Downloads the ERA5 land-sea mask for the given bounding box, then rescales it to [0, 1] and saves it as mask_<country_abbrev>.nc in dest_dir. The mask variable is renamed to country so it is directly compatible with apply_mask.

Usage

```
download_mask(
  country_abbrev = "FRA",
  area = c(51.5, -5.5, 41, 10),
  dest_dir = NULL,
  overwrite = FALSE
)
```

Arguments

country_abbrev	Three-letter ISO 3166-1 alpha-3 country code (e.g. "FRA").
area	Numeric vector c(north, west, south, east). Default: metropolitan France.
dest_dir	Character. Directory for the output file. Default NULL, which resolves to a subdirectory of tempdir().
overwrite	Logical. Overwrite if the file already exists. Default FALSE.

Value

Invisibly, the path to the mask NetCDF file.

Examples

```
## Not run:
cds_set_key("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")

# Produces (by default): <tempdir()>/xaci_era5/FRA/mask_FRA.nc
download_mask(
  country_abbrev = "FRA",
  area      = c(51.5, -5.5, 41.0, 10.0)
)

## End(Not run)
```

---

drought *Drought Component of the ACI*

Description

Computes the drought component of the Actuarial Climate Index via maximum consecutive dry days (CDD).

drought_component *Calculate the drought component of the ACI*

Description

Computes standardised consecutive dry days.

Usage

```
drought_component(
  precipitation_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

precipitation_data_path	Path to the precipitation NetCDF file.
country_abbrev	Three-letter ISO country code (e.g. "FRA"). Used to build save_dir/load_dir defaults and to construct the admin mask when admin_level is not NULL.
reference_period	Character vector c("start", "end").
study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "drought_1980_2020.rds").
mask_path	Path to the country mask NetCDF file, or NULL.
area	Logical. If TRUE return national spatial mean as a named numeric vector. Ignored when admin_level or admin_mask is not NULL. Default FALSE.
admin_level	Integer or NULL. If not NULL, builds the admin mask internally and returns one column per unit. Ignored when admin_mask is supplied directly.
admin_mask	Output of build_admin_mask(), or NULL. When supplied, takes precedence over admin_level (no rebuild).
crs_metric	EPSG code for the metric CRS used when building the admin mask. Default 4326.

computed_components
: Logical. If TRUE, reloads a previously saved .rds file from load_dir instead of recomputing. Default FALSE.

save
: Logical. Default FALSE.

save_dir
: Character. Default NULL, which resolves to a sub-directory of tempdir().

load_dir
: Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Named numeric vector (area = TRUE), standardised list (area = FALSE), or data.frame per admin unit.

---

drought_component_terra

Calculate the drought component of the ACI (terra version)

Description

Drop-in, memory-safe replacement for drought_component().

Usage

```
drought_component_terra(
  precipitation_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

precipitation_data_path
: Path to the precipitation NetCDF file.

country_abbrev
: Three-letter ISO country code (e.g. "FRA"). Used to build save_dir/load_dir defaults and to construct the admin mask when admin_level is not NULL.

reference_period
: Character vector c("start", "end").

study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "drought_1980_2020.rds").
mask_path	Path to the country mask NetCDF file, or NULL.
area	Logical. If TRUE return national spatial mean as a named numeric vector. Ignored when admin_level or admin_mask is not NULL. Default FALSE.
admin_level	Integer or NULL. If not NULL, builds the admin mask internally and returns one column per unit. Ignored when admin_mask is supplied directly.
admin_mask	Output of build_admin_mask(), or NULL. When supplied, takes precedence over admin_level (no rebuild).
crs_metric	EPSG code for the metric CRS used when building the admin mask. Default 4326.
computed_components	Logical. If TRUE, reloads a previously saved .rds file from load_dir instead of recomputing. Default FALSE.
save	Logical. Default FALSE.
save_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Same as drought_component().

`drought_interpolate` *Linearly interpolate annual CDD to monthly resolution*

Description

For each pair of consecutive years k and $k+1$ the monthly CDD for month $m \in 1..12$ of year k is:

$$CDD_m = \frac{12-m}{12} \cdot CDD_k + \frac{m}{12} \cdot CDD_{k+1}$$

The last year repeats its own value for all 12 months.

Usage

```
drought_interpolate(cdd_annual)
```

Arguments

cdd_annual	List returned by max_consecutive_dry_days().

Value

A list with data [lon × lat × months] and monthly time.

drought_terra *Terra-based Drought Component*

Description

Memory-safe equivalents of `max_consecutive_dry_days()` and `drought_component()`. Only the hourly-to-daily reduction runs through terra; `drought_interpolate()` and the consecutive-dry-day scan already operate on small, annual/daily-resolution data, so they are reused as-is via the shared helper `.max_consecutive_dry_days_from_daily()` (defined in `drought.R`) – no logic is duplicated.

load_component *Load a NetCDF dataset and optionally apply a country mask*

Description

This function is a constructor. It reads a NetCDF variable and, if `mask_path` is provided, zeroes out (sets to `NA`) all grid cells where the mask value is below `threshold`.

