Package: climasus4r (via r-universe)

July 10, 2026

Title Integrated Analysis Toolkit for Health, Climate, and Environmental Data in Brazil: Reproducible Workflows for Climate-Health Research

Version 1.0.0

Description climasus4r is a comprehensive R toolkit for integrated analysis of health, climate, and environmental data in Brazil. The package automates the entire data pipeline: (1) importing and cleaning health data from the Brazilian Unified Health System (SUS) across six systems (SIM, SIH, SINAN, SIA, CNES, SINASC); (2) integrating climate data (INMET, ERA5), air quality (AQI), environmental data (MapBiomas), and socioeconomic indicators (IBGE); (3) performing flexible spatiotemporal aggregation at multiple scales; (4) constructing reproducible analytical pipelines (RAPs) that ensure transparency and replicability. Designed for researchers studying climate-health interactions, environmental vulnerability, and adaptation strategies in the Brazilian context. Supports multilingual output (Portuguese, Spanish, English) and parallel processing for large datasets.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr (>= 1.0.0), stringi (>= 1.7.0), cli (>= 3.0.0), lubridate, microdatasus, magrittr, tidyr, fs, digest, stats, rlang, stringr, purrr, data.table, glue, read.dbc, furrr, future, duckdb, tibble, generics, scales

Remotes rfsaldanha/microdatasus

Suggests arrow, nanoparquet, RColorBrewer, ggrepel, DBI, slider, splines, sf, censobr, geobr, geocodebr, sfarrow, duckspatial, knitr, rmarkdown, ggplot2, future.apply, xgboost, zoo, MASS, dlnm, rstudioapi, DT, ggplot2, ggsci, viridisLite, htmltools,

patchwork, plotly, htmlwidgets, httr, httr2, parallelly, jsonlite, gt, leaflet, mvmeta, survival, terra, exactextractr, ncdf4, readxl, spdep (>= 1.3), spatialreg (>= 1.3), CARBayes (>= 6.1), SpatialEpi (>= 1.2), INLA, targets (>= 1.0.0), tarchetypes, shiny (>= 1.7.0), yaml, renv, quarto, strucchange

Additional_repositories https://inla.r-inla-download.org/R/stable

URL https://bymaxanjos.r-universe.dev/climasus4r, https://github.com/ByMaxAnjos/climasus4r

BugReports https://github.com/ByMaxAnjos/climasus4r/issues

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev xz-utils

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

Date/Publication 2026-07-10 16:26:21 UTC

RemoteUrl https://github.com/ByMaxAnjos/climasus4r

RemoteRef HEAD

RemoteSha 6115248fc85ed4a8d5c5e873af7657fa1da1c261

Contents

`[.climasus_df` *Subsetting method for climasus_df*

Description

Subsetting method for climasus_df

Usage

```
## S3 method for class 'climasus_df'
x[i, j, drop = FALSE]
```

Arguments

`x`	A climasus_df object
`i`	Row indices
`j`	Column indices
`drop`	Logical

`[[.climasus_df` *Column extraction for climasus_df*

Description

Column extraction for climasus_df

Usage

```
## S3 method for class 'climasus_df'
x[[i]]
```

Arguments

`x`	A climasus_df object
`i`	Column index or name

$<-.climasus_df *Column assignment for climasus_df*

Description

Column assignment for climasus_df

Usage

```
## S3 replacement method for class 'climasus_df'
x$name <- value
```

Arguments

x	A climasus_df object
name	Column name
value	New column values

as.data.frame.climasus_df *Coerce climasus_df to plain data.frame (strips metadata)*

Description

Coerce climasus_df to plain data.frame (strips metadata)

Usage

```
## S3 method for class 'climasus_df'
as.data.frame(x, ...)
```

Arguments

x	A climasus_df object
...	Additional arguments (ignored)

`coef.climasus_dlnm` *Extract coefficients from a climasus_dlnm model*

Description

Extract coefficients from a climasus_dlnm model

Usage

```
## S3 method for class 'climasus_dlnm'
coef(object, ...)
```

Arguments

`object`	A `climasus_dlnm` object.
`...`	Passed to `stats::coef()`.

Value

Named numeric vector of GLM coefficients.

`coef.climasus_metaregression`
Extract pooled coefficients from a climasus_metaregression object

---

Description

Extract pooled coefficients from a climasus_metaregression object

Usage

```
## S3 method for class 'climasus_metaregression'
coef(object, ...)
```

Arguments

`object`	A `climasus_metaregression` object.
`...`	Passed to `stats::coef()`.

Value

Named numeric vector of meta-regression coefficients.

coef.climasus_pool *Extract pooled coefficients from a climasus_pool object*

Description

Extract pooled coefficients from a climasus_pool object

Usage

```
## S3 method for class 'climasus_pool'
coef(object, ...)
```

Arguments

object	A climasus_pool object.
...	Passed to stats::coef().

Value

Named numeric vector of pooled cross-basis coefficients.

coef.climasus_vi *Extract VI scores as a named numeric vector*

Description

Extract VI scores as a named numeric vector

Usage

```
## S3 method for class 'climasus_vi'
coef(object, ...)
```

Arguments

object	A climasus_vi object.
...	Unused.

Value

Named numeric vector (city → vi_score).

cw_active_days *Return daily rows where at least one coldwave method is active*

Description

Return daily rows where at least one coldwave method is active

Usage

```
cw_active_days(cw_result)
```

Arguments

cw_result	A climasus_cw list returned by sus_climate_compute_coldwaves().

Value

A filtered tibble from cw_result$daily.

cw_count_by_year *Count coldwave events by year, station, and method*

Description

Count coldwave events by year, station, and method

Usage

```
cw_count_by_year(cw_result)
```

Arguments

cw_result	A climasus_cw list returned by sus_climate_compute_coldwaves().

Value

A tibble with columns year, station_code, station_name, method, n_events, total_days_cw, mean_duration.

cw_get_events — *Extract coldwave events table from a climasus_cw result*

Description

Extract coldwave events table from a climasus_cw result

Usage

```
cw_get_events(cw_result, method_filter = NULL)
```

Arguments

`cw_result`	A `climasus_cw` list returned by `sus_climate_compute_coldwaves()`.
`method_filter`	Character vector. If supplied, keep only these methods.

Value

A tibble of coldwave events.

hw_active_days — *Return daily rows where at least one heatwave method is active*

Description

Return daily rows where at least one heatwave method is active

Usage

```
hw_active_days(hw_result)
```

Arguments

`hw_result`	A `climasus_hw` list returned by `sus_climate_compute_heatwaves()`.

Value

A filtered tibble from `hw_result$daily`.

hw_count_by_year *Count heatwave events by year, station, and method*

Description

Count heatwave events by year, station, and method

Usage

```
hw_count_by_year(hw_result)
```

Arguments

hw_result	A climasus_hw list returned by sus_climate_compute_heatwaves().

Value

A tibble with columns year, station_code, station_name, method, n_events, total_days_hw, mean_duration.

hw_get_events *Extract heatwave events table from a climasus_hw result*

Description

Extract heatwave events table from a climasus_hw result

Usage

```
hw_get_events(hw_result, method_filter = NULL)
```

Arguments

hw_result	A climasus_hw list returned by sus_climate_compute_heatwaves().
method_filter	Character vector. If supplied, keep only these methods.

Value

A tibble of heatwave events.

predict.climasus_ml *Generate predictions from a climasus_ml model on new data*

Description

Applies the fitted XGBoost model to a new data frame. The new data must contain all feature columns used during training (see `object$meta$feature_cols`).

Usage

```
## S3 method for class 'climasus_ml'
predict(object, newdata, ...)
```

Arguments

`object`	A `climasus_ml` object from `sus_mod_ml()`.
`newdata`	A data frame with the same feature columns as the training data. Missing features trigger an informative error.
`...`	Unused.

Value

A numeric vector of predictions (in the natural scale of the objective: counts for Poisson, values for squarederror, probabilities for binary logistic).

print.climasus_af *Print a climasus_af object*

Description

Print a climasus_af object

Usage

```
## S3 method for class 'climasus_af'
print(x, ...)
```

Arguments

`x`	A `climasus_af` object.
`...`	Unused.

Value

`x` invisibly.

`print.climasus_burden`
Print a climasus_burden object

Description

Print a climasus_burden object

Usage

```
## S3 method for class 'climasus_burden'
print(x, ...)
```

Arguments

`x`	A `climasus_burden` object from `sus_mod_burden()`.
`...`	Unused.

Value

`x` invisibly.

`print.climasus_casecrossover`
Print a climasus_casecrossover object

Description

Print a climasus_casecrossover object

Usage

```
## S3 method for class 'climasus_casecrossover'
print(x, ...)
```

Arguments

`x`	A `climasus_casecrossover` object from `sus_mod_casecrossover()`.
`...`	Unused.

Value

`x` invisibly.

print.climasus_df *Print method for climasus_df*

Description

Print method for climasus_df

Usage

```
## S3 method for class 'climasus_df'
print(x, n = 10, ...)
```

Arguments

x	A climasus_df object
n	Integer. Number of rows to print
...	Additional arguments passed to print

print.climasus_dlnm *Print a climasus_dlnm model*

Description

Print a climasus_dlnm model

Usage

```
## S3 method for class 'climasus_dlnm'
print(x, ...)
```

Arguments

x	A `climasus_dlnm` object from `sus_mod_dlnm()`.
...	Unused.

Value

`x` invisibly.

`print.climasus_excess` *Print a climasus_excess object*

Description

Print a climasus_excess object

Usage

```
## S3 method for class 'climasus_excess'
print(x, ...)
```

Arguments

`x`	A `climasus_excess` object from `sus_mod_excess()`.
`...`	Unused.

Value

`x` invisibly.

`print.climasus_its` *Print a climasus_its object*

Description

Print a climasus_its object

Usage

```
## S3 method for class 'climasus_its'
print(x, ...)
```

Arguments

`x`	A `climasus_its` object from `sus_mod_its()`.
`...`	Unused.

Value

`x` invisibly.

print.climasus_metaregression *Print a climasus_metaregression object*

Description

Print a climasus_metaregression object

Usage

```
## S3 method for class 'climasus_metaregression'
print(x, ...)
```

Arguments

`x`	A `climasus_metaregression` object from `sus_mod_metaregression()`.
`...`	Unused.

Value

`x` invisibly.

print.climasus_ml *Print a climasus_ml object*

Description

Print a climasus_ml object

Usage

```
## S3 method for class 'climasus_ml'
print(x, n_imp = 10L, ...)
```

Arguments

`x`	A `climasus_ml` object from `sus_mod_ml()`.
`n_imp`	Integer. Number of top features to show. Default `10L`.
`...`	Unused.

Value

`x` invisibly.

`print.climasus_pool` *Print a climasus_pool object*

Description

Print a climasus_pool object

Usage

```
## S3 method for class 'climasus_pool'
print(x, ...)
```

Arguments

`x`	A `climasus_pool` object from `sus_mod_pool()`.
`...`	Unused.

Value

`x` invisibly.

`print.climasus_sensitivity`
Print a climasus_sensitivity object

Description

Print a climasus_sensitivity object

Usage

```
## S3 method for class 'climasus_sensitivity'
print(x, ...)
```

Arguments

`x`	A `climasus_sensitivity` object from `sus_mod_sensitivity()`.
`...`	Unused.

Value

`x` invisibly.

print.climasus_spacetime_bayes

Print method for climasus_spacetime_bayes objects

Description

Displays a formatted summary of the spatiotemporal Bayesian model fit, including model specification, goodness-of-fit statistics, posterior fixed-effect estimates with 95% credible intervals, and a summary of relative risks across the space-time grid.

Usage

```
## S3 method for class 'climasus_spacetime_bayes'
print(x, n_fixed = 10L, ...)
```

Arguments

`x`	A `climasus_spacetime_bayes` object from `sus_mod_spacetime_bayes()`.
`n_fixed`	Integer. Maximum number of fixed-effect rows to display. Default `10L`.
`...`	Additional arguments (ignored).

Value

Invisibly returns `x`.

print.climasus_spacetime_exceedance

Print method for climasus_spacetime_exceedance objects

Description

Displays a formatted summary of exceedance probabilities including the number of municipality-time cells that exceed each RR threshold with posterior probability greater than 0.80.

Usage

```
## S3 method for class 'climasus_spacetime_exceedance'
print(x, n_rows = 10L, ...)
```

Arguments

`x`	A `climasus_spacetime_exceedance` object from `sus_mod_spacetime_exceedance()`.
`n_rows`	Integer. Maximum number of exceedance table rows to display. Default `10L`.
`...`	Additional arguments (ignored).

Value

Invisibly returns `x`.

`print.climasus_spacetime_pred`
Print method for climasus_spacetime_pred objects

Description

Displays a formatted summary of space-time predictions, including the number of predicted observations, the forward horizon, municipality and time-index range, and a preview of the predictions table.

Usage

```
## S3 method for class 'climasus_spacetime_pred'
print(x, n_rows = 6L, ...)
```

Arguments

`x`	A `climasus_spacetime_pred` object from `sus_mod_spacetime_predict()`.
`n_rows`	Integer. Maximum number of prediction rows to display in the preview. Default `6L`.
`...`	Additional arguments (ignored).

Value

Invisibly returns `x`.

`print.climasus_spatial_bayes`
Print method for climasus_spatial_bayes objects

Description

Displays a formatted summary of the Bayesian spatial model fit, including DIC, effective MCMC sample size (or NA for INLA-based models), model specification, and the top fixed effects with posterior means and 95\ credible intervals.

Usage

```
## S3 method for class 'climasus_spatial_bayes'
print(x, n_fixed = 10L, ...)
```

Arguments

x	A climasus_spatial_bayes object from sus_mod_spatial_bayes().
n_fixed	Integer. Maximum number of fixed-effect rows to display. Default 10L.
...	Additional arguments (ignored).

Value

Invisibly returns x.

---

print.climasus_spatial_moran

Print method for climasus_spatial_moran objects

Description

Displays a formatted summary of the global Moran's I test and the LISA cluster counts.

Usage

```
## S3 method for class 'climasus_spatial_moran'
print(x, ...)
```

Arguments

x	A climasus_spatial_moran object from sus_mod_spatial_moran().
...	Additional arguments (ignored).

Value

Invisibly returns x.

---

print.climasus_spatial_reg

Print method for climasus_spatial_reg objects

Description

Displays a formatted summary of the spatial regression model, including the model type, spatial parameters (rho / lambda), coefficient table, AIC comparison with OLS, and the Moran test result on residuals.

Usage

```
## S3 method for class 'climasus_spatial_reg'
print(x, n_coef = 10L, ...)
```

Arguments

x	A climasus_spatial_reg object from sus_mod_spatial_reg().
n_coef	Integer. Maximum number of coefficient rows to display. Default 10.
...	Additional arguments (ignored).

Value

Invisibly returns x.

---

print.climasus_spatial_scan *Print a climasus_spatial_scan object*

Description

Print a climasus_spatial_scan object

Usage

```
## S3 method for class 'climasus_spatial_scan'
print(x, ...)
```

Arguments

x	A climasus_spatial_scan object returned by sus_mod_spatial_scan().
...	Unused.

Value

x invisibly.

---

print.climasus_swot *Print a climasus_swot object*

Description

Print a climasus_swot object

Usage

```
## S3 method for class 'climasus_swot'
print(x, ...)
```

Arguments

x	A climasus_swot object from sus_mod_swot().
...	Ignored.

print.climasus_ts_quality *Print a climasus_ts_quality object*

Description

Print a climasus_ts_quality object

Usage

```
## S3 method for class 'climasus_ts_quality'
print(x, ...)
```

Arguments

`x`	A `climasus_ts_quality` object.
`...`	Unused.

Value

`x` invisibly.

print.climasus_vi *Print a climasus_vi object*

Description

Print a climasus_vi object

Usage

```
## S3 method for class 'climasus_vi'
print(x, ...)
```

Arguments

`x`	A `climasus_vi` object from `sus_mod_vulnerability_index()`.
`...`	Unused.

Value

`x` invisibly.

print.climasus_weights *Print method for climasus_weights*

Description

Displays a compact summary of the spatial weights object including the number of regions, islands, weight style, and neighbour statistics.

Usage

```
## S3 method for class 'climasus_weights'
print(x, ...)
```

Arguments

x	A climasus_weights object.
...	Ignored.

Value

x invisibly.

rbind.climasus_df *Row binding for climasus_df objects*

Description

Row binding for climasus_df objects

Usage

```
## S3 method for class 'climasus_df'
rbind(..., deparse.level = 1)
```

Arguments

...	climasus_df objects to combine
deparse.level	Integer

summary.climasus_af *Summarise a climasus_af object*

Description

Summarise a climasus_af object

Usage

```
## S3 method for class 'climasus_af'
summary(object, ...)
```

Arguments

object	A climasus_af object.
...	Unused.

Value

object invisibly.

summary.climasus_burden *Summarise a climasus_burden object*

Description

Summarise a climasus_burden object

Usage

```
## S3 method for class 'climasus_burden'
summary(object, ...)
```

Arguments

object	A climasus_burden object from sus_mod_burden().
...	Unused.

Value

object invisibly.

`summary.climasus_casecrossover`
Summarise a climasus_casecrossover object

Description

Summarise a climasus_casecrossover object

Usage

```
## S3 method for class 'climasus_casecrossover'
summary(object, ...)
```

Arguments

`object`	A `climasus_casecrossover` object from `sus_mod_casecrossover()`.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_dlnm`
Summarise a climasus_dlnm model

Description

Summarise a climasus_dlnm model

Usage

```
## S3 method for class 'climasus_dlnm'
summary(object, ...)
```

Arguments

`object`	A `climasus_dlnm` object from `sus_mod_dlnm()`.
`...`	Unused.

Value

`object` invisibly.

summary.climasus_excess *Summarise a climasus_excess object*

Description

Summarise a climasus_excess object

Usage

```
## S3 method for class 'climasus_excess'
summary(object, ...)
```

Arguments

`object`	A `climasus_excess` object from `sus_mod_excess()`.
`...`	Unused.

Value

`object` invisibly.

summary.climasus_its *Summarise a climasus_its object*

Description

Summarise a climasus_its object

Usage

```
## S3 method for class 'climasus_its'
summary(object, ...)
```

Arguments

`object`	A `climasus_its` object from `sus_mod_its()`.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_metaregression` *Summarise a climasus_metaregression object*

Description

Summarise a climasus_metaregression object

Usage

```
## S3 method for class 'climasus_metaregression'
summary(object, ...)
```

Arguments

`object`	A `climasus_metaregression` object from `sus_mod_metaregression()`.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_ml` *Summarise a climasus_ml object*

Description

Summarise a climasus_ml object

Usage

```
## S3 method for class 'climasus_ml'
summary(object, ...)
```

Arguments

`object`	A `climasus_ml` object from `sus_mod_ml()`.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_pool` *Summarise a climasus_pool object*

Description

Summarise a climasus_pool object

Usage

```
## S3 method for class 'climasus_pool'
summary(object, ...)
```

Arguments

`object`	A `climasus_pool` object from `sus_mod_pool()`.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_sensitivity` *Summarise a climasus_sensitivity object*

Description

Summarise a climasus_sensitivity object

Usage

```
## S3 method for class 'climasus_sensitivity'
summary(object, ...)
```

Arguments

`object`	A `climasus_sensitivity` object from `sus_mod_sensitivity()`.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_spacetime_bayes`
Summary method for climasus_spacetime_bayes objects

Description

Alias for `print.climasus_spacetime_bayes()`.

Usage

```
## S3 method for class 'climasus_spacetime_bayes'
summary(object, ...)
```

Arguments

`object`	A `climasus_spacetime_bayes` object.
`...`	Additional arguments passed to `print.climasus_spacetime_bayes()`.

Value

Invisibly returns `object`.

`summary.climasus_spacetime_exceedance`
Summary method for climasus_spacetime_exceedance objects

Description

Alias for `print.climasus_spacetime_exceedance()`.

Usage

```
## S3 method for class 'climasus_spacetime_exceedance'
summary(object, ...)
```

Arguments

`object`	A `climasus_spacetime_exceedance` object.
`...`	Additional arguments passed to `print.climasus_spacetime_exceedance()`.

Value

Invisibly returns `object`.

`summary.climasus_spacetime_pred` *Summary method for climasus_spacetime_pred objects*

Description

Alias for `print.climasus_spacetime_pred()`.

Usage

```
## S3 method for class 'climasus_spacetime_pred'
summary(object, ...)
```

Arguments

`object`	A `climasus_spacetime_pred` object.
`...`	Additional arguments passed to `print.climasus_spacetime_pred()`.

Value

Invisibly returns `object`.

`summary.climasus_spatial_bayes` *Summary method for climasus_spatial_bayes objects*

Description

Alias for `print.climasus_spatial_bayes()`.

Usage

```
## S3 method for class 'climasus_spatial_bayes'
summary(object, ...)
```

Arguments

`object`	A `climasus_spatial_bayes` object.
`...`	Additional arguments passed to `print.climasus_spatial_bayes()`.

Value

Invisibly returns `object`.

`summary.climasus_spatial_moran` *Summary method for climasus_spatial_moran objects*

Description

Returns the global statistics data frame and the local LISA data frame.

Usage

```
## S3 method for class 'climasus_spatial_moran'
summary(object, ...)
```

Arguments

`object`	A `climasus_spatial_moran` object.
`...`	Additional arguments (ignored).

Value

A list with elements `global` and `local`, invisibly.

`summary.climasus_spatial_reg` *Summary method for climasus_spatial_reg objects*

Description

Prints the model and returns the key tables invisibly.

Usage

```
## S3 method for class 'climasus_spatial_reg'
summary(object, ...)
```

Arguments

`object`	A `climasus_spatial_reg` object from `sus_mod_spatial_reg()`.
`...`	Additional arguments passed to `print.climasus_spatial_reg()`.

Value

A named list with elements `coefficients`, `impacts`, `moran_residuals`, `aic`, and `lm_aic`, invisibly.

summary.climasus_spatial_scan

Summarise a climasus_spatial_scan object

Description

Summarise a climasus_spatial_scan object

Usage

```
## S3 method for class 'climasus_spatial_scan'
summary(object, ...)
```

Arguments

`object`	A `climasus_spatial_scan` object.
`...`	Unused.

Value

`object` invisibly.

summary.climasus_swot

Summarise a climasus_swot object

Description

Summarise a climasus_swot object

Usage

```
## S3 method for class 'climasus_swot'
summary(object, ...)
```

Arguments

`object`	A `climasus_swot` object from `sus_mod_swot()`.
`...`	Ignored.

`summary.climasus_ts_quality` *Summarise a climasus_ts_quality object*

Description

Summarise a climasus_ts_quality object

Usage

```
## S3 method for class 'climasus_ts_quality'
summary(object, ...)
```

Arguments

`object`	A `climasus_ts_quality` object.
`...`	Unused.

Value

`object` invisibly.

`summary.climasus_vi` *Summarise a climasus_vi object*

Description

Summarise a climasus_vi object

Usage

```
## S3 method for class 'climasus_vi'
summary(object, ...)
```

Arguments

`object`	A `climasus_vi` object from `sus_mod_vulnerability_index()`.
`...`	Unused.

Value

`object` invisibly.

summary.climasus_weights *Summary method for climasus_weights*

Description

Alias for `print.climasus_weights()`.

Usage

```
## S3 method for class 'climasus_weights'
summary(object, ...)
```

Arguments

`object`	A `climasus_weights` object.
`...`	Ignored.

Value

`object` invisibly.

sus_as_arrow *Convert a climasus_df to an Arrow Table with embedded metadata*

Description

Embeds `sus_meta` as a JSON string in the Arrow schema metadata field `"sus_meta"`. This ensures the metadata survives all downstream dplyr verbs (filter, mutate, group_by, etc.) because Arrow schema metadata travels with the data.

Usage

```
sus_as_arrow(df)
```

Arguments

`df`	A `climasus_df` tibble (output of any `sus_data_*` function).

Value

An Arrow Table with `sus_meta` embedded in the schema.

Examples

```
## Not run:
arrow_df <- sus_data_standardize(df) |> sus_as_arrow()
result   <- sus_data_aggregate(arrow_df, time_unit = "month")

## End(Not run)
```

---

sus_as_duckdb *Convert a climasus_df to a DuckDB lazy tbl with embedded metadata*

Description

Copies the data to a DuckDB connection and stores `sus_meta` in a helper table named `.__sus_meta__`. This allows `sus_data_aggregate()` to recover the metadata automatically.

Usage

```
sus_as_duckdb(df, con, name = "sus_data", overwrite = TRUE)
```

Arguments

`df`	A `climasus_df` tibble.
`con`	A DuckDB `DBIConnection` (from `duckdb::dbConnect(duckdb::duckdb())`).
`name`	Character; name for the DuckDB table. Defaults to `"sus_data"`.
`overwrite`	Logical. Overwrite existing table? Default `TRUE`.

Value

A `tbl_dbi` referencing the DuckDB table.

Examples

```
## Not run:
con      <- duckdb::dbConnect(duckdb::duckdb())
duck_tbl <- sus_data_standardize(df) |> sus_as_duckdb(con)
result   <- sus_data_aggregate(duck_tbl, time_unit = "month")

## End(Not run)
```

sus_cache_clear *Clear the Climasus4r cache*

Description

Clear the Climasus4r cache

Usage

```
sus_cache_clear(
  cache_dir = "~/.climasus4r_cache",
  older_than_days = NULL,
  verbose = TRUE
)
```

Arguments

cache_dir	Character. Cache directory to clear. Default is "~/.climasus4r_cache".
older_than_days	Numeric. Only clear files older than this many days. NULL clears all.
verbose	Logical. If TRUE, prints information about what was cleared.

Examples

```
## Not run:
# Clear all cache
sus_cache_clear()

# Clear only files older than 90 days
sus_cache_clear(older_than_days = 90)

## End(Not run)
```

sus_cache_info *Get cache information*

Description

Get cache information

Usage

```
sus_cache_info(cache_dir = "~/.climasus4r_cache", verbose = TRUE)
```

Arguments

cache_dir	Character. Cache directory to inspect. Default is "~/.climasus4r_cache".
verbose	Logical. If TRUE, prints detailed information.

Value

A list with cache statistics.

Examples

```
## Not run:
cache_info <- sus_cache_info()

## End(Not run)
```

---

sus_census_join *Add Census Socioeconomic variables to Health Data*

Description

Enriches health data with socioeconomic indicators from Brazilian Census using the `censobr` package. Provides aggregation of microdata to municipality level. Supports multiple census datasets (population, households, families, mortality, emigration) with intelligent caching and optimized performance for large datasets using Arrow/Parquet format.

Usage

```
sus_census_join(
  df,
  dataset = "population",
  tracts_dataset = "Basico",
  year = 2010,
  census_vars = NULL,
  aggregation_fun = "sum",
  join_muni_col = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/census",
  lang = "pt",
  translate_columns = TRUE,
  standardize_values = TRUE,
  verbose = TRUE
)
```

Arguments

df	A data.frame or sf object containing health data. Typically the output from sus_spatial_join() function.
dataset	Character string specifying the census dataset to use. Options: • "population" - Population microdata (default) • "households" - Household microdata • "families" - Family microdata • "mortality" - Mortality microdata • "emigration" - Emigration microdata • "tracts" - Census tract aggregate data (setores censitarios). Use tracts_dataset to select which tract table to download.
tracts_dataset	Character string. Relevant only when dataset = "tracts". Selects which censobr tract table to download. Options: "Basico" (default), "Domicilio01", "Domicilio02", "DomicilioRenda", "Pessoa01" through "Pessoa13", "ResponsavelRenda", "ResponsavelRendaSexo". See censobr::read_tracts() for full variable listings per table.
year	Integer specifying census year. Options: 2010 (default) or 2000. Note: Dataset availability varies by year.
census_vars	Character vector specifying census variables to add. Use sus_census_select() to select available variables. If NULL, returns all available variables (not recommended for large datasets).
aggregation_fun	Character. Method to aggregate microdata to municipality level: • "sum" - Sums the selected variables by municipality (e.g., for total population). • "mean" - Averages the selected variables (e.g., for income). • "median", "min", "max", "sd", "q25", "q75", "q95", and "q99".
join_muni_col	Character string specifying the column in df containing the **6 or 7-digit IBGE municipality code**. If NULL, detects common SUS patterns (e.g., code_muni).
use_cache	Logical. If TRUE (default), uses censobr's caching system to store downloaded data locally for faster subsequent access.
cache_dir	Character string specifying cache directory path. Defaults to "~/.climasus4r_cache/census".T function automatically calls censobr::set_censobr_cache_dir(cache_dir) to ensure consistency across the ecosystem.
lang	Character string specifying language for messages. Options: "pt" (Portuguese, default), "en" (English), "es" (Spanish).
translate_columns	Logical. If TRUE, translates column names. Default is TRUE.
standardize_values	Logical. If TRUE, standardizes categorical values. Default is TRUE.
verbose	Logical. If TRUE (default), prints progress messages and download progress bar.

Details

Integration with censobr package: This function is a wrapper around censobr's dataset-specific functions (`read_population()`, `read_households()`, etc.), providing seamless integration with the climasus4r ecosystem.

Geographic Columns: The function automatically inherits and matches geographic columns following the geobr/censobr standard:

- `code_muni` - 7-digit municipality code
- `code_state` - 2-digit state code
- `abbrev_state` - State abbreviation (e.g., "AM")
- `name_state` - State name
- `code_region` - Region code
- `name_region` - Region name
- `code_weighting` - Weighting area code

Automatic Column Detection: If `join_muni_col = NULL`, the function automatically detects the appropriate code municipality column based on common SUS patterns:

- **Municipality**: `residence_municipality_code`, `municipality_code`, `codigo_municipio`, `CODMUNRES`, etc.

Performance Optimization: The function uses Arrow/Parquet format for efficient larger-than-memory dataset handling:

- Downloads data only once (cached locally)
- Filters municipalities BEFORE loading to RAM
- Uses `dplyr::collect()` only after filtering
- Uses `sfarrow` pak for spatial filtering

Spatial Data Support: If `df` is an `sf` object from `sus_spatial_join()`, geometries are preserved in the output.

Value

Returns the input `data.frame` or `sf` object with additional columns. If the input is an sf object, the spatial geometry and CRS are strictly preserved through a join.

References

Pereira, Rafael H. M.; Barbosa, Rogerio J. (2023) censobr: Download Data from Brazil's Population Census. R package version v0.4.0, https://CRAN.R-project.org/package=censobr. DOI: 10.32614/CRAN.package.censobr.

Examples

```
## Not run:
library(climasus4r)

# Prepare spatial health data
sf_sim <- sus_data_import(uf = "SP", year = 2023, system = "SIM-DO") %>%
  sus_data_standardize(lang = "pt") %>%
  sus_spatial_join(level = "munic", lang = "pt")

# Add census population data
sf_enriched <- sus_census_join(
  df = sf_sim,
  dataset = "population",
  census_vars = c("V0001", "V0002"),
  year = 2010,
  lang = "pt"
)

## End(Not run)
```

---

sus_census_select *Interactive Census Variables Explorer*

Description

Opens an interactive HTML interface to explore Brazilian Census variables and copy codes for use with sus_census_join().

Usage

```
sus_census_select(
  dataset = "all",
  year = 2010,
  lang = "pt",
  output = "browser",
  verbose = TRUE
)
```

Arguments

`dataset` Character string specifying the census dataset. Options:

- `"all"` - All datasets (default)
- `"population"` - Population microdata
- `"households"` - Household microdata
- `"families"` - Family microdata
- `"mortality"` - Mortality microdata
- `"emigration"` - Emigration microdata

- `"tracts"` - Census tract aggregate data

`year` Integer specifying census year. Options: `2010` (default) or `2000`.

`lang` Character string specifying language. Options: `"pt"` (Portuguese, default), `"en"` (English), `"es"` (Spanish).

`output` Character string specifying output format. Options:

- `"browser"` - Interactive HTML in web browser (default)
- `"console"` - Simple list in R console
- `"codes"` - Returns only variable codes as character vector

`verbose` Logical. If `TRUE` (default), prints informative messages.

Details

This function helps users discover which census variables are available. The interactive interface allows easy copying of variable codes for use in `sus_census_join(vars = ...)`.

Value

Depending on `output`:

- `"browser"`: Opens HTML interface, returns invisible data.frame
- `"console"`: Prints summary, returns invisible data.frame
- `"codes"`: Returns character vector of variable codes

Examples

```
## Not run:
# Open interactive explorer for all datasets
sus_census_select()

# Explore only population variables
sus_census_select(dataset = "population")

# Get variable codes for programmatic use
vars <- sus_census_select(
  dataset = "population",
  output = "codes",
  lang = "en"
)

# Use in sus_census_join
data <- sus_census_join(
  df = my_data,
  vars = vars,
  dataset = "population",
  year = 2010
)

## End(Not run)
```

sus_chat *Launch the climasus4r AI Assistant*

Description

Launch the climasus4r AI Assistant

Usage

```
sus_chat()
```

sus_climate_aggregate *Integration of Climate and Health Data*

Description

`sus_climate_aggregate()` aggregates meteorological data to DATASUS health data using epidemiologically rigorous temporal strategies. The function links each health record to the nearest climate station (by Euclidean distance) and applies the requested temporal window.

Usage

```
sus_climate_aggregate(
  health_data,
  climate_data,
  climate_var = "all",
  time_unit = "day",
  temporal_strategy = "exact",
  climate_region = "auto",
  window_days = NULL,
  lag_days = NULL,
  offset_days = NULL,
  temp_base = NULL,
  gdd_temp_var = "tair_dry_bulb_c",
  min_obs = 0.7,
  threshold_value = NULL,
  threshold_direction = "above",
  weights = NULL,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`health_data` A `climasus_df` object produced by `sus_spatial_join()`. Must contain columns `date` (Date), `code_muni` (character), and geometry column `geom`.

`climate_data` A `climasus_df` object produced by `sus_climate_fill_inmet()`. Must contain `date` (Date or POSIXct), `station_code`, `latitude`, `longitude`, and climate variables.

`climate_var` Character vector with climate variables to aggregate. Use `"all"` (default) to include all available variables.

Available variables are grouped as follows:

Atmospheric pressure:

- `"patm_mb"`, `"patm_max_mb"`, `"patm_min_mb"`

Air temperature:

- `"tair_dry_bulb_c"`, `"tair_max_c"`, `"tair_min_c"`

Dew point temperature:

- `"dew_tmean_c"`, `"dew_tmax_c"`, `"dew_tmin_c"`

Relative humidity:

- `"rh_mean_porc"`, `"rh_max_porc"`, `"rh_min_porc"`

Precipitation:

- `"rainfall_mm"`

Wind:

- `"ws_2_m_s"` (mean wind speed),
- `"ws_gust_m_s"` (wind gust),
- `"wd_degrees"` (wind direction)

Solar radiation:

- `"sr_kj_m2"`

Biometeorological indices:

- `"wbgt_c"` (Wet Bulb Globe Temperature)
- `"hi_c"` (Heat Index)
- `"thi_c"` (Temperature-Humidity Index)
- `"wcet_c"` (Wind Chill Equivalent Temperature)
- `"wct_c"` (Wind Chill Temperature)
- `"et_c"` (Effective Temperature)
- `"utci_c"` (Universal Thermal Climate Index)
- `"pet_c"` (Physiological Equivalent Temperature)

Thermal indices (degree-days):

- `"cdd_c"` (Cooling Degree Days)
- `"hdd_c"` (Heating Degree Days)
- `"gdd_c"` (Growing Degree Days)

Derived variables:

- `"diurnal_range_c"` (daily temperature range)
- `"vapor_pressure_kpa"` (saturation vapor pressure)

@section Notes:

- Not all variables may be available in `climate_data`. Use `names(climate_data)` to inspect available columns.
- When `"all"` is used, only existing variables in the dataset are selected.
- Derived and index variables depend on prior processing with `sus_climate_fill_inmet()` or feature engineering steps.

`time_unit` Temporal aggregation unit for raw climate data before join. Options: `"day"` (default), `"week"`, `"month"`, `"quarter"`, `"year"`, `"season"`. Relevant only when input data are hourly resolution.

`temporal_strategy`

Temporal matching strategy. Options:

`"exact"` Exact date match (same-day temperature for acute heat-related mortality).

`"discrete_lag"` Climate value exactly L days before event.

`"moving_window"` Mean/sum of sliding window (t-W, t). **RECOMMENDED** for cumulative exposure.

`"offset_window"` Aggregates historical interval (t-W2, t-W1), ignoring recent days.

`"distributed_lag"` Creates lag matrix 0 to L for DLNM modeling.

`"degree_days"` Calculates Growing Degree Days (GDD) for thermal stress.

`"seasonal"` Aggregates by climate season (DJF, MAM, JJA, SON).

`"threshold_exceedance"` Counts days exceeding threshold (e.g., heatwaves).

`"cold_wave_exceedance"` Counts days below threshold (e.g., cold extremes in southern regions).

`"weighted_window"` Weighted mean with decay function.

`climate_region`

Character. Climate classification for parameter adaptation. Options: `"auto"` (default, auto-detect by latitude), `"tropical"` (North, Northeast), `"subtropical"` (Center-West, Southeast), `"temperate"` (South).

`window_days` Integer. Number of days in window for `moving_window`, `offset_window`, `degree_days`, `threshold_exceedance`, `cold_wave_exceedance`, or `weighted_window`. Required for these strategies.

`lag_days` Integer vector. Specific lags for `discrete_lag` or maximum lag for `distributed_lag`. Required for these strategies.

`offset_days` Integer vector of length 2 for `offset_window`. Defines historical interval: `c(W1, W2)` aggregates from t-W2 to t-W1.

`temp_base` Numeric. Temperature base for `degree_days` calculation. If `NULL` (default), uses region-specific default: 20degC (tropical), 18degC (subtropical), 15degC (temperate). Use 11 degC for *Aedes aegypti* development, 10degC for *Plasmodium*.

`gdd_temp_var` Character. Temperature column for `degree_days`. Default: `"tair_dry_bulb_c"`.

`min_obs` Numeric (0 to 1). Minimum proportion of valid observations required within window. Default: 0.7 (70%).

`threshold_value` Numeric. Threshold for `threshold_exceedance` or `cold_wave_exceedance`. If `NULL`, uses region-specific default.

`threshold_direction` Character: `"above"` (default, > threshold) or `"below"` (< threshold). For `cold_wave_exceedance`, automatically set to `"below"`.

`weights` Numeric vector (optional) for `weighted_window`. Defines weight for each day within window, from most recent (position 1) to oldest (position W+1). If `NULL`, linear decreasing weights are generated automatically.

`lang` Language for messages: `"pt"` (Portuguese), `"en"` (English), or `"es"` (Spanish). Default: `"pt"`.

`verbose` Logical. If `TRUE`, displays progress messages. Default: `TRUE`.

Details

Temporal Strategies Epidemiological Foundations:

The choice of strategy should reflect the hypothesized biological mechanism:

- **exact**: Immediate effects (heat stroke, hemorrhagic stroke)
- **discrete_lag**: Known delayed effect (e.g., temperature 7 days before influences dengue)
- **moving_window**: Cumulative exposure without specific lag.
- **offset_window**: Defined incubation period (e.g., temperature 14-7 days before death)
- **distributed_lag**: Distributed lag analysis (DLNM); generates exposure matrix for `dlnm::crossbasis()`
- **degree_days**: Thermal threshold for vector development. GDD above temperature base accumulates over window_days.
- **seasonal**: Seasonal climate patterns for ecological or long-term studies
- **threshold_exceedance**: Counts days exceeding threshold (ideal for heatwaves)
- **cold_wave_exceedance**: Counts days below threshold (ideal for cold extremes in southern regions)
- **weighted_window**: Weighted mean modeling biological decay of exposure

Regional Climate Considerations:

The function automatically adapts to Brazil's diverse climates:

Tropical (North, Northeast): Mean temperature 24-28degC, recommended temp_base 20degC (human health), heatwave threshold Tmax > 35degC, high autocorrelation.

Subtropical (Center-West, Southeast): Mean temperature 18-26degC, recommended temp_base 18degC, heatwave threshold Tmax > 32degC, moderate autocorrelation.

Temperate (South): Mean temperature 12-24degC, recommended temp_base 15degC, heatwave threshold Tmax > 30degC, **coldwave threshold Tmin < 5degC** (critical), low autocorrelation.

Causal Inference & Look-Ahead Bias:

This function uses **retroactive windows** (t-W, t), never symmetric windows (t-W, t+W). The climate of day t+7 cannot cause a health event on day t. This design prevents look-ahead bias, a common methodological error in environmental epidemiology.

Value

A `climasus_df` (tibble) with original health data and integrated climate variables as new columns. Geometry is preserved if input is `sf`.

References

Gasparrini, A. (2011). Distributed lag linear and non-linear models in R: the package dlnm. *Journal of Statistical Software*, 43(8), 1-20.

Silveira, I. H., et al. (2023). Heat waves and mortality in the Brazilian Amazon. *International Journal of Hygiene and Environmental Health*, 248, 114109.

Marengo, J. A., et al. (2023). A cold wave of winter 2021 in central South America. *Climate Dynamics*, 61, 3-4.

Examples

```
## Not run:
# Prepare data
df_health <- sus_data_import() |>
  sus_data_clean_encoding() |>
  sus_data_standardize() |>
  sus_spatial_join(level = "munic")

df_climate <- sus_climate_inmet(years = 2023, uf = "AM") |>
  sus_climate_fill_inmet(target_var = "all", parallel = TRUE)

# Example 1: Exact match (same-day temperature)
df_exact <- sus_climate_aggregate(
  health_data = df_health,
  climate_data = df_climate,
  climate_var = "tair_dry_bulb_c",
  temporal_strategy = "exact"
)

# Example 2: Moving window (cumulative exposure, RECOMMENDED)
df_moving <- sus_climate_aggregate(
  health_data = df_health,
  climate_data = df_climate,
  temporal_strategy = "moving_window",
  window_days = 14
)

# Example 3: Regional adaptation (auto-detect)
df_regional <- sus_climate_aggregate(
  health_data = df_health,
  climate_data = df_climate,
```

```
  climate_region = "auto",
  temporal_strategy = "threshold_exceedance",
  window_days = 7
)

# Example 4: Cold waves (southern regions)
df_coldwave <- sus_climate_aggregate(
  health_data = df_health_south,
  climate_data = df_climate_south,
  climate_region = "temperate",
  temporal_strategy = "cold_wave_exceedance",
  window_days = 7,
  threshold_value = 5
)

## End(Not run)
```

`sus_climate_anomaly` *Climate Anomaly Computation vs. INMET Climatological Normals*

Description

Computes climate anomalies by comparing observed meteorological data from `sus_climate_inmet()` against 30-year climatological normals from `sus_climate_normals()`. Three anomaly types are supported: absolute (obs − normal), relative ((obs − normal) / |normal| × 100 %), and standardized z-score ((obs − normal) / _obs). The result is a `climasus_df` ready for epidemiological modelling with `sus_mod_dlnm()`, `sus_climate_compute_heatwaves()`, or `sus_mod_vulnerability_index()`.

Usage