Usage

```
load_component(data_path, var_name, mask_path = NULL, threshold = 0.8)
```

Arguments

`data_path`	Path to the NetCDF data file.
`var_name`	Name of the primary variable to load.
`mask_path`	Path to the mask NetCDF file, or `NULL` (default).
`threshold`	Numeric threshold for the mask. Default `0.8`.

Value

A list with elements `data` (3-D array [lon × lat × time]), `lon`, `lat`, `time` (POSIXct vector), and `var_name`.

load_component_terra *Load a NetCDF variable and optionally apply a country mask (terra version)*

Description

Drop-in, memory-safe replacement for `load_component()` intended for full-resolution, grid-cell-level ("area = FALSE") workflows over long historical periods.

Usage

```
load_component_terra(
  data_path,
  var_name,
  mask_path = NULL,
  threshold = 0.8,
  chunk_size = 20000
)
```

Arguments

`data_path`	Path to the NetCDF file.
`var_name`	Name of the variable to extract.
`mask_path`	Path to the mask NetCDF file, or `NULL` (default).
`threshold`	Numeric threshold for the mask. Default `0.8`.
`chunk_size`	Passe a `apply_mask_terra()` (voir sa doc) pour le cas ou `nlyr(r)` depasse la limite de 65535 couches.

Value

A `terra::SpatRaster`.

load_netcdf_terra *Lazily load a NetCDF variable as a terra SpatRaster*

Description

Equivalent of `load_netcdf()`, but never materialises pixel values in RAM: only metadata (dimensions, CRS, time) is read up front.

Usage

```
load_netcdf_terra(path, var_name)
```

Arguments

path	Path to the NetCDF file.
var_name	Name of the variable to extract.

Value

A terra::SpatRaster with one layer per time step and terra::time() attached.

---

load_psmsl_data *Load the bundled PSMSL station metadata*

Description

Load the bundled PSMSL station metadata

Usage

```
load_psmsl_data()
```

Value

A data.frame of PSMSL tide-gauge station information.

---

max_consecutive_dry_days *Calculate maximum consecutive dry days (CDD) per year*

Description

A dry day is defined as total daily precipitation < 1 mm. The function returns the annual maximum number of consecutive dry days for each grid cell.

Usage

```
max_consecutive_dry_days(dataset)
```

Arguments

dataset	List returned by load_component() for tp (total precipitation), with at least daily time resolution.

Value

A list with data [lon × lat × years] and annual time.

max_consecutive_dry_days_terra *Calculate maximum consecutive dry days (terra version)*

Description

Calculate maximum consecutive dry days (terra version)

Usage

```
max_consecutive_dry_days_terra(r, mask_path = NULL, threshold = 0.8)
```

Arguments

r	A terra::SpatRaster, hourly (or sub-daily) resolution precipitation, with terra::time() set. **Non masque** : le masquage se fait ici, une fois les donnees reduites a la resolution journaliere (voir note ci-dessous).
mask_path	Path to the mask NetCDF file, or NULL (no masking).
threshold	Numeric threshold for the mask. Default 0.8.

Value

Same structure as max_consecutive_dry_days().

plot_aci_components *Plot the ACI component decomposition*

Description

Produces either a **bar chart** (one panel per component, type = "bar") or a **stacked area chart** (type = "stacked") showing the six standardised components that make up the ACI.

Usage

```
plot_aci_components(
  aci_df,
  type = c("stacked", "line", "bar"),
  components = NULL,
  title = "ACI Component Contributions",
  subtitle = NULL
)
```

Arguments

aci_df	A data.frame returned by calculate_aci() with area = TRUE.
type	"stacked" (default), "line" or "bar".
components	Character vector of component names to include. Default: all six c("t90","t10","precipitation","d
title	Plot title.
subtitle	Character or NULL. Optional subtitle displayed below the title. Default NULL.

Value

A ggplot object.

Examples

```
# Toy monthly ACI data.frame (see plot_aci_timeseries() for details)
n     <- 60L
dates <- format(seq(as.Date("2015-01-01"), by = "month", length.out = n),
                "%Y-%m")
aci_df <- data.frame(
  t90           = 20 + stats::rnorm(n, sd = 2),
  t10           = 5  + stats::rnorm(n, sd = 2),
  precipitation = stats::rnorm(n, sd = 1),
  drought       = stats::rnorm(n, sd = 1),
  wind          = stats::rnorm(n, sd = 1),
  sealevel      = stats::rnorm(n, sd = 1),
  row.names     = dates
)
aci_df$ACI <- with(aci_df, (t90 - t10 + precipitation + drought +
                                wind + sealevel) / 5)

plot_aci_components(aci_df, type = "bar")
plot_aci_components(aci_df, type = "stacked", components = c("t90", "t10"))
```

---

plot_aci_dashboard *Plot a summary dashboard of the ACI results*

Description

Arranges the four core plots (time series, component facets, boxplots and density) into a single patchwork figure. Requires the **patchwork** package to be installed.

Usage

```
plot_aci_dashboard(aci_df, title = "ACI Summary Dashboard")
```

Arguments

aci_df	A data.frame returned by calculate_aci() with area = TRUE.
title	Overall figure title. Default "ACI Summary Dashboard".

Value

A **patchwork** object (inherits from ggplot).

Examples

```
# Toy monthly ACI data.frame (see plot_aci_timeseries() for details)
n     <- 60L
dates <- format(seq(as.Date("2015-01-01"), by = "month", length.out = n),
                "%Y-%m")
aci_df <- data.frame(
  t90           = 20 + stats::rnorm(n, sd = 2),
  t10           = 5  + stats::rnorm(n, sd = 2),
  precipitation = stats::rnorm(n, sd = 1),
  drought       = stats::rnorm(n, sd = 1),
  wind          = stats::rnorm(n, sd = 1),
  sealevel      = stats::rnorm(n, sd = 1),
  row.names     = dates
)
aci_df$ACI <- with(aci_df, (t90 - t10 + precipitation + drought +
                              wind + sealevel) / 5)
plot_aci_dashboard(aci_df)
```

---

plot_aci_distribution *Plot the distribution of ACI components*

Description

Displays the statistical distribution of each standardised component (and optionally the global ACI) as **boxplots** (type = "boxplot"), **violin plots** (type = "violin"), or **density curves** (type = "density").

Usage

```
plot_aci_distribution(
  aci_df,
  components = NULL,
  include_aci = FALSE,
  type = c("boxplot", "violin", "density"),
  title = "ACI Component Distributions",
  subtitle = NULL
)
```

Arguments

aci_df	A data.frame returned by calculate_aci() with area = TRUE.
components	Character vector of component columns to include.
include_aci	Logical. If TRUE, adds the composite ACI column to the distribution plot alongside the components. Default FALSE.
type	"boxplot" (default), "violin", or "density".
title	Plot title.
subtitle	Character or NULL. Optional subtitle displayed below the title. Default NULL.

Value

A ggplot object.

Examples

```
# Toy monthly ACI data.frame (see plot_aci_timeseries() for details)
n     <- 60L
dates <- format(seq(as.Date("2015-01-01"), by = "month", length.out = n),
                "%Y-%m")
aci_df <- data.frame(
  t90           = 20 + stats::rnorm(n, sd = 2),
  t10           = 5  + stats::rnorm(n, sd = 2),
  precipitation = stats::rnorm(n, sd = 1),
  drought       = stats::rnorm(n, sd = 1),
  wind          = stats::rnorm(n, sd = 1),
  sealevel      = stats::rnorm(n, sd = 1),
  row.names     = dates
)
aci_df$ACI <- with(aci_df, (t90 - t10 + precipitation + drought +
                              wind + sealevel) / 5)
plot_aci_distribution(aci_df, type = "violin")
plot_aci_distribution(aci_df, type = "density")
```

---

plot_aci_map *Plot a spatial map of the ACI or one of its components*

Description

Handles three types of input, at any spatial aggregation level produced by calculate_aci() (grid-cell, admin level 1, admin level 2, ...):

- **Bare array** (e.g. grid$t90, grid$ACI): a single component array as stored in the grid-cell list returned by calculate_aci(area = FALSE). Self-describing via its attached lon/lat/time/country_abbrev attributes, so it can be passed on its own. Renders a raster map. Since ACI and its components share the exact same array structure, this works identically whether variable is a component or the ACI itself.

- **Grid-cell list** (the full output of `calculate_aci(area = FALSE)`, or of `load_component()`): `variable` selects which field of the list to map. Renders a raster map.
- **Administrative** `data.frame` (output of `calculate_aci(admin_level = N)`): renders a choropleth map. `country_abbrev`/`admin_level`/`crs_metric` are read from the data.frame's attributes when not supplied explicitly.

Usage

```
plot_aci_map(
  data,
  variable = "ACI",
  time_index = "mean",
  country_abbrev = NULL,
  admin_level = NULL,
  crs_metric = NULL,
  var_label = "Standardised anomaly",
  title = "ACI Spatial Distribution",
  palette = "RdBu",
  reverse = TRUE,
  n_breaks = 9,
  borders = TRUE
)
```

Arguments

`data`	A bare array (component or ACI, with attributes), a named list (grid-cell / raw dataset), or a `data.frame` (admin output).
`variable`	Name of the variable to map when `data` is a list. For grid-cell output use e.g. `"ACI"`, `"drought"`, `"t90"`. For admin output use `"ACI"` (columns must be named `ACI_<unit>`). Ignored when `data` is already a bare array. Default `"ACI"`.
`time_index`	Integer (1-based time slice) or `"mean"` (default) to plot the temporal average.
`country_abbrev`	Three-letter ISO code. Required for admin maps and for overlaying country/region borders on raster maps. If `NULL` and `data` carries a `country_abbrev` attribute (as set by `calculate_aci()`), that value is used automatically.
`admin_level`	Integer or `NULL`. Required for admin maps; read from `data`'s attributes when not supplied.
`crs_metric`	EPSG code used for admin choropleth projection only. If `NULL`, read from `data`'s attributes, falling back to `4326`. Ignored (with a warning if explicitly set) in raster (grid-cell) mode: the grid-cell data stays in its native EPSG:4326 (lon/lat), since reprojecting a fixed regular raster to a different CRS would distort/break it; the basemap is reprojected to 4326 to match it instead.
`var_label`	Colour-bar label. Default `"Standardised anomaly"`.
`title`	Plot title. Default `"ACI Spatial Distribution"`.
`palette`	RColorBrewer palette. Default `"RdBu"`.
`reverse`	Logical. Reverse colour scale? Default `TRUE`.

n_breaks	Number of colour breaks. Default 9.
borders	Logical. Overlay administrative borders? Default TRUE.