```
sus_climate_anomaly(
  observed,
  normals,
  vars = NULL,
  method = c("absolute", "relative", "standardized", "all"),
  time_scale = c("monthly", "decadal"),
  station_col = "station_code",
  date_col = "date",
  lang = c("pt", "en", "es"),
  verbose = TRUE
)
```

Arguments

`observed`	A `climasus_df` from `sus_climate_inmet()` or `sus_climate_aggregate()`. At minimum must contain: a station identifier column (see `station_col`),

a date column (see `date_col`), and at least one climate variable column covered by the `vars` mapping.

`normals` A `climasus_df` from `sus_climate_normals()`. Must contain columns `codigo`, `mes`, `decada`, `var_code`, and `valor`. Run `sus_climate_normals_meta()` to see available `var_code` values.

`vars` Named character vector mapping **observed column names** to **normals `var_code` values**, e.g. `c(tair_max_c = "t_max", rainfall_mm = "precipitation")`. `NULL` (default) auto-detects variables using the built-in map for standard INMET columns.

`method` Character. Anomaly computation method: `"absolute"` (default), `"relative"`, `"standardized"`, or `"all"`.

`time_scale` Character. Temporal resolution for the anomaly: `"monthly"` (default) averages normals across the three decades of each month and aggregates observations by calendar month; `"decadal"` preserves the 10-day decade structure of INMET normals and is more precise for variables such as temperature extremes and precipitation.

`station_col` Character. Name of the station identifier column in `observed`. Default `"station_code"`. Must contain INMET codes matching the `codigo` column in `normals`.

`date_col` Character. Name of the date or datetime column in `observed`. Default `"date"`. Accepts `Date` or `POSIXct`.

`lang` Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_df` at stage `"climate"` and type `"anomaly"`. One row per station × year × month (or × decade when `time_scale = "decadal"`). Key columns:

`station_col` Station identifier (same as `station_col` argument).

`year` Calendar year of the observation.

`month_num` Month number (1–12).

`month_name` Portuguese month name, ASCII.

`decade_num` Decade within month (1, 2, or 3). Present only when `time_scale = "decadal"`.

`{v}_obs` Observed monthly/decadal value (mean or sum depending on variable type).

`{v}_normal` Climatological normal value.

`{v}_anomaly` Absolute anomaly: obs − normal.

`{v}_anomaly_pct` Relative anomaly (\ |normal| × 100. Only when `method  c("relative", "all")`.

`{v}_anomaly_std` Standardized anomaly (z-score): (obs − normal) / _obs, where is the inter-annual SD of monthly observations for each station × month. Only when `method  c("standardized", "all")`.

Why anomalies matter for climate-health with DATASUS

- **Multi-city comparability**: standardized anomalies make DLNM exposure-response curves poolable across cities with very different baseline climates (Manaus vs. Porto Alegre).
- **Extreme event attribution**: anomaly peaks pinpoint the dates to correlate with DATASUS hospitalization or mortality surges.
- **Climate trend detection**: a positive trend in temperature anomalies captures local warming for the IPCC Exposure pillar in `sus_mod_vulnerability_index()`.
- **Vector-borne and waterborne diseases**: precipitation anomalies detect flood/drought episodes — with a 2–4 week lag — relevant to dengue, leptospirosis, and diarrheal disease in SINAN.
- **Thermal comfort indices**: anomaly-adjusted temperature feeds DLNM models without confounding by mean inter-city temperature differences.

Anomaly methods

Method	Formula	Primary use
`"absolute"`	obs − normal	Temperature anomalies (°C)
`"relative"`	(obs − normal) / |normal| × 100	Precipitation anomaly (%)
`"standardized"`	(obs − normal) / _obs	Multi-city pooling, DLNM input
`"all"`	All three	Exploratory analysis

See Also

`sus_climate_normals()`, `sus_climate_normals_meta()`, `sus_climate_inmet()`, `sus_climate_aggregate()`, `sus_mod_dlnm()`, `sus_mod_vulnerability_index()`

Examples

```
## Not run:
# 1. Download data
obs  <- sus_climate_inmet(years = 2018:2023, uf = "RJ")
norm <- sus_climate_normals(period = "1991-2020",
                            target_var = c("t_max", "t_min",
                                           "t_mean_comp", "precipitation"))

# 2. Monthly absolute anomalies (default)
anom <- sus_climate_anomaly(obs, norm)

# 3. Standardized anomalies for multi-city DLNM pooling
anom_std <- sus_climate_anomaly(obs, norm,
                                method     = "standardized",
                                vars       = c(tair_dry_bulb_c = "t_mean_comp"))

# 4. Decadal precipitation anomalies - dengue/leptospirosis lag studies
```

```
anom_10d <- sus_climate_anomaly(obs, norm,
                                method     = "relative",
                                time_scale = "decadal",
                                vars       = c(rainfall_mm = "precipitation"))

# 5. All methods at once - exploratory
anom_all <- sus_climate_anomaly(obs, norm, method = "all")

## End(Not run)
```

`sus_climate_compute_coldwaves`
Detect coldwaves using multiple standard methodologies

Description

Applies seven coldwave detection methods (WHO, WMO, INMET, ECF, UTCI, WBGT, HI) to a `climasus_df` at stage `"climate"` produced by `sus_climate_inmet()`, `sus_climate_fill_gap()`, or `sus_climate_compute_indicators()`. Aggregates hourly data to daily scale, computes smoothed historical percentile thresholds from a reference baseline, identifies coldwave days and discrete events per method, and returns a named list with events, daily-flag, and annual-summary tables — all carrying `sus_meta` at stage `"climate"` / type `"coldwaves"`.

Usage

```
sus_climate_compute_coldwaves(
  df,
  method = c("WHO", "WMO", "INMET", "EHF", "UTCI", "WBGT", "HI"),
  baseline_start = NULL,
  baseline_end = NULL,
  percentile = 10,
  min_duration = NULL,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` at stage `"climate"` with type `"inmet"`, `"filled"`, or `"indicators"`. Produced by `sus_climate_inmet()`, `sus_climate_fill_gap()`, or `sus_climate_compute_indicators()`. Must contain columns `date` and `station_code`. Temperature columns `tair_max_c`, `tair_min_c`, `tair_dry_bulb_c` are required for WHO/WMO/ INMET/ECF; `utci_c`, `wbgt_c`, `hi_c` are required for UTCI/WBGT/HI (available after `sus_climate_compute_indicators()`).
`method`	Character vector. One or more of `"WHO"`, `"WMO"`, `"INMET"`, `"EHF"`, `"UTCI"`, `"WBGT"`, `"HI"`. Use `"all"` to run every method. Methods whose required columns are absent emit a warning and are skipped. Note: the internal algorithm for the `"EHF"` slot uses the Excess Cold Factor (ECF) formula.

`baseline_start`
: Date or character (e.g., `"1981-01-01"`). Start of the reference period for percentile thresholds. `NULL` uses the full series.

`baseline_end`
: Date or character. End of the reference period. `NULL` uses the full series.

`percentile`
: Numeric (0-100, default `10`). Percentile used for WHO, WMO, UTCI, WBGT, and HI threshold calculations (lower tail for coldwaves).

`min_duration`
: Integer or `NULL`. Minimum consecutive days to qualify as a coldwave. `NULL` uses method defaults: WHO=3, WMO=5, INMET=5, EHF=3, UTCI=3, WBGT=3, HI=3.

`lang`
: Character. Output language: `"pt"` (default), `"en"`, `"es"`.

`verbose`
: Logical. Print progress messages. Default `TRUE`.

Value

A named list with class `climasus_cw` and `sus_meta` attribute (stage `"climate"`, type `"coldwaves"`):

`events` One row per detected event: `event_id`, `station_code`, `method`, `start_date`, `end_date`, `duration_days`, `temp_mean`, `temp_peak` (coldest), `anomaly_mean`, `anomaly_cumulative`, `severity_index`, `ecf_peak`, `ecf_mean`, `intensity_class`, and station metadata.

`daily` Daily aggregated data with logical flag columns `cw_<method>` and `cw_any`.

`summary` Annual summary by station and method: `year`, `n_events`, `total_days_cw`, `mean_duration`, `max_duration`, `mean_intensity`, `max_intensity`, `mean_anomaly`, `severity_total`.

Implemented methods

1. WHO `Tmin < P10(Tmin)` for >= N consecutive days (default N=3). Threshold smoothed over a 15-day rolling window by calendar day-of-year. Reference: WMO & WHO (2015).

2. WMO `Tmin < P10(Tmin) AND Tmax < P10(Tmax)` for >= N days (default N=5). Double threshold ensures both daytime and nighttime extremes are captured. Reference: Perkins & Alexander (2013).

3. INMET `Tmin < mean(Tmin_hist) - 5 degC` for >= N days (default N=5). Climatological mean from the baseline period. Designed for Brazilian context.

4. ECF Excess Cold Factor: `ECF = ECI_sig x max(1, ECI_acc)` where `ECI_sig = P5(Tmean) - T3` and `ECI_acc = T30 - T3`. Intensity classified by ECF85 (85th percentile of positive ECF values). Adaptation of Nairn & Fawcett (2015) for cold extremes. Positive `ECI_acc` means the preceding month was warmer than the recent 3-day mean (body acclimatised to heat, amplifying cold stress).

5. UTCI `UTCImin < P10(UTCImin)` for >= N days (default N=3). Requires `utci_c` column from `sus_climate_compute_indicators()`.

6. WBGT `WBGTmin < P10(WBGTmin)` for >= N days (default N=3). Requires `wbgt_c` column from `sus_climate_compute_indicators()`.

7. HI `HImin < P10(HImin)` for >= N days (default N=3). Requires `hi_c` column from `sus_climate_compute_indicators()`.

References

- Perkins, S.E. & Alexander, L.V. (2013). On the measurement of heat waves. *Journal of Climate*, 26(13), 4500-4517. (Adapted for coldwaves)
- WMO & WHO (2015). *Heatwaves and Health: Guidance on Warning-System Development*. Geneva. (Adapted for coldwaves)
- Nairn, J.R. & Fawcett, R.J.B. (2015). The Excess Heat Factor. *Int. J. Environ. Res. Public Health*, 12(1), 227-253. (Adapted for cold) doi:10.3390/ijerph120100227
- INMET (2009). *Normais Climatologicas do Brasil 1961-1990*. Brasilia.

See Also

`sus_climate_inmet()`, `sus_climate_compute_indicators()`, `sus_climate_plot_coldwaves()`

Examples

```
## Not run:
cw <- df_indicators |>
  sus_climate_compute_coldwaves(
    method         = c("WHO", "INMET", "EHF"),
    baseline_start = "2000-01-01",
    baseline_end   = "2020-12-31",
    lang           = "pt"
  )

cw$events
cw$summary
cw$daily[cw$daily$cw_any, ]

## End(Not run)
```

`sus_climate_compute_heatwaves`
Detect heatwaves using multiple standard methodologies

Description

Applies seven heatwave detection methods (WHO, WMO, INMET, EHF, UTCI, WBGT, HI) to a `climasus_df` at stage `"climate"` produced by `sus_climate_inmet()`, `sus_climate_fill_gap()`, or `sus_climate_compute_indicators()`. Aggregates hourly data to daily scale, computes smoothed historical percentile thresholds from a reference baseline, identifies heatwave days and discrete events per method, and returns a named list with event, daily-flag, and annual-summary tables — all carrying `sus_meta` at stage `"climate"` / type `"heatwaves"`.

Usage

```
sus_climate_compute_heatwaves(
  df,
  method = c("WHO", "WMO", "INMET", "EHF", "UTCI", "WBGT", "HI"),
  baseline_start = NULL,
  baseline_end = NULL,
  percentile = 90,
  min_duration = NULL,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` at stage `"climate"` with type `"inmet"`, `"filled"`, or `"indicators"`. Produced by `sus_climate_inmet()`, `sus_climate_fill_gap()`, or `sus_climate_compute_indicators()`. Must contain columns `date` and `station_code`. Temperature columns `tair_max_c`, `tair_min_c`, `tair_dry_bulb_c` are required for WHO/WMO/ INMET/EHF; `utci_c`, `wbgt_c`, `hi_c` are required for UTCI/WBGT/HI (available after `sus_climate_compute_indicators()`).
`method`	Character vector. One or more of `"WHO"`, `"WMO"`, `"INMET"`, `"EHF"`, `"UTCI"`, `"WBGT"`, `"HI"`. Use `"all"` to run every method. Methods whose required columns are absent emit a warning and are skipped.
`baseline_start`	Date or character (e.g., `"1981-01-01"`). Start of the reference period for percentile thresholds. `NULL` uses the full series.
`baseline_end`	Date or character. End of the reference period. `NULL` uses the full series.
`percentile`	Numeric (0–100, default `90`). Percentile used for WHO, WMO, UTCI, WBGT, and HI threshold calculations.
`min_duration`	Integer or `NULL`. Minimum consecutive days to qualify as a heatwave. `NULL` uses method defaults: WHO=3, WMO=5, INMET=5, EHF=3, UTCI=3, WBGT=3, HI=3.
`lang`	Character. Output language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A named list with class `climasus_hw` and `sus_meta` attribute (stage `"climate"`, type `"heatwaves"`):

`events` One row per detected event: `event_id`, `station_code`, `method`, `start_date`, `end_date`, `duration_days`, `temp_mean`, `temp_peak`, `anomaly_mean`, `anomaly_cumulative`, `severity_index`, `ehf_peak`, `ehf_mean`, `intensity_class`, and station metadata.

`daily` Daily aggregated data with logical flag columns `hw_<method>` and `hw_any`.

`summary` Annual summary by station and method: `year`, `n_events`, `total_days_hw`, `mean_duration`, `max_duration`, `mean_intensity`, `max_intensity`, `mean_anomaly`, `severity_total`.

Implemented methods

1. WHO `Tmax > P90(Tmax)` for >= N consecutive days (default N=3). Threshold smoothed over a 15-day rolling window by calendar day-of-year. Reference: WMO & WHO (2015).

2. WMO `Tmax > P90(Tmax) AND Tmin > P90(Tmin)` for >= N days (default N=5). Double threshold ensures both daytime and nighttime extremes are captured. Reference: Perkins & Alexander (2013).

3. INMET `Tmax > mean(Tmax_hist) + 5 degC` for >= N days (default N=5). Climatological mean from the baseline period. Designed for Brazilian context. Reference: INMET (2009); MCTI/Gov.br (2025).

4. EHF Excess Heat Factor: `EHF = EHI_sig x max(1, EHI_acc)` where `EHI_sig = T3 - P95(Tmean)` and `EHI_acc = T3 - T30`. Intensity classified by EHF85 (85th percentile of positive EHF values). Reference: Nairn & Fawcett (2015).

5. UTCI `UTCImax > P90(UTCImax)` for >= N days (default N=3). Requires `utci_c` column from `sus_climate_compute_indicators()`.

6. WBGT `WBGTmax > P90(WBGTmax)` for >= N days (default N=3). Requires `wbgt_c` column from `sus_climate_compute_indicators()`.

7. HI `HImax > P90(HImax)` for >= N days (default N=3). Requires `hi_c` column from `sus_climate_compute_indicators()`.

References

- Perkins, S.E. & Alexander, L.V. (2013). On the measurement of heat waves. *Journal of Climate*, 26(13), 4500-4517.
- WMO & WHO (2015). *Heatwaves and Health: Guidance on Warning-System Development*. Geneva.
- Nairn, J.R. & Fawcett, R.J.B. (2015). The Excess Heat Factor. *Int. J. Environ. Res. Public Health*, 12(1), 227-253. doi:10.3390/ijerph120100227
- Broede, P. et al. (2012). Deriving the operational procedure for UTCI. *Int. J. Biometeorology*, 56(3), 481-494. doi:10.1007/s0048401104541
- INMET (2009). *Normais Climatologicas do Brasil 1961-1990*. Brasilia.

See Also

`sus_climate_inmet()`, `sus_climate_compute_indicators()`

Examples

```
## Not run:
hw <- df_indicators |>
  sus_climate_compute_heatwaves(
    method         = c("WHO", "INMET", "EHF"),
    baseline_start = "2000-01-01",
    baseline_end   = "2020-12-31",
    lang           = "pt"
  )
```

```
hw$events
hw$summary
hw$daily[hw$daily$hw_any, ]

## End(Not run)
```

`sus_climate_compute_indicators` *Compute Bioclimatic and Thermal Stress Indicators*

Description

`sus_climate_compute_indicators()` calculates a comprehensive set of bioclimatic and thermal-stress indices from INMET station data (output of `sus_climate_inmet()` or `sus_climate_fill_inmet()`).

Scientific robustness highlights:

- WBGT uses a dual wet-bulb estimate (Liljegren + Bernard/Pourmoghani) averaged for numerical robustness; solar radiation `NA` treated as 0 (no solar load at night) so all hours yield a valid value.
- HI implements full Rothfusz (1990) regression with both NWS humidity adjustments and a regionally adapted validity threshold.
- UTCI uses a Fiala-polynomial-inspired multi-term regression (vapour pressure, logarithmic wind, radiation linear + non-linear, wind vs radiation interaction).
- PET is based on a MEMI-style energy-balance approximation with seasonal clothing adjustment and regional correction.
- Solar radiation undergoes optional physical consistency verification against extraterrestrial radiation limits.
- Diurnal range is a true daily aggregate (Tmax - Tmin per calendar day) broadcast to all hourly rows of that day.
- Multi-level confidence flags (`_flag_extreme`, `_flag_high`, `_flag_low`) replace a single `_out_of_range` flag.
- Optional 200-simulation Monte Carlo uncertainty estimation.

Available indicators:

Code	Output column	Requires
`wbgt`	`wbgt_c`	T, RH, SR, WS
`hi`	`hi_c`	T, RH
`thi`	`thi_c`	T, RH
`wcet`	`wcet_c`	T, WS
`wct`	`wct_c`	T, WS
`et`	`et_c`	T, RH, WS
`utci`	`utci_c`	T, RH, WS, SR
`pet`	`pet_c`	T, RH, WS, SR

`cdd`	`cdd_c`	T
`hdd`	`hdd_c`	T
`gdd`	`gdd_c`	T
`diurnal_range`	`diurnal_range_c`	T (daily agg.)
`vapor_pressure`	`vapor_pressure_kpa`	T, RH
`heat_stress_risk`	`heat_stress_risk`	T, RH, WS, SR
`koppen_humidity`	`koppen_humidity`	RH

(Where: T=`tair_dry_bulb_c`*, RH=*`rh_mean_porc`*, SR=*`sr_kj_m2`*, WS=*`ws_2_m_s`*)*

Usage

```
sus_climate_compute_indicators(
  df,
  indicators = "all",
  region = "auto",
  datetime_col = NULL,
  station_col = NULL,
  apply_validity_mask = TRUE,
  confidence_flags = NULL,
  custom_thresholds = NULL,
  keep_source_vars = FALSE,
  verify_physics = TRUE,
  compute_uncertainty = FALSE,
  verbose = TRUE,
  lang = "pt",
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/spatial"
)
```

Arguments

`df`	Data frame or tibble from `sus_climate_inmet()` or `sus_climate_fill_inmet()`.
`indicators`	Character vector of indicator codes or `"all"` (default).
`region`	Character. `"auto"` (default, standard thresholds) or a Brazilian biome: `"amazon"`, `"cerrado"`, `"caatinga"`, `"atlantic_forest"`, `"pampa"`, `"pantanal"`, `"southeast"`, `"south"`.
`datetime_col`	Character or `NULL` (auto-detected).
`station_col`	Character or `NULL` (auto-detected).
`apply_validity_mask`	Logical (default `TRUE`). When `region=="none"`, masks HI, WCET, WCT outside their valid meteorological domains with `NA`.
`confidence_flags`	Logical. Add `_flag_extreme`, `_flag_high`, `_flag_low` columns. Default `TRUE` when region is set, `FALSE` otherwise.

`custom_thresholds`
: Named list to override regional defaults.

`keep_source_vars`
: Logical (default `FALSE`).

`verify_physics`
: Logical (default `TRUE`). Check solar radiation against extraterrestrial limits; requires `latitude` column.

`compute_uncertainty`
: Logical (default `FALSE`). Monte Carlo 95% CI.

`verbose`
: Logical (default `TRUE`).

`lang`
: Character: `"pt"` (default), `"en"`, `"es"`.

`use_cache`
: Logical. If `TRUE` (default), uses cached spatial data to avoid re-downloads and improve performance.

`cache_dir`
: Character string specifying the directory to store cached files. Default is `"~/.climasus4r_cache/spatial"`.

Value

A `tibble` of class `climasus_df` containing indicator columns, optional flag/CI columns, and a `sus_meta` attribute.

Scientific References

- Liljegren et al. (2008) *J. Occup. Environ. Hyg.* 5(10):645-655.
- Bernard & Pourmoghani (1999) *AIHAJ* 60(1):32-37.
- Rothfusz (1990) NWS Tech. Attachment SR/SSD 90-23.
- Thom (1959) *Weatherwise* 12(2):57-59.
- Environment Canada (2001) Wind Chill Index.
- Brode et al. (2012) *Int. J. Biometeorol.* 56(4):481-494.
- Matzarakis et al. (1999) *Int. J. Biometeorol.* 43:76-84.
- Magnus (1844) / Tetens (1930) saturation vapour pressure.

Examples

```
## Not run:
result <- sus_climate_compute_indicators(df_clima)

result <- sus_climate_compute_indicators(
  df_clima, region = "amazon", confidence_flags = TRUE
)

result <- sus_climate_compute_indicators(
  df_clima,
  region              = "auto",
  verify_physics      = TRUE,
  compute_uncertainty = TRUE,
  keep_source_vars    = TRUE
```

```
)

df_clima |>
  sus_climate_fill_inmet(target_var = "all") |>
  sus_climate_compute_indicators(indicators = c("wbgt", "hi", "utci"))

## End(Not run)
```

`sus_climate_compute_spei` *Compute Standardized Precipitation-Evapotranspiration Index (SPEI)*

Description

`sus_climate_compute_spei()` calculates the Standardized Precipitation- Evapotranspiration Index (SPEI; Vicente-Serrano et al., 2010) at multiple timescales from monthly precipitation and potential evapotranspiration (PET) already aggregated to municipalities.

SPEI extends `sus_climate_compute_spi()` by accounting for atmospheric water demand (temperature-driven evapotranspiration) in addition to precipitation supply. It uses the climatic water balance `D = P - PET` and fits a 3-parameter log-logistic distribution, making it sensitive to warming- amplified drought. Health applications in Brazil:

- Drought (SPEI-3 to SPEI-12): malnutrition, diarrhoeal disease, heat-amplified mortality, vector-borne diseases in the semi-arid Northeast and Amazônia
- Wet spells (positive SPEI): leptospirosis, hepatitis A, dengue

PET options (choose one):

- `pet_method = "column"` (default): pass a pre-computed PET column via `pet_var`. ERA5-Land potential evapotranspiration (`"evap"`) or the FAO Penman-Monteith result are ideal.
- `pet_method = "thornthwaite"`: compute PET internally using only monthly mean temperature (`temp_var`). Suitable when only temperature data is available (e.g., INMET or ERA5 T2m).

Usage

```
sus_climate_compute_spei(
  df,
  rain_var = "rainfall_chirps_mm",
  pet_var = "pet_mm",
  pet_method = c("column", "thornthwaite"),
  temp_var = "tair_dry_bulb_c",
  scales = c(1L, 3L, 6L, 12L),
  ref_start = NULL,
  ref_end = NULL,
  min_n = 24L,
```

```
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` at `stage = "climate"` with monthly data. Must contain `code_muni` (character), `date` (Date, first day of each month), and the precipitation column named by `rain_var`.
`rain_var`	Character. Name of the monthly precipitation column (mm). Default `"rainfall_chirps_mm"`. Other common value: `"rainfall_mm"`.
`pet_var`	Character. Name of the monthly potential evapotranspiration column (mm). Used when `pet_method = "column"`. Default `"pet_mm"`. Ignored when `pet_method = "thornthwaite"`.
`pet_method`	Character. How to obtain PET values. One of: • `"column"` (default): use the column named by `pet_var` directly. • `"thornthwaite"`: compute Thornthwaite (1948) PET from monthly mean temperature in the column named by `temp_var`. Only needs temperature; less accurate than energy-balance methods but practical when only INMET or ERA5 T2m is available.
`temp_var`	Character. Name of the monthly mean temperature column (°C). Used only when `pet_method = "thornthwaite"`. Default `"tair_dry_bulb_c"`.
`scales`	Integer vector. SPEI timescales in months. Default `c(1L, 3L, 6L, 12L)`.
`ref_start`	Date or `NULL`. Start of the calibration period for log-logistic distribution fitting. `NULL` uses all available data. WMO recommends at least 30 years.
`ref_end`	Date or `NULL`. End of the calibration period. `NULL` uses all available data.
`min_n`	Integer. Minimum number of non-NA values required to fit the log-logistic distribution per location. Default `24L` (2 years). Locations with fewer values receive `NA`.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

The input `climasus_df` with additional columns:

- `spei_{s}mo` — SPEI at timescale `s` months (numeric, dimensionless). For example, `scales = c(3, 6, 12)` adds `spei_3mo`, `spei_6mo`, `spei_12mo`.

Metadata: `stage = "climate"`, `type = "spei"`.

SPEI vs SPI

SPEI is more sensitive to climate change because rising temperatures increase PET even when precipitation is unchanged. Use SPEI-3 for short-term drought and health outcomes with 1-4 week lag (diarrhoea, dengue). Use SPEI-6 or SPEI-12 for water supply and agricultural drought (malnutrition, livelihood loss).

Algorithm

For each municipality and timescale `s`:

1. Compute climate water balance: `D = P - PET` (monthly).
2. Accumulate over `s` months via rolling sum.
3. Fit 3-parameter log-logistic distribution via L-moments (Vicente-Serrano et al., 2010). Falls back to empirical ECDF for numerically unstable cases.
4. Transform to standard normal.

References

Vicente-Serrano, S.M., Beguería, S., López-Moreno, J.I. (2010). A multiscalar drought index sensitive to global warming: the Standardized Precipitation Evapotranspiration Index. *Journal of Climate*, 23(7), 1696–1718. doi:10.1175/2009JCLI2909.1

Thornthwaite, C.W. (1948). An approach toward a rational classification of climate. *Geographical Review*, 38(1), 55–94.

See Also

`sus_climate_compute_spi()`, `sus_grid_chirps()`, `sus_grid_era5()`, `sus_grid_join()`, `sus_mod_dlnm()`

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Option A: SPEI from CHIRPS + Thornthwaite PET (ERA5-Land temperature)
chirps_era5 <- sus_grid_era5(
  years = 1990:2022, vars = c("t2m"), municipalities = mt_mun
) |> sus_grid_join(
  sus_grid_chirps(resolution="monthly", years=1990:2022, municipalities=mt_mun)
)

spei_mt <- sus_climate_compute_spei(
  df         = chirps_era5,
  rain_var   = "rainfall_chirps_mm",
  pet_method = "thornthwaite",
  temp_var   = "tair_dry_bulb_c",
  scales     = c(3L, 6L, 12L),
  lang       = "pt"
)

# Option B: SPEI from pre-computed PET column
spei_mt2 <- sus_climate_compute_spei(
  df       = climate_df,         # has rainfall_mm and pet_mm columns
  rain_var = "rainfall_mm",
  pet_var  = "pet_mm",
  pet_method = "column",
```

```
  scales   = c(3L, 6L, 12L)
)

sus_meta(spei_mt, "stage")  # "climate"
sus_meta(spei_mt, "type")   # "spei"
names(spei_mt)              # ..., spei_3mo, spei_6mo, spei_12mo

## End(Not run)
```

`sus_climate_compute_spi` *Compute Standardized Precipitation Index (SPI)*

Description

`sus_climate_compute_spi()` calculates the Standardized Precipitation Index (SPI; McKee et al., 1993) at multiple timescales from monthly cumulative precipitation already aggregated to municipalities.

SPI is a dimensionless index of precipitation anomaly relative to a long-term distribution: negative values indicate drought (SPI $\leq$ -1 = moderate, $\leq$ -1.5 = severe, $\leq$ -2 = extreme) and positive values indicate wet conditions. Multi-scale SPI is a leading indicator for several health outcomes in Brazil:

- Drought (SPI-3, SPI-6): diarrhoeal disease, malnutrition, vector-borne diseases during water scarcity
- Wet spells (SPI-1, SPI-3): leptospirosis, hepatitis A, dengue

The function requires **monthly** precipitation data (one row per municipality $\times$ month). The best upstream source is `sus_grid_chirps()` with `resolution = "monthly"` (0.05°, 1981–present). INMET-derived `rainfall_mm` from `sus_climate_aggregate()` also works.

Usage

```
sus_climate_compute_spi(
  df,
  var = "rainfall_chirps_mm",
  scales = c(1L, 3L, 6L, 12L),
  ref_start = NULL,
  ref_end = NULL,
  min_n = 24L,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` at `stage = "climate"` with monthly precipitation. Must contain `code_muni` (character), `date` (Date, first day of each month), and the column named by `var`.
`var`	Character. Name of the monthly precipitation column. Default `"rainfall_chirps_mm"`. Other common value: `"rainfall_mm"`.
`scales`	Integer vector. SPI timescales in months. Default `c(1L, 3L, 6L, 12L)`. Common choices: 1 (soil moisture), 3 (short-term), 6 (agriculture/water supply), 12 (groundwater/streamflow).
`ref_start`	Date or `NULL`. Start of the calibration period for gamma distribution fitting. `NULL` (default) uses all available data. WMO recommends at least 30 years; CHIRPS 1981–2010 is ideal.
`ref_end`	Date or `NULL`. End of the calibration period. `NULL` (default) uses all available data.
`min_n`	Integer. Minimum number of non-NA, non-zero precipitation values required to fit the gamma distribution per location. Default `24L` (2 years). Locations with fewer values receive `NA` for all SPI columns.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

The input `climasus_df` with additional columns:

- `spi_{s}mo` — SPI at timescale `s` months (numeric, dimensionless)

For example, `scales = c(1, 3, 6, 12)` adds `spi_1mo`, `spi_3mo`, `spi_6mo`, `spi_12mo`. Metadata: `stage = "climate"`, `type = "spi"`.

SPI classification

```
SPI   2.0 : Extremely wet
SPI  1.5 to  1.99 : Very wet
SPI  1.0 to  1.49 : Moderately wet
SPI -0.99 to  0.99 : Near normal
SPI -1.0 to -1.49 : Moderately dry (D1)
SPI -1.5 to -1.99 : Severely dry   (D2)
SPI  -2.0 : Extremely dry (D3-D4)
```

Algorithm

For each municipality and timescale `s`:

1. Compute `s`-month rolling sum of precipitation using `slider::slide_dbl()`.
2. Estimate gamma distribution parameters (shape α, rate β) via method of moments from the calibration period, handling zero-inflation with a mixed distribution (McKee et al., 1993).
3. Transform to standard normal: `SPI = qnorm(p0 + (1-p0) * pgamma(x, shape, rate))`, where `p0` is the proportion of zeros.

References

McKee, T.B., Doesken, N.J., Kleist, J. (1993). The relationship of drought frequency and duration to time scales. *Proceedings of the 8th Conference on Applied Climatology*. American Meteorological Society.

See Also

sus_grid_chirps(), sus_climate_anomaly(), sus_grid_join(), sus_mod_dlnm()

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Download monthly CHIRPS for Mato Grosso, 1990-2022
chirps_mt <- sus_grid_chirps(
  resolution    = "monthly",
  years         = 1990:2022,
  municipalities = mt_mun
)

# Compute SPI at 1-, 3-, 6- and 12-month timescales
spi_mt <- sus_climate_compute_spi(
  df         = chirps_mt,
  var        = "rainfall_chirps_mm",
  scales     = c(1L, 3L, 6L, 12L),
  ref_start  = as.Date("1991-01-01"),
  ref_end    = as.Date("2020-12-31"),
  lang       = "pt"
)

sus_meta(spi_mt, "stage")  # "climate"
sus_meta(spi_mt, "type")   # "spi"
names(spi_mt)              # ..., spi_1mo, spi_3mo, spi_6mo, spi_12mo

# Join to health data
combined <- sus_grid_join(health_mt, spi_mt)

## End(Not run)
```

`sus_climate_fill_inmet`

Fill gaps in INMET climate time series using XGBoost

Description

`sus_climate_fill_inmet()` imputes missing values in **INMET automatic station data** using station-wise XGBoost models with automated feature engineering.

Key features:

- **Multi-target support**: Impute one, many, or ALL numeric variables in a single call
- **Station-wise modeling**: Separate model per station per variable
- **Temporal features**: Automatic creation of lags and rolling statistics
- **Quality control**: Stations with >`quality_threshold` missing are excluded
- **Parallel processing**: Stations processed in parallel; workers resolved once and reused across all variable iterations
- **Evaluation mode**: Assess accuracy by creating artificial gaps

Important: This function is designed exclusively for data imported by `sus_climate_inmet()` and works with the standard INMET variable set.

Usage

```
sus_climate_fill_inmet(
  df,
  target_var,
  datetime_col = NULL,
  station_col = NULL,
  quality_threshold = 0.4,
  run_evaluation = FALSE,
  gap_percentage = 0.2,
  keep_features = FALSE,
  parallel = TRUE,
  workers = NULL,
  verbose = TRUE,
  lang = "pt"
)
```

Arguments

`df`	A data frame (or tibble) containing climate data, typically from `sus_climate_inmet()`. Must contain: • A datetime column (POSIXct or convertible) • A station identifier column • The target numeric column(s) to be imputed
`target_var`	Character vector of column name(s) to impute, **or** the special string `"all"` to impute every numeric column that is not the datetime or station column. Examples: • Single variable: `target_var = "tair_dry_bulb_c"` • Multiple variables: `target_var = c("tair_dry_bulb_c", "rh_pct", "ws_2_m_s")`

- All numeric variables: `target_var = "all"`

`datetime_col` Character. Name of the datetime column. If `NULL` (default), auto-detected.

`station_col` Character. Name of the station identifier column. If `NULL`, auto-detected.

`quality_threshold`
Numeric (0-1). Maximum allowed missing proportion per station. Stations exceeding this are excluded. Default: `0.4` (40%).

For INMET data: Recommended values:

- `0.3` (30%) - Conservative, high-quality stations only
- `0.4` (40%) - Default, balanced approach
- `0.6` (60%) - Lenient, includes stations with significant gaps

`run_evaluation`
Logical. If `TRUE`, runs in evaluation mode:

- Creates artificial MCAR gaps in observed data
- Imputes and compares predictions with true values
- Returns metrics by station

Default: `FALSE` (production mode).

`gap_percentage`
Numeric (0-1). Proportion of data to set as missing in evaluation mode. The `"MCAR"` Missing Completely At Random is used as default. Default: `0.2` (20%).

`keep_features` Logical. If `TRUE`, retains engineered features (lags, rolling stats). Default: `FALSE` (returns only original columns + `is_imputed`).

`parallel` Logical. If `TRUE` (default), processes stations in parallel using `furrr`.

`workers` Integer. Number of parallel workers. If `NULL`, uses `max(1, availableCores() - 1)`. The value is resolved **once** at the start and reused across all variable iterations, avoiding repeated plan switches.

`verbose` Logical. If `TRUE` (default), prints progress messages.

`lang` Character. Message language: `"pt"` (Portuguese), `"en"` (English), or `"es"` (Spanish). Default: `"pt"`.

Value

Production mode (`run_evaluation = FALSE`):

- **Single variable**: a `tibble` (same class as input) with imputed values in `target_var` and an `is_imputed_<var>` flag column.
- **Multiple variables / `"all"`**: the same tibble with all requested variables imputed sequentially, each with its own `is_imputed_<var>` flag column.
- A `sus_meta` attribute records all imputed variables and their rates.

Evaluation mode (`run_evaluation = TRUE`): Returns a **named list** (one element per variable) of class `climasus_eval`, each with:

- `$data`: Data frame with artificial gaps and predictions

- `$metrics`: A `tibble` with per-station performance metrics:
  - `station`: Station identifier
  - `rmse`: Root Mean Squared Error
  - `mae`: Mean Absolute Error
  - `r_squared`: R-squared (lower than 1, higher is better)
  - `smape`: Symmetric MAPE (0-200%, lower is better)
  - `slope_bias`: Should be close to 1.0, indicating underestimate and overestimate
  - `n_gaps`: Number of artificial gaps

INMET Variable Set

The function is optimized for the following 17 INMET variables:

- **Atmospheric Pressure**: `patm_mb`, `patm_max_mb`, `patm_min_mb`
- **Temperature**: `tair_dry_bulb_c`, `tair_max_c`, `tair_min_c`
- **Dew Point**: `dew_tmean_c`, `dew_tmax_c`, `dew_tmin_c`
- **Relative Humidity**: `rh_max_porc`, `rh_min_porc`, `rh_mean_porc`
- **Precipitation**: `rainfall_mm`
- **Wind**: `ws_gust_m_s`, `ws_2_m_s`, `wd_degrees`
- **Solar Radiation**: `sr_kj_m2`

When `target_var = "all"`, the function automatically detects and imputes **only these 17 variables** if present in the data.

Methodological Notes

Important limitations:

- **Not forecasting**: Predicts only where data are missing
- **No future data**: Uses only past information (lags)
- **Station independence**: Models don't share information
- **Quality filter**: Stations with >`quality_threshold` missing are skipped

Feature engineering: Automatically creates:

- Time features: hour, day, month, year, cyclic transforms
- Lag features: 1,2,3,6,12,24,48,72,168 periods
- Rolling statistics: mean and sd over windows 3,6,12,24,48,72

Evaluation Mode Details

When `run_evaluation = TRUE`, the function:

1. Creates artificial gaps (MCAR by default)
2. Runs imputation on the data with gaps
3. Compares predictions with true values
4. Returns per-station performance

This helps assess model accuracy before production use.

Examples

```
## Not run:
# ===== PRODUCTION MODE - single variable =====
filled_temp <- sus_climate_fill_inmet(
  df = climate_data,
  target_var = "tair_dry_bulb_c",
  quality_threshold = 0.3,
  parallel = TRUE
)

# ===== PRODUCTION MODE - multiple variables =====
filled_multi <- sus_climate_fill_inmet(
  df = climate_data,
  target_var = c("tair_dry_bulb_c", "rh_pct", "ws_2_m_s"),
  quality_threshold = 0.3,
  parallel = TRUE,
  workers = 4
)

# ===== PRODUCTION MODE - all numeric variables =====
filled_all <- sus_climate_fill_inmet(
  df = climate_data,
  target_var = "all",
  parallel = TRUE
)

# ===== EVALUATION MODE - single variable =====
eval_results <- sus_climate_fill_inmet(
  df = climate_data,
  target_var = "ws_2_m_s",
  run_evaluation = TRUE,
  gap_percentage = 0.2,
  workers = 4
)
eval_results$ws_2_m_s$metrics

# ===== EVALUATION MODE - multiple variables =====
eval_multi <- sus_climate_fill_inmet(
  df = climate_data,
  target_var = c("tair_dry_bulb_c", "ws_2_m_s"),
  run_evaluation = TRUE,
  gap_percentage = 0.2
)
eval_multi$tair_dry_bulb_c$metrics
eval_multi$ws_2_m_s$metrics

## End(Not run)
```

`sus_climate_inmet` *Import and Process INMET Meteorological Data*

Description

`sus_climate_inmet()` downloads, imports, standardizes, and quality-controls Brazilian meteorological data from the National Institute of Meteorology (INMET).

The function implements a **comprehensive processing pipeline**:

1. **Download**: Multi-year data with automatic retry and backoff
2. **Parsing**: Handles INMET's CSV format with metadata extraction
3. **Standardization**: Renames columns to canonical names (see **Variables**)
4. **Quality Control**: Physical consistency checks (see **QC Details**)
5. **Caching**: Two-level (memory + disk) with Parquet format
6. **Parallel Processing**: Both between and within years

Usage

```
sus_climate_inmet(
  years = NULL,
  uf = NULL,
  station_code = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/climate",
  parallel = TRUE,
  workers = 4,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Numeric vector of year(s) to import. Examples: `2020`, `2020:2024`, `c(2019, 2021, 2023)`. Must be between 2000 and current year. If `NULL`, imports last 2 years.
`uf`	Character vector of Brazilian state codes (e.g., `"AM"`, `c("RJ", "MG")`). Case insensitive. If `NULL` (default), imports **all 27 states** (may be slow - see **Performance**).
`station_code`	Character vector of INMET station codes to filter (e.g., `c("A101", "A122")`). Optional. If provided, the function downloads data for the requested `years`/`uf` and then filters to include only the specified stations. If `NULL` (default), all stations in the selected states are returned. Station codes are matched case-insensitively.
`use_cache`	Logical. If `TRUE` (default), implements two-level caching: • **Session cache**: In-memory cache (MD5 hash of parameters) • **Disk cache**: Parquet files with Zstandard compression Use `unlink(cache_dir, recursive = TRUE)` to clear all caches.
`cache_dir`	Character. Directory path for disk cache. Default: `"~/.climasus4r_cache/climate"`. Created automatically.

`parallel` Logical. If `TRUE` (default), enables **two levels** of parallelism:

- **Between years**: Multiple years downloaded simultaneously
- **Within year**: CSV files for each year parsed in parallel

For single-year imports, only within-year parallelization applies.

`workers` Integer. Number of parallel workers. Default: `4`. Ignored if `parallel = FALSE`. Uses `future::multisession` backend. Automatically capped at `availableCores() - 1`.

`lang` Character. Message language. One of:

- `"pt"`: Portuguese (default)
- `"en"`: English
- `"es"`: Spanish

`verbose` Logical. If `TRUE` (default), prints detailed progress including:

- Cache hits/misses
- Download progress with retry attempts
- QC modifications (rows corrected/removed)
- Final statistics

Value

A `climasus_df` object (subclass of `tibble`) with class hierarchy: `climasus_df` > `tbl_df` > `tbl` > `data.frame`

Data Columns:

`station_code` Character. INMET 8-digit station identifier (e.g., "A001")

`station_name` Character. Full station name

`region` Character. Brazilian region (Norte, Nordeste, etc.)