Value

A ggplot object.

Offline vs. online examples

The first two examples below use a small synthetic 10x8x5 (lon x lat x time) grid and run offline (borders = FALSE). Overlaying administrative borders, or plotting an admin-level choropleth, additionally requires downloading boundary polygons via geodata::gadm() and therefore a working internet connection; the corresponding examples are wrapped in dontrun and not run automatically by R CMD check.

Examples

```
lon <- seq(-5, 8, length.out = 10)
lat <- seq(42, 51, length.out = 8)
arr <- array(stats::rnorm(10 * 8 * 5), dim = c(10, 8, 5))
arr <- structure(arr, lon = lon, lat = lat,
                  time = seq_len(5), country_abbrev = "FRA")

# Bare array, self-describing thanks to its attributes
plot_aci_map(arr, time_index = "mean", borders = FALSE)

# Equivalent, via the parent list and `variable`
grid <- list(t90 = arr, lon = lon, lat = lat, time = seq_len(5))
plot_aci_map(grid, variable = "t90", time_index = 1, borders = FALSE)

## Not run:
plot_aci_map(arr, time_index = "mean", country_abbrev = "FRA")

admin <- calculate_aci(
  temperature_data_path   = "t2m_1960-2020.nc",
  precipitation_data_path = "tp_1960-2020.nc",
  wind_u10_data_path      = "u10_1960-2020.nc",
  wind_v10_data_path      = "v10_1960-2020.nc",
  country_abbrev          = "FRA",
  mask_data_path          = "mask_FRA.nc",
  study_period            = c("1961-01-01", "2020-12-31"),
  reference_period        = c("1961-01-01", "1990-12-31"),
  granularity             = "year",
  area                    = FALSE,
  admin_level             = 1
)
plot_aci_map(admin, variable = "ACI", time_index = 1)

## End(Not run)
```

plot_aci_map_mean *Plot the temporal mean of an ACI variable over a sub-period*

Description

Convenience wrapper around plot_aci_map that computes the mean over a user-defined date range rather than a single time index.

Usage

```
plot_aci_map_mean(
  data,
  variable = "ACI",
  period = NULL,
  country_abbrev = NULL,
  admin_level = NULL,
  crs_metric = NULL,
  var_label = "Standardised anomaly",
  title = NULL,
  palette = "RdBu",
  reverse = TRUE,
  n_breaks = 9,
  borders = TRUE
)
```

Arguments

data	Bare array, grid-cell list, or admin data.frame.
variable	Variable to map. Default "ACI". Ignored when data is already a bare array.
period	Character vector c("start", "end") in the same format as the time field of data (e.g. c("2000-01", "2010-12") for monthly). If NULL (default), uses the full time range.
country_abbrev	Three-letter ISO code.
admin_level	Integer or NULL.
crs_metric	EPSG code. Default NULL (read from attributes, falling back to 4326).
var_label	Colour-bar label.
title	Plot title.
palette	RColorBrewer palette. Default "RdBu".
reverse	Logical. Default TRUE.
n_breaks	Integer. Default 9.
borders	Logical. Default TRUE.

Value

A ggplot object.

`plot_aci_timeseries` *Plot the ACI global time series*

Description

Draws a line chart of the ACI values returned by `calculate_aci`, with an optional smoothed trend and a horizontal zero reference line.

Usage

```
plot_aci_timeseries(
  aci_df,
  smooth = TRUE,
  span = 0.2,
  title = "Actuarial Climate Index (ACI)",
  colour = "#1F77B4",
  fill_area = TRUE
)
```

Arguments

`aci_df`	A `data.frame` returned by `calculate_aci()` with `area = TRUE`. Must contain at least the column `ACI` and have month-end dates as row names.
`smooth`	Logical. If `TRUE` (default), a LOESS smoothing curve is overlaid in red.
`span`	Numeric span for the LOESS smoother. Default `0.2`.
`title`	Plot title. Default `"Actuarial Climate Index (ACI)"`.
`colour`	Line colour for the raw ACI series. Default `"#1F77B4"` (blue).
`fill_area`	Logical. If `TRUE` (default), fills the area between the ACI curve and zero.

Value

A `ggplot` object.

Examples

```
# Toy monthly ACI data.frame (not from calculate_aci(), just for
# illustration), with the columns and "YYYY-MM" row names that
# calculate_aci(area = TRUE) would return.
n     <- 60L
dates <- format(seq(as.Date("2015-01-01"), by = "month", length.out = n),
                "%Y-%m")
aci_df <- data.frame(
  t90           = 20 + stats::rnorm(n, sd = 2),
  t10           = 5  + stats::rnorm(n, sd = 2),
  precipitation = stats::rnorm(n, sd = 1),
  drought       = stats::rnorm(n, sd = 1),
  wind          = stats::rnorm(n, sd = 1),
```

```
  sealevel      = stats::rnorm(n, sd = 1),
  row.names     = dates
)
aci_df$ACI <- with(aci_df, (t90 - t10 + precipitation + drought +
                              wind + sealevel) / 5)
plot_aci_timeseries(aci_df)
```

`precipitation` *Precipitation Component of the ACI*

Description

Computes the precipitation component of the Actuarial Climate Index.

`precipitation_component`
Calculate the precipitation component of the ACI

Description

Computes the standardised anomaly of maximum monthly precipitation over a rolling 5-day window.

Usage

```
precipitation_component(
  precipitation_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  var_name = "tp",
  window_size = 5L,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

precipitation_data_path	Path to the precipitation NetCDF file.
country_abbrev	Three-letter ISO country code.
reference_period	Character vector c("start", "end").
study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "precipitation_1980_2020.rds").
mask_path	Path to the country mask NetCDF file, or NULL.
var_name	Variable name in the NetCDF. Default "tp".
window_size	Rolling window in days. Default 5.
area	Logical. Default FALSE.
admin_level	Integer or NULL.
admin_mask	Output of build_admin_mask(), or NULL.
crs_metric	EPSG code. Default 4326.
computed_components	Logical. Default FALSE.
save	Logical. Default FALSE.
save_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Named numeric vector, standardised list, or data.frame per admin unit.

---

precipitation_component_terra

Calculate the precipitation component of the ACI (terra version)

Description

Drop-in, memory-safe replacement for precipitation_component().

Usage

```
precipitation_component_terra(
  precipitation_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  var_name = "tp",
  window_size = 5L,
```

```
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

precipitation_data_path	Path to the precipitation NetCDF file.
country_abbrev	Three-letter ISO country code.
reference_period	Character vector c("start", "end").
study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "precipitation_1980_2020.rds").
mask_path	Path to the country mask NetCDF file, or NULL.
var_name	Variable name in the NetCDF. Default "tp".
window_size	Rolling window in days. Default 5.
area	Logical. Default FALSE.
admin_level	Integer or NULL.
admin_mask	Output of build_admin_mask(), or NULL.
crs_metric	EPSG code. Default 4326.
computed_components	Logical. Default FALSE.
save	Logical. Default FALSE.
save_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Same as precipitation_component().

---

precipitation_terra *Terra-based Precipitation Component*

Description

Memory-safe equivalents of calculate_maximum_precipitation_over_window() and precipitation_component(). Only the hourly-to-daily reduction runs through terra; the rolling-sum and monthly-max logic already operates on small, daily-resolution data, reused as-is via the shared helper .max_precipitation_from_daily() (defined in precipitation.R) – no logic is duplicated.

reduce_dataarray_to_dataframe

Reduce a standardised spatial metric to a data frame

Description

If admin_mask is NULL, averages over all non-NA cells (national mean). Otherwise computes a weighted mean per administrative unit using surface-fraction weights.

Usage

```
reduce_dataarray_to_dataframe(metric, column_name = "value", admin_mask = NULL)
```

Arguments

metric	List with fields data [lon x lat x time] and time.
column_name	Character. Column name prefix. Default "value".
admin_mask	Output of build_admin_mask(), or NULL (default) for national mean.

Value

A data.frame with one column per administrative unit (or one column for the national mean), indexed by month-start dates.

request_sealevel_data *Download PSMSL tide-gauge data for a country*

Description

Reads the bundled psmsl_data.csv to identify stations for the given country abbreviation, downloads the corresponding data files from the PSMSL website, and stores them locally.

Usage

```
request_sealevel_data(country_abbrev, dest_dir = NULL)
```

Arguments

country_abbrev	Three-letter ISO country code (e.g. "FRA").
dest_dir	Destination directory. If NULL (default), resolves to a sub-directory of tempdir().

Value

Invisibly, the path to the destination directory.

resample_daily_terra *Resample a SpatRaster to daily resolution (terra version)*

Description

Equivalent of `resample_daily()`. Groups layers by calendar day and applies `fun`, writing the result to disk chunk-by-chunk via `terra::tapp()` rather than holding the full input in RAM.

Usage

```
resample_daily_terra(r, fun = "mean", filename = "")
```

Arguments

`r`	A `terra::SpatRaster` with `terra::time()` set (sub-daily time steps expected).
`fun`	Aggregation function name understood by `terra::tapp()` (e.g. `"mean"`, `"sum"`, `"max"`, `"min"`).
`filename`	Optional path to write the result directly to disk (highly recommended for large jobs). Default `""` (terra decides, using a temp file if the result doesn't fit in memory).

Value

A `terra::SpatRaster` with one layer per day.

sealevel *Sea Level Component of the ACI*

Description

Processes PSMSL tide-gauge data and computes the sea-level component of the Actuarial Climate Index.

sealevel_clean_data *Replace PSMSL sentinel values (-99999) with NA*

Description

Replace PSMSL sentinel values (-99999) with NA

Usage

```
sealevel_clean_data(df)
```

Arguments

df	data.frame of sea-level measurements.