`UF` Character. State abbreviation

`latitude`, `longitude`, `altitude` Numeric. Station coordinates (WGS84)

`date` POSIXct. Observation timestamp **UTC** (always hourly)

`year` Integer. Year extracted from date

Climate variables Numeric. Standardized names (see **Variables**)

Metadata (accessible via `attr(x, "sus_meta")`):

`version` Package version used

`timestamp` Import date/time

`source` "INMET"

`years` Years imported

`ufs` States imported

`station_codes` Station codes filtered (or NULL if all)

`cache_used` Whether cache was used

`qc_stats` List with quality control statistics

`n_stations` Number of unique stations

`n_observations` Total rows

`temporal_coverage` List with `start` and `end` dates

`history` Character vector of processing steps

Standardized Meteorological Variables

INMET raw column names vary by year. This function automatically detects and renames them to the following canonical names:

Canonical Name	Description	Unit	Physical Range
`rainfall_mm`	Precipitation total	mm	0 - 500
`patm_mb`	Mean atmospheric pressure	mb	700 - 1100
`patm_max_mb`	Max atmospheric pressure	mb	700 - 1100
`patm_min_mb`	Min atmospheric pressure	mb	700 - 1100
`sr_kj_m2`	Solar radiation	kJ/m²	0 - 40000
`tair_dry_bulb_c`	Mean air temperature	°C	-90 - 60
`tair_max_c`	Max air temperature	°C	-90 - 60
`tair_min_c`	Min air temperature	°C	-90 - 60
`dew_tmean_c`	Mean dew point	°C	-90 - 60
`dew_tmax_c`	Max dew point	°C	-90 - 60
`dew_tmin_c`	Min dew point	°C	-90 - 60
`rh_mean_porc`	Mean relative humidity	%	0 - 100
`rh_max_porc`	Max relative humidity	%	0 - 100
`rh_min_porc`	Min relative humidity	%	0 - 100
`ws_2_m_s`	Wind speed at 2m	m/s	0 - 100
`ws_gust_m_s`	Wind gust	m/s	0 - 100
`wd_degrees`	Wind direction	degrees	0 - 360

Quality Control Details

The function applies **automatic physical consistency checks**:

1. Physical Range Validation: Values outside physically possible ranges are set to `NA`:

- **Temperature**: -90°C to 60°C
- **Pressure**: 700 mb to 1100 mb
- **Humidity**: 0% to 100%
- **Precipitation**: 0 mm to 500 mm
- **Solar radiation**: 0 to 40000 kJ/m²
- **Wind speed**: 0 to 100 m/s
- **Wind direction**: 0° to 360°

2. Dew Point Consistency: Calculates theoretical dew point from T and RH using Magnus formula. If |observed - calculated| > 3°C, observed is set to NA.

3. Solar Radiation: Nighttime values (18h-6h) are set to 0 for physical consistency.

Caching System Details

Disk Cache:

- **Format**: Apache Parquet with Zstandard compression (level 6)
- **Partitioning**: By `year` and `UF` for fast filtering
- **Backup**: Compressed CSV as fallback if Parquet corrupted
- **Location**: `~/.climasus4r_cache/climate/inmet_parquet/`

Note

- **Data frequency**: Always **hourly**. Use `sus_climate_aggregate()` for daily/weekly.
- **Timezone**: All timestamps are **UTC**. Convert if needed.
- **Missing data**: Represented as `NA`. Use `sus_climate_fill_inmet()` for imputation.
- **Encoding**: All strings converted to UTF-8.
- **Decimals**: Converted from comma (`,`) to point (`.`).

See Also

- `sus_climate_fill_inmet()` for ML-based gap filling
- `sus_spatial_join()` for preparing municipality data

Examples

```
## Not run:
# Basic import - single state, single year
climate_am <- sus_climate_inmet(
  years = 2023,
  uf = "AM"
)

# Filter by specific station codes
climate_sp <- sus_climate_inmet(
  years = 2023,
  uf = "SP",
  station_code = c("A701", "A711")
)

# Multi-year import with parallel processing
climate_sp <- sus_climate_inmet(
  years = 2020:2024,
  uf = "SP",
  parallel = TRUE,
  workers = 4
)

# Import all Southeast states
climate_se <- sus_climate_inmet(
  years = 2023,
  uf = c("SP", "RJ", "MG", "ES"),
```

```
  verbose = TRUE
)

# Inspect available stations
climate_df |>
  dplyr::distinct(station_code, station_name, latitude, longitude)

## End(Not run)
```

`sus_climate_normals` *Download and Process INMET Climate Normals*

Description

`sus_climate_normals()` retrieves 30-year climatological normals from the Brazilian National Institute of Meteorology (INMET) for one of three reference periods (1961-1990, 1981-2010, 1991-2020). It loads the variable catalogue from the bundled `normal_meta.parquet` dictionary—with transparent disk caching and automatic fallback download—then fetches each selected variable's Excel file from the INMET portal, parses the decade-level monthly structure, and returns a tidy long-format `climasus_df`.

Usage

```
sus_climate_normals(
  period = "1991-2020",
  target_var = NULL,
  cache_dir = "~/.climasus4r_cache/normals",
  use_cache = TRUE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`period`	Character. Reference climatological period. One of `"1961-1990"`, `"1981-2010"`, or `"1991-2020"` (default).
`target_var`	Character vector of `var_code` values to download (e.g. `c("t_max", "precipitation")`). `NULL` (default) downloads all variables available for the selected `period`. Run `sus_climate_normals_meta()` to inspect the catalogue.
`cache_dir`	Character. Local directory for disk-cached files. Default: `"~/.climasus4r_cache/normals"`.
`use_cache`	Logical. If `TRUE` (default), loads from and saves to `cache_dir`. Set to `FALSE` to force a fresh download.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_df` object at stage `"climate"` and type `"normals"`, tidy long format with columns:

`codigo` Character. INMET station identifier.

`nome_estacao` Character. Station name.

`uf` Character. Brazilian state abbreviation.

`mes` Character. Month name (Portuguese, lower-case ASCII).

`decada` Character. Decade within month (`"1"`, `"2"`, `"3"`).

`valor` Numeric. Climatological normal value.

`var_code` Character. Variable identifier from `normal_meta`.

`variable_pt` Character. Variable label in Portuguese.

`variable_en` Character. Variable label in English.

`period` Character. Reference period (e.g. `"1991-2020"`).

Public-health and environmental relevance

Climate normals serve as the **climatological baseline** against which observed exposures are compared in epidemiological studies:

- **Heat-health**: temperature normals reveal chronically hot municipalities for heat-wave attribution and heat-related mortality studies.
- **Precipitation extremes**: wet-day and dry-spell normals support flood and drought exposure assessments linked to diarrhoeal disease, leptospirosis, and waterborne outbreaks.
- **Vector-borne diseases**: humidity and temperature normals define the climatic envelope of *Aedes* mosquito habitat, informing dengue, chikungunya and Zika risk mapping.
- **Relative-risk baselines**: DLNM models built with `sus_mod_dlnm()` require a reference exposure; normals provide the station-level climatological mean for that reference.
- **Vulnerability indices**: combined with `sus_mod_vulnerability_index()`, normals quantify chronic exposure as an IPCC *hazard* pillar component.

See Also

`sus_climate_normals_meta()`, `sus_climate_inmet()`, `sus_mod_dlnm()`, `sus_mod_vulnerability_index()`

Examples

```
## Not run:
# Full 1991-2020 normals (all variables)
normals <- sus_climate_normals(period = "1991-2020")

# Only temperature variables
temp_normals <- sus_climate_normals(
  period     = "1991-2020",
  target_var = c("t_max", "t_min", "t_mean_comp"),
```

```
  lang       = "pt"
)

# Browse available variables first
meta <- sus_climate_normals_meta(period = "1991-2020")

## End(Not run)
```

`sus_climate_normals_meta` *Browse the INMET Climate Normals Catalogue*

Description

Returns the `normal_meta` dictionary as a tibble so you can inspect available variables and their `var_code` values before calling `sus_climate_normals()`.

Usage

```
sus_climate_normals_meta(
  period = NULL,
  lang = "pt",
  cache_dir = "~/.climasus4r_cache/normals",
  use_cache = TRUE,
  verbose = FALSE
)
```

Arguments

`period`	Character. Filter to a specific period, or `NULL` (default) to return the full catalogue.
`lang`	Character. Language for the returned label column: `"pt"` (default), `"en"`, `"es"`.
`cache_dir`	Character. Cache directory. Default: `"~/.climasus4r_cache/normals"`.
`use_cache`	Logical. Use cached catalogue. Default `TRUE`.
`verbose`	Logical. Print messages. Default `FALSE`.

Value

A tibble with columns `var_code`, `variable_label`, `period`, `var_slug`, and `code_link`.

Examples

```
## Not run:
sus_climate_normals_meta()
sus_climate_normals_meta(period = "1991-2020", lang = "en")

## End(Not run)
```

---

sus_climate_plot_aggregate

Visualise Climate-Health Aggregate Data

Description

`sus_climate_plot_aggregate()` produces exploratory visualisations for the output of `sus_climate_aggregate()`. Six complementary plot types cover the full exploratory workflow before modelling: time-series overlay, scatter with smooth, cross-correlation (CCF), distribution, correlation matrix, and seasonal patterns. Modelling-specific plots (DLNM surfaces, residual diagnostics, RR tables) are deliberately excluded to avoid duplication with the dedicated `sus_mod_plot_*` family.

Usage

```
sus_climate_plot_aggregate(
  df,
  outcome_col = NULL,
  climate_cols = NULL,
  plot_type = "timeseries",
  smooth_method = "loess",
  max_lag = 30L,
  alpha = 0.05,
  interactive = FALSE,
  lang = "pt",
  verbose = TRUE,
  title = NULL,
  source = NULL
)
```

Arguments

`df`	A `climasus_df` at `stage = "climate"` produced by `sus_climate_aggregate()`. Must contain at least `date`, `code_muni`, a health outcome column, and one or more climate exposure columns.
`outcome_col`	Character. Name of the health-outcome column (e.g. `"n_obitos"`, `"n_internacoes"`). If `NULL` (default) the first integer column that is not a climate variable is used.

`climate_cols` Character vector of climate column names to visualise. If `NULL` (default) all columns that match the naming conventions of `sus_climate_aggregate()` are detected automatically (`lag*`, `mvwin*`, `off*to*`, `gdd*`, `_lag*`, or known INMET variable names).

`plot_type` Character. One of:

- `"timeseries"` Dual-axis time-series: rescaled climate exposure (ribbon) overlaid on health outcome (line + points). When multiple climate columns are provided and `patchwork` is installed, one panel per variable is stacked vertically via `patchwork::wrap_plots()`. Falls back to the first column only when `patchwork` is not available.
- `"scatter"` Scatter plot with loess/GAM smooth: climate exposure on x-axis, health outcome on y-axis. Shows the marginal association ignoring temporal structure.
- `"ccf"` Cross-Correlation Function bars from −30 to `+max_lag` days. Highlights lags with significant correlation (|r| > 2/sqrt(n)) in a contrasting colour. Useful for identifying the optimal lag before modelling.
- `"distribution"` Histogram + density overlay of each climate column. For `discrete_lag` outputs the multiple lag columns are overlaid for comparison.
- `"corr_matrix"` Heatmap of Spearman correlations between all climate columns and the outcome. Especially informative when the input was produced with `temporal_strategy = "discrete_lag"`.
- `"seasonal"` Monthly boxplots of the climate exposure and the outcome incidence, side-by-side, to reveal seasonal co-variation.

Default `"timeseries"`.

`smooth_method` Character. Smoothing method for the scatter plot: `"loess"` (default) or `"gam"`.

`max_lag` Integer. Maximum lag (days) shown in the CCF plot. Default `30L`.

`alpha` Numeric. Confidence level for CCF significance bounds (default `0.05`; bounds drawn at $\pm 2/\sqrt{n}$).

`interactive` Logical. If `TRUE`, returns a `plotly` interactive version built with `plotly::plot_ly()` for proper dual-axis support (`plotly::ggplotly()` is avoided as it breaks secondary axes for `"timeseries"`). Requires `plotly` to be installed. Default `FALSE`.

`lang` Character. Language for axis labels and titles: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. Print progress messages. Default `TRUE`.

`title` Character or `NULL`. Override the auto-generated plot title. Default `NULL` uses the built-in multilingual title.

`source` Character or `NULL`. Data source attribution prepended to the plot caption. Default `NULL`.

Value

A ggplot2 object, a patchwork object (when multiple climate columns are supplied and patchwork is installed for "timeseries" or "scatter" plot types), or a plotly object when interactive = TRUE.

See Also

sus_climate_aggregate(), sus_climate_plot_fill(), sus_climate_plot_heatwaves(), sus_mod_plot_dlnm()

Examples

```
## Not run:
# Build aggregate data (exact strategy)
df_agg <- sus_climate_aggregate(
  health_data       = sf_sim_spatial,
  climate_data      = df_inmet,
  climate_var       = "tair_dry_bulb_c",
  temporal_strategy = "exact"
)

# Time-series overlay
sus_climate_plot_aggregate(df_agg, plot_type = "timeseries", lang = "pt")

# Discrete lag: correlation matrix across all lag columns
df_lag <- sus_climate_aggregate(
  health_data       = sf_sim_spatial,
  climate_data      = df_inmet,
  climate_var       = "tair_dry_bulb_c",
  temporal_strategy = "discrete_lag",
  lag_days          = c(7, 14, 21)
)
sus_climate_plot_aggregate(df_lag, plot_type = "corr_matrix", lang = "en")

# Cross-correlation
sus_climate_plot_aggregate(df_agg, plot_type = "ccf", max_lag = 21L)

## End(Not run)
```

---

sus_climate_plot_coldwaves

Plot and Analyze Coldwave Events

Description

Visualizes and analyzes the results from sus_climate_compute_coldwaves(). Provides multiple visualization types including a Gantt-style timeline of events, a calendar heatmap, an intensity-vs-duration scatter plot, and an annual trend bar chart.

Usage

```
sus_climate_plot_coldwaves(
  cw_result,
  type = c("timeline", "calendar", "intensity", "trend"),
  station_code = NULL,
  method = NULL,
  year = NULL,
  interactive = TRUE,
  color_palette = "npg",
  lang = "pt",
  save_plot = NULL
)
```

Arguments

`cw_result`	List. The output from `sus_climate_compute_coldwaves()`.
`type`	Character. Type of plot to generate: • `"timeline"`: Gantt-style timeline of coldwave events. • `"calendar"`: Calendar heatmap showing days with coldwaves. • `"intensity"`: Scatter plot of duration vs. coldest temperature. • `"trend"`: Bar chart of number of events per year.
`station_code`	Character. Optional. Filter by a specific station code.
`method`	Character. Optional. Filter by a specific method (e.g., `"EHF"`, `"INMET"`).
`year`	Numeric. Optional. Filter by a specific year (useful for calendar plots).
`interactive`	Logical. If `TRUE`, returns an interactive Plotly chart. If `FALSE`, returns a static ggplot2 chart. Default is `TRUE`.
`color_palette`	Character. Name of the ggsci palette to use. Default is `"npg"`. Blue tones are extracted for cold-event emphasis.
`lang`	Character. Language for labels: `"pt"` (default), `"en"`, `"es"`.
`save_plot`	Character. Optional file path to save the plot (e.g., `"plot.html"` or `"plot.png"`).

Value

A ggplot or plotly object.

See Also

`sus_climate_compute_coldwaves()`

Examples

```
## Not run:
cw <- df_indicators |> sus_climate_compute_coldwaves(method = c("WHO", "EHF"))
sus_climate_plot_coldwaves(cw, type = "timeline", lang = "pt")
sus_climate_plot_coldwaves(cw, type = "trend", interactive = FALSE)
```

```
## End(Not run)
```

`sus_climate_plot_fill` *Visualise time-series gap-filling from sus_climate_fill()*

Description

Unified visualisation for gap-filling outputs of the `climasus4r` pipeline. Automatically detects **production mode** (filled tibble) vs **evaluation mode** (list with `data` and `metrics` components) and renders appropriate interactive (plotly) or static (ggplot2) plots plus DT metric tables.

Production mode displays:

- Time-series of observed (with gaps) vs imputed series.
- Imputed segments as distinct markers. Consecutive imputed blocks are connected by a dotted line; isolated single points appear as dots only.
- Interactive range-slider with zoom presets (plotly only).

Evaluation mode displays a 2×2 diagnostic dashboard:

- Observed vs Predicted scatter with 1:1 reference line.
- Residual distribution histogram.
- Residuals over time (temporal-autocorrelation diagnostic).
- Residuals vs observed (heteroscedasticity diagnostic).

Usage

```
sus_climate_plot_fill(
  df_filled,
  df_original = NULL,
  target_var,
  interactive = FALSE,
  output_type = c("plot", "table", "metrics", "all"),
  save_plot = NULL,
  lang = c("en", "pt", "es"),
  color_palette = c("npg", "aaas", "nejm", "lancet", "jama", "bmj", "jco", "frontiers",
    "gsea", "uchicago", "primer", "atlassian", "observable", "d3", "igv", "cosmic",
    "locuszoom", "ucscgb", "startrek", "tron", "futurama", "rickandmorty", "simpsons",
    "flatui", "bs5", "material", "tw3"),
  verbose = FALSE
)
```

Arguments

df_filled	Output of sus_climate_fill(). In production mode this is a <climasus_df> tibble; in evaluation mode it is the named list returned by sus_climate_fill(evaluation = TRUE).
df_original	Raw data before gap filling, e.g. from sus_climate_inmet(). Required in production mode; ignored in evaluation mode.
target_var	Character vector of variable(s) to visualise.
interactive	Logical. TRUE → plotly; FALSE → ggplot2 (default: TRUE).
output_type	What to return: "plot", "table", "metrics", or "all" (default: "all").
save_plot	Optional file path to save the plot (e.g. "plot.png" or "plot.html").
lang	Language: "en", "pt", or "es" (default: "en").
color_palette	ggsci palette name (default: "npg").
verbose	Print diagnostic information (default: FALSE).

Value

Depending on output_type, returns the plot, table, metrics, or a named list containing all four components: plot, table, metrics, data.

---

sus_climate_plot_heatwaves

Plot and Analyze Heatwave Events

Description

Visualizes and analyzes the results from sus_climate_compute_heatwaves(). It provides multiple visualization types including time series of events, calendar heatmaps, intensity distributions, and annual summaries.

Usage

```
sus_climate_plot_heatwaves(
  hw_result,
  type = c("timeline", "calendar", "intensity", "trend"),
  station_code = NULL,
  method = NULL,
  year = NULL,
  interactive = TRUE,
  color_palette = "npg",
  lang = "en",
  save_plot = NULL
)
```

Arguments

`hw_result`	List. The output from `sus_climate_compute_heatwaves()`.
`type`	Character. Type of plot to generate: • `"timeline"`: Gantt-style timeline of heatwave events. • `"calendar"`: Calendar heatmap showing days with heatwaves. • `"intensity"`: Scatter plot of duration vs. intensity (EHF peak). • `"trend"`: Bar chart of number of events per year.
`station_code`	Character. Optional. Filter by a specific station code.
`method`	Character. Optional. Filter by a specific method (e.g., "EHF", "INMET").
`year`	Numeric. Optional. Filter by a specific year (useful for calendar plots).
`interactive`	Logical. If `TRUE`, returns an interactive Plotly chart. If `FALSE`, returns a static ggplot2 chart. Default is `TRUE`.
`color_palette`	Character. Name of the ggsci color palette to use. Default is `"npg"`.
`lang`	Character. Language for labels and titles: `"en"` (English), `"pt"` (Portuguese), or `"es"` (Spanish). Default is `"en"`.
`save_plot`	Character. Optional file path to save the plot (e.g., "plot.html" or "plot.png").

Value

A ggplot or plotly object.

`sus_climate_uniplu` *Import UNIPLU-BR: Unified Brazilian Rainfall Dataset*

Description

`sus_climate_uniplu()` downloads, caches, and imports data from the **Unified Brazilian Rainfall Dataset (UNIPLU-BR)** — the most comprehensive national rainfall database for Brazil, covering **21,000+ gauges** from five monitoring networks over **140 years (1885–2025)**.

The function implements a full processing pipeline:

1. **Download**: Single 1.6 GB ZIP from Zenodo (first call only)
2. **Extraction**: Two Parquet files cached locally for instant re-use
3. **Standardization**: Column names aligned with climasus4r conventions
4. **Filtering**: By year, state (UF), and/or monitoring network
5. **Aggregation**: Sub-daily data optionally aggregated to daily/monthly/yearly totals

Usage

```
sus_climate_uniplu(
  years = NULL,
  uf = NULL,
  network = NULL,
  aggregate_to = "day",
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/climate",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Integer vector of year(s) to import. Examples: `2020`, `2020:2024`, `c(2015, 2020, 2024)`. Must be between 1885 and the current year. If `NULL` (default), imports the last 2 years.
`uf`	Character vector of Brazilian state codes (e.g., `"RN"`, `c("SP", "RJ", "MG")`). Case insensitive. If `NULL` (default), all 27 states are returned.
`network`	Character vector of monitoring network(s) to include. One or more of: `"Hidroweb"`, `"INMET"`, `"ICEA"`, `"CEMADEN"`, `"Telemetria"`. Case insensitive. If `NULL` (default), all networks are returned.
`aggregate_to`	Character. Temporal resolution for aggregating sub-daily observations into totals. One of: • `"day"` (default): Daily rainfall totals (`sum(rain_mm)`) • `"month"`: Monthly rainfall totals • `"year"`: Annual rainfall totals • `"none"`: Return raw observations at original resolution When `"none"`, `rainfall_mm` reflects the raw measurement interval (10 min to 1440 min depending on network). Use `"day"` for compatibility with `sus_climate_aggregate()`.
`use_cache`	Logical. If `TRUE` (default), skips download when the two Parquet files already exist in `cache_dir/uniplu/`. The full ZIP (~1.6 GB) is only downloaded once. Use `use_cache = FALSE` to force a re-download (e.g., after a new dataset version is published on Zenodo).
`cache_dir`	Character. Directory path for the disk cache. Default: `"~/.climasus4r_cache/climate"`. Created automatically if absent. Use `unlink(file.path(cache_dir, "uniplu"), recursive = TRUE)` to clear the UNIPLU cache.
`lang`	Character. Message language. One of: • `"pt"`: Portuguese (default) • `"en"`: English • `"es"`: Spanish
`verbose`	Logical. If `TRUE` (default), prints progress messages including cache status, download progress, filtering counts, and aggregation summary.

Value

A `climasus_df` object (subclass of `tibble`) at `stage = "climate"`, `type = "uniplu"`. Columns:

`station_code` Character. Unique gauge identifier (from `gauge_code`)

`station_name` Character. City name of the gauge (from `city`)

`uf` Character. State abbreviation (from `state`)

`latitude` Numeric. Decimal latitude, WGS84 (from `lat`)

`longitude` Numeric. Decimal longitude, WGS84 (from `long`)

`altitude` Numeric. Elevation in metres above sea level (from `elevation`)

`network` Character. Source monitoring network

`time_step_min` Integer. Original temporal resolution in minutes (e.g., 10, 60, 1440). Only present when `aggregate_to = "none"`.

`utc_offset` Numeric. Timezone offset from UTC (from `utc`). Only present when `aggregate_to = "none"`.

`date` POSIXct or Date. Observation timestamp / aggregation period

`rainfall_mm` Numeric. Precipitation in millimetres (sum over aggregation period, or raw value when `aggregate_to = "none"`)

Metadata (accessible via `sus_meta()`):

`stage` "climate"

`type` "uniplu"

`source` "UNIPLU-BR"

`doi` "10.5281/zenodo.18883358"

`years` Years retained after filtering

`ufs` States retained after filtering

`networks` Networks retained after filtering

`n_stations` Number of unique gauges

`n_observations` Total rows

`temporal` List with `start`, `end`, `unit` (aggregate_to)

Monitoring Networks

Network	Operator	Period	Resolution
Hidroweb	ANA	1885–2025	Daily (1440 min)
INMET daily	INMET	1889–2025	Daily (1440 min)
ICEA	ICEA	1951–2025	Various
CEMADEN	CEMADEN	2014–2025	10–15 min
Telemetria	Various	2014–2025	10–60 min
INMET sub-daily	INMET	2000–2025	60 min

Note

- **Data quality**: The dataset is strictly structurally standardized. **No outlier removal or physical consistency checks were applied** by the data producers. Extreme or implausible values from source agencies remain. Consider applying your own quality filters for scientific analyses.
- **Download size**: ~1.6 GB compressed. Only performed once; subsequent calls read from cached Parquet files (typically a few seconds).
- **Rainfall only**: UNIPLU-BR contains precipitation exclusively. For multi-variable climate data (temperature, humidity, radiation), use `sus_climate_inmet()`.
- **License**: CC-BY-4.0. Cite the dataset and paper when publishing.

References

Dataset: Das Neves Almeida, C., Bertrand, G. F., Lemos, F. C., et al. (2026). *Unified Brazilian Rainfall Dataset (UNIPLU-BR): A Standardized National Database of Point Precipitation from Major Brazilian Monitoring Networks (1885–2025)*. Zenodo. doi:10.5281/zenodo.18883358

Related paper: Das Neves Almeida, C., Bertrand, G. F., Lemos, F. C., et al. (2025). The design of the Brazilian Sub-Daily Rainfall dataset (BR-SDR): two decades of high-time-resolution data in Brazil. *Hydrological Sciences Journal*, 70(11), 1850–1862. doi:10.1080/02626667.2025.2506193

GitHub repository: https://github.com/LARHENA/UNIPLU-BR

See Also

- `sus_climate_inmet()` for multi-variable meteorological data (INMET)
- `sus_climate_aggregate()` for health-climate temporal integration
- `sus_climate_fill_inmet()` for ML-based gap filling

Examples

```
## Not run:
# Daily rainfall for Rio Grande do Norte, last 2 years (default)
rain_rn <- sus_climate_uniplu(uf = "RN")

# Multiple states, specific years, daily totals
rain_ne <- sus_climate_uniplu(
  years = 2015:2024,
  uf    = c("RN", "CE", "PB", "PE"),
  aggregate_to = "day"
)

# Only ANA/Hidroweb network, monthly totals
rain_hidroweb <- sus_climate_uniplu(
  years   = 2000:2020,
  network = "Hidroweb",
  aggregate_to = "month"
)
```

```
# Raw sub-daily data (no aggregation)
rain_raw <- sus_climate_uniplu(
  years        = 2023,
  uf           = "SP",
  network      = "CEMADEN",
  aggregate_to = "none"
)

# Check metadata
sus_meta(rain_rn)

# Inspect stations
dplyr::distinct(rain_rn, station_code, station_name, latitude, longitude)

# Integrate with health data (requires prior sus_spatial_join())
health_climate <- sus_climate_aggregate(
  health_data  = health_spatial,
  climate_data = rain_rn,
  climate_var  = "rainfall_mm",
  temporal_strategy = "moving_window",
  window_days  = 14
)

## End(Not run)
```

`sus_data_aggregate` *Aggregate Health Data into Time Series*

Description

Aggregates individual-level health data into time series counts by specified time units and grouping variables. This function is essential for preparing data for time series analysis, DLNM models, and other temporal epidemiological methods.

Usage

```
sus_data_aggregate(
  df,
  time_unit = "day",
  fun = "count",
  group_by = NULL,
  value_col = NULL,
  complete_dates = FALSE,
  date_col = NULL,
  backend = "arrow",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A data frame containing health data (output from `sus_data_standardize()`, or `sus_data_filter*()`).
`time_unit`	Character string specifying the temporal aggregation unit. **Standard units**: `"day"`, `"week"`, `"month"`, `"quarter"`, `"year"` **Multi-day/week/month**: `"2 days"`, `"5 days"` (pentads), `"14 days"` (fortnightly), `"3 months"` (trimester), `"6 months"` (semester). **Special**: `"season"` (Brazilian seasons: DJF, MAM, JJA, SON). Default is `"day"`.
`fun`	Character string or list of functions specifying the aggregation function(s). Options: `"count"` (default), `"sum"`, `"mean"`, `"median"`, `"min"`, `"max"`, `"sd"`, `"q25"` (25th percentile), `"q75"`, `"q95"`, and `"q99"`. Can also be a named list for multiple aggregations, e.g., `list(mean_temp = "mean", max_temp = "max")`.
`group_by`	Character vector with names of columns to group by (e.g., `c("sex", "age_group", "race")`). If `NULL` (default), aggregates across `"municipality_code"` records.
`value_col`	Character string with the name of the column to aggregate when using functions other than `"count"`. Required for `"sum"`, `"mean"`, etc. For example, `"temperature"`, `"precipitation"`, `"pm25"`.
`complete_dates`	Logical. If `TRUE` (default), fills in missing time periods with zero counts to create a complete time series without gaps.
`date_col`	Character string with the name of the date column to use for aggregation. If `NULL` (default), the function will attempt to auto-detect the date column based on common patterns.
`backend`	Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets). • `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows. • `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets. If not specified, the function may automatically choose the backend based on the input data type.
`lang`	Character string specifying the language for messages. Options: `"en"` (English), `"pt"` (Portuguese, default), `"es"` (Spanish).
`verbose`	Logical. If `TRUE` (default), prints progress messages.

Details

New Features:

- **Multiple aggregation functions**: Beyond counting, you can now calculate mean, sum, median, percentiles, etc., useful for climate and environmental data.
- **Smart column naming**: The aggregated column is automatically named based on the health system (e.g., `n_deaths` for SIM, `n_hospitalizations` for SIH-RD, `n_births` for SINASC, `n_cases` for SINAN, `n_procedures` for SIA, and `n_establishments`, for CNES).

Epidemiological Use Cases:

- **Daily/Weekly**: Standard time series analysis, DLNM for short-term effects
- **Pentads (5 days)**: Heat wave analysis, smoothing daily noise
- **Fortnightly (14 days)**: Diseases with longer incubation periods
- **Monthly**: Seasonal patterns, long-term trends
- **Quarterly**: SUS management reports, policy evaluation
- **Seasonal**: Dengue, Influenza, respiratory diseases aligned with Brazilian climate
- **Yearly**: Long-term trend analysis, climate change impacts

Brazilian Seasons (when `time_unit = "season"`):

- **Summer (Verao)**: December-January-February (DJF)
- **Autumn (Outono)**: March-April-May (MAM)
- **Winter (Inverno)**: June-July-August (JJA)
- **Spring (Primavera)**: September-October-November (SON)

Value

A tibble with aggregated data containing:

- `date`: The aggregated date (start of period)
- Grouping columns (if `group_by` was specified)
- Aggregated value column(s) with smart names based on system and function

Examples

```
## Not run:
library(climasus4r)

# Basic daily aggregation
df_daily <- sus_data_import(uf = "SP", year = 2023, system = "SIM-DO") %>%
  sus_data_standardize() %>%
  sus_data_filter_cid(disease_group = "respiratory") %>%
  sus_data_aggregate(time_unit = "day")

# Pentad aggregation (5-day periods) for heat wave analysis
df_pentad <- sus_data_aggregate(df, time_unit = "5 days")

# Fortnightly aggregation for diseases with longer incubation
df_fortnightly <- sus_data_aggregate(df, time_unit = "14 days")
```

```
# Monthly aggregation by municipality
df_monthly <- sus_data_aggregate(
  df,
  time_unit = "month",
  group_by = c("race", "sex"),
  lang = "pt"
)

# Quarterly aggregation for SUS reports
df_quarterly <- sus_data_aggregate(df, time_unit = "quarter")

# Seasonal aggregation for dengue analysis (Brazilian seasons)
df_seasonal <- sus_data_aggregate(
  df,
  time_unit = "season"
)

# Weekly aggregation by age group and sex
df_weekly <- sus_data_aggregate(
  df,
  time_unit = "week",
  group_by = c("age_group", "sex") #age_group comes from `sus_data_create_variables()`
)

## End(Not run)
```

`sus_data_cid_select` *Interactive Disease Groups Explorer*

Description

Opens an interactive HTML interface to explore disease groups and climate factors for use with `sus_data_filter_cid()`.

Usage

```
sus_data_cid_select(
  lang = "pt",
  output = "browser",
  filter_climate = FALSE,
  verbose = TRUE
)
```

Arguments

`lang`	Character string specifying language. Options: `"pt"` (Portuguese, default), `"en"` (English), `"es"` (Spanish).
`output`	Character string specifying output format. Options:

- `"browser"` - Interactive HTML in web browser (default)
- `"console"` - Simple list in R console

`filter_climate`
: Logical. If `TRUE`, shows only climate-sensitive diseases.

`verbose`
: Logical. If `TRUE` (default), prints informative messages.

Details

This function helps users discover available disease groups without needing to know specific ICD-10 codes. The interactive interface allows:

- Multi-select disease groups (Ctrl/Cmd + Click)
- Filter by climate factors (temperature, precipitation, etc.)
- Copy group names for use in `sus_data_filter_cid()`
- View ICD-10 codes and descriptions

Value

Depending on `output`:

- `"browser"`: Opens HTML interface, returns invisible data.frame
- `"console"`: Prints summary, returns invisible data.frame

Examples

```
## Not run:
# Open interactive explorer
sus_data_cid_select()

# Explore only climate-sensitive diseases
sus_data_cid_select(filter_climate = TRUE)

# Get disease group names for programmatic use
groups <- sus_data_cid_select(
  lang = "pt"
)

# Use in sus_data_filter_cid
data <- sus_data_filter_cid(
  df = my_data,
  disease_group = groups[1]
)

## End(Not run)
```

`sus_data_clean_encoding` *Detect and correct character encoding issues*

Description

This function scans text columns in a data.frame and corrects common encoding problems (e.g., "Sao Paulo") that occur when Latin1 data is incorrectly read as UTF-8. It acts as a final auditor to ensure all text data is properly encoded, complementing the preprocessing done by `microdatasus`. Supports multilingual output messages (English, Portuguese, Spanish).

Usage

```
sus_data_clean_encoding(df, backend = "arrow", lang = "pt", verbose = TRUE)
```

Arguments

`df`	A `data.frame` or `tibble` to be cleaned.
`backend`	Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets). • `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows. • `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets. If not specified, the function may automatically choose the backend based on the input data type.
`lang`	Character. Language for UI messages. Options: "en" (English), "pt" (Portuguese, default), "es" (Spanish).
`verbose`	Logical. If TRUE, prints a report of columns checked and corrected. Default is TRUE.

Value

A `data.frame` with corrected text columns.

Examples

```
## Not run:
# Create a sample dataset with encoding issues
# In real data, this might happen with Brazilian Portuguese text
df_problem <- data.frame(
```

```
  id = 1:3,
  city = c("Sao Paulo", "Rio de Janeiro", "Belo Horizonte"),
  state = c("SP", "RJ", "MG"),
  stringsAsFactors = FALSE
)

# Simulate encoding issue (for demonstration only)
# In practice, this happens when Latin1 text is read as UTF-8

# Correct encoding with English messages
df_clean_en <- sus_data_clean_encoding(df_problem, lang = "en")

# Correct encoding with Portuguese messages
df_clean_pt <- sus_data_clean_encoding(df_problem, lang = "pt")

# Correct encoding with Spanish messages
df_clean_es <- sus_data_clean_encoding(df_problem, lang = "es")

# Use in a pipeline
df_clean <- sus_data_import(uf = "RJ", year = 2022, system = "SIM") %>%
  sus_data_clean_encoding(lang = "pt")

## End(Not run)
```

`sus_data_create_variables`

Create Derived Variables for Epidemiological Analysis

Description

Creates commonly used derived variables from health data across ALL SUS systems (SIM, SINAN, SIH, SIA, SINASC, CNES), including age groups, calendar variables, and other epidemiologically relevant categorizations. Features age calculation that handles different data formats across systems.

Usage

```
sus_data_create_variables(
  df,
  create_age_groups = TRUE,
  age_breaks = c(0, 5, 15, 60, Inf),
  age_labels = NULL,
  create_calendar_vars = TRUE,
  create_climate_vars = TRUE,
  climate_region = NULL,
  date_col = NULL,
  age_col = NULL,
  hemisphere = "south",
  backend = "arrow",
```

```
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A data frame containing health data from any SUS system.
`create_age_groups`	Logical. If `TRUE`, derives age-based variables. When enabled, the function will attempt to determine age using the following hierarchy (in order of preference): 1. An existing age column supplied via `age_col`. 2. Calculation from birth date and event date columns (gold standard). 3. Decoding of DATASUS age code variables (fallback method). If age is successfully determined, the following variables may be created: • **User-defined age groups**: A categorical variable based on `age_breaks` and `age_labels`, created using `cut()`. This variable is always created when `create_age_groups = TRUE`. • **Climate risk age group**: A coarse age classification designed for climate–health analyses, with three categories: **0–4** High risk **5–64** Standard risk **65+** High risk The variable name depends on the selected language: – English: `climate_risk_group` – Portuguese: `grupo_risco_climatico` – Spanish: `grupo_riesgo_climatico` • **IBGE quinquennial age groups**: A standardized 17-group age classification following the Brazilian Institute of Geography and Statistics (IBGE) quinquennial structure: `0-4, 5-9, ..., 75-79, 80+`. This variable ensures national and international comparability and is particularly useful for demographic and epidemiological analyses. Variable names by language: – English: `ibge_age_group` – Portuguese: `faixa_etaria_ibge` – Spanish: `grupo_edad_ibge` All age group variables are created only if age can be reliably determined. If age cannot be inferred, the function will stop with an informative error.
`age_breaks`	Numeric vector specifying the breakpoints for age groups. Default is `c(0, 5, 15, 65, Inf)` for standard epidemiological categories.
`age_labels`	Character vector with labels for age groups. If `NULL` (default), generates labels automatically from breaks.
`create_calendar_vars`	Logical. If `TRUE`, creates calendar variables (day of week, month, season, etc.). Default is `FALSE`.

`create_climate_vars`
: Logical. Se `TRUE`, cria variaveis climaticas e sazonais.

`climate_region`
: Character. Regiao climatica para calculos sazonais. Opcoes: ”norte”, ”nordeste”, ”centro-oeste”, ”sudeste”, ”sul”.

`date_col`
: Character string with the name of the date column. If `NULL` (default), auto-detects the date column.

`age_col`
: Character string with the name of the age column (in years). If `NULL` (default), auto-detects and calculates age using hierarchical logic:

  1. Direct age column (if exists)
  2. Calculate from dates (event_date - birth_date)
  3. Decode DATASUS age codes (NU_IDADE_N, IDADE)

`hemisphere`
: Character string specifying the hemisphere for season calculation. Options: `"south"` (default, for Brazil), `"north"`.

`backend`
: Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets).

  - `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows.
  - `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets.

  If not specified, the function may automatically choose the backend based on the input data type.

`lang`
: Character string specifying the language for variable labels and messages. Options: `"en"` (English), `"pt"` (Portuguese, default), `"es"` (Spanish).

`verbose`
: Logical. If `TRUE` (default), prints progress messages.

Details

Age Calculation (Hierarchical Logic):

The function uses a 3-tier hierarchy to ensure age is calculated correctly across all SUS systems:

1. **Direct Age Column** (Fastest): If a column with age in years already exists (common in SIM after microdatasus processing), uses it directly.
2. **Date Calculation** (Gold Standard): If birth date and event date are available, calculates exact age as: `interval(birth_date, event_date) / years(1)`. This is the most accurate method and works for:
   - SINAN: Has `DTNASC` and `DT_NOTIFIC`
   - SIH: Has `NASC` and `DT_INTER`
   - SINASC: Has `DTNASC` (mother) and `DTNASC` (newborn)

3. **DATASUS Code Decoder** (Fallback): If dates are missing (common in anonymized data), decodes the composite age code used by DATASUS:
   - Codes starting with `1`: Hours (converted to 0 years)
   - Codes starting with `2`: Days (converted to 0 years)
   - Codes starting with `3`: Months (converted to 0 years for <12 months)
   - Codes starting with `4`: Years (e.g., 4035 = 35 years)
   - Codes starting with `5`: 100+ years (e.g., 5105 = 105 years)

Age Groups: Creates a factor variable `age_group` based on the specified breaks and labels. Common epidemiological categories:

- Pediatric: 0-4, 5-14, 15-19
- Adult: 20-39, 40-59, 60+
- Elderly: 65-74, 75-84, 85+
- Climate-Health: 0-4, 5-64, 65+ (vulnerable populations)

Calendar Variables (when `create_calendar_vars = TRUE`):

- `day_of_week`: Day of the week (1 = Monday, 7 = Sunday)
- `day_of_week_name`: Day name (e.g., "Monday", "Segunda-feira")
- `month`: Month number (1-12)
- `month_name`: Month name (e.g., "January", "Janeiro")
- `year`: Year
- `quarter`: Quarter (1-4)
- `season`: Season (Summer, Autumn, Winter, Spring)
- `is_weekend`: Logical indicating if date is weekend
- `day_of_year`: Day of year (1-365/366)
- `semester`: Semester (1 or 2)

Seasons are calculated based on hemisphere:

- **Southern Hemisphere** (Brazil): Summer (Dec-Feb), Autumn (Mar-May), Winter (Jun-Aug), Spring (Sep-Nov)
- **Northern Hemisphere**: Summer (Jun-Aug), Autumn (Sep-Nov), Winter (Dec-Feb), Spring (Mar-May)

Value

The input data frame with additional columns for the created variables.

Examples

```
## Not run:
library(climasus4r)

# ===== EXAMPLE 1: SIM (Mortality) - Age already calculated =====
df_sim <- sus_data_import(uf = "SP", year = 2023, system = "SIM-DO") |>
  sus_data_standardize(lang = "en") |>
  sus_data_create_variables(
    create_age_groups = TRUE,
    age_breaks = c(0, 5, 65, Inf),
    age_labels = c("0-4", "5-64", "65+"),
    create_calendar_vars = TRUE,
    lang = "en"
  )
# Uses direct age column (fastest)

# ===== EXAMPLE 2: SINAN (Dengue) - Calculate from dates =====
df_sinan <- sus_data_import(uf = "RJ", year = 2023, system = "SINAN-DENGUE") |>
  sus_data_standardize(lang = "pt") |>
  sus_data_create_variables(
    create_age_groups = TRUE,
    age_breaks = c(0, 15, 60, Inf),
    create_calendar_vars = TRUE,
    lang = "pt"
  )
# Calculates age from DTNASC and DT_NOTIFIC (gold standard)

# ===== EXAMPLE 3: SIH (Hospitalizations) - Decode age codes =====
df_sih <- sus_data_import(uf = "MG", year = 2023, system = "SIH-RD") |>
  sus_data_standardize(lang = "es") |>
  sus_data_create_variables(
    create_age_groups = TRUE,
    age_breaks = c(0, 18, 60, Inf),
    age_labels = c("0-17", "18-59", "60+"),
    create_calendar_vars = TRUE,
    lang = "es"
  )
# Decodes DATASUS age codes if dates are missing (fallback)

# ===== EXAMPLE 4: Custom age groups for elderly analysis =====
df_elderly <- sus_data_create_variables(
  df,
  create_age_groups = TRUE,
  age_breaks = c(60, 70, 80, 90, Inf),
  age_labels = c("60-69", "70-79", "80-89", "90+"),
  lang = "pt"
)

# ===== EXAMPLE 5: Calendar variables and climate variables =====
df_calendar_climate <- sus_data_create_variables(
  df,
  create_calendar_vars = TRUE,
```

```
  create_semester = TRUE,
  create_climate_vars = TRUE,
  climate_region = "Norte"
)

## End(Not run)
```

`sus_data_export` *Export Processed Health Data with Metadata*

Description

Exports processed health data to a file with optional metadata documentation to ensure reproducibility. Supports multiple file formats optimized for different use cases, including GeoArrow/GeoParquet for spatial data.

Usage

```
sus_data_export(
  df,
  file_path,
  format = NULL,
  include_metadata = TRUE,
  metadata = NULL,
  compress = TRUE,
  compression_level = 6,
  overwrite = FALSE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A data frame or sf object containing the processed health data to export.
`file_path`	Character string specifying the output file path. The file extension determines the format if `format` is not explicitly specified.
`format`	Character string specifying the output format. Options: `"rds"` (default, R binary format), `"arrow"` (Apache Arrow/Parquet), `"geoparquet"` (GeoArrow/GeoParquet for spatial data), `"shapefile"` (ESRI Shapefile), `"GeoPackage"`, and `"csv"` (comma-separated values). If `NULL`, infers from `file_path` extension and data type (auto-detects sf objects).
`include_metadata`	Logical. If `TRUE` (default), saves a companion metadata file (`.txt` or `.json`) with processing information.
`metadata`	Named list containing custom metadata to save. Common fields: • `source_system`: Health system (e.g., "SIM-DO")

- `states`: Vector of state codes
- `years`: Vector of years
- `filters_applied`: Description of filters
- `disease_groups`: Disease groups included
- `processing_date`: Date of processing
- `package_version`: climasus4r version
- `author`: Analyst name
- `notes`: Additional notes

If `NULL`, generates basic metadata automatically.

`compress` Logical. If `TRUE` (default for RDS and Arrow), compresses the output file. Compression level can be specified for some formats.

`compression_level`
Integer specifying compression level (1-9). Higher values = smaller files but slower. Default is 6. Only applies to formats that support compression.

`overwrite` Logical. If `TRUE`, overwrites existing files. If `FALSE` (default), stops with an error if file exists.

`lang` Character string specifying the language for messages. Options: `"en"` (English, default), `"pt"` (Portuguese), `"es"` (Spanish).

`verbose` Logical. If `TRUE` (default), prints export summary.

Details

File Formats:

- **RDS** (`.rds`): Native R format. Fast, compressed, preserves all R object attributes. Best for R-only workflows.
- **Parquet** (`.parquet`): Columnar format. Excellent compression, fast reading, language-agnostic. Best for large datasets and interoperability with Python, Spark, etc.
- **GeoParquet** (`.geoparquet`, `.parquet` for sf objects): Optimized columnar format for spatial data. Combines benefits of Parquet with efficient geometry storage. 50-90% smaller than shapefiles, 10-100x faster.
- **Shapefile** (`.shp` or `.gpkg`): Traditional GIS format. Widely supported but inefficient for large datasets. Multiple files generated (.shp, .shx, .dbf, etc.).
- **CSV** (`.csv`): Universal text format. Human-readable, compatible with all software. Best for sharing with non-R users. Larger file size. Note: Geometries are exported as WKT (Well-Known Text) for spatial data.

Automatic Format Detection: If `format = NULL`, the function automatically detects the best format:

- If input is an `sf` object and extension is `.parquet` → `"geoparquet"`
- If input is an `sf` object and extension is `.shp` → `"shapefile"`
- Otherwise, infers from file extension

Spatial Data Export: When exporting `sf` objects (spatial data from `sus_spatial_join()`):

- **Recommended**: Use GeoParquet format for optimal performance
- GeoParquet preserves CRS, geometry types, and all attributes
- Compatible with QGIS, Python (geopandas), and other GIS software
- Significantly faster and smaller than shapefiles

Value

Invisibly returns the file path of the exported data. If metadata was saved, also returns the metadata file path as an attribute.

Examples

```
## Not run:
library(climasus4r)

# Export regular data frame to RDS
sus_data_export(df_final, "output/data.rds")

# Export spatial data to GeoParquet (RECOMMENDED)
sf_result <- sus_spatial_join(df, level = "munic")
sus_data_export(
  sf_result,
  file_path = "output/spatial_data.geoparquet",
  format = "geoparquet"  # Auto-detected if extension is .parquet
)

# Export spatial data to Shapefile (traditional)
sus_data_export(
  sf_result,
  file_path = "output/spatial_data.shp",
  format = "shapefile"
)

# Export to Arrow with custom metadata
sus_data_export(
  df_final,
  file_path = "output/respiratory_sp_2023.parquet",
  format = "parquet",
  metadata = list(
    source_system = "SIM-DO",
    states = "SP",
    years = 2023,
    disease_groups = "respiratory",
    author = "Max Anjos"
  )
)

## End(Not run)
```

`sus_data_filter_cid` *Filter SUS health data by ICD-10 codes or disease groups with multilingual support*

Description

Filters Brazilian Unified Health System (SUS) data based on ICD-10 codes. (International Classification of Diseases, 10th Revision) or predefined epidemiological disease groups. This function supports complex filtering scenarios including specific codes, code ranges, chapters, and 50+ disease groups relevant to epidemiological research in Brazil. Includes specialized support for SUS-specific coding practices and multilingual interface (English, Portuguese, Spanish).

Usage

```
sus_data_filter_cid(
  df,
  icd_codes = NULL,
  disease_group = NULL,
  icd_column = NULL,
  match_type = "starts_with",
  backend = "arrow",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df` A `data.frame` or `tibble` containing SUS health data with ICD-10 codes. Typically obtained from DATASUS systems (SIM, SIH, SINAN). The data should contain at least one column with ICD-10 codes in standard format (e.g., "A00.0", "I10", "C50.9").

`icd_codes` A character vector of ICD-10 codes, ranges, or categories. Multiple syntaxes are supported:

Basic filtering:

- Single code: `"J18.9"` (Pneumonia, unspecified)
- Multiple codes: `c("I10", "I11.0", "I11.9")` (Hypertensive diseases)

Range filtering:

- Complete range: `"J00-J99"` (All diseases of respiratory system)
- Partial range: `"J09-J18"` (Influenza and pneumonia only)

Chapter filtering:

- Full chapter: `"I"` (All diseases of circulatory system, codes I00-I99)
- Chapter group: `"C00-D48"` (All neoplasms)

Special SUS categories:

- `"causas_externas"` or `"external_causes"`: V01-Y98 (External causes)
- `"causas_maternas"` or `"maternal_causes"`: O00-O99 (Pregnancy/childbirth)
- `"causas_infantis"` or `"infant_causes"`: P00-P96 (Perinatal conditions)
- `"doencas_infecciosas"` or `"infectious_diseases"`: A00-B99 (Infectious)
- `"doencas_respiratorias"` or `"respiratory_diseases"`: J00-J99 (Respiratory)
- `"doencas_cardiovasculares"` or `"cardiovascular_diseases"`: I00-I99 (Cardio)
- `"neoplasias"` or `"neoplasms"`: C00-D48 (Neoplasms)

Brazilian epidemiological priorities:

- `"dengue_like"`: A90-A91 (Dengue) + A92.0-A92.9 (Other viral fevers)
- `"zika_chik"`: A92.8 (Zika) + A92.0 (Chikungunya)
- `"tb_respiratoria"`: A15-A16 (Respiratory tuberculosis)
- `"covid19"`: U07.1 (COVID-19) + U07.2 (Suspected COVID-19)
- `"violencia"`: X85-Y09 (Assault) + Y35-Y36 (Legal intervention)

Note: Either `icd_codes` OR `disease_group` must be provided, not both.

`disease_group` Character. Name of predefined disease group (e.g., "dengue", "cardiovascular", "respiratory"). Use `list_disease_groups()` to see all available groups. Mutually exclusive with `icd_codes`.

`icd_column` Character. Name of the column containing ICD-10 codes. If NULL (default), the function attempts auto-detection from common SUS column names in this priority order:

1. `"CAUSABAS"` - Underlying cause (primary cause of death in SIM)
2. `"DIAG_PRINC"` - Main diagnosis (SIH hospitalizations)
3. `"DIAG_SECUN"` - Secondary diagnosis
4. `"CAUSAOBITO"` - Cause of death (alternative SIM field)
5. `"DIAGNOSTIC"` - Diagnosis (SINAN notifiable diseases)
6. `"linha_a"` through `"linha_f"` - Multiple cause lines (SIM)

`match_type` Character. Type of matching algorithm:

- `"exact"`: Exact code match (e.g., "I10" matches only "I10")
- `"starts_with"`: Match codes starting with pattern (default, e.g., "I10" matches "I10", "I10.0", "I10.9")
- `"range"`: Match codes within specified ranges (e.g., "I10-I15" matches I10-I15.9)
- `"chapter"`: Match entire ICD-10 chapter (e.g., "I" matches I00-I99.9)
- `"fuzzy"`: Allow for common SUS coding variations (e.g., "I10" matches "I10", "I10 ", "I10X")

`backend` Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets).

- `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows.
- `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets.

If not specified, the function may automatically choose the backend based on the input data type.

`lang` Character. Language for user interface messages, warnings, and documentation. Options:

- `"en"`: English
- `"pt"`: Portuguese (default, recommended for Brazilian users)
- `"es"`: Spanish Affects all console output and documentation of matched codes.

`verbose` Logical. If TRUE (default), prints detailed filtering information including: records processed, match statistics, common coding issues detected, and time elapsed.

Details

Automatic ICD Column Detection:

The function automatically identifies the appropriate ICD column based on the health system

Disease Groups:

The function includes 50+ predefined epidemiological groups organized by:

- **ICD Chapters**: All major disease categories (A00-Y98)
- **Climate-Sensitive Diseases**: Vector-borne, waterborne, heat-related, etc.
- **Specific Conditions**: Dengue, malaria, cardiovascular, respiratory, etc.
- **Syndromic Groups**: Fever, respiratory, diarrheal syndromes
- **Age-Specific Groups**: Pediatric, elderly populations

Each group includes:

- ICD code ranges
- Multilingual labels and descriptions
- Climate sensitivity flag
- Associated climate factors

Use `list_disease_groups()` to see all available groups and their details.

Value

A filtered `data.frame` or `tibble` containing only records matching the specified ICD-10 codes or disease group. The output preserves all original columns

References

1. World Health Organization. (2016). ICD-10 International Statistical Classification of Diseases and Related Health Problems. 10th Revision.
2. Brazilian Ministry of Health. (2023). Classificacao Estatistica Internacional de Doencas e Problemas Relacionados a Saude - CID-10. DATASUS. http://datasus.saude.gov.br/cid10

Examples

```
## Not run:
# Example 1: Filter by explicit ICD codes
df_cardio <- sus_data_filter_cid(
  sim_data,
  icd_codes = "I00-I99",
  lang = "en"
)

# Example 2: Filter by disease group (easier!)
df_dengue <- sus_data_filter_cid(
  sinan_data,
  disease_group = "dengue",
  lang = "pt"
)

# Example 3: Climate-sensitive diseases
df_climate <- sus_data_filter_cid(
  sim_data,
  disease_group = "climate_sensitive_all",
  lang = "en"
)

# Example 4: Multiple specific codes
df_ami_stroke <- sus_data_filter_cid(
  sih_data,
  icd_codes = c("I21", "I22", "I63", "I64"),
  lang = "es"
)

# Example 5: Respiratory diseases in children
df_pediatric <- sus_data_filter_cid(
  sih_data,
  disease_group = "pediatric_respiratory",
  lang = "pt"
)

# List all available disease groups
list_disease_groups(lang = "pt")

# List only climate-sensitive groups
list_disease_groups(climate_sensitive_only = TRUE, lang = "en")

# Get details about a specific group
```

```
get_disease_group_details("dengue", lang = "pt")

## End(Not run)
```

`sus_data_filter_demographics` *Filter Health Data by Demographic Variables*

Description

Filters health data based on demographic characteristics such as sex, race, age range, education level, region, and municipality. This function complements `sus_data_filter_cid()` by enabling stratified analyses by population subgroups.

Usage

```
sus_data_filter_demographics(
  df,
  sex = NULL,
  race = NULL,
  age_range = NULL,
  education = NULL,
  region = NULL,
  city = NULL,
  municipality_code = NULL,
  drop_ignored = FALSE,
  backend = "arrow",
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/spatial",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` object containing health data (output of the `climasus4r` pipeline).
`sex`	Character vector specifying sex categories to include. Accepts values in English, Portuguese, or Spanish (e.g., `"Male"`, `"Masculino"`, `"Masculino"`). If `NULL` (default), includes all sexes.
`race`	Character vector specifying race/color categories to include. Accepts IBGE standard categories in multiple languages. If `NULL` (default), includes all races.
`age_range`	Numeric vector of length 2 specifying the age range `c(min_age, max_age)`. Use `Inf` for no upper limit. If `NULL` (default), includes all ages.

`education` Character vector specifying education levels to include. If `NULL` (default), includes all education levels.

`region` A string indicating a predefined group of states or regions (supports multilingual names PT, EN, ES). See Details.

`city` Character vector of municipality names (e.g., `"Sao Paulo"`, `"Natal"`) or IBGE codes (6 or 7-digit, e.g., `"3550308"`, `"2408102"`). Case-insensitive; accents are normalised for matching. Partial typos trigger fuzzy suggestions. If `NULL` (default), no additional municipality filter is applied. Results are merged (union) with any codes in `municipality_code`.

`municipality_code` Character or numeric vector specifying municipality codes (IBGE 6 or 7-digit codes) to include. If `NULL` (default), includes all municipalities.

`drop_ignored` Logical. If `TRUE`, explicitly removes rows where demographic variables (sex, race, education) contain missing values (`NA`) or DATASUS ignored codes (e.g., `"9"`, `"Ignorado"`). Default is `FALSE`.

`backend` Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets).

- `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows.
- `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets.

If not specified, the function may automatically choose the backend based on the input data type.

`use_cache` Logical. If `TRUE` (default), uses cached spatial data to avoid re-downloads and improve performance. Only relevant when `city` is provided.

`cache_dir` Character string specifying the directory to store cached files. Default is `"~/.climasus4r_cache/spatial"`.

`lang` Character string specifying the language for messages. Options: `"en"` (English), `"pt"` (Portuguese, default), `"es"` (Spanish).

`verbose` Logical. If `TRUE` (default), prints filtering summary.

Details

The function automatically detects column names in different languages and standardisations. It handles both original DATASUS column names and standardised names from `sus_data_standardize()`.

Sex categories (case-insensitive):

- English: `"Male"`, `"Female"`
- Portuguese: `"Masculino"`, `"Feminino"`
- Spanish: `"Masculino"`, `"Femenino"`

Race/Color categories (IBGE standard):

- English: `"White"`, `"Black"`, `"Yellow"`, `"Brown"`, `"Indigenous"`
- Portuguese: `"Branca"`, `"Preta"`, `"Amarela"`, `"Parda"`, `"Indigena"`
- Spanish: `"Blanca"`, `"Negra"`, `"Amarilla"`, `"Parda"`, `"Indigena"`

IBGE Macro-regions:

- `"norte"`: c("AC", "AP", "AM", "PA", "RO", "RR", "TO")
- `"nordeste"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE")
- `"centro_oeste"`: c("DF", "GO", "MT", "MS")
- `"sudeste"`: c("ES", "MG", "RJ", "SP")
- `"sul"`: c("PR", "RS", "SC")

Biomes (Ecological Borders):

- `"amazonia_legal"`: c("AC", "AP", "AM", "PA", "RO", "RR", "MT", "MA", "TO")
- `"mata_atlantica"`: c("AL", "BA", "CE", "ES", "GO", "MA", "MG", "MS", "PB", "PE", "PI", "PR", "RJ", "RN", "RS", "SC", "SE", "SP")
- `"caatinga"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE", "MG")
- `"cerrado"`: c("BA", "DF", "GO", "MA", "MG", "MS", "MT", "PA", "PI", "PR", "RO", "SP", "TO")
- `"pantanal"`: c("MT", "MS")
- `"pampa"`: c("RS")

Hydrography & Climate:

- `"bacia_amazonia"`: c("AC", "AM", "AP", "MT", "PA", "RO", "RR")
- `"bacia_sao_francisco"`: c("AL", "BA", "DF", "GO", "MG", "PE", "SE")
- `"bacia_parana"`: c("GO", "MG", "MS", "PR", "SP")
- `"bacia_tocantins"`: c("GO", "MA", "PA", "TO")
- `"semi_arido"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE", "MG")

Health, Agriculture & Geopolitics:

- `"matopiba"`: c("MA", "TO", "PI", "BA")
- `"arco_desmatamento"`: c("RO", "AC", "AM", "PA", "MT", "MA")
- `"dengue_hyperendemic"`: c("GO", "MS", "MT", "PR", "RJ", "SP")
- `"sudene"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE", "MG", "ES")
- `"fronteira_brasil"`: c("AC", "AM", "AP", "MT", "MS", "PA", "PR", "RO", "RR", "RS", "SC")

Value

A `climasus_df` filtered by all specified demographic criteria.

Examples

```
## Not run:
library(climasus4r)

# Filter by sex only
df_women <- sus_data_filter_demographics(df, sex = "Female")

# Filter by age range (elderly, 65+)
df_elderly <- sus_data_filter_demographics(df, age_range = c(65, Inf))

# Filter by city name (with accent tolerance)
df_natal <- sus_data_filter_demographics(df, city = "Natal", lang = "pt")

# Filter by multiple city names
df_capitals <- sus_data_filter_demographics(
  df,
  city = c("Sao Paulo", "Rio de Janeiro", "Fortaleza"), #Usar acentos se preferir
  lang = "pt"
)

# Mix: city names + explicit codes (union)
df_subset <- sus_data_filter_demographics(
  df,
  city            = "Natal",
  municipality_code = "3550308",
  lang            = "pt"
)

# Complex filtering
df_children <- sus_data_filter_demographics(
  df,
  age_range = c(0, 5),
  region    = "Norte",
  lang      = "pt"
)

## End(Not run)
```

`sus_data_import` *Import and preprocess data from DATASUS with intelligent caching*

Description

This function acts as a wrapper for `microdatasus::fetch_datasus`, simplifying the download and reading of data from Brazilian public health information systems (SIM, SINAN, SIH, SIA, CNES, SINASC). It includes parallel processing, caching, and user-friendly CLI feedback.

Usage

```
sus_data_import(
  uf = NULL,
  region = NULL,
  year,
  month = NULL,
  system,
  backend = "arrow",
  city = NULL,
  municipality_code = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/data",
  force_redownload = FALSE,
  parallel = FALSE,
  workers = 4,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`uf` A string or vector of strings with state abbreviations (**igonered if 'region' is provided**) (e.g., "AM", c("SP", "RJ")). Valid UF codes: AC, AL, AP, AM, BA, CE, DF, ES, GO, MA, MT, MS, MG, PA, PB, PR, PE, PI, RJ, RN, RS, RO, RR, SC, SP, SE, TO.

`region` A string indicating a predefined group of states (supports multilingual names PT, EN, ES). Available regions:

IBGE Macro-regions:

- `"norte"`: c("AC", "AP", "AM", "PA", "RO", "RR", "TO")
- `"nordeste"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE")
- `"centro_oeste"`: c("DF", "GO", "MT", "MS")
- `"sudeste"`: c("ES", "MG", "RJ", "SP")
- `"sul"`: c("PR", "RS", "SC")

Biomes (Ecological Borders):

- `"amazonia_legal"`: c("AC", "AP", "AM", "PA", "RO", "RR", "MT", "MA", "TO")
- `"mata_atlantica"`: c("AL", "BA", "CE", "ES", "GO", "MA", "MG", "MS", "PB", "PE", "PI", "PR", "RJ", "RN", "RS", "SC", "SE", "SP")
- `"caatinga"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE", "MG")
- `"cerrado"`: c("BA", "DF", "GO", "MA", "MG", "MS", "MT", "PA", "PI", "PR", "RO", "SP", "TO")
- `"pantanal"`: c("MT", "MS")
- `"pampa"`: c("RS")

Hydrography & Climate:

- `"bacia_amazonia"`: c("AC", "AM", "AP", "MT", "PA", "RO", "RR")
- `"bacia_sao_francisco"`: c("AL", "BA", "DF", "GO", "MG", "PE", "SE")
- `"bacia_parana"`: c("GO", "MG", "MS", "PR", "SP")
- `"bacia_tocantins"`: c("GO", "MA", "PA", "TO")
- `"semi_arido"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE", "MG")

Health, Agriculture & Geopolitics:

- `"matopiba"`: c("MA", "TO", "PI", "BA")
- `"arco_desmatamento"`: c("RO", "AC", "AM", "PA", "MT", "MA")
- `"dengue_hyperendemic"`: c("GO", "MS", "MT", "PR", "RJ", "SP")
- `"sudene"`: c("AL", "BA", "CE", "MA", "PB", "PE", "PI", "RN", "SE", "MG", "ES")
- `"fronteira_brasil"`: c("AC", "AM", "AP", "MT", "MS", "PA", "PR", "RO", "RR", "RS", "SC")

`year` An integer or vector of integers with the desired years (4 digits).

`month` An integer or vector of integers with the desired months (1-12). This argument is only used with monthly-based health information systems: SIH, CNES, and SIA. For annual systems (SIM, SINAN, SINASC), this parameter is ignored.

`system` A string indicating the information system. Available systems:

Mortality Systems (SIM - Mortality Information System):

- `"SIM-DO"`: Death certificates (Declaracoes de Obito) - Complete dataset
- `"SIM-DOFET"`: Fetal deaths (Obitos Fetais)
- `"SIM-DOEXT"`: External causes deaths (Obitos por Causas Externas)
- `"SIM-DOINF"`: Infant deaths (Obitos Infantis)
- `"SIM-DOMAT"`: Maternal deaths (Obitos Maternos)

Hospitalization Systems (SIH - Hospital Information System):

- `"SIH-RD"`: Hospital Admission Authorizations (AIH - Autorizacoes de Internacao Hospitalar)
- `"SIH-RJ"`: Hospital Admission Authorizations - Rio de Janeiro specific
- `"SIH-SP"`: Hospital Admission Authorizations - Sao Paulo specific
- `"SIH-ER"`: Emergency Room Records (Prontuarios de Emergencia)

Notifiable Diseases (SINAN - Notifiable Diseases Information System):

- `"SINAN-DENGUE"`: Dengue fever cases
- `"SINAN-CHIKUNGUNYA"`: Chikungunya cases
- `"SINAN-ZIKA"`: Zika virus cases
- `"SINAN-MALARIA"`: Malaria cases
- `"SINAN-CHAGAS"`: Chagas disease cases
- `"SINAN-LEISHMANIOSE-VISCERAL"`: Visceral leishmaniasis cases

- `"SINAN-LEISHMANIOSE-TEGUMENTAR"`: Cutaneous leishmaniasis cases
- `"SINAN-LEPTOSPIROSE"`: Leptospirosis cases

Outpatient Systems (SIA - Outpatient Information System):

- `"SIA-AB"`: Primary Care (Atencao Basica)
- `"SIA-ABO"`: Dental Procedures (Procedimentos Odontologicos)
- `"SIA-ACF"`: Pharmaceutical Assistance (Assistencia Farmaceutica)
- `"SIA-AD"`: High Complexity (Alta Complexidade/Diferenciada)
- `"SIA-AN"`: Home Care (Atencao Domiciliar)
- `"SIA-AM"`: Medical Specialties (Ambulatorio de Especialidades)
- `"SIA-AQ"`: Strategic Actions (Acoes Estrategicas)
- `"SIA-AR"`: Regulation (Regulacao)
- `"SIA-ATD"`: Urgency/Emergency (Urgencia/Emergencia)
- `"SIA-PA"`: Hospital Outpatient (Procedimentos Ambulatoriais em Hospital)
- `"SIA-PS"`: Psychosocial Care (Atencao Psicossocial)
- `"SIA-SAD"`: Specialized Care (Atencao Especializada)

Health Establishments (CNES - National Health Establishment Registry):

- `"CNES-LT"`: Beds (Leitos)
- `"CNES-ST"`: Health Professionals (Profissionais de Saude)
- `"CNES-DC"`: Equipment (Equipamentos) - Detailed
- `"CNES-EQ"`: Equipment (Equipamentos) - Summary
- `"CNES-SR"`: Specialized Services (Servicos Especializados)
- `"CNES-HB"`: Hospital Beds (Leitos Hospitalares)
- `"CNES-PF"`: Health Professionals Detailed (Pessoal Fisico)
- `"CNES-EP"`: Teaching Participants (Participantes do Ensino)
- `"CNES-RC"`: Hospital Class (Classificacao Hospitalar)
- `"CNES-IN"`: Hospital Indicators (Indicadores Hospitalares)
- `"CNES-EE"`: Educational Entities (Entidades de Ensino)
- `"CNES-EF"`: Teaching Facilities (Instalacoes de Ensino)
- `"CNES-GM"`: Management/Support (Gestao e Apoio)

Live Births (SINASC - Live Birth Information System):

- `"SINASC"`: Live Birth Declarations (Declaracoes de Nascidos Vivos)

`backend` Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets).

- `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows.
- `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets.

If not specified, the function may automatically choose the backend based on the input data type.

`city` Character vector of municipality names (e.g., `"Porto Velho"`, `"São Paulo"`). Case-insensitive; accents are normalised. Typos trigger fuzzy suggestions. Union with `municipality_code`. Applied after download to the combined dataset.

`municipality_code` Character or numeric vector of 6 or 7-digit IBGE municipality codes. Applied independently from `city`; both 6-digit (DATASUS) and 7-digit (IBGE) forms are always included.

`use_cache` Logical. If TRUE (default), will use cached data to avoid re-downloads. Cache is based on UF, year, month, and system parameters.

`cache_dir` Character. Directory to store cached files. Default is "~/.climasus4r_cache/data".

`force_redownload` Logical. If TRUE, ignores cache and re-downloads everything. Useful when you suspect cached data is corrupted or outdated.

`parallel` Logical. If TRUE (default), will use parallel processing for multiple UF/year combinations. Significantly speeds up bulk downloads.

`workers` Integer. Number of parallel workers to use. Default is 4. Set to 1 to disable parallel processing.

`lang` Character string specifying the language for variable labels and messages. Options: `"en"` (English), `"pt"` (Portuguese, default), `"es"` (Spanish).

`verbose` Logical. If TRUE (default), prints detailed progress information including cache status, download progress, and time estimates.

Details

Data Sources:

All data is sourced from the Brazilian Ministry of Health's DATASUS portal (http://datasus.saude.gov.br).

Caching System:

The cache uses SHA-256 hashing of parameters to create unique cache keys. Cached files are stored as compressed RDS files and include metadata about the download date and parameter combination. Cache is automatically invalidated after 30 days for dynamic systems (CNES, SIA, SIH) and 365 days for static systems (SIM, SINAN, SINASC).

Parallel Processing:

When downloading data for multiple states or years, parallel processing can reduce download time by up to 70%. The function uses `future.apply` internally. For large downloads (>100 files), consider increasing `workers` up to 8 (if your system has sufficient cores and memory).

Value

A `tibble` (or `data.frame`) with the requested data, combining multiple UFs/years when requested. The output includes:

- All original variables from the DATASUS system
- Additional metadata columns: `source_system`, `download_timestamp`
- Standardized date formats (Date objects instead of strings)
- UTF-8 encoded character variables

Note: Large datasets (especially SIA and SIH) may require significant memory (1GB+ for national annual data).

References

Brazilian Ministry of Health. DATASUS. http://datasus.saude.gov.br

SALDANHA, Raphael de Freitas; BASTOS, Ronaldo Rocha; BARCELLOS, Christovam. Microdatasus: pacote para download e pre-processamento de microdados do Departamento de Informatica do SUS (DATASUS). Cad. Saude Publica, Rio de Janeiro , v. 35, n. 9, e00032419, 2019. Available from https://doi.org/10.1590/0102-311x00032419.

See Also

- Official DATASUS documentation: http://datasus.saude.gov.br
- Microdatasus package: https://github.com/rfsaldanha/microdatasus

Examples

```
## Not run:
# Basic example: Mortality data for Rio de Janeiro in 2022
df_sim <- sus_data_import(
  uf = "RJ",
  year = 2022,
  system = "SIM-DO",
  use_cache = TRUE
)

# Dengue cases for two states with parallel processing
df_dengue <- sus_data_import(
  uf = c("SP", "MG"),
  year = 2023,
  system = "SINAN-DENGUE",
  parallel = TRUE,
  workers = 3
)

# Hospitalizations with monthly specification
df_hospital <- sus_data_import(
  uf = "SP",
  year = 2024,
  month = 1:6,  # January to June
  system = "SIH-RD",
  verbose = TRUE
)

# Force re-download ignoring cache
```

```
df_births <- sus_data_import(
  uf = "BA",
  year = 2020:2022,
  system = "SINASC",
  use_cache = TRUE,
  force_redownload = TRUE  # Refresh cached data
)

## End(Not run)
```

`sus_data_plot_aggregate_map` *Plot Municipal Map of Aggregated Health Data*

Description

Renders a bubble or choropleth map showing the spatial distribution of health events (counts or incidence rates) at the Brazilian municipal level. Designed for data produced by `sus_data_aggregate()`.

Bubble map (`map_type = "bubble"`) places proportionally sized, colour-encoded circles at each municipality centroid. State boundaries are drawn as a background polygon layer.

Choropleth map (`map_type = "choropleth"`) fills municipality polygons downloaded via `geobr::read_municipality()`. Requires `geobr` and `sf`. If these packages are unavailable the function falls back to a bubble map with a warning.

Usage

```
sus_data_plot_aggregate_map(
  df,
  value_col = NULL,
  map_type = c("bubble", "choropleth", "quantile_choropleth"),
  rate_per_100k = FALSE,
  period = NULL,
  city = NULL,
  top_n = NULL,
  state_borders = TRUE,
  show_labels = TRUE,
  palette = "YlOrRd",
  log_scale = TRUE,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  theme_style = "publication",
  base_size = 11,
  interactive = FALSE,
  use_cache = TRUE,
```

```
  cache_dir = "~/.climasus4r_cache/spatial",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

df	A climasus_df at stage = "aggregate" or later (output of sus_data_aggregate()).
value_col	Character. Name of the health-count column to map. If NULL (default) the column is auto-detected using the same priority order as sus_climate_plot_aggregate(): n_obitos, n_internacoes, n_nascimentos, n_casos, n_procedimentos, n_estabelecimentos and their English / Spanish equivalents.
map_type	Character. "bubble" (default), "choropleth" or "quantile_choropleth"
rate_per_100k	Logical. If TRUE, divides total_cases by pop_25 (from municipio_meta) and multiplies by 1e5 to compute incidence per 100,000 inhabitants. Default FALSE.
period	Date scalar or c(start, end) vector. When supplied, rows in df whose date column falls outside the interval are removed before aggregation. NULL (default) uses all rows.
city	Character vector of municipality names or 6-/7-digit IBGE codes. When supplied, the map is restricted to those municipalities. Names are resolved through municipio_meta with accent tolerance.
top_n	Integer. If supplied, only the top N municipalities by total_cases are labelled on the map. NULL disables top-N labels.
state_borders	Logical. Overlay state boundary polygons (downloaded via geobr). Default TRUE.
show_labels	Logical. Annotate capital cities with text labels (bubble map only). Default TRUE.
palette	Character. Colour palette for the fill / bubble fill scale. Any RColorBrewer::brewer.pal() sequential/diverging palette name (e.g. "YlOrRd", "Blues", "RdYlBu") or one of the ggsci aliases "lancet" / "nejm". Default "YlOrRd".
log_scale	Logical. Apply log1p transformation to the colour scale. Default TRUE.
title	Character. Map title. NULL uses a built-in multilingual default.
subtitle	Character. Map subtitle. NULL auto-generates one from the number of municipalities, metric and period.
caption	Character. Figure caption. NULL uses the DATASUS source string for the selected language.
theme_style	Character. Reserved for future theme variants. Currently only "publication" (default) is implemented.
base_size	Numeric. Base font size for theme_void(). Default 11.
interactive	Logical. If TRUE, wraps the ggplot2 object with plotly::ggplotly(). Requires plotly. Default FALSE.
use_cache	Logical. Cache municipality metadata to disk. Default TRUE.

`cache_dir`	Character. Directory for the disk cache. Default `"~/.climasus4r_cache/spatial"`.
`lang`	Character. Language for messages and axis labels: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `ggplot2` object (class `"gg"` / `"ggplot"`), or a `plotly` object when `interactive = TRUE`. The function does **not** modify `df` or advance the pipeline `stage`.

See Also

`sus_data_aggregate()`, `sus_climate_plot_aggregate()`

Examples

```
## Not run:
library(climasus4r)

# Monthly deaths by municipality
df_agg <- sus_data_import(uf = "RN", year = 2022, system = "SIM-DO") |>
  sus_data_standardize() |>
  sus_data_aggregate(time_unit = "month",
                     group_by  = "codigo_municipio_residencia")

# Default bubble map (Portuguese labels)
sus_data_plot_aggregate_map(df_agg, lang = "pt")

# Choropleth with incidence rate per 100k, English labels
sus_data_plot_aggregate_map(
  df_agg,
  map_type      = "choropleth",
  rate_per_100k = TRUE,
  palette       = "Blues",
  lang          = "en"
)

# Interactive plotly bubble map
sus_data_plot_aggregate_map(df_agg, interactive = TRUE, lang = "pt")

## End(Not run)
```

`sus_data_plot_aggregate_ts`

Time-Series Plot of Aggregated DATASUS Health Outcomes

Description

Produces publication-quality time-series visualisations (epidemic curve, seasonal boxplot, calendar heatmap, or annual trend) from a `climasus_df` dataset at stage `"aggregate"` or later. Colour palettes are powered by ggsci::ggsci (Lancet, NEJM, JCO, UChicago). An interactive plotly::plotly widget can be returned with `interactive = TRUE`.

Usage

```
sus_data_plot_aggregate_ts(
  df,
  value_col = NULL,
  group_col = NULL,
  facet_col = NULL,
  facet_ncol = 3L,
  plot_type = c("epidemic", "seasonal", "heatmap", "trend"),
  smooth_method = "loess",
  smooth_span = 0.25,
  log_transform = FALSE,
  free_scales = TRUE,
  palette = "lancet",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  theme_style = "publication",
  date_labels = "%b/%y",
  year_breaks = "3 months",
  base_size = 11,
  interactive = FALSE,
  city = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/spatial",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` at stage `"aggregate"` or later (output of `sus_data_aggregate()`).
`value_col`	Character. Name of the outcome column to plot. If `NULL` (default), the function auto-detects the first matching column from: `n_obitos`, `n_internacoes`, `n_nascimentos`, `n_casos`, `n_procedimentos`, `n_estabelecimentos`, and their English / Spanish equivalents.
`group_col`	Character. Optional column used for colour-grouping in `"epidemic"` and `"seasonal"` plots.
`facet_col`	Character. Optional column for `facet_wrap()` panels.
`facet_ncol`	Integer. Number of facet columns. Default `3L`.

`plot_type`	Character. One or more of `"epidemic"`, `"seasonal"`, `"heatmap"`, `"trend"`. When multiple types are supplied, plots are assembled with patchwork::patchwork (one per row). Default: `"epidemic"`.
`smooth_method`	Character. Smoothing method for the `"epidemic"` plot: `"loess"` (default), `"gam"`, or `"lm"`.
`smooth_span`	Numeric. Span for LOESS smoothing. Default `0.25`.
`log_transform`	Logical. Apply `log1p` transformation on the y-axis for `"epidemic"` and `"seasonal"` plots, and to the fill scale in `"heatmap"`. Default `FALSE`.
`free_scales`	Logical. Allow free y-axis scales in faceted plots. Default `TRUE`.
`palette`	Character. Colour palette. One of `"lancet"` (default), `"nejm"`, `"jco"`, `"uchicago"`, or `"viridis"`.
`title`	Character. Plot title. Auto-generated if `NULL`.
`subtitle`	Character. Plot subtitle. `NULL` auto-generates one from row count, outcome column name and date range.
`caption`	Character. Figure caption. `NULL` uses the DATASUS source string for the selected language.
`theme_style`	Character. Reserved for future theme variants. Currently only `"publication"` (default) is implemented.
`date_labels`	Character. `strftime` format string for x-axis date labels. Default `"%b/%y"`.
`year_breaks`	Character. `date_breaks` string for the x-axis. Default `"3 months"`.
`base_size`	Numeric. Base font size for the ggplot2 theme. Default `11`.
`interactive`	Logical. Return a `plotly::ggplotly()` widget instead of a static ggplot2 object. Default `FALSE`.
`city`	Character vector of municipality names or 6-/7-digit IBGE codes. When supplied, the plot is filtered to those municipalities before rendering. Names are resolved through `municipio_meta`, with accent tolerance, using the same cache as `sus_data_filter_demographics()`.
`use_cache`	Logical. Cache municipality metadata to disk when city filtering or municipality labels are needed. Default `TRUE`.
`cache_dir`	Character. Directory for municipality metadata cache. Default `"~/.climasus4r_cache/spatial`
`lang`	Character. Language for labels and messages: `"pt"` (default), `"en"`, or `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

Invisibly returns the ggplot2 object (or plotly widget when `interactive = TRUE`). Side effect: prints the plot in the active graphics device.

Supported systems

Automatically adapts y-axis labels and titles to SINAN, SIM, SIH, SINASC, SIA, and CNES via outcome column auto-detection.

Examples

```
## Not run:
# Epidemic curve, Portuguese labels
sus_data_plot_aggregate_ts(df_agg, lang = "pt")

# Seasonal boxplot, log-transformed
sus_data_plot_aggregate_ts(
  df_agg,
  plot_type     = "seasonal",
  log_transform = TRUE,
  lang          = "en"
)

# Calendar heatmap by group
sus_data_plot_aggregate_ts(
  df_agg,
  plot_type  = "heatmap",
  facet_col  = "municipio",
  facet_ncol = 2L,
  lang       = "pt"
)

# Multi-panel: epidemic + trend
sus_data_plot_aggregate_ts(
  df_agg,
  plot_type = c("epidemic", "trend"),
  lang      = "en"
)

# Interactive plotly widget
sus_data_plot_aggregate_ts(df_agg, interactive = TRUE)

## End(Not run)
```

`sus_data_plot_demographics`

Visualise Demographic Profiles from DATASUS Systems

Description

Produces publication-quality tables, charts, and composite dashboards summarising the demographic and climate-risk composition of any standardised `climasus_df` dataset (SIM, SIH, SINAN, CNES, SIA, SINASC). Visual style follows The Lancet / Nature Medicine guidelines. Colour palettes are powered by ggsci::ggsci.

Usage

```
sus_data_plot_demographics(
```

```
  df,
  type = "table",
  var = NULL,
  time_unit = "month",
  fill_var = NULL,
  palette = "lancet",
  heatmap_row = NULL,
  heatmap_col = NULL,
  fill_metric = "pct_row",
  show_ci = FALSE,
  benchmark = NULL,
  interactive = FALSE,
  base_size = 11,
  lang = "pt",
  subtitle = NULL,
  caption = NULL,
  theme_style = "publication",
  caption_suffix = NULL,
  save_path = NULL,
  width = NULL,
  height = NULL,
  dpi = 300,
  verbose = TRUE,
  ...
)
```

Arguments

`df`	A `climasus_df` object at stage `"filter_demo"` or later (output of `sus_data_filter_demographi`
`type`	Character. Visualisation type. One of: • `"table"` – Frequency table (gt / DT). • `"bar"` – Horizontal bar chart for one variable. • `"pyramid"` – Age-sex population pyramid. • `"heatmap"` – Cross-demographic tile matrix (two variables). • `"temporal"` – Epidemic curve / time-series. • `"climate"` – Climate-risk group distribution (bar + season heatmap). • `"race_equity"`– Race/colour equity diverging plot vs. IBGE 2022 Census. • `"dashboard"` – Composite Lancet-layout panel.
`var`	Character. Demographic variable (required for `"bar"` and single-variable `"table"`). One of `"sex"`, `"race"`, `"age_group"`, `"education"`, `"climate_risk"`, `"region"`.
`time_unit`	Character. Temporal resolution for `type = "temporal"`. One of `"month"` (default), `"epi_week"`, `"year"`, `"quarter"`, `"semester"`.
`fill_var`	Character. Optional stratification variable for temporal plots (e.g., `"sex"`, `"age_group"`, `"climate_risk_group"`).

palette	Character. Colour palette powered by ggsci::ggsci. One of "lancet" (default), "nature", "nejm", "jco", "aaas", "sus", "viridis", "colorblind".
heatmap_row	Character. Row variable for type = "heatmap". One of "age_group", "sex", "race", "education", "climate_risk", "region". Defaults to "age_group" (auto-detected).
heatmap_col	Character. Column variable for type = "heatmap". Defaults to "race" (auto-detected). Must differ from heatmap_row.
fill_metric	Character. What to show in each tile for type = "heatmap". One of "pct_row" (default – % within each row, e.g. % of each age group that is Parda), "pct_col" (% within each column), "pct_total" (% of all records), "count" (raw count).
show_ci	Logical. Add 95% Poisson confidence intervals in temporal plots. Default FALSE.
benchmark	Numeric named vector. Reference proportions for the race-equity plot. If NULL (default), IBGE 2022 Census proportions are used where available.
interactive	Logical. Return interactive plotly::ggplotly() widget instead of static ggplot2/gt objects. Default FALSE.
base_size	Numeric. Base font size for ggplot2 theme. Default 11.
lang	Character. Language for labels and messages. One of "pt" (default), "en", "es".
subtitle	Character. Figure subtitle. NULL uses no subtitle.
caption	Character. Figure caption. NULL auto-generates a DATASUS source string for the selected language.
theme_style	Character. Reserved for future theme variants. Currently only "publication" (default) is implemented.
caption_suffix	Character. Additional text appended to the figure caption (e.g., study period, DOI). Default NULL.
save_path	Character. File path to save output (PNG, PDF, SVG, or HTML). Default NULL (no file saved).
width, height	Numeric. Output dimensions in inches. Defaults: 7 x 5 for single plots; 12 x 9 for dashboards.
dpi	Numeric. Resolution for raster output. Default 300.
verbose	Logical. Print progress messages. Default TRUE.
...	Additional arguments passed to underlying plot helpers.

Value

Invisibly returns the primary plot or list of plots. Side effect: renders in the active graphics device (or interactive widget).

Systems supported

Automatically adapts column detection to SINAN, SIM, SIH, CNES, SIA, and SINASC. Run sus_data_create_variables() before this function to generate age_group, year, month, epidemiological_week, and climate_risk_group.

Climate integration

When `climate_risk_group` is present (created by `sus_data_create_variables()`), `type = "climate"` produces a combined bar + seasonal heatmap designed for climate-health submissions to *The Lancet Planetary Health*.

Examples

```
## Not run:
# Age-sex pyramid (Lancet palette, Portuguese)
sus_data_plot_demographics(df, type = "pyramid", lang = "pt")

# Composite dashboard saved as PDF
sus_data_plot_demographics(
  df,
  type      = "dashboard",
  palette   = "lancet",
  lang      = "en",
  save_path = "figures/fig1_demographics.pdf",
  width = 12, height = 9
)

# Epidemic curve stratified by climate risk group (with CI)
sus_data_plot_demographics(
  df,
  type      = "temporal",
  time_unit = "epi_week",
  fill_var  = "climate_risk_group",
  show_ci   = TRUE,
  lang      = "pt"
)

# Race equity diverging plot
sus_data_plot_demographics(df, type = "race_equity", lang = "pt")

# Cross-demographic heatmap: age group x race, % within each age group
sus_data_plot_demographics(
  df,
  type        = "heatmap",
  heatmap_row = "age_group",
  heatmap_col = "race",
  fill_metric = "pct_row",
  lang        = "pt"
)

# Cross-demographic heatmap: education x climate risk (absolute counts)
sus_data_plot_demographics(
  df,
  type        = "heatmap",
  heatmap_row = "education",
  heatmap_col = "climate_risk",
  fill_metric = "count",
  palette     = "nejm"
```

```
)

## End(Not run)
```

`sus_data_quality_report` *Pipeline-aware data quality report for health data*

Description

Reads `sus_meta` processing history to identify which climasus4r pipeline functions were applied, then produces a section-by-section quality assessment covering completeness, demographic distributions, date validity, ICD-10 codes, geographic coverage, and derived variables. Returns an overall quality score (0-100) and supports four output formats.