Value

Cleaned data.frame.

sealevel_component *Calculate the sea-level component of the ACI*

Description

Calculate the sea-level component of the ACI

Usage

```
sealevel_component(
  country_abbrev,
  study_period,
  reference_period,
  mask_path = NULL,
  area = TRUE,
  max_dist_km = 500,
  sealevel_dir = NULL,
  admin_level = NULL,
  admin_assignment = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

country_abbrev	Three-letter country code (e.g. "FRA").
study_period	Character vector c("start", "end"). Required to filter and/or download PSMSL tide-gauge data (unlike ERA5-based components which read the full NetCDF file and filter in downstream steps).
reference_period	Character vector c("start", "end").
mask_path	Path to the country mask NetCDF. Used to extract the ERA5 grid when area = FALSE.
area	Logical. Mirrors the area argument of all other component functions: if FALSE (default), interpolates the standardised anomalies onto the ERA5 grid and returns a list with a [lon x lat x t] array — consistent with the grid-cell output of temperature_component(), precipitation_component(), etc. If TRUE, returns a data.frame of aggregated anomalies (national or per admin unit, depending on admin_level).
max_dist_km	Numeric. Maximum distance (km) for IDW interpolation. Only used when area = FALSE. Default 500.
sealevel_dir	Character or NULL. Path to the directory containing PSMSL .txt files. Named sealevel_dir for consistency with the sealevel_dir argument of calculate_aci(). If NULL or the directory does not exist, data are downloaded automatically.
admin_level	Integer or NULL. If not NULL, returns one column per admin unit. Ignored when admin_assignment is supplied.
admin_assignment	Output of assign_sealevel_to_admin(), or NULL. When supplied, takes precedence over admin_level.
crs_metric	EPSG code used when building admin_assignment internally. Default 4326.
computed_components	Logical. If TRUE, reloads a previously saved .rds file from load_dir. Default FALSE.
save	Logical. If TRUE, saves the processed result to save_dir. Default FALSE.
save_dir	Character. Directory for saving results. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Directory from which to reload a cached result when computed_components = TRUE. Default NULL, which resolves to a sub-directory of tempdir().

Value

If area = FALSE: a list with a [lon x lat x t] array, lon, lat, time — same structure as other grid-cell components. If area = TRUE: a data.frame of station anomalies (national or per admin unit).

sealevel_compute_monthly_stats

Compute monthly reference statistics for sea-level data

Description

Compute monthly reference statistics for sea-level data

Usage

```
sealevel_compute_monthly_stats(df, reference_period, stats)
```

Arguments

df	Clean data.frame (row names = "YYYY-MM-DD").
reference_period	Character vector c("start", "end").
stats	"means" or "std".

Value

A numeric matrix, 12 rows (calendar months "1"-"12" as row names) x one column per station (station names as returned by colnames(df)). Each station is standardised against its own monthly reference statistics, independently of the other stations – consistent with how the other ACI components (ERA5 grid cells, administrative units) are each standardised against their own reference, not a value pooled across the whole spatial domain.

sealevel_correct_date_format

Correct PSMSL float date format to Date objects

Description

PSMSL encodes dates as YYYY.fraction where the fraction identifies the month.

Usage

```
sealevel_correct_date_format(df)
```

Arguments

df	data.frame with numeric row names (PSMSL date format).

Value

The same data.frame with Date row names ("YYYY-MM-01"), rows with unrecognised dates removed.

sealevel_load_data *Load sea-level txt files from a directory*

Description

Load sea-level txt files from a directory

Usage

```
sealevel_load_data(directory)
```

Arguments

directory	Path to the directory containing PSMSL `.txt` files.

Value

A `data.frame` with one column per station and a numeric date as row names.

sealevel_process *Full sea-level processing pipeline*

Description

Full sea-level processing pipeline

Usage

```
sealevel_process(directory, study_period, reference_period)
```

Arguments

directory	Path to the directory with PSMSL `.txt` files.
study_period	Character vector `c("start", "end")`.
reference_period	Character vector `c("start", "end")`.

Value

A named list with:

`data` Standardised `data.frame` of anomalies `[time x stations]`, row names `"YYYY-MM-DD"`.

`coords` A `data.frame` with columns `station_id`, `lon`, `lat`, one row per station present in `data`.

sealevel_standardize_data *Standardise sea-level data over the study period*

Description

Standardise sea-level data over the study period

Usage

```
sealevel_standardize_data(df, monthly_means, monthly_std_devs, study_period)
```

Arguments

df	Clean data.frame.
monthly_means	Numeric matrix, 12 rows (months "1"-"12") x one column per station, as returned by sealevel_compute_monthly_stats(..., stats = "means").
monthly_std_devs	Same shape as monthly_means, for stats = "std".
study_period	Character vector c("start", "end").

Value

A data.frame of standardised anomalies for the study period, with rows containing all-NA removed. Each station's values are standardised against its own monthly reference (see sealevel_compute_monthly_stats() not a value pooled across stations.

standardize_metric *Standardise a monthly metric relative to a reference period*

Description

For each calendar month, computes the mean and standard deviation over the reference period, then returns (x - mean) / sd. Optionally averages over the spatial dimensions first.

Usage