Usage

```
sus_data_quality_report(
  df,
  output_format = "console",
  output_file = NULL,
  check_icd = TRUE,
  check_dates = TRUE,
  top_n = 10,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` or plain `data.frame`. Arrow objects are materialised automatically.
`output_format`	Character. One of `"console"` (default), `"gt"`, `"markdown"`, or `"html"`. The `"gt"` format returns a `gt_tbl` object suitable for Quarto / R Markdown documents.
`output_file`	Character. File path for `"markdown"` and `"html"` formats. Defaults to a timestamped filename in the working directory.
`check_icd`	Logical. Include ICD-10 quality section. Default `TRUE`.
`check_dates`	Logical. Include date validation section. Default `TRUE`.
`top_n`	Integer. Rows shown in frequency tables. Default `10`.
`lang`	Character. Output language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

Invisibly returns a named list of quality metrics (always available regardless of `output_format`):

`$meta` Pipeline metadata from `sus_meta`

`$pipeline` Functions detected in processing history

`$overview` Row/column counts, duplicates, column types

`$missing` Per-column missing values with quality flags

`$demographics` Frequency tables for sex, race, age, education

`$dates` Date range and validity per date column

`$icd` ICD-10 frequency and validity

`$geographic` Municipality and state coverage

`$derived` Presence of expected derived variables

`$score` Overall quality score 0-100

Examples

```
## Not run:
# Console report (default)
sus_data_quality_report(df, lang = "pt")

# gt table (for Quarto / R Markdown)
sus_data_quality_report(df, output_format = "gt", lang = "en")

# Save complete Markdown report
sus_data_quality_report(df, output_format = "markdown",
                        output_file = "reports/quality.md", lang = "pt")

# Save HTML report
sus_data_quality_report(df, output_format = "html",
                        output_file = "reports/quality.html")

## End(Not run)
```

`sus_data_read` *Read Processed Health Data with Batch and Parallel Support*

Description

Smartly reads one or multiple health data files exported by `sus_data_export()`. Supports automatic format detection, batch processing, parallel execution, spatial data, metadata loading, and data validation.

Usage

```
sus_data_read(
  path,
  format = NULL,
  parallel = FALSE,
  workers = 4,
  read_metadata = FALSE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`path`	Character vector of file paths, or a single directory path. If a directory is provided, all matching files will be read.
`format`	Character string specifying the input format. Options: `"dbf"`, `"dbc"`, `"rds"`, `"parquet"`, `"geoparquet"`, `"shapefile"`, `"gpkg"`, `"geojson"`, `"csv"`. If `NULL` (default), automatically detects format from file extension.
`parallel`	Logical. If `TRUE`, uses parallel processing for multiple files. Requires `future` and `future.apply` packages. Default: `FALSE`.
`workers`	Integer. Number of parallel workers when `parallel = TRUE`. Default: 4.
`read_metadata`	Logical. If `TRUE` (default), loads companion metadata files and attaches them as attributes.
`lang`	Character string specifying the language for messages. Options: `"en"` (English), `"pt"` (Portuguese, default), `"es"` (Spanish).
`verbose`	Logical. If `TRUE` (default), prints progress and summary.

Details

Batch Processing: Pass a vector of file paths or a directory path to read multiple files at once. All files are automatically combined into a single object.

Parallel Processing: When `parallel = TRUE`, files are read simultaneously using `future.apply`. This significantly speeds up batch reads of large files.

Format Detection: Automatically detects format from file extension. For `.parquet` files, automatically determines if it's GeoParquet (spatial) or regular Parquet.

Memory Efficiency: For very large datasets (>50 GB), consider using chunked processing or reading files individually instead of batch mode.

Value

A data frame or sf object (for spatial data) containing the loaded data. For batch reads, all files are combined with `dplyr::bind_rows()`. Metadata is attached as attributes:

- Single file: `attr(df, "metadata")`
- Batch: `attr(df, "batch_metadata")` (list of metadata from each file)
- Batch: `attr(df, "n_files_combined")` (number of files)

Examples

```
## Not run:
library(climasus4r)

# Single file
df <- sus_data_read("output/data.parquet")

# Multiple files (vector)
df <- sus_data_read(c("output/2020.parquet", "output/2021.parquet"))

# Directory (all Parquet files)
df <- sus_data_read("output/", format = "parquet")

# Parallel batch read
df <- sus_data_read("output/", format = "dbf",
                    parallel = TRUE, workers = 6)

# Access batch metadata
batch_meta <- attr(df, "batch_metadata")
n_files <- attr(df, "n_files_combined")

## End(Not run)
```

`sus_data_standardize` *Standardize SUS data column names and values*

Description

This function standardizes column names and categorical values in SUS datasets, ensuring consistency across different years and versions. It supports three languages: English (en), Portuguese (pt), and Spanish (es).

Usage

```
sus_data_standardize(
  df,
  lang = "pt",
  translate_columns = TRUE,
  standardize_values = TRUE,
  keep_original = FALSE,
  backend = "arrow",
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` or `tibble` to be standardized (typically output from `sus_data_import()`).

`lang` Character. Output language for column names and values. Options: "en" (English), "pt" (Portuguese, Default), "es" (Spanish).

`translate_columns`
Logical. If TRUE, translates column names. Default is TRUE.

`standardize_values`
Logical. If TRUE, standardizes categorical values. Default is TRUE.

`keep_original` Logical. If TRUE, keeps original columns alongside standardized ones. Default is FALSE.

`backend` Character string specifying the data processing backend. Use `"arrow"` for out-of-memory, lazy processing (recommended for large datasets), or `"tibble"` for in-memory processing (recommended for small to medium datasets).

- `"arrow"`: operations are performed lazily using the Apache Arrow engine, avoiding loading the full dataset into memory. Ideal for large files (e.g., Parquet, Feather) and high-performance workflows.
- `"tibble"`: data is fully loaded into memory as a tibble and processed eagerly using dplyr. Simpler and more predictable, but may be slow or fail for large datasets.

If not specified, the function may automatically choose the backend based on the input data type.

`verbose` Logical. If TRUE, prints a report of standardization actions. Default is TRUE.

Details

The function builds upon the preprocessing done by `microdatasus`, adding an additional layer of standardization specifically designed for climate-health research workflows.

Value

A `data.frame` with standardized column names and values in the specified language.

References

Brazilian Ministry of Health. DATASUS. http://datasus.saude.gov.br

SALDANHA, Raphael de Freitas; BASTOS, Ronaldo Rocha; BARCELLOS, Christovam. Microdatasus: pacote para download e pre-processamento de microdados do Departamento de Informatica do SUS (DATASUS). Cad. Saude Publica, Rio de Janeiro , v. 35, n. 9, e00032419, 2019. Available from https://doi.org/10.1590/0102-311x00032419.

Examples

```
## Not run:
# Standardize to English (default)
df_en <- sus_data_standardize(df_raw, lang = "en")

# Standardize to Portuguese
df_pt <- sus_data_standardize(df_raw, lang = "pt")
```

```
# Standardize to Spanish
df_es <- sus_data_standardize(df_raw, lang = "es")

# Keep original columns for comparison
df_both <- sus_data_standardize(
  df_raw,
  lang = "pt",
  keep_original = TRUE
)

# Only translate column names (not values)
df_cols_only <- sus_data_standardize(
  df_raw,
  lang = "en",
  translate_columns = TRUE,
  standardize_values = FALSE
)

# Complete pipeline
df_analysis_ready <- sus_data_import(uf = "SP", year = 2023, system = "SIM-DO") |>
  sus_data_clean_encoding() |>
  sus_data_standardize(lang = "pt")

## End(Not run)
```

`sus_data_ts_quality` *Time-Series Quality Control for Daily Municipal Health Counts*

Description

Evaluates the quality of daily health event time series at the municipal level, producing per-municipality flags for:

Usage

```
sus_data_ts_quality(
  data,
  outcome_col = "n_obitos",
  muni_col = "code_muni",
  date_col = "date",
  min_completeness = 0.9,
  max_gap = 7L,
  break_alpha = 0.05,
  max_outlier_months = 3L,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`data`	A `climasus_df` at stage `"aggregate"`, or any `data.frame` with a municipality identifier column, a date column, and a daily count column.
`outcome_col`	Character. Name of the daily count column. Default: `"n_obitos"`.
`muni_col`	Character. Name of the municipality identifier column. Default: `"code_muni"`.
`date_col`	Character. Name of the date column. Default: `"date"`.
`min_completeness`	Numeric (0-1). Minimum completeness to recommend inclusion. Default: `0.90` (90% of expected days present).
`max_gap`	Integer. Maximum tolerated consecutive zero/missing days before flagging a temporal gap. Default: `7L`.
`break_alpha`	Numeric. Significance level for structural break test. Default: `0.05`. Ignored if `strucchange` is not installed.
`max_outlier_months`	Integer. Maximum number of outlier months allowed before exclusion. Default: `3L`.
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Details

- **Completeness**: fraction of expected days with non-missing records.
- **Structural breaks**: abrupt level shifts detectable by the CUSUM-based Zeileis et al. (2002) test (requires the `strucchange` package).
- **Monthly outliers**: months with counts $>$ Q3 + 1.5 x IQR (Tukey fence).
- **Temporal gaps**: runs of consecutive days with zero *or* missing counts longer than `max_gap`.

A composite inclusion score (0-100) and a binary `include` recommendation are returned, matching the exclusion criteria used in large Brazilian heat-wave studies (Lowe et al. 2021, Anjos & Targino 2023).

Value

A `climasus_ts_quality` list with:

`$flags` Tibble (one row per municipality) with columns: `muni_col`, `n_obs`, `n_expected`, `completeness`, `has_break`, `break_pval`, `n_outlier_months`, `n_gaps_gt{max_gap}d`, `score`, `include`.

`$recommend_include` Character vector of municipality codes recommended for inclusion.

`$recommend_exclude` Character vector of municipality codes recommended for exclusion, with reason.

`$params` List of QC parameters used.

References

Zeileis, A., Leisch, F., Hornik, K., & Kleiber, C. (2002). strucchange: An R Package for Testing for Structural Change in Linear Regression Models. *Journal of Statistical Software*, 7(2). doi:10.18637/jss.v007.i02

See Also

`sus_data_aggregate()`, `sus_mod_casecrossover()`, `sus_mod_dlnm()`

Examples

```
## Not run:
qc <- sus_data_ts_quality(
  df_agg,
  outcome_col       = "n_obitos",
  min_completeness  = 0.90,
  max_gap           = 7L
)
print(qc)

# Filter series to include only QC-passing municipalities
df_clean <- df_agg[df_agg$code_muni %in% qc$recommend_include, ]

## End(Not run)
```

---

sus_grid_chirps *Import CHIRPS Rainfall Data for Brazilian Municipalities*

Description

`sus_grid_chirps()` downloads CHIRPS v2.0 precipitation data, crops it to Brazil, spatially aggregates to municipalities, and returns a `climasus_df` compatible with `sus_mod_dlnm()` and `sus_climate_anomaly()`.

CHIRPS (Climate Hazards Group InfraRed Precipitation with Station data) is a quasi-global, 0.05° (~5 km) daily rainfall dataset from UCSB CHC covering 1981 to present. It combines satellite imagery with station data and is considered the best freely available high-resolution rainfall product for Brazil — particularly for leptospirosis, diarrheal disease, and dengue analyses where precipitation is the key exposure.

No authentication is required.

Usage

```
sus_grid_chirps(
  resolution = "monthly",
  years = NULL,
  months = 1:12,
  municipalities = NULL,
```

```
  agg_fun = "mean",
  crop_brazil = TRUE,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/chirps",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`resolution`	Character. Temporal resolution of source files: • `"monthly"` (default) — one file per month (~5–20 MB each); returns mm/month. • `"daily"` — one file per day (~3 MB each); returns mm/day. Recommended only for short periods; a full year = 365 downloads. • `"annual"` — one file per year; returns mm/year.
`years`	Integer vector. Years to download. Coverage: 1981 to present for daily/monthly; 1981–2024 for annual. `NULL` defaults to the last two complete years.
`months`	Integer vector (1–12). Months to include for daily and monthly resolutions. Ignored for annual. Default `1:12`.
`municipalities`	An `sf` POLYGON object (e.g., from `geobr::read_municipality()`). When provided, rasters are aggregated and a `climasus_df` is returned. If `NULL`, a named character vector of cached file paths is returned instead.
`agg_fun`	Character. Spatial aggregation function for `exactextractr::exact_extract()`. Default `"mean"` (area-weighted mean). For rainfall totals over a municipality use `"sum"`.
`crop_brazil`	Logical. Crop global rasters to Brazil's bounding box before aggregation. Reduces memory usage significantly. Default `TRUE`.
`use_cache`	Logical. Reuse previously downloaded raster files and aggregated Parquet caches. Default `TRUE`.
`cache_dir`	Character. Root cache directory. Default `"~/.climasus4r_cache/chirps"`.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

- If `municipalities` is provided: a `climasus_df` with columns `code_muni` (character), `date` (Date), and `rainfall_chirps_mm` (numeric). Metadata: `stage = "climate"`, `type = "chirps"`.
- If `municipalities = NULL`: a named character vector of paths to the cached GeoTIFF/gz files.

Units

- Daily: mm/day
- Monthly: mm/month (cumulative)
- Annual: mm/year (cumulative)

CHIRPS missing/no-data value (-9999) is automatically converted to `NA`.

Data source

Funk, C. et al. (2015). The climate hazards infrared precipitation with stations — a new environmental record for monitoring extremes. *Scientific Data*, 2, 150066. doi:10.1038/sdata.2015.66
Data: https://data.chc.ucsb.edu/products/CHIRPS-2.0/

Packages required

Spatial aggregation requires `terra` and `exactextractr` (both in `Suggests`). Install with `install.packages(c("terra", "exactextractr"))`.

See Also

`sus_grid_era5()`, `sus_climate_anomaly()`, `sus_mod_dlnm()`

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Monthly rainfall for Mato Grosso, 2020
chirps <- sus_grid_chirps(
  resolution    = "monthly",
  years         = 2020,
  municipalities = mt_mun,
  lang          = "pt"
)
sus_meta(chirps, "stage")  # "climate"
sus_meta(chirps, "type")   # "chirps"

# Daily rainfall, fire season
chirps_daily <- sus_grid_chirps(
  resolution    = "daily",
  years         = 2020,
  months        = 7:9,
  municipalities = mt_mun,
  agg_fun       = "mean",
  lang          = "en"
)

# Annual totals for all Brazil (no spatial aggregation)
paths <- sus_grid_chirps(
```

```
  resolution = "annual",
  years      = 2018:2022
)

## End(Not run)
```

`sus_grid_era5` *Import ERA5-Land Daily Climate Data for Brazilian Municipalities*

Description

`sus_grid_era5()` downloads pre-processed ERA5-Land daily climate aggregates for Latin America from Zenodo and, optionally, spatially aggregates them to Brazilian municipalities. No API key is required.

The data source is the **ERA5-Land Daily Aggregates for Latin America** project (Saldanha, rfsaldanha.github.io), which provides NetCDF files (one per variable × month × year) at ~10 km spatial resolution for 1950–2025. Files are hosted on Zenodo under CC-BY 4.0.

When `municipalities` is provided, the function returns a `climasus_df` at `stage = "climate"`, `type = "era5_land"` — directly compatible with `sus_climate_anomaly()`, `sus_climate_aggregate()`, and `sus_mod_dlnm()`.

Usage

```
sus_grid_era5(
  years,
  months = 1:12,
  vars = c("t2m", "tp"),
  municipalities = NULL,
  agg_fun = "mean",
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/era5",
  parallel = FALSE,
  workers = 2,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Numeric vector of years to import. Must be between 1950 and 2025. Example: `2010:2023`, `c(2019, 2021)`.
`months`	Integer vector of months to import (1–12). Default: `1:12` (all months).
`vars`	Character vector of variable aliases to download. Allowed values: • `"t2m"` — 2 m temperature mean (→ `tair_dry_bulb_c`, °C)

- `"t2m_max"` — 2 m temperature maximum (→ `tair_max_c`, °C)
- `"t2m_min"` — 2 m temperature minimum (→ `tair_min_c`, °C)
- `"td2m"` — 2 m dewpoint temperature mean (→ `dew_point_c`, °C)
- `"u10"` — 10 m u-component of wind mean (→ `ws_10m_u_m_s`, m/s)
- `"v10"` — 10 m v-component of wind mean (→ `ws_10m_v_m_s`, m/s)
- `"sp"` — surface pressure mean (→ `patm_hpa`, hPa)
- `"tp"` — total precipitation daily sum (→ `rainfall_mm`, mm)
- `"all"` — all variables above

Default: `c("t2m", "tp")`.

`municipalities`
: An `sf` object with polygon geometries representing the areas to which raster data will be aggregated (typically Brazilian municipalities from `geobr::read_municipality()`). Must contain a column identifying each polygon — the function auto-detects `code_muni`, `CD_MUN`, or `CD_GEOCMU`. If `NULL`, returns a named list of downloaded NetCDF file paths instead of a `climasus_df`.

`agg_fun`
: Character. Spatial aggregation function applied over raster pixels within each polygon. Any function supported by `exactextractr::exact_extract()`: `"mean"` (default, area-weighted), `"sum"`, `"median"`, `"min"`, `"max"`. For precipitation, `"mean"` gives the representative areal value.

`use_cache`
: Logical. If `TRUE` (default), downloaded NetCDF files are stored in `cache_dir` and reused on subsequent calls.

`cache_dir`
: Character. Directory for cached NetCDF files. Default: `"~/.climasus4r_cache/era5"`. Created automatically if needed.

`parallel`
: Logical. If `TRUE`, downloads run in parallel using the active `future::plan()`. Do not set a plan inside this function; configure it beforehand. Default: `FALSE`.

`workers`
: Integer. Number of parallel workers when `parallel = TRUE`. Default: `2`.

`lang`
: Character. Language for messages: `"pt"` (default), `"en"`, or `"es"`.

`verbose`
: Logical. If `TRUE` (default), prints progress messages.

Value

- If `municipalities` is provided: a `climasus_df` tibble with columns `code_muni` (character), `date` (Date), and one column per requested variable (e.g., `tair_dry_bulb_c`, `rainfall_mm`). Metadata: `stage = "climate"`, `type = "era5_land"`.
- If `municipalities = NULL`: a named character vector of paths to the downloaded NetCDF files (named by `"{year}_{month}_{var}"`).

Data Source

Saldanha, R. ERA5-Land Daily Aggregates for Latin America (1950–2025). Zenodo. https://zenodo.org/doi/10.5281/zenodo.10013254. CC-BY 4.0.

Unit Conversions Applied

- Temperature (K → °C): subtract 273.15
- Precipitation (m → mm): multiply by 1000
- Pressure (Pa → hPa): divide by 100
- Wind components (m/s): no conversion needed

See Also

sus_climate_anomaly(), sus_climate_aggregate(), sus_mod_dlnm()

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Temperature and precipitation for Mato Grosso, Q1 2020
era5 <- sus_grid_era5(
  years          = 2020,
  months         = 1:3,
  vars           = c("t2m", "tp"),
  municipalities = mt_mun,
  lang           = "pt"
)

sus_meta(era5, "stage")  # "climate"
sus_meta(era5, "type")   # "era5_land"

# All variables, no spatial aggregation (returns file paths)
paths <- sus_grid_era5(years = 2020, months = 1, vars = "all")

## End(Not run)
```

---

sus_grid_fires *Import Fire Hotspot Data for Brazilian Municipalities*

Description

`sus_grid_fires()` downloads fire hotspot (active fire) data from INPE Queimadas or NASA FIRMS, optionally aggregates counts and Fire Radiative Power (FRP) to Brazilian municipalities, and returns a `climasus_df` compatible with sus_mod_dlnm() and sus_climate_anomaly().

Fire data is critical for respiratory health analyses in Brazil: smoke from agricultural burning (cerrado, cana-de-acucar) and deforestation fires (Amazonia) are associated with acute respiratory outcomes in SIH/SINAN data.

Sources available:

- `"inpe"` (default) — INPE Queimadas portal. No authentication. Historical data from 1998 to present. Brazil only. DOI: doi:10.2312/inpe.2022.009
- `"firms_modis"` — NASA FIRMS MODIS Collection 6.1. Requires a free MAP KEY from `https://firms.modaps.eosdis.nasa.gov/api/map_key/`. Global, 1 km resolution, 2000 to present.
- `"firms_viirs"` — NASA FIRMS VIIRS SNPP. Same key as FIRMS MODIS. 375 m resolution, 2012 to present.

Usage

```
sus_grid_fires(
  years,
  months = 1:12,
  uf = NULL,
  bbox = NULL,
  source = c("inpe", "firms_modis", "firms_viirs"),
  municipalities = NULL,
  agg_fun = "count",
  biome = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/fires",
  firms_key = NULL,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Numeric vector of years. INPE: 1998–present. FIRMS MODIS: 2000–present. FIRMS VIIRS: 2012–present.
`months`	Integer vector (1–12). Default `1:12`.
`uf`	Character vector of state codes (e.g., `c("MT", "PA")`). Case-insensitive. `NULL` = all Brazil. Used to filter INPE queries and as a shortcut to define `bbox` for FIRMS.
`bbox`	Numeric vector `c(xmin, ymin, xmax, ymax)` in WGS84 longitude/ latitude. Used by FIRMS for spatial subsetting. If `NULL` and `uf` is provided, the bbox is computed from Brazilian state centroids. If both are `NULL`, Brazil's bounding box is used.
`source`	Character. Data source: `"inpe"` (default), `"firms_modis"`, or `"firms_viirs"`.
`municipalities`	An `sf` POLYGON object (e.g., from `geobr::read_municipality()`). When provided, fire points are spatially joined to polygons and aggregated to daily municipality-level counts. If `NULL`, the raw fire point data are returned as a `climasus_df`.
`agg_fun`	Character. Aggregation strategy when `municipalities` is provided. Currently `"count"` is supported (count of fire hotspots per municipality per day). Future versions may support `"frp_sum"`.

biome	Character vector. INPE biome filter. Allowed values: "Amazonia", "Cerrado", "Mata Atlantica", "Caatinga", "Pampa", "Pantanal". NULL = no filter.
use_cache	Logical. Reuse previously downloaded CSV/JSON files. Default TRUE.
cache_dir	Character. Root cache directory. Default "~/.climasus4r_cache/fires".
firms_key	Character. NASA FIRMS MAP KEY. Defaults to Sys.getenv("FIRMS_MAP_KEY"). Required when source %in% c("firms_modis", "firms_viirs").
lang	Character. Message language: "pt" (default), "en", "es".
verbose	Logical. Print progress. Default TRUE.

Value

- If municipalities is provided: a climasus_df with columns date (Date), code_muni (character), n_fires (integer count of hotspots), and frp_mean (mean Fire Radiative Power in MW, NA when not available). Metadata: stage = "climate", type = "fires".
- If municipalities = NULL: a climasus_df with raw fire point columns: date, lat, lon, frp, biome, estado, source.

INPE Queimadas API

Queries https://queimadas.dgi.inpe.br/api/focos/ monthly. Each monthly request is cached as a JSON file. Large states in fire-season months may return truncated results due to the API's record limit; use uf to narrow the query when needed.

NASA FIRMS API

Queries https://firms.modaps.eosdis.nasa.gov/api/area/csv/{key}/{product}/{bbox}/10/{date} in 10-day chunks (maximum allowed per request). A free MAP KEY is required and must be activated at https://firms.modaps.eosdis.nasa.gov/api/map_key/.

See Also

sus_grid_pollution_cams(), sus_climate_anomaly(), sus_mod_dlnm()

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# INPE fire count for Mato Grosso municipalities, fire season 2020
fires <- sus_grid_fires(
  years          = 2020,
  months         = 7:10,
  uf             = "MT",
  municipalities = mt_mun,
  lang           = "pt"
)
```

```
sus_meta(fires, "stage")  # "climate"
sus_meta(fires, "type")   # "fires"

# Raw fire points for Amazonia, no aggregation
fires_pts <- sus_grid_fires(
  years  = 2022,
  months = 8,
  biome  = "Amazonia",
  lang   = "en"
)

# FIRMS MODIS with API key
Sys.setenv(FIRMS_MAP_KEY = "my_key")
fires_firms <- sus_grid_fires(
  years          = 2023,
  months         = 9,
  uf             = "MT",
  source         = "firms_modis",
  municipalities = mt_mun
)

## End(Not run)
```

`sus_grid_join` *Join Gridded Environmental Data to Health Data*

Description

`sus_grid_join()` merges the municipality × date output of any `sus_grid_*()` function (ERA5, CHIRPS, GHAP, CAMS, fires, PRODES, …) with a health `climasus_df` produced by `sus_spatial_join()` or `sus_data_aggregate()`. The result is a single `climasus_df` at `stage = "climate"` ready for `sus_mod_dlnm()`, `sus_climate_anomaly()`, and `sus_mod_vulnerability_index()`.

This function is the bridge between the gridded environmental pipeline (`sus_grid_*`) and the health pipeline (`sus_data_import` → … → `sus_spatial_join`), equivalent to the role `sus_climate_aggregate()` plays for station-based INMET data.

Usage

```
sus_grid_join(
  health_data,
  grid_data,
  by = c("code_muni", "date"),
  type_out = NULL,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

health_data	A climasus_df at stage "spatial", "aggregate", or "census". Must contain code_muni (character) and date (Date) columns.
grid_data	A climasus_df at stage "climate" produced by any sus_grid_*() function. Must contain code_muni and date columns and at least one environmental variable column.
by	Character vector. Join key columns. Default c("code_muni", "date"). Change to c("code_muni") if joining annual grid data (e.g., PRODES) to monthly health data — the join will broadcast the annual value to all months.
type_out	Character. Overrides sus_meta$type in the result. NULL (default) inherits the type from grid_data.
lang	Character. Message language: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default TRUE.

Value

A climasus_df with all columns from health_data plus the environmental variable columns from grid_data. Metadata: stage = "climate", type inherited from grid_data (or type_out). Rows from health_data with no matching grid row receive NA for grid columns.

Temporal granularity mismatch

If health_data has monthly dates (YYYY-MM-01) and grid_data has daily dates, the join will produce NAs for most rows. In that case, first aggregate grid_data to monthly resolution with sus_climate_aggregate() or by passing by = c("code_muni") (annual grid data only) or summarising manually before calling sus_grid_join().

See Also

sus_grid_era5(), sus_grid_chirps(), sus_grid_fires(), sus_grid_prodes(), sus_climate_aggregate(), sus_mod_dlnm()

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Health data (SIH respiratory hospitalisations)
sih_agg <- sus_data_import(system = "SIH", years = 2020, uf = "MT") |>
  sus_data_clean_encoding() |>
  sus_data_standardize() |>
  sus_data_filter_cid(cid_group = "respiratorio") |>
  sus_data_aggregate() |>
  sus_spatial_join(municipalities = mt_mun)

# Gridded climate data
era5 <- sus_grid_era5(
```

```
  years          = 2020,
  municipalities = mt_mun,
  vars           = c("t2m", "tp")
)

# Join: one step to connect health + climate
combined <- sus_grid_join(sih_agg, era5)
sus_meta(combined, "stage")  # "climate"
sus_meta(combined, "type")   # "era5_land"

# Annual deforestation joined to monthly health data (no date in by)
prodes <- sus_grid_prodes(years = 2020, biomes = "Amazon",
                          uf = "MT", municipalities = mt_mun)
combined2 <- sus_grid_join(sih_agg, prodes, by = "code_muni")

## End(Not run)
```

`sus_grid_koppen` *Assign Kopppen-Geiger Climate Zones to Brazilian Municipalities*

Description

Adds a `zona_koppen` column to a `climasus_df` (or plain `data.frame`) with municipality-level Koppen-Geiger climate classification following Alvares et al. (2013), who produced the first high-resolution (1 km) Koppen map for Brazil.

Usage

```
sus_grid_koppen(
  data,
  mode = "approx",
  koppen_sf = NULL,
  as_factor = TRUE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`data`	A `climasus_df` or `data.frame` containing a `code_muni` column (6- or 7-digit IBGE municipality code), **or** an `sf` object of Brazilian municipalities.
`mode`	Character. Assignment mode: `"approx"` (default, no extra dependencies) or `"exact"` (requires `sf` and `koppen_sf`).
`koppen_sf`	An `sf` polygon object with a column named `koppen` containing the zone codes (`"Af"`, `"Am"`, etc.). Required when `mode = "exact"`. The Alvares et al. (2013) shapefile is available at https://doi.org/10.1127/0941-2948/2013/0507. Ignored for `mode = "approx"`.

as_factor	Logical. Return zona_koppen as an ordered factor with canonical level order (Af < Am < As < Aw < BSh < Cf < Cw)? Default TRUE. Set to FALSE to get a plain character column.
lang	Character. Language for messages: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default TRUE.

Details

Two assignment modes are available:

- "approx" (default): rule-based assignment from municipality centroid coordinates stored in the built-in municipio_meta table. Fast and dependency-free, with ~85% accuracy at the municipal level.
- "exact": spatial join between municipality centroids and a user- supplied sf polygon layer (koppen_sf). Produces exact results when the original Alvares et al. (2013) shapefile is provided.

Value

The input object with an additional column zona_koppen. If the input was a climasus_df, the returned object preserves its class and sus_meta attribute with the history updated.

Koppen zones for Brazil (Alvares et al. 2013)

Af Tropical humid (equatorial) – Amazon / NE coast.

Am Tropical monsoon – North / Center-West.

As Tropical summer dry – NE coast.

Aw Tropical winter dry (savanna) – Central Brazil.

BSh Hot semi-arid – NE *sertao*.

Cf Subtropical humid – South / coastal SE.

Cw Subtropical winter dry – SE plateau / Center-West uplands.

References

Alvares, C.A., Stape, J.L., Sentelhas, P.C., Goncalves, J.L.M., & Sparovek, G. (2013). Koppen's climate classification map for Brazil. *Meteorologische Zeitschrift*, 22(6), 711-728. doi:10.1127/09412948/2013/0507

See Also

sus_grid_era5(), sus_mod_casecrossover(), sus_mod_dlnm()

Examples

```
## Not run:
library(climasus4r)

# With a climasus_df that has code_muni
df_with_koppen <- sus_grid_koppen(df_aggregated, mode = "approx")
table(df_with_koppen$zona_koppen)

# Exact mode with Alvares et al. 2013 shapefile
library(sf)
kop_sf <- sf::read_sf("koppen_brazil_alvares2013.shp")
names(kop_sf)[names(kop_sf) == "zone"] <- "koppen"  # rename if needed
df_exact <- sus_grid_koppen(df_aggregated, mode = "exact", koppen_sf = kop_sf)

## End(Not run)
```

`sus_grid_pdsi` *Import Palmer Drought Severity Index (PDSI) for Brazilian Municipalities*

Description

`sus_grid_pdsi()` downloads Palmer Drought Severity Index (PDSI) data, crops it to Brazil, spatially aggregates to municipalities, and returns a `climasus_df` compatible with `sus_mod_dlnm()` and `sus_climate_anomaly()`.

PDSI (Palmer, 1965) quantifies cumulative moisture departures relative to local climate normals using a two-layer soil water balance model. Values range from approximately -10 (extreme drought) to +10 (extreme wet), with operational categories at -2 (moderate drought) to -4 (extreme drought). Health applications in Brazil:

- Drought (PDSI $<$ -2): malnutrition, diarrheal disease, mental health stress, vector-borne disease in semi-arid Northeast and Amazônia
- Wet periods (PDSI $>$ 2): leptospirosis, hepatitis A, flooding

Available sources:

- `"terraclimate"` (default): TerraClimate monthly PDSI (Abatzoglou et al., 2018; University of Idaho). Resolution 1/24° (~4 km), 1950–2025, WGS84, no authentication. One NetCDF file (~165 MB) per year.
- `"noaa_psl"`: Dai (2011) self-calibrated PDSI from NOAA Physical Sciences Laboratory. Resolution 2.5°, 1850–2018, single global file (~40 MB). Coarser but longer record.

Usage

```
sus_grid_pdsi(
  years = NULL,
  months = 1:12,
  source = c("terraclimate", "noaa_psl"),
  municipalities = NULL,
  agg_fun = "mean",
  crop_brazil = TRUE,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/pdsi",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Integer vector. Years to download. TerraClimate: 1950–2025. NOAA PSL: 1850–2018. `NULL` (default) = last two complete years.
`months`	Integer vector (1–12). Months to include. Default `1:12`.
`source`	Character. Data source: `"terraclimate"` (default) or `"noaa_psl"`.
`municipalities`	An `sf` POLYGON object (e.g., from `geobr::read_municipality()`). When provided, rasters are aggregated and a `climasus_df` is returned. If `NULL`, returns cached NetCDF file paths.
`agg_fun`	Character. Spatial aggregation for `exactextractr::exact_extract()`. Default `"mean"` (area-weighted).
`crop_brazil`	Logical. Crop global rasters to Brazil's bounding box. Default `TRUE`.
`use_cache`	Logical. Reuse cached NetCDF files and Parquet results. Default `TRUE`.
`cache_dir`	Character. Root cache directory. Default `"~/.climasus4r_cache/pdsi"`.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

- If `municipalities` is provided: a `climasus_df` with columns `code_muni`, `date` (Date, first day of month), and `pdsi` (numeric, unitless). Metadata: `stage = "climate"`, `type = "pdsi"`.
- If `municipalities = NULL`: a named character vector of cached NetCDF file paths.

PDSI classification

```
PDSI >= +4.0 : Extremely wet
PDSI  +3.0 to +3.99 : Very wet
PDSI  +2.0 to +2.99 : Moderately wet
PDSI  -1.99 to +1.99 : Near normal
PDSI  -2.0 to -2.99 : Moderate drought  (D1)
```

```
PDSI  -3.0 to -3.99 : Severe drought    (D2)
PDSI <= -4.0 : Extreme drought           (D3-D4)
```

Data sources

- TerraClimate: Abatzoglou, J.T. et al. (2018). TerraClimate, a high- resolution global dataset of monthly climate and climatic water balance from 1958–2015. *Scientific Data*, 5, 170191. doi:10.1038/sdata.2017.191. URL: https://climate.northwestknowledge.net/TERRACLIMATE-DATA/
- NOAA PSL: Dai, A. (2011). Characteristics and trends in various forms of the PDSI during 1900–2008. *J. Geophys. Res.*, 116, D12115. doi:10.1029/2010JD015541. URL: https://downloads.psl.noaa.gov/Datasets/dai_pdsi/

Packages required

`terra` and `exactextractr` (both in `Suggests`) are required when `municipalities` is provided.

See Also

`sus_climate_compute_spi()`, `sus_climate_compute_spei()`, `sus_grid_chirps()`, `sus_grid_join()`, `sus_mod_dlnm()`

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# TerraClimate PDSI for Mato Grosso, 2015-2022
pdsi_mt <- sus_grid_pdsi(
  years          = 2015:2022,
  months         = 1:12,
  municipalities = mt_mun,
  lang           = "pt"
)
sus_meta(pdsi_mt, "stage")  # "climate"
sus_meta(pdsi_mt, "type")   # "pdsi"

# Longer history with NOAA PSL (2.5 deg)
pdsi_hist <- sus_grid_pdsi(
  years          = 1960:2018,
  source         = "noaa_psl",
  municipalities = mt_mun,
  lang           = "en"
)

# Join to health data for DLNM
combined <- sus_grid_join(sih_mt, pdsi_mt)

## End(Not run)
```

`sus_grid_pollution_cams` *Import CAMS Pollution Data for Brazilian Municipalities*

Description

`sus_grid_pollution_cams()` downloads pre-processed CAMS (Copernicus Atmosphere Monitoring Service) daily pollution data for Brazilian municipalities from Zenodo. No API key is required.

Data cover six pollutants — PM2.5, PM10, CO, O3, NO2 and SO2 — aggregated to daily means/maxima/minima per municipality from 2003 to 2024. Files are hosted on Zenodo (CC-BY) as Parquet and are downloaded once and cached locally.

The return value is a `climasus_df` at `stage = "climate"`, `type = "pollution_cams"`, directly compatible with `sus_climate_anomaly()`, `sus_climate_aggregate()`, and `sus_mod_dlnm()`.

Usage

```
sus_grid_pollution_cams(
  pollutants = c("pm25", "pm10"),
  metric = "mean",
  years = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/cams",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`pollutants`	Character vector. Pollutants to include. Allowed values: `"pm25"`, `"pm10"`, `"co"`, `"o3"`, `"no2"`, `"so2"`, or `"all"`. Default: `c("pm25", "pm10")`.
`metric`	Character. Daily aggregation statistic. One of: `"mean"` (default), `"max"`, `"min"`, or `"all"` (all three).
`years`	Integer vector. Years to include (2003–2024). `NULL` (default) returns all available years.
`use_cache`	Logical. Re-use previously downloaded Parquet files. Default `TRUE`.
`cache_dir`	Character. Directory for cached Parquet files. Default `"~/.climasus4r_cache/cams"`.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_df` tibble with columns:

- `code_muni` — 7-digit IBGE municipality code (character).

- `date` — Date.
- `{pollutant}_{metric}` — one numeric column per requested pollutant × metric combination (e.g. `pm25_mean`, `no2_max`).

Metadata: `stage = "climate"`, `type = "pollution_cams"`.

Data source

Saldanha, R. et al. CAMS pollution daily averages for Brazilian municipalities (2003–2024). Zenodo. CC-BY 4.0.
PM2.5: doi:10.5281/zenodo.16374139 | PM10: doi:10.5281/zenodo.16419737 | CO: doi:10.5281/zenodo.18641834 | O3: doi:10.5281/zenodo.18641945 | NO2: doi:10.5281/zenodo.18642048 | SO2: doi:10.5281/zenodo.18642198

Units

- PM2.5, PM10, O3, NO2, SO2: µg/m³
- CO: ppm

See Also

`sus_climate_anomaly()`, `sus_climate_aggregate()`, `sus_mod_dlnm()`

Examples

```
## Not run:
# PM2.5 and NO2 daily means for 2019-2022
cams <- sus_grid_pollution_cams(
  pollutants = c("pm25", "no2"),
  metric     = "mean",
  years      = 2019:2022,
  lang       = "pt"
)

sus_meta(cams, "stage")  # "climate"
sus_meta(cams, "type")   # "pollution_cams"

# All pollutants, all metrics, all years
cams_full <- sus_grid_pollution_cams(
  pollutants = "all",
  metric     = "all",
  lang       = "en"
)

## End(Not run)
```

`sus_grid_pollution_ghap` *Import GHAP High-Resolution Pollution Data for Brazilian Municipalities*

Description

`sus_grid_pollution_ghap()` downloads, crops, spatially aggregates, and caches GHAP (GlobalHighAirPollutants) raster data to Brazilian municipalities. No API key is required.

GHAP provides AI-generated, seamless ground-level air pollutant fields at 1 km (PM2.5, CO) and 10 km (O3) resolution. Files are downloaded as NetCDF from Zenodo, cropped to Brazil, aggregated with area-weighted extraction, and the result is stored as a Parquet cache for fast subsequent access.

Available pollutants and temporal coverage:

- **PM2.5** (`"pm25"`) — daily, monthly & annual, 2017–2022, 1 km, $\mu g/m^3$
- **O3** (`"o3"`) — annual only, 2000–2020, 10 km, ppb
- **CO** (`"co"`) — annual only, 2019–2022, 1 km, mg/m^3
- **NO2** (`"no2"`) — data not yet publicly released

When `municipalities` is provided the function returns a `climasus_df` at `stage = "climate"`, `type = "pollution_ghap"`, compatible with `sus_climate_anomaly()`, `sus_climate_aggregate()`, and `sus_mod_dlnm()`.

Usage

```
sus_grid_pollution_ghap(
  pollutants = "pm25",
  resolution = "monthly",
  years = NULL,
  months = 1:12,
  municipalities = NULL,
  agg_fun = "mean",
  crop_brazil = TRUE,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/ghap",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`pollutants`	Character vector. Pollutants to download. Accepted: `"pm25"`, `"o3"`, `"co"`. Default: `"pm25"`. Note: `"no2"` is not yet publicly available.
`resolution`	Character. Temporal aggregation of source files. One of `"daily"`, `"monthly"`, or `"annual"`. Default: `"monthly"`.

- `"daily"` — available for PM2.5 only (2017–2022). Each monthly ZIP (~3 GB) is downloaded, extracted to a temp directory, each daily NetCDF is aggregated, and the result is cached as Parquet. Subsequent calls read from Parquet without re-extracting.
- `"monthly"` — PM2.5 only (2017–2022).
- `"annual"` — PM2.5 (2017–2022), O3 (2000–2020), CO (2019–2022).

O3 and CO automatically fall back to `"annual"` regardless of this parameter.

`years` Integer vector. Years to download. Availability depends on pollutant and resolution:

- PM2.5 daily, monthly & annual: 2017–2022
- O3 annual: 2000–2020
- CO annual: 2019–2022

`NULL` (default) returns all available years for the selected pollutant.

`months` Integer vector (1–12). Months to include when `resolution = "monthly"`. Ignored for annual data. Default: `1:12`.

`municipalities`
An `sf` object with polygon geometries (e.g., from `geobr::read_municipality()`). If provided, raster data are aggregated to these polygons and a `climasus_df` is returned. If `NULL`, a named character vector of cached NetCDF file paths is returned instead.

`agg_fun` Character. Spatial aggregation function applied by `exactextractr::exact_extract()`. Default: `"mean"` (area-weighted). Other options: `"sum"`, `"median"`, `"min"`, `"max"`.

`crop_brazil` Logical. Crop global rasters to Brazil's bounding box before extraction, significantly reducing memory use. Default `TRUE`.

`use_cache` Logical. If `TRUE` (default), reuse previously downloaded NetCDF files and previously computed municipality-level Parquet caches.

`cache_dir` Character. Root directory for all cached files. Default `"~/.climasus4r_cache/ghap"`. Sub-directories are created per pollutant.

`lang` Character. Message language: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. Print progress messages. Default `TRUE`.

Value

- If `municipalities` is provided: a `climasus_df` tibble with columns `code_muni`, `date`, and one column per pollutant (e.g., `pm25_mean`, `o3_mean`). Metadata: `stage = "climate"`, `type = "pollution_ghap"`.
- If `municipalities = NULL`: a named character vector of paths to the cached NetCDF files.

Data source

Wei, J. et al. (2023). Estimating 1-km-resolution PM2.5 concentrations across China using the space-time random forest approach. *Remote Sensing of Environment*, 231, 111221. DOI: doi:10.5281/zenodo.10800980
Wei, J. et al. GlobalHighAirPollutants (GHAP) v2. Zenodo. CC-BY 4.0.
PM2.5: doi:10.5281/zenodo.10800980 | O3: doi:10.5281/zenodo.10208188 | CO: doi:10.5281/zenodo.14207363

Packages required

Spatial aggregation requires `terra` and `exactextractr` (both in `Suggests`). The Parquet cache requires `arrow` (in `Imports`). Install with `install.packages(c("terra", "exactextractr"))`.

See Also

`sus_climate_anomaly()`, `sus_climate_aggregate()`, `sus_grid_era5()`, `sus_grid_pollution_cams()`, `sus_mod_dlnm()`

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# PM2.5 monthly for Mato Grosso, Jan-Mar 2020
ghap <- sus_grid_pollution_ghap(
  pollutants    = "pm25",
  resolution    = "monthly",
  years         = 2020,
  months        = 1:3,
  municipalities = mt_mun,
  lang          = "pt"
)

sus_meta(ghap, "stage")  # "climate"
sus_meta(ghap, "type")   # "pollution_ghap"

# O3 annual, all available years
ghap_o3 <- sus_grid_pollution_ghap(
  pollutants    = "o3",
  resolution    = "annual",
  municipalities = mt_mun,
  lang          = "en"
)

## End(Not run)
```

`sus_grid_pollution_merra2`
Import MERRA-2 Pollution Data for Brazilian Municipalities

Description

`sus_grid_pollution_merra2()` downloads, spatially aggregates, and caches NASA MERRA-2 aerosol and air quality data for Brazilian municipalities.