```
standardize_metric(metric, reference_period, area = FALSE)
```

Arguments

metric	A list with fields data [lon x lat x time] or [time], time, and optionally lon/lat.
reference_period	Character vector of length 2: start and end dates ("YYYY-MM-DD").
area	Logical. If TRUE the spatial mean is computed before standardising and a named numeric vector is returned. Default FALSE.

Value

If `area = TRUE`: a named numeric vector (names = dates). Otherwise: a list with the same structure as `metric` but standardised values.

`temperature` *Temperature Component of the ACI*

Description

Computes the temperature component of the Actuarial Climate Index.

`temperature_component` *Calculate the full temperature component of the ACI*

Description

Combines day and night half-day components (equal weighting), then standardises relative to the reference period.

Usage

```
temperature_component(
  temperature_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  percentile = 90,
  extremum = "max",
  above_thresholds = TRUE,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

temperature_data_path	Path to the hourly t2m NetCDF file.
country_abbrev	Three-letter ISO country code.
reference_period	Character vector c("start", "end"). Climatological baseline used for standardisation.
study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "temperature_t90_1980_2020.rds"), so that distinct runs over different study windows don't collide or get mixed up.
mask_path	Path to the country mask NetCDF file.
percentile	Percentile for the threshold. Default 90.
extremum	"max" (hot) or "min" (cold).
above_thresholds	Logical. Default TRUE.
area	Logical. Default FALSE.
admin_level	Integer or NULL.
admin_mask	Output of build_admin_mask(), or NULL.
crs_metric	EPSG code. Default 4326.
computed_components	Logical. Default FALSE.
save	Logical. Default FALSE.
save_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Named numeric vector, standardised list, or data.frame per admin unit.

---

temperature_component_terra

Compute the temperature ACI component (terra version)

Description

Drop-in, memory-safe replacement for temperature_component(), intended for full-resolution, grid-cell-level ("area = FALSE") workflows over long historical periods (40+ years hourly).

Usage

```
temperature_component_terra(
  temperature_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  percentile = 90,
  extremum = "max",
  above_thresholds = TRUE,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  cores = 1L,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

temperature_data_path	Path to the hourly t2m NetCDF file.
country_abbrev	Three-letter ISO country code.
reference_period	Character vector c("start", "end"). Climatological baseline used for standardisation.
study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "temperature_t90_1980_2020.rds"), so that distinct runs over different study windows don't collide or get mixed up.
mask_path	Path to the country mask NetCDF file.
percentile	Percentile for the threshold. Default 90.
extremum	"max" (hot) or "min" (cold).
above_thresholds	Logical. Default TRUE.
area	Logical. Default FALSE.
admin_level	Integer or NULL.
admin_mask	Output of build_admin_mask(), or NULL.
crs_metric	EPSG code. Default 4326.
cores	Passed through to calculate_halfday_component_terra() / calculate_percentiles_terra() (see its performance note on terra::roll() having no built-in parallelism). Default 1.

computed_components
: Logical. Default FALSE.

save
: Logical. Default FALSE.

save_dir
: Character. Default NULL, which resolves to a sub-directory of tempdir().

load_dir
: Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Same as temperature_component(): a standardized metric list, or a data frame if admin_mask/admin_level is provided.

---

temperature_terra *Terra-based Temperature Component*

Description

Memory-safe equivalents of calculate_halfday_component() and temperature_component(), for full-resolution, grid-cell-level workflows over long historical periods (see R/component_terra.R for the underlying loading/reduction primitives).

Only the hourly-scale steps (temp_extremum_terra(), calculate_percentiles_terra()) run through terra. Once reduced to daily/monthly resolution, the exact same .crossing_frequency() helper used by the base-R pipeline takes over – no logic is duplicated.

---

temp_extremum *Compute daily temperature extremum (min or max) for day or night hours*

Description

Compute daily temperature extremum (min or max) for day or night hours

Usage

```
temp_extremum(dataset, extremum, period)
```

Arguments

dataset
: List returned by load_component() for the t2m variable (sub-daily, hourly data expected).

extremum
: "min" or "max".

period
: "day" (hours 6–21) or "night" (hours 0–5 and 22–23).

Value

A list with data [lon × lat × days] and daily time.

temp_extremum_terra *Compute daily temperature extremum (min or max) for day or night hours (terra version)*

Description

Equivalent of `temp_extremum()`. Filters layers by hour-of-day (a cheap, lazy operation on a SpatRaster – no data is read), then reduces to daily resolution via `resample_daily_terra()`.

Usage

```
temp_extremum_terra(r, extremum, period, filename = "")
```

Arguments

`r`	A `terra::SpatRaster`, hourly resolution, with `terra::time()` set.
`extremum`	`"min"` or `"max"`.
`period`	`"day"` (hours 6-21) or `"night"` (hours 0-5 and 22-23).
`filename`	Optional output path (see `resample_daily_terra()`).

Value

A `terra::SpatRaster` with one layer per day.

utils *Utility Functions for ACI Package*

Description

Helper functions to manipulate and merge climate data.

visualization *Visualization Functions for the ACI Package*

Description

A collection of ggplot2-based plotting functions to visualize the Actuarial Climate Index (ACI) and its components. Four families of plots are provided:

1. **Time series** of the global ACI (`plot_aci_timeseries`)
2. **Component decomposition** stacked / faceted chart (`plot_aci_components`)
3. **Geographic maps** from a NetCDF-derived array (`plot_aci_map`)
4. **Distribution plots** - boxplots and density ridges (`plot_aci_distribution`)

wind — *Wind Component of the ACI*

Description

Computes the wind component of the Actuarial Climate Index.

wind_component — *Calculate the wind component of the ACI*

Description

Calculate the wind component of the ACI

Usage