MERRA-2 (Modern-Era Retrospective Analysis for Research and Applications v2) from NASA GMAO provides global atmospheric reanalysis from **1980 to present** at 0.625×0.5° (~55 km). It is the longest available reanalysis record for aerosol and pollution studies, making it ideal for historical trend analysis and cross-validation with higher- resolution products such as CAMS or GHAP.

Supported variables:

- **PM2.5** (`"pm25"`) — derived from aerosol mass components ($\mu g/m^3$): `DUSMASS25 + SSSMASS25 + BCSMASS + 1.4*OCSMASS + SO4SMASS`, each multiplied by 1×10^9 to convert kg/m^3 to $\mu g/m^3$.
- **AOD** (`"aod"`) — Total aerosol optical depth at 550 nm (`TOTEXTTAU`), dimensionless, proxy for total aerosol column.
- **SO2** (`"so2"`, experimental) — sulfate surface mass concentration from M2I3NVAER ($\mu g/m^3$).

Usage

```
sus_grid_pollution_merra2(
  pollutants = c("pm25", "aod"),
  resolution = "monthly",
  years = NULL,
  months = 1:12,
  municipalities = NULL,
  agg_fun = "mean",
  earthdata_user = Sys.getenv("EARTHDATA_USER"),
  earthdata_pass = Sys.getenv("EARTHDATA_PASSWORD"),
  netrc_path = NULL,
  crop_brazil = TRUE,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/merra2",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`pollutants`	Character vector. Variables to download. Allowed: `"pm25"`, `"aod"`, `"so2"`. Default: `c("pm25", "aod")`.

`resolution` Character. Temporal resolution of source files. One of:

- `"monthly"` (default) — uses `M2TMNXAER`, one file per month, already monthly-averaged. Recommended for most analyses.
- `"daily"` — uses `M2T1NXAER` (hourly), one file per day; 24 hourly layers are aggregated to a single daily value using `agg_fun`. Note: `so2` only available in monthly resolution.

`years` Integer vector. Years to download (1980 to current year). `NULL` defaults to the last two complete years.

`months` Integer vector (1–12). Months to include. Default `1:12`.

`municipalities` An `sf` POLYGON object (e.g., from `geobr::read_municipality()`). If provided, raster data are aggregated and a `climasus_df` is returned. If `NULL`, returns cached NetCDF paths.

`agg_fun` Character. Spatial aggregation function for `exactextractr::exact_extract()`. Default `"mean"` (area-weighted).

`earthdata_user` Character. Earthdata username. Defaults to `Sys.getenv("EARTHDATA_USER")`.

`earthdata_pass` Character. Earthdata password. Defaults to `Sys.getenv("EARTHDATA_PASSWORD")`.

`netrc_path` Character. Path to a `.netrc` file with Earthdata credentials. If provided, takes precedence over `earthdata_user` / `earthdata_pass`.

`crop_brazil` Logical. Crop rasters to Brazil's bounding box before extraction to save memory. Default `TRUE`.

`use_cache` Logical. Reuse previously downloaded NetCDF files and previously computed Parquet caches. Default `TRUE`.

`cache_dir` Character. Root cache directory. Default `"~/.climasus4r_cache/merra2"`.

`lang` Character. Message language: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. Print progress messages. Default `TRUE`.

Value

- If `municipalities` is provided: a `climasus_df` with columns `code_muni`, `date`, and one column per pollutant (e.g., `pm25_merra2`, `aod_merra2`). Metadata: `stage = "climate"`, `type = "pollution_merra2"`.
- If `municipalities = NULL`: a named character vector of paths to cached NetCDF files.

Authentication

A **free NASA Earthdata Login** account is required (https://urs.earthdata.nasa.gov). Set credentials in one of two ways:

- **Environment variables** (recommended):

```
Sys.setenv(EARTHDATA_USER = "my_user",
           EARTHDATA_PASSWORD = "my_pass")
```

- `.netrc` **file** (for pipelines), then pass its path via `netrc_path`:

```
machine urs.earthdata.nasa.gov login my_user password my_pass
```

After registering, activate GES DISC access at https://disc.gsfc.nasa.gov/earthdata-login.

Data source

Gelaro, R. et al. (2017). The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2). *Journal of Climate*, 30(14), 5419–5454. doi:10.1175/JCLID160758.1
NASA/GSFC/EPS GMAO. MERRA-2 tavg1_2d_aer_Nx (M2T1NXAER v5.12.4). NASA Goddard Earth Sciences DISC. doi:10.5067/KLICLTZ8EM9D

See Also

sus_grid_pollution_cams(), sus_grid_pollution_ghap(), sus_climate_anomaly(), sus_mod_dlnm()

Examples

```
## Not run:
# Set credentials
Sys.setenv(EARTHDATA_USER = "user", EARTHDATA_PASSWORD = "pass")

library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Monthly PM2.5 + AOD, Q1 2020
merra2 <- sus_grid_pollution_merra2(
  pollutants    = c("pm25", "aod"),
  resolution    = "monthly",
  years         = 2020,
  months        = 1:3,
  municipalities = mt_mun,
  lang          = "pt"
)

sus_meta(merra2, "stage")  # "climate"
sus_meta(merra2, "type")   # "pollution_merra2"

# Historical PM2.5 trend 2000-2010, annual summary
merra2_hist <- sus_grid_pollution_merra2(
  pollutants = "pm25", resolution = "monthly",
  years = 2000:2010, municipalities = mt_mun, lang = "en"
)

## End(Not run)
```

`sus_grid_prodes` *Import PRODES Deforestation Data from TerraBrasilis for Brazilian Municipalities*

Description

`sus_grid_prodes()` downloads INPE PRODES annual deforestation data from the TerraBrasilis WFS API, spatially intersects deforestation polygons with municipality boundaries, and returns municipality-level annual deforestation areas as a `climasus_df` compatible with `sus_mod_dlnm()` and `sus_climate_anomaly()`.

PRODES (Projeto de Monitoramento do Desmatamento na Amazônia Legal por Satélite) is INPE's primary deforestation monitoring program. Deforestation exposure is a key driver of frontier malaria (SINAN), dengue, leishmaniasis, and respiratory disease from biomass burning in Brazil.

Usage

```
sus_grid_prodes(
  years,
  biomes = c("Amazon", "Cerrado", "MataAtlantica", "Caatinga", "Pampa", "Pantanal"),
  uf = NULL,
  municipalities = NULL,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/prodes",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Integer vector. Years to download. Availability by biome: Amazon 2007–present; all others 2000–present.
`biomes`	Character vector. Biomes to include. Any combination of: `"Amazon"`, `"Cerrado"`, `"MataAtlantica"`, `"Caatinga"`, `"Pampa"`, `"Pantanal"`. Default: all six biomes.
`uf`	Character vector. Optional state filter (e.g., `c("MT", "PA")`). Reduces download size. `NULL` = all states in the selected biome(s).
`municipalities`	An `sf` POLYGON object (e.g., from `geobr::read_municipality()`). When provided, deforestation polygons are spatially intersected with municipality boundaries and annual areas are aggregated. If `NULL`, raw deforestation polygons are returned as an sf `climasus_df`.
`use_cache`	Logical. Reuse previously downloaded GeoJSON files and aggregated Parquet caches. Default `TRUE`.
`cache_dir`	Character. Root cache directory. Default `"~/.climasus4r_cache/prodes"`.

`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress. Default `TRUE`.

Value

- If `municipalities` is provided: a `climasus_df` with columns `code_muni` (character), `date` (Date, Jan 1 of the PRODES year), `year` (integer), `deforested_area_km2` (numeric, total area of deforestation polygons intersecting the municipality), `n_patches` (integer, count of distinct deforestation patches), and `biome` (character). Metadata: `stage = "climate"`, `type = "prodes"`.
- If `municipalities = NULL`: the raw deforestation polygon sf object as a `climasus_df` with columns `year`, `state`, `area_km`, `biome`.

PRODES year convention

A PRODES year N covers August 1 of year N-1 through July 31 of year N. The `date` column is set to `YYYY-01-01` by convention (the publication year). For time-series analyses, note that most deforestation peaks occur during the dry season (July–October).

Data source

INPE PRODES via TerraBrasilis WFS. No authentication required.
All biomes accessed via https://terrabrasilis.dpi.inpe.br/geoserver/wfs

Packages required

`sf` is required for spatial intersection when `municipalities` is provided. `httr2` and `jsonlite` are used for WFS downloads (in `Suggests`).

See Also

`sus_grid_fires()`, `sus_climate_anomaly()`, `sus_mod_dlnm()`

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Annual deforestation for Mato Grosso municipalities, Amazon biome
prodes <- sus_grid_prodes(
  years          = 2015:2022,
  biomes         = "Amazon",
  uf             = "MT",
  municipalities = mt_mun,
  lang           = "pt"
)
sus_meta(prodes, "stage")  # "climate"
sus_meta(prodes, "type")   # "prodes"

# Raw polygon data (no aggregation)
```

```
prodes_raw <- sus_grid_prodes(
  years  = 2022,
  biomes = "Amazon",
  uf     = "PA",
  lang   = "en"
)

## End(Not run)
```

`sus_grid_smvi` *Import Soil Moisture Volatility Index (SMVI / Flash Drought) Data*

Description

`sus_grid_smvi()` downloads the global flash drought inventory based on the Soil Moisture Volatility Index (SMVI; Osman et al., 2024), spatially assigns events to Brazilian municipalities, and returns annual or monthly flash drought statistics as a `climasus_df` compatible with `sus_mod_dlnm()` and `sus_mod_vulnerability_index()`.

What is SMVI? SMVI detects *flash droughts* — rapid, high-impact soil moisture deficits that develop within days to weeks. Unlike SPI/SPEI/PDSI (which measure cumulative drought severity), SMVI captures the *speed* of soil moisture decline by comparing a 5-day running average of root-zone soil moisture (0–100 cm) against a 20-day running average. A flash drought event begins when the 5-day average drops below the 20-day average AND reaches the 20th percentile deficit threshold.

Why SMVI is different from SPI/SPEI/PDSI:

- SPI/SPEI/PDSI measure *how dry* a period is (severity)
- SMVI measures *how fast* a drought develops (onset speed) — flash droughts can devastate crops, trigger wildfires, and compromise water supply with little warning

Health connections in Brazil:

- **Malnutrition & food insecurity**: flash droughts destroy crops within weeks; 1–3 month lag to increased child malnutrition in semi-arid Northeast and Amazônia
- **Leptospirosis**: rapid soil rewetting after flash drought events concentrates rodents near water, increasing transmission; 1–4 week lag
- **Leishmaniasis**: drought-driven human migration to water sources increases sandfly contact; 2-year lag documented in Bahia
- **Waterborne disease**: water shortage during flash drought increases dependence on unclean sources

Data source: Global Flash Drought Inventory (Osman et al., 2024). Based on NASA GLDAS-2 root-zone soil moisture (1990–2021). No authentication required. Events file (109 MB) covers the full global domain at 0.25° resolution and is downloaded once and cached.

Usage

```
sus_grid_smvi(
  years = NULL,
  municipalities = NULL,
  aggregate_by = c("year", "month"),
  brazil_only = TRUE,
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/smvi",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`years`	Integer vector. Years to include. Must be within 1990–2021. `NULL` (default) returns all available years.
`municipalities`	An `sf` POLYGON object (e.g., from `geobr::read_municipality()`). When provided, flash drought events are spatially assigned to municipalities and statistics are aggregated. If `NULL`, returns the raw event data frame for Brazil filtered to the requested years.
`aggregate_by`	Character. Temporal resolution of the output when `municipalities` is provided. One of `"year"` (default) or `"month"`. Annual aggregation counts flash drought events and their total duration per municipality per year.
`brazil_only`	Logical. Filter the global dataset to Brazil's bounding box (-75 to -28 lon, -35 to 6 lat) before processing. Default `TRUE`.
`use_cache`	Logical. Reuse cached files. Default `TRUE`.
`cache_dir`	Character. Root cache directory. Default `"~/.climasus4r_cache/smvi"`.
`lang`	Character. Message language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

- If `municipalities` is provided: a `climasus_df` with columns:
  - `code_muni` — 7-digit IBGE municipality code
  - `date` — Date (Jan 1 for annual; first day of month for monthly)
  - `n_fd_events` — integer count of flash drought events
  - `fd_total_days` — total days under flash drought
  - `fd_mean_severity` — mean severity (SV) across events
  - `fd_max_severity` — maximum severity (SV) in the period

  Metadata: `stage = "climate"`, `type = "smvi"`.
- If `municipalities = NULL`: a `climasus_df` with raw event data columns: `cell_id`, `Lon`, `Lat`, `fstdate`, `lstdate`, `SV`, `duration_days`.

Algorithm

SMVI flash drought detection (Osman et al., 2024):

1. Compute 5-day and 20-day running averages of root-zone soil moisture (RZSM, 0–100 cm) from NASA GLDAS-2 data.
2. A flash drought *onset* occurs when the 5-day average drops below the 20-day average.
3. A flash drought *event* is confirmed when the RZSM deficit relative to the 20th percentile threshold is exceeded.
4. Severity (SV) quantifies the cumulative soil moisture deficit during the event.

Data source

Osman, M. et al. (2024). A global flash drought inventory based on soil moisture volatility. *Scientific Data*, 11, 916. doi:10.1038/s41597024038099
Data: https://www.hydroshare.org/resource/642ff72592404a17bb85a8a92b4dbcd6/ (CC BY-SA 4.0, no authentication required)

See Also

sus_climate_compute_spi(), sus_climate_compute_spei(), sus_grid_pdsi(), sus_grid_join(), sus_mod_dlnm()

Examples

```
## Not run:
library(geobr)
mt_mun <- read_municipality(code_muni = "MT", year = 2020)

# Annual flash drought statistics for Mato Grosso
smvi_mt <- sus_grid_smvi(
  years          = 2000:2021,
  municipalities = mt_mun,
  aggregate_by   = "year",
  lang           = "pt"
)
sus_meta(smvi_mt, "stage")  # "climate"
sus_meta(smvi_mt, "type")   # "smvi"

# Monthly resolution
smvi_mo <- sus_grid_smvi(
  years          = 2015:2021,
  municipalities = mt_mun,
  aggregate_by   = "month"
)

# Join to health data for DLNM
combined <- sus_grid_join(sih_mt, smvi_mt)

## End(Not run)
```

`sus_install_deps` *Install Optional Dependencies for climasus4r*

Description

Installs optional packages needed for specific feature groups. Tries multiple installation strategies, including Posit Package Manager (PPM) for pre-compiled binaries on Posit Cloud / Linux.

Usage

```
sus_install_deps(features = "all", force = FALSE, lang = "pt")
```

Arguments

`features`	Character vector. Feature groups to install. One or more of `"parquet"`, `"spatial"`, `"plot"`, `"models"`, `"all"`. Default `"all"`.
`force`	Logical. Re-install even if the package is already available. Default `FALSE`.
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.

Details

`"parquet"` Installs `arrow` (fastest Parquet) and/or `nanoparquet` (pure-R fallback, no C++ required).

`"spatial"` Installs `sf`, `geobr`, `geocodebr`, `sfarrow`, `censobr`.

`"plot"` Installs `ggplot2`, `plotly`, `patchwork`, `ggsci`.

`"models"` Installs `dlnm`, `mvmeta`, `survival`, `xgboost`.

On Posit Cloud, PPM binaries are used automatically for faster installation.

Examples

```
## Not run:
# Install all optional dependencies
sus_install_deps()

# Install only Parquet backend (no C++ required path)
sus_install_deps("parquet")

# Force reinstall arrow
sus_install_deps("parquet", force = TRUE)

## End(Not run)
```

`sus_meta` *Manage climasus_df S3 Class Metadata and Storage Backends*

Description

Unified interface for getting, setting, and managing metadata in `climasus_df` objects, with native support for three storage backends: in-memory **tibble** (default), columnar **Parquet** files via the **arrow** package, and analytical **DuckDB** databases via the **duckdb** package.

A single, unified interface for reading and writing metadata on `climasus_df` objects (including Arrow Tables and DuckDB results). Minimises namespace pollution by replacing the previous family of ten accessor functions.

Usage

```
sus_meta(
  x = NULL,
  field = NULL,
  system = NULL,
  stage = NULL,
  type = NULL,
  backend = NULL,
  add_history = NULL,
  print_history = FALSE,
  valid_values = NULL,
  ...
)

sus_meta(
  x = NULL,
  field = NULL,
  system = NULL,
  stage = NULL,
  type = NULL,
  backend = NULL,
  add_history = NULL,
  print_history = FALSE,
  valid_values = NULL,
  ...
)
```

Arguments

`x`	A `climasus_df` object, Arrow Table, DuckDB result, or `NULL` when using `valid_values`.
`field`	Character. When provided as the only extra argument, returns the value of that metadata field (`"system"`, `"stage"`, `"type"`, `"spatial"`, `"temporal"`, `"backend"`, `"created"`, `"modified"`, `"history"`, `"user"`).

`system`	Character. DATASUS system identifier (e.g. `"SIM"`).
`stage`	Character. Pipeline stage (e.g. `"filter_cid"`).
`type`	Character. Data type (e.g. `"raw"`).
`backend`	Character. Storage backend: `"tibble"` (default), `"parquet"`, or `"duckdb"`.
`add_history`	Character. A single string appended (with timestamp) to the processing history.
`print_history`	Logical. If `TRUE`, prints the history and returns `invisible(NULL)`.
`valid_values`	Character. When provided, returns the allowed values for that vocabulary (`"system"`, `"stage"`, `"type"`, `"backend"`).
`...`	Additional named metadata fields to update.

Details

Operation Dispatch Order:

1. `valid_values` — vocabulary query (no `x` required)
2. `from_parquet` — read Parquet file and reconstruct
3. `from_duckdb` — read DuckDB table and reconstruct
4. `print_history` — print processing history
5. `add_history` — append timestamped history entry
6. `to_parquet` — write to Parquet file
7. `to_duckdb` — register in DuckDB
8. `field` getter — return single metadata field
9. `...` updates — update metadata fields
10. Default — return all metadata as list

Metadata Persistence Across Backends:

When writing to Parquet, `sus_meta` is serialised as JSON and stored in the Arrow schema metadata under the key `"climasus_meta"`, making it fully recoverable after reading the file back.

When writing to DuckDB, `sus_meta` is stored in a companion table named `<view_name>__meta` within the same connection, enabling SQL-level introspection of pipeline provenance.

Supports multiple backends:

- **climasus_df**: metadata stored as `sus_meta` attribute
- **Arrow Table**: metadata embedded in schema under `"climasus_meta"` key
- **DuckDB result**: metadata extracted from companion `<table>__meta` table

Value

Depends on the operation:

- **Get all metadata**: Named list with all metadata fields.
- **Get specific field**: Value of the requested field.
- **Update metadata**: Updated `climasus_df` object.
- **Add history**: Updated `climasus_df` with new history entry.
- **Print history**: `invisible(NULL)` after printing.
- **Valid values**: Character vector of allowed values.
- **to_parquet**: Updated `climasus_df` (`backend = "parquet"`).
- **from_parquet**: Reconstructed `climasus_df` from file.
- **to_duckdb**: Updated `climasus_df` (`backend = "duckdb"`).
- **from_duckdb**: Reconstructed `climasus_df` from DuckDB.

The updated `climasus_df` object, the requested field value, or `invisible(NULL)` when `print_history = TRUE`.

Backend Operations

The following named arguments trigger backend-specific operations when passed via `...`:

`to_parquet = "<path>"` Converts the `climasus_df` to an Arrow Table, embeds `sus_meta` as JSON in the Parquet schema, and writes to `<path>`. Returns the updated `climasus_df` with `backend = "parquet"`.

`from_parquet = "<path>"` Reads a Parquet file written by `to_parquet` and reconstructs a fully-featured `climasus_df` with all original metadata. Returns the reconstructed object. `x` is ignored.

`to_duckdb = <DBI connection>` Registers the `climasus_df` as a DuckDB table (default name: `"climasus_data"`) and stores `sus_meta` in a companion table `<duckdb_view>__meta`. Returns the updated `climasus_df` with `backend = "duckdb"`.

`from_duckdb = <DBI connection>` Reads a table from a DuckDB connection and reconstructs a `climasus_df`. Combine with `duckdb_view` and `duckdb_query` for fine-grained control.

`duckdb_view = "<name>"` Name of the DuckDB table/view to read or write (default: `"climasus_data"`).

`duckdb_query = "<SQL>"` Optional SQL `WHERE` clause or full `SELECT` statement applied when reading from DuckDB.

See Also

`write_parquet_climasus`, `write_duckdb_climasus`

Examples

```
## Not run:
#   Basic metadata operations

# Get all metadata
meta <- sus_meta(df)

# Get specific field
sus_meta(df, "stage")     # "filter_cid"
sus_meta(df, "backend")   # "tibble"

# Update metadata
df <- sus_meta(df, stage = "clean", type = "clean")

# Add to processing history
df <- sus_meta(df, add_history = "Removed missing values")

# Print history
sus_meta(df, print_history = TRUE)

# Query controlled vocabulary
sus_meta(valid_values = "backend")   # "tibble" "parquet" "duckdb"
sus_meta(valid_values = "system")    # "SIM" "SIH" ...

#   Arrow / Parquet backend

# Write to Parquet (metadata embedded in schema)
df <- sus_meta(df, to_parquet = "data/sim_respiratory.parquet")
sus_meta(df, "backend")  # "parquet"

# Read back as climasus_df (metadata fully restored)
df2 <- sus_meta(from_parquet = "data/sim_respiratory.parquet")
sus_meta(df2, "stage")   # "filter_cid"

#   DuckDB backend

library(duckdb)
con <- duckdb::dbConnect(duckdb::duckdb())

# Register as DuckDB view
df <- sus_meta(df, to_duckdb = con, duckdb_view = "sim_respiratory")

# SQL query directly on the view
DBI::dbGetQuery(con, "SELECT sexo, COUNT(*) AS n FROM sim_respiratory GROUP BY sexo")

# Read back as climasus_df (with optional SQL filter)
df3 <- sus_meta(from_duckdb = con,
                duckdb_view  = "sim_respiratory",
                duckdb_query = "WHERE ano_obito = 2020")
sus_meta(df3, "stage")   # "filter_cid"

duckdb::dbDisconnect(con)
```

```
## End(Not run)

## Not run:
# With climasus_df
sus_meta(df)
sus_meta(df, "stage")
df <- sus_meta(df, stage = "filter_cid", type = "filter_cid")

# With Arrow Table (auto-extracts from schema metadata)
arrow_tbl <- as_arrow_climasus(df)
sus_meta(arrow_tbl)  # Extracts from Arrow schema
sus_meta(arrow_tbl, "system")

# With DuckDB result (if metadata table exists)
con <- duckdb::dbConnect(duckdb::duckdb())
as_duckdb_climasus(df, con, "my_data")
result <- DBI::dbGetQuery(con, "SELECT * FROM my_data")
sus_meta(result)  # Extracts from companion __meta table

# Query valid values (no object needed)
sus_meta(valid_values = "backend")

## End(Not run)
```

`sus_mod_af` *Attributable Fraction and Number from a DLNM Fit*

Description

Computes population attributable fractions (AF) and attributable numbers (AN) from a `climasus_dlnm` object returned by `sus_mod_dlnm()`. Decomposes the total burden into heat and cold components, reports AF at user-defined exposure percentile ranges, and optionally breaks the burden down by month, year, or season.

Usage

```
sus_mod_af(
  fit,
  threshold = NULL,
  range = NULL,
  pred_at = c(0.75, 0.9, 0.95, 0.99),
  by = NULL,
  nsim = 1000L,
  alpha = 0.05,
  lang = c("pt", "en", "es"),
  verbose = TRUE
)
```

Arguments

fit	A climasus_dlnm object produced by sus_mod_dlnm().
threshold	Numeric or NULL. Exposure value splitting heat (above) and cold (below) components. NULL (default) uses fit$meta$ref_value.
range	Numeric vector c(low, high) or NULL. When provided, AF is also computed for this custom exposure range.
pred_at	Numeric vector (0-1). Quantiles defining the percentile-band table. For each q, the AF above the q-th quantile (hot) and below the (1-q)-th quantile (cold) is reported. Default c(0.75, 0.90, 0.95, 0.99).
by	Character or NULL. Temporal grouping for the period table: "month", "year", or "season". NULL (default) skips the breakdown.
nsim	Integer. Number of Monte Carlo simulations for CI. Default 1000L. Reduce to 200L for quick exploratory runs; increase to 5000L for publication-quality CI.
alpha	Numeric. Significance level for CI. Default 0.05 (95%).
lang	Character. Language for messages: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default TRUE.

Value

A climasus_af object (named list) with:

$total One-row-per-component tibble (total/heat/cold) with AF, AN, and 95% CI.

$by_quantile AF/AN at each percentile band in pred_at.

$by_period Monthly/yearly/seasonal point estimates; NULL when by = NULL.

$daily Per-day date + exposure + cases + AN + AF for mapping.

$custom AF/AN for the custom range; NULL when not provided.

$meta Named list of all parameters used.

Methodology

Point estimates are obtained by calling dlnm::crosspred() at the observed exposure values and computing $AN_t = n_t \times (1 - 1/RR_t)$ for each day. Confidence intervals use Monte Carlo simulation: nsim coefficient vectors are drawn from $N(\hat{\beta}_{cb}, \widehat{V}_{cb})$ via MASS::mvrnorm, crosspred is re-evaluated for each draw, and AF quantiles yield the CI. When **MASS** is unavailable the function falls back to delta-method bounds from the original crosspred object.

References

Gasparrini, A., & Armstrong, B. (2013). The impact of heat waves on mortality. *Epidemiology*, 24(1), 64-71. doi:10.1097/EDE.0b013e3182770cb4

Gasparrini, A., *et al.* (2017). Attributable causes of heat-related mortality across 739 locations. *The Lancet Planetary Health*, 1(9), e360-e367. doi:10.1016/S25425196(17)30156-0

See Also

sus_mod_dlnm(), sus_mod_plot_dlnm()

Examples

```
## Not run:
fit <- sus_mod_dlnm(df_dl, outcome_col = "n_obitos", lang = "pt")

# Total burden with heat/cold split
af <- sus_mod_af(fit, lang = "pt")
print(af)
tidy(af)

# Monthly temporal breakdown
af_m <- sus_mod_af(fit, by = "month", nsim = 200L, lang = "en")
af_m$by_period

# Pool across multiple cities
dplyr::bind_rows(tidy(af_city1), tidy(af_city2))

## End(Not run)
```

---

sus_mod_burden *Ranked Disease Burden Table Across Cities or Strata*

Description

Aggregates city-level attributable fraction or excess mortality/morbidity results into a ranked burden table showing each city's contribution to the total population burden. Accepts `climasus_af`, `climasus_excess`, or `climasus_dlnm` objects (DLNM fits are automatically converted to AF via sus_mod_af()). Produces a concentration curve suitable for Lorenz-style inequality analysis: how unevenly burden is distributed across cities.

Usage

```
sus_mod_burden(
  fits,
  component = "total",
  rank_by = NULL,
  top_n = NULL,
  nsim = 0L,
  alpha = 0.05,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

fits	Named list of climasus_af, climasus_excess, or climasus_dlnm objects. Names become city/stratum labels in the output. All elements must be the same class; mixing types is not allowed.
component	Character. For climasus_af inputs: which heat/cold component to display and rank. One of "total" (default), "heat", "cold", or "all" (all three components; ranking is always derived from the "total" component). Ignored for climasus_excess inputs.
rank_by	Character or NULL. Metric for ranking cities. For AF inputs: "an" (attributable number, default) or "af_pct" (attributable fraction percent). For excess inputs: "excess" (default) or "excess_pct". NULL selects the default for the detected input type.
top_n	Integer or NULL. Keep only the top-N cities after ranking. NULL (default) retains all cities.
nsim	Integer. Monte Carlo simulations used when auto-computing AF from climasus_dlnm inputs. 0L (default) uses the faster delta-method CI.
alpha	Numeric. Significance level for confidence intervals. Default 0.05.
lang	Character. Language for messages: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default TRUE.

Value

A climasus_burden object (named list) with:

$burden_table Tibble with one row per city (three rows per city when component = "all"). For AF inputs: city, component, n_cases, an, an_lo, an_hi, af_pct, af_pct_lo, af_pct_hi, rank, pct_of_total. For excess inputs: city, n_days, observed, expected, excess, excess_lo, excess_hi, excess_pct, rank, pct_of_total.

$concentration City-level tibble with city, rank, pct_of_total, and cumulative_pct for Lorenz-style concentration analysis. Always one row per city, based on the total component.

$total_burden Named list with aggregate statistics: an_total and af_pct_avg for AF inputs; excess_total and excess_pct_avg for excess inputs. Also contains top_city and the top city's metric.

$meta Named list of all parameters used in this call.

See Also

sus_mod_af(), sus_mod_excess(), sus_mod_pool(), sus_mod_metaregression(), sus_mod_dlnm()

Examples

```
## Not run:
# Rank cities by attributable number (AN)
af_list <- list(
  fortaleza = sus_mod_af(fit_fortaleza, verbose = FALSE),
```

```
  recife    = sus_mod_af(fit_recife,    verbose = FALSE),
  salvador  = sus_mod_af(fit_salvador,  verbose = FALSE)
)
burden <- sus_mod_burden(af_list, lang = "pt")
print(burden)
tidy(burden)

# Show heat/cold split, keep top 5 cities
sus_mod_burden(af_list, component = "all", top_n = 5L)

# From DLNM fits directly (AF auto-computed via delta method)
burden2 <- sus_mod_burden(dlnm_list, nsim = 0L, lang = "en")

# Excess-based ranking
exc_list <- list(
  sp  = sus_mod_excess(fit_sp,  method = "from_dlnm", verbose = FALSE),
  rj  = sus_mod_excess(fit_rj,  method = "from_dlnm", verbose = FALSE)
)
sus_mod_burden(exc_list, rank_by = "excess_pct")

## End(Not run)
```

`sus_mod_casecrossover` *Time-Stratified Case-Crossover Analysis for Climate-Health Data*

Description

Fits a time-stratified case-crossover model to quantify the association between a climate exposure and a daily health outcome. Each time stratum (month or week) acts as its own control, removing long-term trends and seasonal confounding by design. Two fitting methods are supported: `"conditional_poisson"` (GLM with stratum fixed effects, default) for aggregate count data, and `"clogit"` (conditional logistic regression) for binary outcomes or rare events.

Usage

```
sus_mod_casecrossover(
  data,
  outcome_col = "n_obitos",
  exposure_col,
  covariates = NULL,
  stratum = "month",
  lag = 0L,
  method = "conditional_poisson",
  family = "quasipoisson",
  alpha = 0.05,
```

```
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`data`	A `climasus_df` at stage `"aggregate"` or `"climate"` (produced by `sus_climate_aggregate()`), or any `data.frame` containing at minimum a `date` column (Date), an outcome count column, and an exposure column.
`outcome_col`	Character. Name of the daily health outcome count column (e.g., `"n_obitos"`, `"n_internacoes"`). Default: `"n_obitos"`.
`exposure_col`	Character. Name of the exposure column. Required; no auto-detection is performed (unlike `sus_mod_dlnm()`). Examples: `"tair_dry_bulb_c"`, `"pm25_mean"`, `"precip_mm"`.
`covariates`	Character vector or `NULL`. Names of additional columns to include as linear confounders. These should be time-varying within strata (e.g., day-of-week dummy, holiday indicator, relative humidity). Long-term trends are already controlled by the strata. Default: `NULL`.
`stratum`	Character. Time stratum type or name of an existing column: • `"month"` (default): year-month strata (e.g., `"2021-07"`). • `"week"`: ISO year-week strata (e.g., `"2021-W27"`). • Any column name in `data`: uses that column's values as stratum IDs.
`lag`	Integer or integer vector. Lag(s) to apply to the exposure before fitting. A single integer uses that specific lag; a vector uses the mean of exposures at all specified lags. Default: `0L` (same-day exposure).
`method`	Character. Fitting method: • `"conditional_poisson"` (default): GLM with stratum fixed effects. Preferred for aggregate count data; uses `stats::glm()` with the specified `family`. • `"clogit"`: conditional logistic regression via `survival::clogit()`. Requires the `survival` package. Treats outcome as binary (`is_case = count > 0`); best for rare events.
`family`	Character. GLM family for `method = "conditional_poisson"`: `"quasipoisson"` (default, robust to overdispersion) or `"poisson"`. Ignored for `method = "clogit"`.
`alpha`	Numeric. Significance level for confidence intervals. Default: `0.05` (95% CI).
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_casecrossover` list with:

`$model` The fitted model object: a `glm` for `"conditional_poisson"` or a `clogit` object for `"clogit"`.

`$or_table` Tibble with one row per exposure term: `term`, `lag_spec`, `estimate` (log-OR), `or`, `or_lo`, `or_hi`, `p_value`.

`$data` The analysis dataset after lag creation and NA removal: columns `date`, `outcome_val`, `exposure_val`, `stratum_id`, and any covariates.

`$diagnostics` List: `n_obs`, `n_cases`, `n_strata`, `disp_ratio` (for `"conditional_poisson"`), `method`, `family`.

`$meta` List of analysis parameters: `outcome_col`, `exposure_col`, `covariates`, `stratum`, `lag`, `method`, `family`, `alpha`, `call_time`.

Statistical framework

`"conditional_poisson"` **method** (Whitaker et al., 2006):

$$\log(E[Y_t]) = \alpha_s + \beta X_t + \mathbf{z}_t^\top \boldsymbol{\gamma}$$

where α_s is a stratum-specific intercept (absorbed as `factor(stratum_id)` in the GLM), X_t is the (possibly lagged or averaged) exposure on day t, and $\mathbf{z}_t$ are optional time-varying covariates. The rate ratio is $RR = e^{\beta}$.

`"clogit"` **method** (Maclure, 1991; Levy et al., 2001):

Uses `survival::clogit()` with the binary indicator $D_t = \mathbf{1}(Y_t > 0)$ as the outcome. Each case day is matched to all other days within the same stratum as controls. This method is most appropriate for rare events (low daily counts) and binary outcomes.

Lag specification

The `lag` argument controls how the daily exposure is lagged before the association is estimated:

- **Single integer** (e.g., `lag = 0`): uses the exposure exactly `lag` days before the outcome day.
- **Integer vector** (e.g., `lag = 0:6`): uses the arithmetic mean of exposures at all specified lags — a moving average common for temperature (e.g., `lag = 0:6` is the "lag 0-6" or "mean temperature over 7 days").

References

Maclure, M. (1991). The case-crossover design: a method for studying transient effects on the risk of acute events. *American Journal of Epidemiology*, 133(2), 144-153. doi:10.1093/oxfordjournals.aje.a115853

Levy, D., Lumley, T., Sheppard, L., Kaufman, J., & Checkoway, H. (2001). Referent selection in case-crossover analyses of acute health effects of air pollution. *Epidemiology*, 12(2), 186-192.

Whitaker, H.J., Farrington, C.P., Spiessens, B., & Musonda, P. (2006). Tutorial in biostatistics: the self-controlled case series method. *Statistics in Medicine*, 25(10), 1768-1797. doi:10.1002/sim.2302

Armstrong, B.G., Gasparrini, A., & Tobias, A. (2014). Conditional Poisson models: a flexible alternative to conditional logistic case crossover analysis. *BMC Medical Research Methodology*, 14, 122. doi:10.1186/1471228814122

See Also

sus_mod_dlnm(), sus_mod_excess(), sus_mod_af()

Examples

```
## Not run:
# Conditional Poisson with lag-0 temperature
cc <- sus_mod_casecrossover(
  df_daily,
  outcome_col  = "n_obitos",
  exposure_col = "tair_dry_bulb_c",
  stratum      = "month",
  lag          = 0L,
  lang         = "pt"
)
print(cc)
tidy(cc)

# Moving-average lag 0-6
cc_lag06 <- sus_mod_casecrossover(
  df_daily,
  outcome_col  = "n_obitos",
  exposure_col = "tair_dry_bulb_c",
  lag          = 0L:6L,
  lang         = "en"
)

## End(Not run)
```

---

sus_mod_dlnm *Distributed Lag Non-linear Model (DLNM) for Climate-Health Analyses*

Description

Fits a Distributed Lag Non-linear Model (DLNM) to quantify the association between a climate exposure and a daily health outcome count. The function accepts a `climasus_df` at stage `"climate"` / type `"distributed_lag"` produced by `sus_climate_aggregate(temporal_strategy = "distributed_lag")`, constructs the bidimensional `crossbasis`, fits a GLM, and returns a `climasus_dlnm` object with the fitted model, pre-computed `crosspred`, exposure-response and lag-response tables, and diagnostic statistics.

Usage

```
sus_mod_dlnm(
  df,
  outcome_col = "n_obitos",
  climate_col = NULL,
```

```
  lag_max = NULL,
  covariates = NULL,
  argvar = list(fun = "ns", df = 4),
  arglag = list(fun = "ns", df = 3),
  family = "quasipoisson",
  ns_df = NULL,
  dof_per_year = 4L,
  ref_value = NULL,
  pred_at = c(0.25, 0.5, 0.75, 0.9, 0.95, 0.99),
  alpha = 0.05,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` at stage `"climate"` and type `"distributed_lag"`, produced by `sus_climate_aggregate(temporal_strategy = "distributed_lag")`. Must contain a `date` column (Date), the outcome column, and lag columns `{climate_col}_lag0`, ..., `{climate_col}_lag{L}`. Geometry columns (sf) are automatically dropped before modelling.
`outcome_col`	Character. Name of the daily health count column (e.g., `"n_obitos"`, `"n_internacoes"`, `"n_dengue"`). Default: `"n_obitos"`.
`climate_col`	Character or `NULL`. Base name of the climate exposure (without `_lagN` suffix, e.g., `"tair_dry_bulb_c"`). `NULL` auto-detects by priority: air temperature > rainfall > humidity > other known variables.
`lag_max`	Integer or `NULL`. Maximum lag to include in the crossbasis. `NULL` auto-detects from the highest `N` present in `{climate_col}_lagN` columns.
`covariates`	Character vector or `NULL`. Names of additional columns in `df` to include as linear confounders in the GLM formula (e.g., `c("rainfall_mm", "rh_mean_porc")`). These are averaged across municipalities before fitting. Default: `NULL`.
`argvar`	Named list. Arguments for the exposure dimension of the `crossbasis` (passed to `dlnm::crossbasis(argvar = ...)`). Default: `list(fun = "ns", df = 4)`. See **Basis selection** section.
`arglag`	Named list. Arguments for the lag dimension of the `crossbasis` (passed to `dlnm::crossbasis(arglag = ...)`). Default: `list(fun = "ns", df = 3)`. See **Basis selection** section.
`family`	Character. GLM family: `"quasipoisson"` (default, recommended), `"poisson"`, or `"negbin"` (redirected to `"quasipoisson"` with warning).
`ns_df`	Integer or `NULL`. Degrees of freedom for the natural spline time trend (`splines::ns(date, df = ns_df)`). `NULL` (default) triggers automatic computation based on `dof_per_year`. Set to `0` to suppress time control entirely.
`dof_per_year`	Integer. Degrees of freedom per year of data used for automatic `ns_df` computation. Default: `4L`. Ignored when `ns_df` is explicitly set.

`ref_value`	Numeric or `NULL`. Exposure value where RR = 1 (the centring value for `dlnm::crosspred()`). `NULL` (default) uses the sample median of lag-0 exposure values.
`pred_at`	Numeric vector of quantile probabilities (0-1) at which to report the cumulative exposure-response curve in `$exposure_response`. Default: `c(0.25, 0.50, 0.75, 0.90, 0.95, 0.99)`.
`alpha`	Numeric (0-1). Confidence level for intervals: CI = $(1-\alpha)$. Default: `0.05` (95% CI).
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress steps and warnings. Default `TRUE`.

Value

A `climasus_dlnm` list with the following components:

`$model` The fitted `glm` object. Includes the `crossbasis` in its model frame — compatible with `dlnm::crosspred()`.

`$crossbasis` The `dlnm::crossbasis` object encoding both the exposure and lag dimensions.

`$pred` A `dlnm::crosspred` object computed on a 100-point grid from the 1st to 99th percentile of exposure. Contains `allRRfit`, `allRRlow`, `allRRhigh` (cumulative overall effect) and `matRRfit`, `matRRlow`, `matRRhigh` (lag-specific effects).

`$exposure_response` Tibble with cumulative RR at the percentiles specified in `pred_at`: columns `pct`, `exposure`, `rr`, `lo`, `hi`.

`$lag_response` Tibble with lag-specific RR at the 75th percentile of exposure (for `dlnm_lag` plots): columns `lag`, `rr`, `lo`, `hi`, `rr_cum`.

`$models` One-row summary tibble for integration with `plot_climate_health(fit = ...)`: `variable`, `n`, `family`, `lag_max`, `ref_value`, `exposure_p75`, `rr`, `lo`, `hi`, `lag_peak`, `disp_ratio`, `aic_poisson`.

`$data_daily` Aggregated daily tibble used for model fitting (outcome summed; climate averaged across municipalities).

`$diagnostics` List: `disp_ratio`, `disp_category`, `autocorr_pval`, `has_autocorr`, `aic_poisson`, `deviance`, `null_deviance`, `df_residual`.

`$meta` List of modelling parameters plus the input `sus_meta` history for pipeline tracing.

The object also carries `attr(result, "sus_meta")` with an updated history entry.

Statistical framework

The DLNM models the log-linear association between an exposure history $X_{t-0}, X_{t-1}, \ldots, X_{t-L}$ and the daily count outcome $Y_t \sim$ Quasi-Poisson(μ_t):

$$\log(\mu_t) = \alpha + \mathbf{w}_t^\top \boldsymbol{\theta} + \sum_{k=1}^{K} \gamma_k s_k(t)$$

where $\mathbf{w}_t$ is the cross-basis vector (outer product of the exposure and lag basis matrices), and $s_k(t)$ are natural splines controlling seasonal confounding. The **cumulative overall effect** across all lags at exposure level x is:

$$RR_{\text{cum}}(x) = \exp\left\{\sum_{l=0}^{L}[f(x,l) - f(x_0,l)]\right\}$$

where x_0 is the reference value (default: sample median).

Family selection

- `"quasipoisson"` (default, recommended): Robust to overdispersion ($\hat{\phi} > 1$). SE inflated by $\sqrt{\hat{\phi}}$. Recommended when `disp_ratio > 1.5` (Armstrong, 2006).
- `"poisson"`: Valid only when $\hat{\phi} \approx 1$. Provides AIC/BIC but underestimates SE under overdispersion.
- `"negbin"`: Redirects to `"quasipoisson"` with a warning — negative binomial is incompatible with `dlnm::crossbasis()`.

Basis selection (`argvar`, `arglag`)

`argvar`	When to use
`list(fun = "ns", df = 3)`	Mild non-linearity; conservative
`list(fun = "ns", df = 4)`	U/J shape (heat+cold, default)
`list(fun = "ns", df = 5)`	Complex dose-response; large N
`list(fun = "lin")`	Linear hypothesis test

`arglag`	When to use
`list(fun = "ns", df = 3)`	Smooth decay; temperature mortality
`list(fun = "ns", df = 4)`	Bimodal lag; cold effect
`list(fun = "poly", df = 2)`	Conservative; small samples

Temporal confounding (`ns_df`, `dof_per_year`)

A natural spline ns(date, df) controls long-term trends and seasonality that confound the climate-health association (Bhaskaran et al., 2013). When `ns_df = NULL` the function auto-computes $df =$ `dof_per_year` $\times\, n_{\text{years}}$. Recommended:

- `dof_per_year = 4`: 1-2 year series; captures annual seasonality.
- `dof_per_year = 6`: Subtropical Brazil; stronger seasonality.
- `dof_per_year = 8`: Arboviral diseases; sub-seasonal variation.
- `ns_df = NULL` with `dof_per_year = 0`: suppress time control (use only when exposure is already an anomaly from climatology).

References

Gasparrini, A., Armstrong, B., & Kenward, M.G. (2010). Distributed lag non-linear models. *Statistics in Medicine*, 29(21), 2224-2234. doi:10.1002/sim.3940

Gasparrini, A. (2011). Distributed lag linear and non-linear models in R: the package dlnm. *Journal of Statistical Software*, 43(8), 1-20. doi:10.18637/jss.v043.i08

Gasparrini, A., Armstrong, B., & Kenward, M.G. (2014). Reducing and meta-analysing estimates from distributed lag non-linear models. *BMC Medical Research Methodology*, 14, 70. doi:10.1186/147122881470

Armstrong, B. (2006). Models for the relationship between ambient temperature and daily mortality. *Epidemiology*, 17(6), 624-631.

Bhaskaran, K., Gasparrini, A., Hajat, S., Smeeth, L., & Armstrong, B. (2013). Time series regression studies in environmental epidemiology. *International Journal of Epidemiology*, 42(4), 1187-1195. doi:10.1093/ije/dyt092

Perkins, S.E. (2015). A review on the scientific understanding of heatwaves. *Atmospheric Research*, 164-165, 242-267.

See Also

`sus_climate_aggregate()`, `sus_climate_compute_heatwaves()`

Examples

```
## Not run:
# Step 1: produce distributed_lag exposure data
df_dl <- sus_climate_aggregate(
  health_data       = sf_sim_spatial,
  climate_data      = df_inmet_filled,
  climate_var       = "tair_dry_bulb_c",
  temporal_strategy = "distributed_lag",
  lag_days          = 21,
  lang              = "pt"
)

# Step 2: fit DLNM
fit <- sus_mod_dlnm(
  df          = df_dl,
  outcome_col = "n_obitos",
  lag_max     = 21,
  argvar      = list(fun = "ns", df = 4),
  arglag      = list(fun = "ns", df = 3),
  family      = "quasipoisson",
  dof_per_year = 4L,
  lang        = "pt"
)

print(fit)
fit$exposure_response   # RR cumulative at percentile grid
fit$lag_response        # RR by lag at p75
fit$diagnostics$disp_ratio
```

```
# Step 3: visualise (requires plot_climate_health)
# plots <- plot_climate_health(
#   data = df_dl, fit = fit,
#   plot_type = c("dlnm_surface", "dlnm_overall", "dlnm_lag")
# )

## End(Not run)
```

`sus_mod_excess` *Excess Mortality and Morbidity from Climate-Health Time Series*

Description

Estimates excess counts (observed minus expected baseline) for daily health outcome series in the context of climate-health research. Supports three baseline methods: counterfactual extraction from a fitted DLNM (`"from_dlnm"`), quasi-Poisson GLM with natural splines (`"spline"`), and Serfling sinusoidal regression (`"serfling"`). Confidence intervals are derived from the baseline model's prediction standard errors. A z-score flag identifies statistically significant excess events.

Usage

```
sus_mod_excess(
  data,
  outcome_col = NULL,
  date_col = "date",
  control_period = NULL,
  study_period = NULL,
  method = NULL,
  dof_per_year = 8L,
  harmonics = 2L,
  family = "quasipoisson",
  threshold_z = 1.96,
  by = NULL,
  alpha = 0.05,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`data`	A `climasus_dlnm` object (for `method = "from_dlnm"`), a `climasus_df` at stage `"aggregate"` or later, or any `data.frame` with a date column and an integer count column.

`outcome_col` Character. Name of the outcome count column. Required when `data` is a data frame; inferred from `fit$meta$outcome_col` when `data` is a `climasus_dlnm`.

`date_col` Character. Name of the date column. Default `"date"`.

`control_period`
Date vector of length 2 specifying the reference period used to fit the baseline model, e.g. `c(as.Date("2018-01-01"), as.Date("2019-12-31"))`. Default `NULL` uses all available data before `study_period`, or all data when both are `NULL`.

`study_period` Date vector of length 2 specifying the period for which excess is reported. Default `NULL` uses the full data range.

`method` Character. Baseline estimation method:

- `"from_dlnm"` — counterfactual from DLNM (requires `climasus_dlnm`).
- `"spline"` (default for data frames) — quasi-Poisson with ns().
- `"serfling"` — Serfling harmonic regression.

`dof_per_year` Integer. Degrees of freedom per year for the spline baseline. Default `8L`.

`harmonics` Integer. Number of sinusoidal harmonics in the Serfling model. Default `2L`.

`family` Character. GLM family: `"quasipoisson"` (default) or `"poisson"`. Ignored for `method = "from_dlnm"`.

`threshold_z` Numeric. Z-score threshold for flagging excess events. Default `1.96` (95th percentile).

`by` Character or `NULL`. Temporal breakdown variable for the period summary: `"year"`, `"month"`, or `"season"`. Default `NULL`.

`alpha` Numeric. Significance level for confidence intervals. Default `0.05`.

`lang` Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_excess` list with:

`$daily` Tibble with one row per day: `date`, `observed`, `expected`, `expected_lo`, `expected_hi`, `excess`, `excess_lo`, `excess_hi`, `z_score`, `is_excess`.

`$total` One-row tibble: total observed, expected, excess, excess percentage, and their confidence bounds.

`$by_period` Temporal breakdown tibble (when `by` is not `NULL`).

`$model` The fitted baseline GLM object (`NULL` for `method = "from_dlnm"`).

`$meta` Metadata list: `method`, `outcome_col`, `date_col`, `control_period`, `study_period`, `family`, `dof_per_year`, `harmonics`, `threshold_z`, `n_obs`, `call_time`.

Statistical framework

`"spline"` and `"serfling"` methods: A quasi-Poisson GLM is fitted on the `control_period`:

$$\log(\mu_t) = \alpha + f(t)$$

where $f(t)$ is either a natural spline over calendar date (`"spline"`) or a Serfling harmonic: $\beta_1 t + \beta_2 t^2 + \sum_{k=1}^{H}[\gamma_k \sin(2\pi k t/365.25) + \delta_k \cos(2\pi k t/365.25)]$ (`"serfling"`). Expected counts for the study period are predicted from this model.

`"from_dlnm"` method: Given a `climasus_dlnm` object, the counterfactual expected mortality without the climate-exposure effect is:

$$\hat{\mu}_t^{(0)} = \hat{\mu}_t / \widehat{RR}(x_t)$$

where $\hat{\mu}_t$ are fitted values and $\widehat{RR}(x_t)$ is the cumulative relative risk at the observed exposure x_t from `dlnm::crosspred()`. This yields the Gasparrini & Armstrong (2013) temperature-attributable excess.

See Also

`sus_mod_dlnm()`, `sus_mod_af()`, `sus_mod_plot_dlnm()`

Examples

```
## Not run:
# From a fitted DLNM
exc <- sus_mod_excess(fit_dlnm, method = "from_dlnm", lang = "en")

# Standalone spline baseline for a heat wave period
exc <- sus_mod_excess(
  df_daily,
  outcome_col    = "n_obitos",
  control_period = c(as.Date("2018-01-01"), as.Date("2021-12-31")),
  study_period   = c(as.Date("2022-01-01"), as.Date("2022-12-31")),
  method         = "spline",
  lang           = "pt"
)
print(exc)
tidy(exc)

## End(Not run)
```

`sus_mod_its` *Interrupted Time Series Analysis for Health Outcome Counts*

Description

Fits a segmented quasi-Poisson regression to estimate the immediate and sustained effects of one or more interruptions (policies, extreme events, pandemics) on a daily health outcome count series. The model decomposes each interruption into a **level change** (immediate step) and a **slope change** (sustained trend shift), controlling for the underlying time trend and seasonal variation via harmonic terms. A counterfactual projection shows what would have been expected had the interruption not occurred.

Usage

```
sus_mod_its(
  data,
  outcome_col = "n_obitos",
  date_col = "date",
  interruption_dates,
  harmonics = 2L,
  family = "quasipoisson",
  covariates = NULL,
  alpha = 0.05,
  counterfactual = TRUE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`data`	A `climasus_df` at stage `"aggregate"` or `"climate"`, or any `data.frame` with a date column and an integer count column.
`outcome_col`	Character. Name of the daily health outcome count column. Default: `"n_obitos"`.
`date_col`	Character. Name of the date column. Default: `"date"`.
`interruption_dates`	Date or character vector of at least one interruption date (ISO `"YYYY-MM-DD"` format). Must lie strictly within the data range. For a single interruption pass a scalar; for multiple, pass a vector.
`harmonics`	Integer. Number of sinusoidal harmonic pairs for seasonal control (H in the model above). `0` suppresses seasonal terms. Default: `2L`.
`family`	Character. GLM family: `"quasipoisson"` (default, robust to overdispersion) or `"poisson"`.
`covariates`	Character vector or `NULL`. Names of additional columns to include as linear confounders (e.g., `"rh_pct"`, `"holiday"`). Must be numeric or dummy-coded. Default: `NULL`.

`alpha`	Numeric. Significance level for confidence intervals. Default: `0.05` (95% CI).
`counterfactual`	Logical. Compute and return counterfactual predictions. Default: `TRUE`.
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_its` list with:

`$model` The fitted `glm` object.

`$effects` Tibble with one row per interruption: `label`, `interruption_date`, `level_ratio`, `level_ci_lo`, `level_ci_hi`, `level_p`, `slope_daily_log`, `slope_ratio_annual`, `slope_p`.

`$counterfactual` Tibble with daily `date`, `observed`, `predicted` (fitted values), `counterfactual`, `cf_lo`, `cf_hi`, `ratio_to_cf`, `prevented`. `NULL` when `counterfactual = FALSE`.

`$segments` Tibble with one row per segment (pre-interruption and each post-interruption interval): `segment`, `start_date`, `end_date`, `n_days`, `mean_observed`, `mean_predicted`, `mean_counterfactual`.

`$data` The analysis dataset with all model covariates.

`$meta` Analysis parameters and diagnostics.

Statistical framework

For K interruption dates $T_1 < T_2 < \ldots < T_K$, the segmented Poisson GLM is:

$$\log(E[Y_t]) = \beta_0 + \beta_1 t + \sum_{j=1}^{K} \left[\beta_{2j}\, D_j(t) + \beta_{2j+1}\, S_j(t)\right] + \sum_{k=1}^{H} \left[\gamma_k \sin\!\left(\tfrac{2\pi k\, t}{365.25}\right) + \delta_k \cos\!\left(\tfrac{2\pi k\, t}{365.25}\right)\right] + \mathbf{z}_t^{\top} \boldsymbol{\eta}$$

where $D_j(t) = \mathbf{1}(t \geq T_j)$ is the step indicator and $S_j(t) = \max(0,\, t - T_j)$ is the slope-change ("hockey stick") basis. The immediate rate ratio at interruption j is $RR_j = e^{\beta_{2j}}$; the sustained trend shift is $e^{\beta_{2j+1}}$ per day (or $e^{365.25\,\beta_{2j+1}}$ per year).

Counterfactual

The counterfactual is the model prediction with all D_j and S_j set to zero — i.e., the pre-interruption baseline trend projected forward through the study period. The difference `observed - counterfactual` is the estimated total attributable change.

References

Bernal, J.L., Cummins, S., & Gasparrini, A. (2017). Interrupted time series regression for the evaluation of public health interventions: a tutorial. *International Journal of Epidemiology*, 46(1), 348-355. doi:10.1093/ije/dyw098

Penfold, R.B., & Zhang, F. (2013). Use of interrupted time series analysis in evaluating health care quality improvements. *Academic Pediatrics*, 13(6 Suppl), S38-44. doi:10.1016/j.acap.2013.08.002

See Also

`sus_mod_excess()`, `sus_mod_casecrossover()`, `sus_mod_dlnm()`

Examples

```
## Not run:
# Single interruption (COVID-19 pandemic onset)
its <- sus_mod_its(
  df_daily,
  outcome_col       = "n_obitos",
  interruption_dates = "2020-03-17",
  harmonics         = 2L,
  lang              = "pt"
)
print(its)
tidy(its)

# Multiple interruptions
its2 <- sus_mod_its(
  df_daily,
  outcome_col        = "n_obitos",
  interruption_dates = c("2020-03-17", "2021-01-18"),
  lang               = "en"
)

## End(Not run)
```

---

sus_mod_metaregression

Meta-Regression of Pooled DLNM Estimates with City-Level Covariates

---

Description

Extends two-stage multivariate meta-analysis (`sus_mod_pool()`) by including city-level covariates (e.g., mean temperature, poverty index, age structure) to explain between-city heterogeneity in climate-health associations. Covariates are automatically standardized to mean 0, SD 1 so the intercept represents the predicted association for an "average city". Returns covariate Wald tests, residual heterogeneity statistics, and BLUP predictions per city adjusted for covariate information.

Usage

```
sus_mod_metaregression(
  fits,
  covariates,
  covariate_cols = NULL,
  city_col = NULL,
```

```
  pred_at = c(0.75, 0.9, 0.95, 0.99),
  blup = TRUE,
  method = "reml",
  alpha = 0.05,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`fits`	A named list of `climasus_dlnm` objects, one per city or region. All fits must use the same `climate_col`, `lag_max`, `argvar`, and `arglag`. Produced by `sus_mod_dlnm()`.
`covariates`	A `data.frame` with one row per city and city-level predictors. City names must be stored as row names (default) or in the column specified by `city_col`. Cities present in `fits` but absent in `covariates` are excluded from the meta-regression with a warning.
`covariate_cols`	Character vector of column names to use as meta-regression predictors. Default `NULL` uses all numeric columns in `covariates` (after removing the city identifier column if any).
`city_col`	Character. Name of the column in `covariates` that holds city identifiers. Default `NULL` uses row names of `covariates`.
`pred_at`	Numeric vector of quantile probabilities (0–1) for the exposure-response summary table. Default `c(0.75, 0.90, 0.95, 0.99)`.
`blup`	Logical. Compute BLUP city-specific predictions? Default `TRUE`.
`method`	Character. `mvmeta` estimation method: `"reml"` (default), `"ml"`, or `"fixed"`.
`alpha`	Numeric. Significance level for confidence intervals. Default `0.05`.
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_metaregression` list with:

`$mvmeta_fit` The `mvmeta` model with covariates.

`$null_fit` Intercept-only `mvmeta` model (for heterogeneity comparison).

`$pooled_pred` `dlnm::crosspred` for the average city (all standardized covariates = 0, i.e., the intercept).

`$blup_preds` Named list of per-city BLUP `crosspred` objects (when `blup = TRUE`).

`$city_table` Tibble: one row per city with raw and BLUP RRs at the 75th percentile exposure.

`$covariate_tests` Tibble: one row per covariate with Wald chi-square statistic (df = `n_crossbasis_coef`) and p-value.

`$heterogeneity` Tibble comparing null and full-model heterogeneity: Cochran Q, df, p-value, I^2, and proportion of heterogeneity explained by covariates (R^2_het).

`$cov_scales` Tibble with the mean and SD used to standardize each covariate (for back-transformation).

`$meta` Metadata: `climate_col`, `outcome_col`, `lag_max`, `n_cities`, `city_names`, `covariate_cols`, `method`, `alpha`, `call_time`.

Statistical framework

Let $\boldsymbol{\theta}_i$ be the p-dimensional vector of cross-basis coefficients for city i, with variance-covariance matrix S_i. The meta-regression model is:

$$\boldsymbol{\theta}_i \sim N\left(\boldsymbol{B}_0 + \sum_{j=1}^{k} \boldsymbol{B}_j x_{ij}, S_i + \Psi\right)$$

where $\boldsymbol{B}_j$ is the p-vector of regression coefficients for covariate j and Ψ is the residual between-city covariance estimated by REML. Because covariates are standardized before fitting, $\boldsymbol{B}_0$ is the predicted association for a city with mean covariate values.

The **significance** of each covariate is assessed by a multivariate Wald test:

$$W_j = \hat{\boldsymbol{B}}_j^{\top} \hat{V}_j^{-1} \hat{\boldsymbol{B}}_j \;\sim\; \chi^2_p$$

where $\hat{V}_j$ is the estimated covariance of $\hat{\boldsymbol{B}}_j$.

References

Gasparrini, A., et al. (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*, 31(29), 3821-3839. doi:10.1002/sim.5471

Berkey, C.S., et al. (1998). Meta-analysis of multiple outcomes by regression with random effects. *Statistics in Medicine*, 17(22), 2537-2550.

See Also

`sus_mod_pool()`, `sus_mod_dlnm()`, `sus_mod_sensitivity()`

Examples

```
## Not run:
# City-level covariates: mean annual temperature and poverty index
city_meta <- data.frame(
  mean_temp  = c(28.5, 22.1, 19.8),
  poverty_pct = c(42, 28, 15),
  row.names  = c("fortaleza", "belo_horizonte", "curitiba")
)

mr <- sus_mod_metaregression(fits, city_meta, lang = "en")
print(mr)
tidy(mr)
```

```
## End(Not run)
```

`sus_mod_ml` *XGBoost Machine Learning for Climate-Health Outcome Prediction*

Description

Trains an XGBoost gradient-boosted tree model to predict health outcomes (disease counts, mortality, hospitalizations) from climate and socioeconomic features. Accepts any aggregated data frame produced by the climasus4r pipeline (or a plain `data.frame`). Uses k-fold cross-validation to select the optimal number of trees (`nrounds`) and returns out-of-fold (OOF) predictions alongside the final model trained on full data.

Usage

```
sus_mod_ml(
  df,
  outcome_col,
  feature_cols = NULL,
  id_col = NULL,
  objective = "count:poisson",
  nrounds = 500L,
  max_depth = 6L,
  eta = 0.05,
  subsample = 0.8,
  colsample_bytree = 0.8,
  min_child_weight = 1,
  nfold = 5L,
  early_stopping = 50L,
  seed = 42L,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A data frame (plain or `climasus_df`) with one row per observation (e.g., city × day or city × month). Must contain `outcome_col` and at least one numeric feature column.
`outcome_col`	Character. Name of the column to predict (the target variable). Must be numeric.
`feature_cols`	Character vector or `NULL`. Names of columns to use as predictors. When `NULL` (default), all numeric columns except `outcome_col` and `id_col` are used automatically.

`id_col` Character or `NULL`. City or group identifier column used for **group-aware cross-validation**: each group is kept entirely in one fold, preventing data leakage across cities. When `NULL`, standard random k-fold is used. Strongly recommended when data has a city dimension.

`objective` Character. XGBoost objective function. One of `"count:poisson"` (default), `"reg:squarederror"`, or `"binary:logistic"`. See Details.

`nrounds` Integer. Maximum number of boosting rounds. The optimal value is selected via early stopping during cross-validation. Default `500L`.

`max_depth` Integer. Maximum tree depth. Default `6L`. Increase for more complex patterns; decrease to reduce overfitting.

`eta` Numeric. Learning rate (step size shrinkage). Default `0.05`. Smaller values require more rounds but often generalize better.

`subsample` Numeric (0, 1]. Row subsampling ratio per tree. Default `0.8`.

`colsample_bytree` Numeric (0, 1]. Column subsampling ratio per tree. Default `0.8`.

`min_child_weight` Numeric. Minimum sum of instance weight in a leaf. Default `1`. Increase to prevent learning rare patterns.

`nfold` Integer. Number of cross-validation folds. Default `5L`.

`early_stopping` Integer. Stop training if the CV metric does not improve for this many consecutive rounds. Default `50L`.

`seed` Integer. Random seed for reproducibility. Default `42L`.

`lang` Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. Print progress messages. Default `TRUE`.

Details

This function complements the DLNM epidemiological pipeline (`sus_mod_dlnm()`): DLNM models the exposure-response relationship and provides causal inference; `sus_mod_ml()` focuses on predictive accuracy across a wider feature space and can incorporate lagged climate variables, socioeconomic covariates, and spatial predictors simultaneously.

Value

A `climasus_ml` object (named list) with:

`$predictions` Tibble with one row per (non-NA) observation: `observed`, `fitted` (train-set prediction), `cv_predicted` (out-of-fold CV prediction), `residual` (observed - cv_predicted), and the `id_col` and `outcome_col` columns if present.

`$importance` Tibble sorted by `Gain` (descending): `Feature`, `Gain` (average loss reduction per split), `Cover` (average number of samples per split), `Frequency` (proportion of splits using this feature). These are the standard XGBoost importance metrics.

`$performance` Named list: `RMSE_train`, `MAE_train`, `R2_train`, `RMSE_cv`, `MAE_cv`, `R2_cv`, `Pearson_cv` (correlation between observed and OOF predictions), and `best_nrounds`.

`$model` The final XGBoost model (class `xgb.Booster`) trained on the full dataset using `best_nrounds`. Pass to `predict()` to generate forecasts for new data (see `predict.climasus_ml()`).

`$cv_log` Data frame from `xgb.cv()` with per-round train and test metrics. Useful for plotting the learning curve.

`$meta` Named list of all parameters used in this call.

XGBoost objective

The `objective` controls the loss function and the scale of predictions:

`"count:poisson"` Poisson log-linear model for non-negative integer counts (deaths, hospitalizations). Predictions are in count scale (predicted lambda). **Default** — appropriate for most DATASUS outcomes.

`"reg:squarederror"` Squared-error regression for continuous outcomes (rates, indices). Predictions are in the same scale as the outcome.

`"binary:logistic"` Binary outcome (0/1). Predictions are probabilities (0, 1).

See Also

`sus_mod_dlnm()`, `sus_mod_vulnerability_index()`, `sus_climate_aggregate()`, `predict.climasus_ml()`

Examples

```
## Not run:
# From a climasus4r pipeline output (aggregated city x day data with
# climate columns from sus_climate_aggregate())
ml <- sus_mod_ml(
  df          = df_aggregated,
  outcome_col = "n_obitos",
  id_col      = "name_muni",        # group-aware CV by city
  objective   = "count:poisson",
  nfold       = 5L,
  lang        = "pt"
)
print(ml)
tidy(ml)    # predictions tibble

# Forecast for a new climate scenario
new_climate <- data.frame(tair_max_c = 36, rh_pct_mean = 60, ...)
predict(ml, newdata = new_climate)

## End(Not run)
```

`sus_mod_plot_af` *Plots and Tables from an Attributable Fraction Analysis*

Description

Produces bar charts, forest plots, and quantile summaries from a `climasus_af` object returned by `sus_mod_af()`. All visualisation types follow standard environmental epidemiology burden-of-disease reporting.

Usage

```
sus_mod_plot_af(
  fit,
  type = c("bar", "forest", "quantile"),
  output_type = c("plot", "table", "all"),
  interactive = FALSE,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

`fit`	A `climasus_af` object produced by `sus_mod_af()`.
`type`	Character. Plot type: `"bar"` (default), `"forest"`, or `"quantile"`.
`output_type`	Character. `"plot"` (default), `"table"`, or `"all"` (named list `$plot`, `$table`, `$data`).
`interactive`	Logical. `TRUE` converts the ggplot2 output to an interactive **plotly** widget. Default `FALSE`.
`base_size`	Numeric. ggplot2 base font size. Default `12`.
`save_plot`	Character or `NULL`. File path to save the plot (PNG/PDF for static; HTML for interactive).
`lang`	Character. Language for labels: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `FALSE`.

Value

Depending on `output_type`:

- `"plot"` -> a `ggplot` or `plotly` object.
- `"table"` -> a `tibble` of the plotted data.
- `"all"` -> a named list: `$plot`, `$table`, `$data`.

Plot types (type)

type	Description
"bar"	Grouped bar chart of AF% with CI by component (heat/cold/total)
"forest"	Horizontal forest plot of AN +/- CI by component
"quantile"	Bar chart of AF% across exposure quantile bands (requires $by_quantile)

See Also

sus_mod_af(), sus_mod_plot_dlnm()

Examples

```
## Not run:
fit <- sus_mod_dlnm(df_dl, outcome_col = "n_obitos")
af  <- sus_mod_af(fit)

sus_mod_plot_af(af, type = "bar",      lang = "pt")
sus_mod_plot_af(af, type = "forest",   lang = "en")
sus_mod_plot_af(af, type = "quantile", lang = "es")
out <- sus_mod_plot_af(af, output_type = "all")
out$table

## End(Not run)
```

---

sus_mod_plot_burden *Plots and Tables from a City-Level Disease Burden Analysis*

Description

Produces lollipop charts, Lorenz concentration curves, and stacked bar charts from a climasus_burden object returned by sus_mod_burden().

Usage

```
sus_mod_plot_burden(
  x,
  type = c("lollipop", "lorenz", "stacked"),
  output_type = c("plot", "table", "all"),
  interactive = FALSE,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

x	A climasus_burden object produced by sus_mod_burden().
type	Character. Plot type: "lollipop" (default), "lorenz", or "stacked".
output_type	Character. "plot" (default), "table", or "all" (named list $plot, $table, $data).
interactive	Logical. TRUE converts the ggplot2 output to an interactive **plotly** widget. Default FALSE.
base_size	Numeric. ggplot2 base font size. Default 12.
save_plot	Character or NULL. File path to save the plot.
lang	Character. Language for labels: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default FALSE.

Value

Depending on output_type:

- "plot" -> a ggplot or plotly object.
- "table" -> a tibble of the plotted data.
- "all" -> a named list: $plot, $table, $data.

Plot types (type)

type	Description
"lollipop"	Ranked lollipop chart of city AN or excess burden (default)
"lorenz"	Lorenz concentration curve showing burden inequality across cities
"stacked"	Stacked heat/cold bar by city (only for AF input with component = "all")

See Also

sus_mod_burden(), sus_mod_af(), sus_mod_excess()

Examples

```
## Not run:
burden <- sus_mod_burden(af_list, lang = "pt")

sus_mod_plot_burden(burden, type = "lollipop", lang = "pt")
sus_mod_plot_burden(burden, type = "lorenz",   lang = "en")
sus_mod_plot_burden(burden, type = "stacked",  lang = "es")
out <- sus_mod_plot_burden(burden, output_type = "all")
out$table

## End(Not run)
```

`sus_mod_plot_dlnm` *Scientific Plots and Tables from a DLNM Fit*

Description

Produces publication-quality visualisations and statistical summary tables from a `climasus_dlnm` object returned by `sus_mod_dlnm()`. Implements all visualisation types standard in distributed lag non-linear model (DLNM) epidemiology: overall exposure-response, lag-response, bidimensional surface, contour map, lag-stratified slices, exposure distribution, and time series.

Usage

```
sus_mod_plot_dlnm(
  fit,
  type = c("overall", "lag", "surface", "contour", "slice", "distribution", "series"),
  output_type = c("plot", "table", "all"),
  exposure_at = NULL,
  lags_at = c(0L, 3L, 7L, 14L, 21L),
  pred_at = c(0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99),
  interactive = FALSE,
  color_palette = "npg",
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

`fit`	A `climasus_dlnm` object produced by `sus_mod_dlnm()`.
`type`	Character. Plot type (see **Plot types** section). Default `"overall"`.
`output_type`	Character. What to return: `"plot"` (default), `"table"`, or `"all"` (named list `$plot`, `$table`, `$data`).
`exposure_at`	Numeric or `NULL`. Exposure value used as the focal point for the `"lag"` plot. `NULL` (default) uses the 75th percentile of lag-0 exposure.
`lags_at`	Integer vector. Lag times shown in the `"slice"` plot. Default `c(0L, 3L, 7L, 14L, 21L)` — automatically clipped to `lag_max`.
`pred_at`	Numeric vector (0–1). Quantile probabilities at which to report cumulative RR in the statistical table. Default: `c(0.05, 0.10, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99)`.
`interactive`	Logical. `TRUE` → `plotly::ggplotly()` / plotly 3-D for `"surface"`; `FALSE` (default) → static ggplot2.
`color_palette`	Character. Name of a **ggsci** palette. Default `"npg"`. Other useful options: `"lancet"`, `"jama"`, `"nejm"`, `"aaas"`.

`base_size`	Numeric. ggplot2 base font size. Default `12`.
`save_plot`	Character or `NULL`. File path to save the output. PNG/PDF for static; HTML for interactive.
`lang`	Character. Language for labels: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `FALSE`.

Value

Depending on `output_type`:

- `"plot"` → a `ggplot` or `plotly` object.
- `"table"` → a `gt_tbl` (or `tibble` when **gt** is unavailable).
- `"all"` → a named list: `$plot`, `$table`, `$data`.

Plot types (`type`)

`type`	Description	Key reference
`"overall"`	Cumulative exposure-response curve with CI + exposure histogram	Gasparrini et al. (2010)
`"lag"`	Lag-specific and cumulative lag-response at a given exposure	Gasparrini et al. (2010)
`"surface"`	3-D bidimensional exposure × lag response surface	Gasparrini (2011)
`"contour"`	2-D contour heat map of exposure × lag (publication-friendly)	Gasparrini et al. (2014)
`"slice"`	Exposure-response curves at specific lag times (multilag)	Armstrong (2006)
`"distribution"`	Exposure variable histogram/density with quartile markers	Bhaskaran et al. (2013)
`"series"`	Daily outcome count + exposure time series	Bhaskaran et al. (2013)

Statistical table (`output_type = "table"` or `"all"`)

When a table is requested, returns a **gt** object (or tibble fallback) with four sections: (1) model specification, (2) cumulative RR at exposure percentiles, (3) lag peak summary, and (4) model diagnostics following reporting standards in Gasparrini et al. (2014) and Bhaskaran et al. (2013).

References

Gasparrini, A., Armstrong, B., & Kenward, M.G. (2010). Distributed lag non-linear models. *Statistics in Medicine*, 29(21), 2224–2234. doi:10.1002/sim.3940

Gasparrini, A. (2011). Distributed lag linear and non-linear models in R: the package dlnm. *Journal of Statistical Software*, 43(8), 1–20. doi:10.18637/jss.v043.i08

Gasparrini, A., Armstrong, B., & Kenward, M.G. (2014). Reducing and meta-analysing estimates from distributed lag non-linear models. *BMC Medical Research Methodology*, 14, 70. doi:10.1186/147122881470

Armstrong, B. (2006). Models for the relationship between ambient temperature and daily mortality. *Epidemiology*, 17(6), 624–631.

Bhaskaran, K., Gasparrini, A., Hajat, S., Smeeth, L., & Armstrong, B. (2013). Time series regression studies in environmental epidemiology. *International Journal of Epidemiology*, 42(4), 1187–1195. doi:10.1093/ije/dyt092

See Also

sus_mod_dlnm(), sus_climate_plot_heatwaves()

Examples

```
## Not run:
fit <- sus_mod_dlnm(df_dl, outcome_col = "n_obitos", lang = "pt")

# Overall exposure-response curve (static)
sus_mod_plot_dlnm(fit, type = "overall", lang = "pt")

# Interactive lag-response
sus_mod_plot_dlnm(fit, type = "lag", interactive = TRUE, lang = "en")

# 3-D surface (interactive only)
sus_mod_plot_dlnm(fit, type = "surface", interactive = TRUE)

# Full output with table
out <- sus_mod_plot_dlnm(fit, type = "overall", output_type = "all", lang = "pt")
out$plot
out$table

## End(Not run)
```

---

sus_mod_plot_ml *Plots and Tables from an XGBoost Machine Learning Model*

Description

Produces feature importance charts, observed-vs-predicted scatter plots, and cross-validation loss curves from a `climasus_ml` object returned by sus_mod_ml().

Usage

```
sus_mod_plot_ml(
  x,
  type = c("importance", "fit", "cv_log"),
  output_type = c("plot", "table", "all"),
  n_top = 20L,
  interactive = FALSE,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

x	A climasus_ml object produced by sus_mod_ml().
type	Character. Plot type: "importance" (default), "fit", or "cv_log".
output_type	Character. "plot" (default), "table", or "all" (named list $plot, $table, $data).
n_top	Integer. Maximum number of features to show in the importance plot. Default 20L.
interactive	Logical. TRUE converts the ggplot2 output to an interactive **plotly** widget. Default FALSE.
base_size	Numeric. ggplot2 base font size. Default 12.
save_plot	Character or NULL. File path to save the plot.
lang	Character. Language for labels: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default FALSE.

Value

Depending on output_type:

- "plot" -> a ggplot or plotly object.
- "table" -> a tibble of the plotted data.
- "all" -> a named list: $plot, $table, $data.

Plot types (type)

type	Description
"importance"	Horizontal bar chart of XGBoost feature Gain (top n_top features)
"fit"	Observed vs. out-of-fold CV-predicted scatter with R^2 and RMSE annotation
"cv_log"	Train/test loss per boosting round from xgb.cv(), with best-round marker

See Also

sus_mod_ml(), sus_mod_plot_dlnm()

Examples

```
## Not run:
ml <- sus_mod_ml(df, outcome_col = "n_obitos", feature_cols = c("tmax", "pop"))

sus_mod_plot_ml(ml, type = "importance", lang = "pt")
sus_mod_plot_ml(ml, type = "fit",        lang = "en")
sus_mod_plot_ml(ml, type = "cv_log",     lang = "es")
out <- sus_mod_plot_ml(ml, output_type = "all")
out$table
```

```
## End(Not run)
```

`sus_mod_plot_pool` *Plots and Tables from a Pooled DLNM Meta-Analysis*

Description

Produces exposure-response curves, city-specific forest plots, and BLUP spaghetti plots from a `climasus_pool` object returned by `sus_mod_pool()`.

Usage

```
sus_mod_plot_pool(
  x,
  type = c("overall", "forest", "spaghetti"),
  output_type = c("plot", "table", "all"),
  interactive = FALSE,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

`x`	A `climasus_pool` object produced by `sus_mod_pool()`.
`type`	Character. Plot type: `"overall"` (default), `"forest"`, or `"spaghetti"`.
`output_type`	Character. `"plot"` (default), `"table"`, or `"all"` (named list `$plot`, `$table`, `$data`).
`interactive`	Logical. `TRUE` converts the ggplot2 output to an interactive **plotly** widget. Default `FALSE`.
`base_size`	Numeric. ggplot2 base font size. Default `12`.
`save_plot`	Character or `NULL`. File path to save the plot.
`lang`	Character. Language for labels: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `FALSE`.

Value

Depending on `output_type`:

- `"plot"` -> a `ggplot` or `plotly` object.
- `"table"` -> a `tibble` of the plotted data.
- `"all"` -> a named list: `$plot`, `$table`, `$data`.

Plot types (type)

type	Description
"overall"	Pooled exposure-response curve with CI ribbon
"forest"	City-specific RR at exposure p75 with CI (original and BLUP)
"spaghetti"	Per-city BLUP curves overlaid on the pooled curve

See Also

sus_mod_pool(), sus_mod_plot_dlnm()

Examples

```
## Not run:
pool <- sus_mod_pool(fits, lang = "pt")

sus_mod_plot_pool(pool, type = "overall",   lang = "pt")
sus_mod_plot_pool(pool, type = "forest",    lang = "en")
sus_mod_plot_pool(pool, type = "spaghetti", lang = "en")
out <- sus_mod_plot_pool(pool, output_type = "all")
out$table

## End(Not run)
```

---

sus_mod_plot_sensitivity

Plots and Tables from a Multi-Stratum Sensitivity Analysis

Description

Produces exposure-response curve overlays, hot-vs-cold RR scatter plots, and grouped forest plots from a climasus_sensitivity object returned by sus_mod_sensitivity().