```
wind_component(
  wind_u10_data_path,
  wind_v10_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

wind_u10_data_path	Path to the u10 NetCDF file.
wind_v10_data_path	Path to the v10 NetCDF file.
country_abbrev	Three-letter ISO country code.
reference_period	Character vector `c("start", "end")`. Climatological baseline used for standardisation.
study_period	Character vector `c("start", "end")`. Full period covered by the study; used to name the cached grid-cell-level `.rds` file (e.g. `"wind_1980_2020.rds"`).

mask_path	Path to the country mask NetCDF file, or NULL.
area	Logical. Default FALSE.
admin_level	Integer or NULL.
admin_mask	Output of build_admin_mask(), or NULL.
crs_metric	EPSG code. Default 4326.
computed_components	Logical. Default FALSE.
save	Logical. Default FALSE.
save_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Named numeric vector, standardised list, or data.frame per admin unit.

`wind_component_terra` *Calculate the wind component of the ACI (terra version)*

Description

Drop-in, memory-safe replacement for wind_component(), intended for full-resolution, grid-cell-level ("area = FALSE") workflows over long historical periods (40+ years hourly).

Usage

```
wind_component_terra(
  wind_u10_data_path,
  wind_v10_data_path,
  country_abbrev,
  reference_period,
  study_period,
  mask_path = NULL,
  area = FALSE,
  admin_level = NULL,
  admin_mask = NULL,
  crs_metric = 4326,
  computed_components = FALSE,
  save = FALSE,
  save_dir = NULL,
  load_dir = NULL
)
```

Arguments

wind_u10_data_path	Path to the u10 NetCDF file.
wind_v10_data_path	Path to the v10 NetCDF file.
country_abbrev	Three-letter ISO country code.
reference_period	Character vector c("start", "end"). Climatological baseline used for standardisation.
study_period	Character vector c("start", "end"). Full period covered by the study; used to name the cached grid-cell-level .rds file (e.g. "wind_1980_2020.rds").
mask_path	Path to the country mask NetCDF file, or NULL.
area	Logical. Default FALSE.
admin_level	Integer or NULL.
admin_mask	Output of build_admin_mask(), or NULL.
crs_metric	EPSG code. Default 4326.
computed_components	Logical. Default FALSE.
save	Logical. Default FALSE.
save_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().
load_dir	Character. Default NULL, which resolves to a sub-directory of tempdir().

Value

Same as wind_component().

---

wind_power *Calculate daily wind power from u10 and v10 components*

Description

Wind speed = sqrt(u10^2 + v10^2), wind power = 0.5 * rho * ws^3.

Usage

```
wind_power(u10_dataset, v10_dataset, reference_period = NULL)
```

Arguments

u10_dataset	List returned by load_component() for u10.
v10_dataset	List returned by load_component() for v10.
reference_period	Optional character vector c("start","end"). If provided, only the reference sub-period is returned.

Value

A list with data [lon x lat x days] (wind power in W/m²) and daily time.

---

wind_power_terra *Calculate daily wind power from u10 and v10 components (terra version)*

Description

Terra-based equivalent of wind_power(): computes daily mean u/v via resample_daily_terra() (block-by-block, never loading the full hourly cube in RAM), then wind speed/power arithmetic directly on the (now daily, small) SpatRasters before converting back to the package's list format.

Usage

```
wind_power_terra(
  u10_r,
  v10_r,
  reference_period = NULL,
  mask_path = NULL,
  threshold = 0.8
)
```

Arguments

u10_r, v10_r	terra::SpatRaster, hourly resolution, **non masque** – le masquage se fait ici, une fois wp_r reduit a la resolution journaliere (voir note ci-dessous), et une seule fois plutot que separement sur u10_r et v10_r.
reference_period	Optional character vector c("start","end"). If provided, only the reference sub-period is returned.
mask_path	Path to the mask NetCDF file, or NULL (no masking).
threshold	Numeric threshold for the mask. Default 0.8.

Value

Same structure as wind_power(): list(data, time, lon, lat).

wind_terra *Terra-based Wind Component*

Description

Memory-safe equivalents of `wind_power()` and `wind_component()`. Only `wind_power_terra()` touches raw hourly data (via terra); `wind_thresholds()`'s internal logic, threshold-crossing, and monthly aggregation already operate on DAILY resolution data (small even for 40+ years), so they are reused as-is via the shared helpers `.wind_thresholds_from_wp()`, `.days_above_from_wp()`, `.monthly_frequency_from_binary()` defined in `wind.R` – no logic is duplicated.

wind_thresholds *Calculate wind power thresholds (90th percentile per day-of-year)*

Description

Calculate wind power thresholds (90th percentile per day-of-year)

Usage

```
wind_thresholds(u10_dataset, v10_dataset, reference_period)
```

Arguments

`u10_dataset`	List for `u10` variable.
`v10_dataset`	List for `v10` variable.
`reference_period`	Character vector `c("start", "end")`.

Value

A list with `data` [lon x lat x all_days] giving the threshold for each day of the full series, and `time`.

Index