Usage

```
sus_mod_plot_sensitivity(
  x,
  type = c("curves", "scatter", "bar"),
  output_type = c("plot", "table", "all"),
  interactive = FALSE,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

x	A climasus_sensitivity object produced by sus_mod_sensitivity().
type	Character. Plot type: "curves" (default), "scatter", or "bar".
output_type	Character. "plot" (default), "table", or "all" (named list $plot, $table, $data).
interactive	Logical. TRUE converts the ggplot2 output to an interactive **plotly** widget. Default FALSE.
base_size	Numeric. ggplot2 base font size. Default 12.
save_plot	Character or NULL. File path to save the plot.
lang	Character. Language for labels: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default FALSE.

Value

Depending on output_type:

- "plot" -> a ggplot or plotly object.
- "table" -> a tibble of the plotted data.
- "all" -> a named list: $plot, $table, $data.

Plot types (type)

type	Description
"curves"	Overlay of each stratum's exposure-response curve with CI ribbon
"scatter"	Hot RR vs Cold RR per stratum; point size encodes sensitivity index
"bar"	Horizontal forest plot of hot and cold RR per stratum ordered by SI

See Also

sus_mod_sensitivity(), sus_mod_plot_dlnm()

Examples

```
## Not run:
sens <- sus_mod_sensitivity(fits, lang = "pt")

sus_mod_plot_sensitivity(sens, type = "curves",  lang = "pt")
sus_mod_plot_sensitivity(sens, type = "scatter", lang = "en")
sus_mod_plot_sensitivity(sens, type = "bar",     lang = "es")
out <- sus_mod_plot_sensitivity(sens, output_type = "all")
out$table

## End(Not run)
```

`sus_mod_plot_spacetime` *Visualizations for Space-Time Bayesian Disease Mapping Results*

Description

Produces five types of visualization for `climasus_spacetime_bayes` and `climasus_spacetime_exceedance` objects returned by `sus_mod_spacetime_bayes()` and `sus_mod_spacetime_exceedance()`: choropleth maps of relative risk, temporal trend charts, space-time interaction heatmaps, exceedance probability maps, and fixed-effect forest plots. Static ggplot2 output is returned by default; set `interactive = TRUE` for `plotly::ggplotly()` output.

Usage

```
sus_mod_plot_spacetime(
  x,
  type = c("rr_map", "temporal", "interaction", "exceedance", "coef"),
  municipalities = NULL,
  time_point = NULL,
  time_range = NULL,
  threshold = 1,
  facet_time = FALSE,
  palette = "RdYlBu",
  title = NULL,
  base_size = 11,
  interactive = FALSE,
  lang = "pt",
  ...
)
```

Arguments

`x`	A `climasus_spacetime_bayes` object from `sus_mod_spacetime_bayes()` or a `climasus_spacetime_exceedance` object from `sus_mod_spacetime_exceedance()`.
`type`	Character. One of `"rr_map"` (default), `"temporal"`, `"interaction"`, `"exceedance"`, `"coef"`. See **Plot types**.
`municipalities`	An `sf` polygon object with a `code_muni` column (7-digit IBGE code). Required for `type = "rr_map"` and `type = "exceedance"`. Typically from `geobr::read_municipality()`.
`time_point`	Integer or `NULL`. Single time index to display in `type = "rr_map"`. Ignored when `facet_time = TRUE` or for other types. Default `NULL` (aggregate mean across all periods).
`time_range`	Integer vector of length 2 (`c(t_start, t_end)`) to restrict the time axis in `type = "temporal"` and `type = "interaction"`. Default `NULL` (all periods).

`threshold`	Numeric. RR threshold used to select the exceedance layer from `x$exceedance` when `type = "exceedance"`. Default `1.0`.
`facet_time`	Logical. When `TRUE` and `type = "rr_map"`, produces a faceted map with one panel per time period. Default `FALSE`.
`palette`	Character. Name of a diverging RColorBrewer palette for RR maps (e.g. `"RdYlBu"`, `"RdBu"`, `"PuOr"`). Default `"RdYlBu"`.
`title`	Character or `NULL`. Custom plot title. `NULL` uses a multilingual default.
`base_size`	Numeric. Base font size for ggplot2 themes. Default `11`.
`interactive`	Logical. `TRUE` wraps the ggplot object with `plotly::ggplotly()` (requires **plotly**). Default `FALSE`.
`lang`	Character. Language for axis labels and titles: `"pt"` (default), `"en"`, `"es"`.
`...`	Currently unused; reserved for future arguments.

Value

A `ggplot` object (or `plotly` object when `interactive = TRUE`).

Plot types (`type`)

`"rr_map"` Choropleth map of the posterior mean relative risk (`rr_mean`) merged with the `municipalities` sf polygon layer. Uses a diverging colour scale centred at 1.0. Municipalities where the entire 95\ Supply `time_point` to show a single period; set `facet_time = TRUE` to produce one facet per period; leave both as defaults to display the mean across all periods. Requires `municipalities`.

`"temporal"` Line chart of the population-averaged posterior mean RR (aggregated across municipalities) with a shaded 95\ interval ribbon over time. Supply `time_range` to restrict the displayed periods.

`"interaction"` Tile heatmap of the space-time interaction term (`gamma_mean`) with municipalities on the y-axis and time periods on the x-axis. Requires `x$interaction` in the input object. Supply `time_range` to restrict columns.

`"exceedance"` Choropleth map of the exceedance probability P(RR > `threshold`). Requires a `climasus_spacetime_exceedance` object (or a `climasus_spacetime_bayes` that has an `$exceedance` slot). Requires `municipalities`.

`"coef"` Horizontal forest plot of the fixed-effect posterior summaries (`x$fixed`). A dashed vertical reference line marks zero.

References

Knorr-Held, L. (2000). Bayesian modelling of inseparable space-time variation in disease risk. *Statistics in Medicine*, 19(17-18), 2555-2567. doi:10.1002/sim.1030

Blangiardo, M. & Cameletti, M. (2015). *Spatial and Spatio-temporal Bayesian Models with R-INLA*. Wiley. doi:10.1002/9781118950203

See Also

`sus_mod_spacetime_bayes()`, `sus_mod_spacetime_exceedance()`, `sus_mod_plot_spatial_bayes()`

Examples

```
## Not run:
library(climasus4r)
library(geobr)

shp <- geobr::read_municipality(code_muni = "all", year = 2022)

fit <- sus_mod_spacetime_bayes(
  df         = agg,
  outcome    = "deaths",
  covariates = c("temp_mean", "prec_total"),
  offset     = "expected"
)

# Choropleth map: mean RR across all periods
sus_mod_plot_spacetime(fit, municipalities = shp, type = "rr_map")

# Choropleth for a single period (t = 3)
sus_mod_plot_spacetime(fit, municipalities = shp, type = "rr_map",
                       time_point = 3L)

# Faceted map
sus_mod_plot_spacetime(fit, municipalities = shp, type = "rr_map",
                       facet_time = TRUE)

# Temporal trend
sus_mod_plot_spacetime(fit, type = "temporal", lang = "en")

# Space-time interaction heatmap
sus_mod_plot_spacetime(fit, type = "interaction")

# Exceedance probability (requires climasus_spacetime_exceedance input)
exc <- sus_mod_spacetime_exceedance(fit, thresholds = c(1.0, 1.5, 2.0))
sus_mod_plot_spacetime(exc, municipalities = shp, type = "exceedance",
                       threshold = 1.5)

# Fixed-effect forest plot
sus_mod_plot_spacetime(fit, type = "coef", lang = "en")

## End(Not run)
```

`sus_mod_plot_spatial_bayes` *Visualizations for Bayesian Spatial Disease Mapping Results*

Description

Produces three types of visualization for a `climasus_spatial_bayes` object returned by `sus_mod_spatial_bayes()`: a relative risk choropleth map, a credible interval width (uncertainty) choropleth, a forest plot of fixed effects, or a combined panel of both maps.

Usage

```
sus_mod_plot_spatial_bayes(
  x,
  municipalities = NULL,
  type = c("rr", "uncertainty", "coef", "both"),
  title = NULL,
  base_size = 12,
  lang = "pt",
  ...
)
```

Arguments

`x`	A `climasus_spatial_bayes` object from `sus_mod_spatial_bayes()`.
`municipalities`	An `sf` object containing municipality polygons with a `code_muni` column (7-digit IBGE code). Required for `type = "rr"`, `"uncertainty"`, and `"both"`. Typically obtained via `geobr::read_municipality()`.
`type`	Character. One of `"rr"` (default), `"uncertainty"`, `"coef"`, or `"both"`. See **Plot types**.
`title`	Character or `NULL`. Custom plot title. When `NULL` (default), a multilingual default is used.
`base_size`	Numeric. Base font size passed to ggplot2 themes. Default `12`.
`lang`	Character. Language for axis labels and titles: `"pt"` (default), `"en"`, or `"es"`.
`...`	Currently unused; reserved for future arguments.

Value

- For `type = "rr"`, `"uncertainty"`, `"coef"`: a `ggplot` object.
- For `type = "both"`: a `patchwork` object when **patchwork** is installed, otherwise a named list with elements `$rr` and `$uncertainty`.

Plot types (`type`)

`"rr"` Choropleth map of the smoothed posterior mean relative risk (`rr_mean`). Uses a diverging colour scale centred at 1.0 (navy = below, white = 1.0, red = above). Municipalities where the entire 95\ interval lies above 1 (`rr_lower95 > 1`) receive a black border to flag significantly elevated risk.

`"uncertainty"` Choropleth of the 95\ (`rr_upper95 - rr_lower95`), rendered with the plasma palette from `ggplot2::scale_fill_viridis_c()`. Highlights areas with high posterior uncertainty, often due to small populations or sparse data.

`"coef"` Horizontal forest plot of the fixed-effect posterior summaries stored in `x$fixed`. A dashed vertical reference line marks zero. Points show the posterior mean; horizontal bars show the 95\ credible interval.

`"both"` Side-by-side panel combining the `"rr"` and `"uncertainty"` maps. Uses **patchwork** when available; falls back to a named list with `$rr` and `$uncertainty` otherwise.

See Also

sus_mod_spatial_bayes(), sus_mod_spatial_weights()

Examples

```
## Not run:
library(climasus4r)
library(geobr)

shp <- geobr::read_municipality(code_muni = "all", year = 2022)
W   <- sus_mod_spatial_weights(shp, return_matrix = TRUE)

result <- sus_mod_spatial_bayes(
  df         = agg,
  outcome    = "deaths",
  W          = W,
  covariates = c("temp_mean", "prec_total"),
  offset     = "expected_deaths"
)

# Relative risk map
sus_mod_plot_spatial_bayes(result, municipalities = shp, type = "rr")

# Uncertainty map
sus_mod_plot_spatial_bayes(result, municipalities = shp, type = "uncertainty")

# Forest plot of fixed effects
sus_mod_plot_spatial_bayes(result, type = "coef", lang = "en")

# Combined panel (requires patchwork)
sus_mod_plot_spatial_bayes(result, municipalities = shp, type = "both")

## End(Not run)
```

---

sus_mod_plot_spatial_moran

Two-Panel LISA Visualisation: Cluster Map and Moran Scatter Plot

---

Description

Produces a choropleth LISA cluster map and/or a Moran scatter plot from a `climasus_spatial_moran` object returned by sus_mod_spatial_moran().

Usage

```
sus_mod_plot_spatial_moran(
  x,
```

```
  municipalities = NULL,
  type = c("map", "scatter", "both"),
  alpha = 0.05,
  title = NULL,
  lang = "pt",
  ...
)
```

Arguments

`x`	A `climasus_spatial_moran` object from `sus_mod_spatial_moran()`.
`municipalities`	An `sf` object with at least the column `code_muni` (7-digit IBGE code) and polygon geometry. Required when `type` is `"map"` or `"both"`. Ignored for `type = "scatter"`.
`type`	Character. Which panel(s) to produce. One of `"map"` (default), `"scatter"`, or `"both"`.
`alpha`	Numeric in (0, 1). Significance level used for the subtitle annotation. Default `0.05`.
`title`	Character or `NULL`. Custom title overriding the default translated title. Applies to the first / only panel. Default `NULL`.
`lang`	Character. Output language: `"pt"` (default), `"en"`, or `"es"`.
`...`	Additional arguments passed to `ggplot2::ggsave()` when saving, or ignored.

Value

- `type = "map"` -> a `ggplot` object.
- `type = "scatter"` -> a `ggplot` object.
- `type = "both"` -> a `patchwork` / `ggplot` combined object, or a named list `list(map = <ggplot>, scatter =` if **patchwork** is not installed.

Panel types (`type`)

`type`	Description
`"map"`	Choropleth map coloured by LISA quadrant (HH/LL/HL/LH/NS).
`"scatter"`	Moran scatter plot of standardised values vs. spatial lag.
`"both"`	Side-by-side panels via **patchwork** (named list fallback if not installed).

Colour scheme

Quadrant colours follow the conventional LISA palette: HH = `"red3"`, LL = `"blue3"`, HL = `"darkorange"`, LH = `"steelblue"`, NS = `"grey80"`.

See Also

sus_mod_spatial_moran(), sus_mod_spatial_weights()

Examples

```
## Not run:
# Requires spdep, sf, ggplot2 (and optionally patchwork)
library(climasus4r)

# moran_res <- sus_mod_spatial_moran(df, "deaths", W)
# shp <- geobr::read_municipality(code_muni = "all", year = 2020)

sus_mod_plot_spatial_moran(moran_res, municipalities = shp, type = "both")
sus_mod_plot_spatial_moran(moran_res, municipalities = shp, type = "map", lang = "en")
sus_mod_plot_spatial_moran(moran_res, type = "scatter", lang = "es")

## End(Not run)
```

---

sus_mod_plot_spatial_scan

Choropleth Map of Kulldorff Spatial Scan Cluster Results

Description

Renders a choropleth map from a `climasus_spatial_scan` object produced by sus_mod_spatial_scan(). Each municipality polygon is coloured either by its cluster's Relative Risk (RR, continuous diverging scale) or by a categorical cluster label, making it straightforward to communicate the location, extent, and intensity of spatial health clusters.

Usage

```
sus_mod_plot_spatial_scan(
  x,
  municipalities,
  show_rr = TRUE,
  alpha = 0.05,
  title = NULL,
  lang = "pt",
  ...
)
```

Arguments

`x`	A `climasus_spatial_scan` object returned by sus_mod_spatial_scan().
`municipalities`	An `sf` object with POLYGON or MULTIPOLYGON geometry containing a `code_muni` column that matches the identifiers stored in `x$most_likely_cluster$location_i` and `x$secondary_clusters`.

`show_rr`	Logical. `TRUE` (default) colours municipalities by Relative Risk using a continuous diverging scale. `FALSE` uses categorical cluster-membership colours.
`alpha`	Numeric in (0, 1). Significance threshold used to decide which cluster borders to emphasise. Must match the value used when running `sus_mod_spatial_scan()`. Default `0.05`.
`title`	Character or `NULL`. Plot title. If `NULL` (default) a localised default title is used.
`lang`	Character. Output language: `"pt"` (default), `"en"`, `"es"`.
`...`	Currently unused; reserved for future extensions.

Value

A `ggplot` object. Assign to a variable to modify further or pass to `ggplot2::ggsave()`.

Colour encoding

When `show_rr = TRUE` (default), the fill variable is the numeric RR of the cluster that each municipality belongs to. Background municipalities (`cluster_id == 0`) receive `NA` and are rendered in `"grey80"`. A diverging gradient (navy to white to red) centred on RR = 1 is used. Municipalities belonging to a significant cluster (p-value < `alpha`) additionally receive a thicker border (`linewidth = 1.5`) to distinguish them from the most-likely cluster when it is not itself significant.

When `show_rr = FALSE`, municipalities are filled with categorical colours: the most-likely cluster, up to nine secondary clusters, and the background each receive a distinct colour via `ggplot2::scale_fill_manual()`.

See Also

`sus_mod_spatial_scan()`, `sus_spatial_join()`, `sus_mod_plot_vulnerability()`

Examples

```
## Not run:
library(geobr)
library(dplyr)

muni_sf <- geobr::read_municipality(code_muni = 31, year = 2020)
muni_sf <- dplyr::rename(muni_sf, code_muni = code_muni)

set.seed(1)
df_cases <- tibble::tibble(
  code_muni  = as.character(muni_sf$code_muni),
  cases      = rpois(nrow(muni_sf), 5),
  population = sample(5000:200000, nrow(muni_sf), replace = TRUE)
)

result <- sus_mod_spatial_scan(
  df             = df_cases,
  cases          = "cases",
```

```
  population     = "population",
  municipalities = muni_sf,
  n_simulations  = 199L,
  lang           = "pt"
)

# Continuous RR map (default)
sus_mod_plot_spatial_scan(result, municipalities = muni_sf, lang = "pt")

# Categorical cluster-membership map
sus_mod_plot_spatial_scan(result, municipalities = muni_sf,
                          show_rr = FALSE, lang = "en")

## End(Not run)
```

`sus_mod_plot_swot` *Plots and Tables from a Climate-Health SWOT Analysis*

Description

Produces radar (spider), matrix (2×2 SWOT board), and bar charts from a `climasus_swot` object returned by `sus_mod_swot()`.

Usage

```
sus_mod_plot_swot(
  x,
  type = c("radar", "matrix", "bar"),
  output_type = c("plot", "table", "all"),
  interactive = FALSE,
  entities = NULL,
  top_n = 3L,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

`x`	A `climasus_swot` object from `sus_mod_swot()`.
`type`	Character. Plot type: `"radar"` (default), `"matrix"`, or `"bar"`.
`output_type`	Character. `"plot"` (default), `"table"`, or `"all"` (named list `$plot`, `$table`).
`interactive`	Logical. Convert the ggplot2 output to an interactive **plotly** widget. Default `FALSE`.

entities	Character vector or NULL. Subset of entity names to plot. NULL (default) plots all entities. For type = "matrix", only the first entity in the selection is shown.
top_n	Integer. For type = "matrix", maximum number of indicator bullets to display per quadrant. Default 3L.
base_size	Numeric. ggplot2 base font size. Default 12.
save_plot	Character or NULL. File path to save the plot (PNG/PDF).
lang	Character. Language for labels: "pt" (default), "en", "es".
verbose	Logical. Print progress messages. Default FALSE.

Value

Depending on output_type:

- "plot" → a ggplot or plotly object.
- "table" → a tibble of the plotted data.
- "all" → a named list $plot, $table.

Plot types (type)

type	Description
"radar"	Spider/radar chart of the four quadrant scores (0–100) per entity
"matrix"	Classic 2×2 SWOT board: score, category label, and top indicators for each quadrant. Best for a singl
"bar"	Grouped horizontal bars comparing all four quadrant scores per entity

See Also

sus_mod_swot(), sus_mod_plot_vulnerability()

Examples

```
## Not run:
swot <- sus_mod_swot(vulnerability = vi, af = af_res, lang = "pt")

sus_mod_plot_swot(swot, type = "matrix", lang = "pt")
sus_mod_plot_swot(swot, type = "radar",  lang = "en")
sus_mod_plot_swot(swot, type = "bar",    interactive = TRUE)
out <- sus_mod_plot_swot(swot, output_type = "all")

## End(Not run)
```

`sus_mod_plot_vulnerability` *Plots and Tables from an IPCC Vulnerability Index Analysis*

Description

Produces city vulnerability rankings, IPCC pillar decompositions, and Lorenz concentration curves from a `climasus_vi` object returned by `sus_mod_vulnerability_index()`.

Usage

```
sus_mod_plot_vulnerability(
  x,
  type = c("ranking", "pillars", "lorenz"),
  output_type = c("plot", "table", "all"),
  interactive = FALSE,
  base_size = 12L,
  save_plot = NULL,
  lang = c("pt", "en", "es"),
  verbose = FALSE
)
```

Arguments

`x`	A `climasus_vi` object produced by `sus_mod_vulnerability_index()`.
`type`	Character. Plot type: `"ranking"` (default), `"pillars"`, or `"lorenz"`.
`output_type`	Character. `"plot"` (default), `"table"`, or `"all"` (named list `$plot`, `$table`, `$data`).
`interactive`	Logical. `TRUE` converts the ggplot2 output to an interactive **plotly** widget. Default `FALSE`.
`base_size`	Numeric. ggplot2 base font size. Default `12`.
`save_plot`	Character or `NULL`. File path to save the plot.
`lang`	Character. Language for labels: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `FALSE`.

Value

Depending on `output_type`:

- `"plot"` -> a `ggplot` or `plotly` object.
- `"table"` -> a `tibble` of the plotted data.
- `"all"` -> a named list: `$plot`, `$table`, `$data`.

Plot types (type)

type	Description
"ranking"	Horizontal lollipop of VI score per city, most vulnerable at top
"pillars"	Stacked horizontal bar of exposure/sensitivity/AC pillar scores
"lorenz"	Lorenz concentration curve of VI inequality across cities

See Also

sus_mod_vulnerability_index(), sus_mod_plot_dlnm()

Examples

```
## Not run:
vi <- sus_mod_vulnerability_index(exposure_df = exp_df, lang = "pt")

sus_mod_plot_vulnerability(vi, type = "ranking", lang = "pt")
sus_mod_plot_vulnerability(vi, type = "pillars", lang = "en")
sus_mod_plot_vulnerability(vi, type = "lorenz",  lang = "es")
out <- sus_mod_plot_vulnerability(vi, output_type = "all")
out$table

## End(Not run)
```

---

sus_mod_pool *Two-Stage Multi-City Pooling of DLNM Estimates*

Description

Performs a two-stage multivariate meta-analysis to pool Distributed Lag Non-linear Model (DLNM) estimates across multiple cities or regions. Stage 1 city-specific estimates (coefficients and variance-covariance matrices from sus_mod_dlnm()) are combined using multivariate random-effects meta-analysis via mvmeta, producing pooled exposure-response and lag-response curves, heterogeneity statistics, and Best Linear Unbiased Predictors (BLUPs) for each city.

Usage

```
sus_mod_pool(
  fits,
  exposure_range = NULL,
  n_grid = 100L,
  pred_at = c(0.75, 0.9, 0.95, 0.99),
  blup = TRUE,
```

```
  method = "reml",
  lang = "pt",
  verbose = TRUE
)
```

Arguments

fits	A named list of `climasus_dlnm` objects, one per city or region. All fits must have been produced with the same `climate_col`, `argvar`, `arglag`, and `lag_max`.
exposure_range	Numeric vector of length 2 specifying the exposure grid range for the pooled prediction. Default `NULL` uses the combined range across all city datasets.
n_grid	Integer. Number of exposure grid points for the pooled prediction curve. Default `100L`.
pred_at	Numeric vector of quantile probabilities (0-1) for the pooled exposure-response summary table. Default `c(0.75, 0.90, 0.95, 0.99)`.
blup	Logical. Compute BLUP city-specific predictions? Default `TRUE`.
method	Character. `mvmeta` estimation method: `"reml"` (default, recommended), `"ml"`, or `"fixed"` (fixed-effects, no heterogeneity).
lang	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
verbose	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_pool` list with components:

`mvmeta_fit` The raw `mvmeta` model object.

`pooled_pred` `dlnm::crosspred` object with pooled coefficients.

`exposure_response` Tibble of pooled RR at quantile grid points.

`lag_response` Tibble of pooled RR by lag at the 75th percentile exposure.

`blup_preds` Named list of city-specific BLUP `crosspred` objects (when `blup = TRUE`).

`city_table` Tibble with one row per city showing city-specific and BLUP-adjusted RRs at the 75th percentile.

`heterogeneity` Tibble with Cochran Q, df, p-value, and I^2 statistics.

`meta` List of metadata: `climate_col`, `outcome_col`, `lag_max`, `n_cities`, `city_names`, `method`, `argvar`, `arglag`, `ref_value`, `pred_at`, `call_time`.

Statistical framework

Let $\boldsymbol{\theta}_i$ be the p-dimensional vector of cross-basis coefficients for city i, estimated with variance S_i. The multivariate random-effects model is:

$$\boldsymbol{\theta}_i \sim N(\boldsymbol{\theta}, S_i + \Psi)$$

where Ψ is the between-city covariance matrix estimated by REML. The pooled estimate $\hat{\boldsymbol{\theta}}$ and its variance are passed to `dlnm::crosspred()` to produce a pooled exposure-response surface identical in format to the single-city output.

BLUPs shrink city-specific estimates toward the pooled mean:

$$\tilde{\boldsymbol{\theta}}_i = \hat{\boldsymbol{\theta}} + \hat{\Psi}(\hat{\Psi} + S_i)^{-1}(\hat{\boldsymbol{\theta}}_i - \hat{\boldsymbol{\theta}})$$

Examples

```
## Not run:
fits <- list(
  city_a = sus_mod_dlnm(df_a, ...),
  city_b = sus_mod_dlnm(df_b, ...),
  city_c = sus_mod_dlnm(df_c, ...)
)
pool <- sus_mod_pool(fits, lang = "en")
print(pool)
plot_pool <- sus_mod_plot_dlnm(pool, lang = "en")

## End(Not run)
```

`sus_mod_sensitivity` *Stratified Climate-Health Sensitivity Analysis from DLNM Fits*

Description

Compares exposure-response curves and cumulative relative risks across population strata (e.g., age groups, sexes, income quintiles, municipalities) to identify which subgroups are most sensitive to a climate exposure. Takes a named list of `climasus_dlnm` fits — one per stratum — and extracts the cumulative RR at user-specified hot and cold percentiles for each, producing a ranked comparison table and full stratum-specific ERCs for plotting.

Usage

```
sus_mod_sensitivity(
  fits,
  hot_percentile = 0.99,
  cold_percentile = 0.01,
  stratum_labels = NULL,
  alpha = 0.05,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`fits`	A named list of `climasus_dlnm` objects, one per stratum. All fits must use the same `climate_col` and were produced by `sus_mod_dlnm()`. If the list is unnamed, strata are labelled `"Stratum 1"`, `"Stratum 2"`, etc.
`hot_percentile`	Numeric in (0, 1). Quantile of the observed exposure distribution used as the "hot" comparison point. Default: `0.99` (99th percentile — extreme heat). Must be greater than `cold_percentile`.
`cold_percentile`	Numeric in (0, 1). Quantile of the observed exposure distribution used as the "cold" comparison point. Default: `0.01` (1st percentile — extreme cold). Must be less than `hot_percentile`.
`stratum_labels`	Named character vector or `NULL`. Custom display labels for the strata, with names matching `names(fits)`. If `NULL`, the list names are used as-is. Useful for plotting (e.g., `c(age_65plus = "65+ years", age_1864 = "18-64 years")`).
`alpha`	Numeric. Significance level for confidence intervals. Default: `0.05`.
`lang`	Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_sensitivity` list with:

`$rr_table` Tibble with one row per (stratum, component): `stratum`, `label`, `component` (`"hot"` or `"cold"`), `quantile_prob`, `exposure`, `rr`, `rr_lo`, `rr_hi`, `ref_exposure`.

`$comparison` Tibble ranked by `sensitivity_index` (descending): one row per stratum with `hot_rr`, `hot_rr_lo`, `hot_rr_hi`, `cold_rr`, `cold_rr_lo`, `cold_rr_hi`, `sensitivity_index`, `hot_rank`, `cold_rank`.

`$stratum_curves` Tibble with the full exposure-response curve for each stratum (100-point grid from each `$pred`): columns `stratum`, `label`, `exposure`, `rr`, `rr_lo`, `rr_hi`. Suitable for overlaid ERC plots.

`$meta` List: `climate_col`, `n_strata`, `stratum_names`, `hot_percentile`, `cold_percentile`, `alpha`, `call_time`.

Statistical framework

For each stratum s with fitted DLNM $\hat{f}_s$, the cumulative RR at exposure quantile q (relative to the stratum's reference $x_{0,s}$) is:

$$RR_s(q) = \exp\left\{\sum_{l=0}^{L}\left[\hat{f}_s\big(F_s^{-1}(q),\, l\big) - \hat{f}_s(x_{0,s},\, l)\right]\right\}$$

where $F_s^{-1}(q)$ is the q-th quantile of the observed exposure distribution in stratum s. RRs are extracted by linear interpolation from the pre-computed `dlnm::crosspred` grid stored in each `climasus_dlnm` object — no re-fitting or additional `dlnm` calls are needed.

The **sensitivity index** for stratum s is defined as:

$$SI_s = \log RR_s(\texttt{hot\_percentile}) + \log RR_s(\texttt{cold\_percentile})$$

a combined measure of total climate-attributable vulnerability (heat + cold).

References

Baccini, M., et al. (2011). Heat effects on mortality in 15 European cities. *Epidemiology*, 22(1), 68-77. doi:10.1097/EDE.0b013e3181fdcd5e

Benmarhnia, T., et al. (2015). A difference-in-differences approach to assess the effect of a heat action plan on heat-related mortality, and differences in effectiveness according to sex, age, and socioeconomic status. *Environmental Health Perspectives*, 124(11), 1694-1699. doi:10.1289/ehp.1510173

Armstrong, B.G., et al. (2017). The role of humidity in associations of high temperature with mortality. *Epidemiology*, 28(6), 781-789.

See Also

`sus_mod_dlnm()`, `sus_mod_af()`, `sus_mod_pool()`

Examples

```
## Not run:
# Fit separate DLNMs per age group
fits <- list(
  elderly = sus_mod_dlnm(df_65plus, ...),
  adults  = sus_mod_dlnm(df_1864,   ...),
  youth   = sus_mod_dlnm(df_under18, ...)
)

sens <- sus_mod_sensitivity(fits, lang = "pt")
print(sens)          # ranked comparison table
tidy(sens)           # one-row-per-stratum tibble

## End(Not run)
```

`sus_mod_spacetime_bayes`

Bayesian Spatiotemporal Hierarchical Model with INLA

Description

Fits a full Bayesian spatiotemporal hierarchical model using INLA (Integrated Nested Laplace Approximation). The model decomposes disease risk into structured spatial effects (BYM2, BYM, Besag, or IID), structured temporal effects (RW1, RW2, AR1, or IID), and optionally a space-time interaction following the four Knorr-Held (2000) interaction types. Climate and environmental covariates from a `climasus_df` can be included as fixed effects.

Usage

```
sus_mod_spacetime_bayes(
  df,
  outcome,
  W,
  time_col = "date",
  time_unit = c("year", "month", "week", "auto"),
  covariates = NULL,
  offset = NULL,
  family = c("poisson", "nbinomial", "binomial", "gaussian"),
  spatial_model = c("bym2", "bym", "besag", "iid"),
  temporal_model = c("rw1", "rw2", "ar1", "iid_time"),
  interaction_type = c("none", "I", "II", "III", "IV"),
  pc_prior_u = 0.5,
  pc_prior_alpha = 0.01,
  compute_waic = TRUE,
  compute_cpo = FALSE,
  exceedance_threshold = 1,
  n_samples = 1000L,
  seed = 42L,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` or `climasus_df` at the `aggregate`, `spatial`, or `climate` pipeline stage. Must contain `code_muni` (7-digit IBGE code), the `outcome` column, the `time_col` column, all columns named in `covariates`, and optionally an `offset` column. The data must form a panel (one row per area-time combination).
`outcome`	Character. Name of the response (count) column in `df`.
`W`	A `climasus_weights` object produced by `sus_mod_spatial_weights()`. Must contain either a `$W` dense matrix or a `$nb` neighbour list used to build the INLA graph.
`time_col`	Character. Name of the column containing time information. Can be a `Date`, `integer`, or `character` column. Default `"date"`.
`time_unit`	Character. Temporal aggregation unit used to build the integer time index. One of `"year"`, `"month"`, `"week"`, or `"auto"` (automatic detection based on date range). Default `"auto"`.
`covariates`	Character vector of covariate column names in `df` to include as fixed effects. Default `NULL` (intercept-only model).
`offset`	Character. Name of a **population or expected counts** column in `df` (natural scale). Entered as `log(offset)` in the linear predictor. Default `NULL`.
`family`	Character. Likelihood family. One of `"poisson"` (default), `"nbinomial"` (negative binomial), `"binomial"`, or `"gaussian"`.

`spatial_model` Character. Spatial random effect structure. One of `"bym2"` (default), `"bym"`, `"besag"`, or `"iid"`. See **Spatial model options**.

`temporal_model`
Character. Temporal random effect structure. One of `"rw1"` (default), `"rw2"`, `"ar1"`, or `"iid_time"`.

`interaction_type`
Character. Space-time interaction type following Knorr-Held (2000). One of `"none"` (default), `"I"`, `"II"`, `"III"`, or `"IV"`. See **Knorr-Held interaction types**.

`pc_prior_u` Positive numeric. PC prior parameter u for the BYM2 spatial precision: $P(\sigma > u) = \alpha$. Default `0.5`.

`pc_prior_alpha`
Numeric in (0, 1). PC prior parameter α (tail probability). Default `0.01`.

`compute_waic` Logical. If `TRUE` (default), compute WAIC via `control.compute = list(waic = TRUE)`.

`compute_cpo` Logical. If `TRUE`, compute CPO/PIT values via `control.compute = list(cpo = TRUE)`. Default `FALSE` (slower).

`exceedance_threshold`
Positive numeric. Compute $P(RR > threshold)$ for each space-time cell using the posterior marginals. Default `1.0`.

`n_samples` Positive integer. Number of posterior samples drawn via `INLA::inla.posterior.sample()` to compute RR credible intervals and exceedance probabilities. Default `1000L`.

`seed` Integer. Random seed for reproducibility. Default `42L`.

`lang` Character. Language for CLI messages: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. If `TRUE` (default), emit progress messages via **cli**.

Value

A named list of class `c("climasus_spacetime_bayes", "list")` with the following slots:

`$fixed` `data.frame` with columns `term`, `mean`, `sd`, `lower95`, `upper95`, `mode` (posterior fixed-effect summaries).

`$rr` `data.frame` with columns `code_muni`, `time_idx`, `rr_mean`, `rr_lower95`, `rr_upper95`, `p_exceed` (relative risk per space-time cell with exceedance probability).

`$spatial_re` `data.frame` with columns `code_muni`, `phi_mean`, `phi_sd` (posterior summaries of structured spatial random effect).

`$temporal_re` `data.frame` with columns `time_idx`, `psi_mean`, `psi_sd` (posterior summaries of structured temporal random effect).

`$interaction_re` `data.frame` with columns `code_muni`, `time_idx`, `gamma_mean`, `gamma_sd`, or `NULL` if `interaction_type = "none"`.

`$fitted` Numeric vector of posterior mean fitted values (response scale).

`$waic` Named list with `$waic` (scalar) and `$p_eff` (effective number of parameters), or `NULL` if `compute_waic = FALSE`.

`$dic` Named list with `$dic` (scalar) and `$p_d` (effective parameters), or `NULL`.

`$model_spec` Named list with `$spatial_model`, `$temporal_model`, `$interaction_type`, `$family`.

`$call` The matched call.

`$n_areas` Integer. Number of unique municipalities.

`$n_times` Integer. Number of unique time periods.

Model structure

The linear predictor for area i at time t is:

$$\eta_{it} = \alpha + \mathbf{x}_{it}^{\top}\boldsymbol{\beta} + \phi_i + \psi_t + \delta_{it} + \log(E_{it})$$

where ϕ_i is the spatial random effect, ψ_t the temporal random effect, δ_{it} the optional space-time interaction, and E_{it} the expected counts offset.

Spatial model options

`"bym2"` BYM2 model (Riebler et al. 2016): scaled ICAR + IID with a single precision and a mixing parameter. Recommended for disease mapping. PC priors applied.

`"bym"` Original BYM model (Besag, York & Mollie 1991): ICAR + independent Gaussian components with separate precision parameters.

`"besag"` Intrinsic CAR (Besag 1974): purely structured spatial random effect.

`"iid"` Independent normal random effects (no spatial structure).

Knorr-Held interaction types

`"none"` No space-time interaction.

`"I"` IID interaction: $\delta_{it} \sim N(0, \sigma_\delta^2)$.

`"II"` Structured in time, unstructured in space: separate RW1 per area.

`"III"` Structured in space, unstructured in time: separate ICAR per time period.

`"IV"` Structured in both space and time (inseparable interaction).

PC priors

For the BYM2 spatial model, penalised-complexity (PC) priors are used for the precision hyperparameter: $P(\sigma > u) = \alpha$ where `pc_prior_u` and `pc_prior_alpha` control the tail probability. Default values `u = 0.5`, `alpha = 0.01` are weakly informative and suitable for standardised disease mapping.

References

Knorr-Held, L. (2000). Bayesian modelling of inseparable space-time variation in disease risk. *Statistics in Medicine*, 19(17-18), 2555-2567.

Riebler, A., Sorbye, S. H., Simpson, D., & Rue, H. (2016). An intuitive Bayesian spatial model for disease mapping that accounts for scaling. *Statistical Methods in Medical Research*, 25(4), 1145-1165.

Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models using integrated nested Laplace approximations. *Journal of the Royal Statistical Society B*, 71(2), 319-392.

Blangiardo, M., & Cameletti, M. (2015). *Spatial and Spatio-temporal Bayesian Models with R-INLA*. Wiley.

Besag, J., York, J., & Mollie, A. (1991). Bayesian image restoration, with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43(1), 1-20.

See Also

`sus_mod_spatial_weights()` to build the required adjacency object, `sus_mod_spatial_bayes()` for purely cross-sectional spatial Bayesian models (CARBayes), `sus_mod_dlnm()` for time-series DLNM models.

Examples

```
## Not run:
library(climasus4r)
library(geobr)

# Build spatial weights for Rio Grande do Norte
shp <- geobr::read_municipality(code_muni = "RN", year = 2022)
W   <- sus_mod_spatial_weights(shp, return_matrix = TRUE)

# Prepare aggregated health-climate panel data (areas x time periods)
# agg <- sus_data_aggregate(cleaned_df) |>
#          sus_climate_inmet() |>
#          sus_climate_aggregate(temporal_strategy = "exact")

# BYM2 spatiotemporal model with Poisson family
result <- sus_mod_spacetime_bayes(
  df               = agg,
  outcome          = "deaths",
  W                = W,
  time_col         = "date",
  time_unit        = "month",
  covariates       = c("temp_mean", "prec_total"),
  offset           = "population",
  family           = "poisson",
  spatial_model    = "bym2",
  temporal_model   = "rw1",
  interaction_type = "I",
  pc_prior_u       = 0.5,
  pc_prior_alpha   = 0.01,
```

```
  compute_waic     = TRUE,
  n_samples        = 1000L,
  seed             = 42L,
  lang             = "pt"
)

print(result)
result$fixed
result$rr
result$spatial_re
result$temporal_re
result$waic

## End(Not run)
```

`sus_mod_spacetime_exceedance` *Posterior Exceedance Probabilities from a Spatiotemporal Bayesian Model*

Description

Given a fitted `climasus_spacetime_bayes` object, computes for each municipality × time cell the posterior probability that the relative risk (RR) exceeds one or more critical thresholds: $P(RR > t \mid \text{data})$.

Exceedance probabilities are a cornerstone of Bayesian disease surveillance because they translate uncertain posterior estimates into decision-relevant quantities. For example, $P(RR > 1.5) > 0.80$ can trigger an alert for a municipality in a given week, whereas the raw posterior mean alone may understate uncertainty.

Usage

```
sus_mod_spacetime_exceedance(
  fit,
  thresholds = c(1, 1.5, 2),
  aggregate_time = NULL,
  municipalities = NULL,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`fit`	A `climasus_spacetime_bayes` object produced by `sus_mod_spacetime_bayes()`. Must contain a `$rr` data frame with columns `code_muni`, `time_idx`, `rr_mean`, `rr_lower95`, `rr_upper95`.

`thresholds` Numeric vector of positive RR thresholds for which $P(RR > t)$ is computed. Default `c(1.0, 1.5, 2.0)`.

`aggregate_time`
Character or `NULL`. If `"year"` or `"month"`, aggregate the temporal dimension before reporting exceedances. Requires that `time_idx` be coercible to `Date` (formats `"YYYY-MM-DD"` or `"YYYY-MM"`). Default `NULL` (no aggregation).

`municipalities`
An `sf` object with at least a `code_muni` column and geometry. If provided, geometry is joined to the exceedance table so the result can be mapped directly. Requires the **sf** package. Default `NULL`.

`lang` Character. Language for CLI messages: `"pt"` (default), `"en"`, or `"es"`.

`verbose` Logical. If `TRUE` (default), print progress messages via **cli**.

Value

A named list of class `c("climasus_spacetime_exceedance", "list")` with the following slots:

`$exceedance data.frame` with columns `code_muni`, `time_idx` (or `time_period` if aggregated), `rr_mean`, and one `p_gt_X` column per threshold (e.g. `p_gt_1_0`, `p_gt_1_5`, `p_gt_2_0`).

`$thresholds` Numeric vector of thresholds used.

`$n_exceed data.frame` with columns `threshold` and `n_cells_exceed` counting how many cells have $P(RR > t) > 0.80$.

`$call` The matched call.

Approximation method

Exact computation requires access to the full MCMC sample or INLA posterior marginals. When only the posterior mean and 95\ stored (as in the `$rr` slot), exceedance probabilities are approximated via a **log-normal approximation**:

$$\sigma_{\log} = \frac{\log(\mathrm{RR}_{\mathrm{upper95}}) - \log(\mathrm{RR}_{\mathrm{lower95}})}{2 \times 1.96}$$

$$P(RR > t) \approx \Phi\left(\frac{\log(\mathrm{RR}_{\mathrm{mean}}) - \log(t)}{\sigma_{\log}}\right)$$

where Φ is the standard normal CDF. This is equivalent to assuming that $\log(RR)$ is approximately normally distributed in the posterior, which is a standard approximation for count-data hierarchical models (Richardson et al., 2004).

For exact computation when the full posterior is available, use `INLA::inla.pmarginal()` on the fitted INLA marginals directly.

Temporal aggregation

When `aggregate_time` is `"year"` or `"month"`, the function averages $\log(RR_{\text{mean}})$ and $\sigma_{\log}$ across cells within each municipality $\times$ time period, then recomputes exceedance probabilities for the aggregated values. This is appropriate for surveillance summaries but not for causal inference.

References

Richardson, S., Thomson, A., Best, N., & Elliott, P. (2004). Interpreting posterior relative risk estimates in disease-mapping studies. *Environmental Health Perspectives*, 112(9), 1016-1025.

Lawson, A. B. (2018). *Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology* (3rd ed.). CRC Press.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian Data Analysis* (3rd ed.). CRC Press.

See Also

`sus_mod_spacetime_bayes()` to fit the spatiotemporal Bayesian model, `sus_mod_spatial_bayes()` for the purely spatial (cross-sectional) version, `sus_mod_spatial_moran()` for exploratory spatial autocorrelation analysis.

Examples

```
## Not run:
library(climasus4r)

# Fit a spatiotemporal Bayesian model (requires climasus_spacetime_bayes)
# fit <- sus_mod_spacetime_bayes(df = agg, outcome = "deaths", ...)

# Compute exceedance probabilities for RR thresholds 1.0, 1.5, and 2.0
exc <- sus_mod_spacetime_exceedance(
  fit        = fit,
  thresholds = c(1.0, 1.5, 2.0),
  lang       = "pt"
)

print(exc)
head(exc$exceedance)
exc$n_exceed

# Aggregate to annual exceedances and join municipality geometry
library(geobr)
shp <- geobr::read_municipality(code_muni = "all", year = 2022)
exc_annual <- sus_mod_spacetime_exceedance(
  fit            = fit,
  thresholds     = c(1.0, 1.5, 2.0),
  aggregate_time = "year",
  municipalities = shp,
  lang           = "pt"
```

```
)

## End(Not run)
```

`sus_mod_spacetime_predict` *Generate Predictions from a Fitted Space-Time Bayesian Model*

Description

Produces approximate predictions at new or future space-time points from a `climasus_spacetime_bayes` object fitted by `sus_mod_spacetime_bayes()`. This function is designed for **disease surveillance projections** and **counterfactual scenario analysis** (e.g. "what if temperature was 2 °C higher?").

Usage

```
sus_mod_spacetime_predict(
  fit,
  newdata = NULL,
  horizon = 0L,
  covariates_new = NULL,
  include_ci = TRUE,
  return_samples = FALSE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`fit`	A `climasus_spacetime_bayes` object from `sus_mod_spacetime_bayes()`.
`newdata`	A `data.frame` with at least columns `code_muni` (7-digit IBGE code) and `time_idx` (integer period index, 1-based, matching the encoding used in the original fit) plus any covariate columns required by the model. If `NULL` and `horizon > 0`, a synthetic grid is constructed by crossing all fitted municipalities with the next `horizon` time steps, using covariate values from the last observed period (stored in `fit$data_last_obs`, if available).
`horizon`	Non-negative integer. Number of periods ahead to predict when `newdata` is `NULL`. If `newdata` is provided, `horizon` must be `0L` (default).
`covariates_new`	Named list of covariate overrides for scenario analysis. Each element must be a scalar (applied to all rows) or a vector of length `nrow(newdata)`. Applied after `newdata` columns are extracted. Default `NULL` (no overrides).

`include_ci` Logical. If `TRUE` (default), compute 95\ credible intervals via variance propagation from posterior standard deviations in `fit$fixed`, `fit$spatial_re`, and `fit$temporal_re`.

`return_samples` Logical. If `TRUE`, attempt to draw posterior predictive samples from the stored INLA fit object (`fit$inla_fit`). Requires the model to have been fitted with `return_inla_fit = TRUE` and the **INLA** package to be installed. Default `FALSE`.

`lang` Character. Language for CLI messages: `"pt"` (default), `"en"`, or `"es"`.

`verbose` Logical. If `TRUE` (default), print progress messages via **cli**.

Value

A named list of class `c("climasus_spacetime_pred", "list")` with:

`$predictions` `data.frame` with columns `code_muni`, `time_idx`, `pred_mean`, and (if `include_ci = TRUE`) `pred_lower95`, `pred_upper95`.

`$n_predicted` Integer. Total number of predicted observations.

`$horizon` Integer. Horizon used (0 when `newdata` is supplied).

`$call` The matched call.

If `return_samples = TRUE` and INLA is available, an additional `$samples` slot contains raw INLA posterior samples (list from `INLA::inla.posterior.sample()`).

Algorithm

Because re-running INLA for new data is computationally expensive, this function uses an **approximate linear-predictor approach**:

1. Extract posterior mean fixed effects $\hat{\beta}$ and their standard deviations σ_β from `fit$fixed`.
2. For new covariate matrix X_{new}, compute the fixed-effect contribution: $\eta^{(f)} = X_{\text{new}}\hat{\beta}$.
3. Spatial random effect: reuse `fit$spatial_re` for municipalities present in the fitted model. For municipalities not in the training data, the spatial random effect is set to `NA` with a warning.
4. Temporal random effect: for in-sample time points, reuse `fit$temporal_re` directly. For future periods (`horizon > 0`), extrapolate using the last observed increment (RW1: repeat last increment; RW2: project second differences) with uncertainty propagation.
5. Full linear predictor: $\hat{\eta} = \eta^{(f)} + \phi_i + \gamma_t$.
6. Variance propagation:
$$\text{Var}(\hat{\eta}) \approx x^T V_\beta x + \sigma_\phi^2 + \sigma_\gamma^2$$
where $V_\beta = \text{diag}(\sigma_\beta^2)$ (diagonal approximation).
7. Point prediction: $\hat{\mu} = \exp(\hat{\eta})$ (Poisson / NB) or $\hat{\mu} = \hat{\eta}$ (Gaussian).
8. 95\ $\exp(\hat{\eta} \pm 1.96\sqrt{\text{Var}(\hat{\eta})})$.

Important: this is an approximation. For exact posterior predictive distributions, re-run `sus_mod_spacetime_bayes()` with the full dataset including future periods.

Covariate override (scenario analysis)

`covariates_new` accepts a named list of scalar or vector overrides, e.g.:

```
covariates_new = list(temp_mean = new_grid$temp_mean + 2)
```

Overrides are applied to the resolved covariate matrix **after** `newdata` columns are extracted, so they can reference vectors of length `nrow(newdata)`.

References

Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2), 319-392.

Blangiardo, M., & Cameletti, M. (2015). *Spatial and Spatio-temporal Bayesian Models with R-INLA*. Wiley.

Held, L., & Schrödle, B. (2010). Posterior and cross-validatory predictive checks: a comparison of MCMC and INLA. In *Statistical Modelling and Regression Structures* (pp. 91-110). Physica-Verlag HD.

See Also

`sus_mod_spatial_bayes()` for cross-sectional Bayesian disease mapping, `sus_mod_spatial_weights()` to build the spatial adjacency object, `sus_mod_spatial_moran()` for exploratory spatial autocorrelation analysis.

Examples

```
## Not run:
library(climasus4r)

# Assume `st_fit` is a climasus_spacetime_bayes object
# fitted on monthly data for 2015-2020

# 1. In-sample predictions for validation
pred_insample <- sus_mod_spacetime_predict(
  fit     = st_fit,
  newdata = training_data,
  lang    = "pt"
)
head(pred_insample$predictions)

# 2. Out-of-sample: 6 months ahead
pred_future <- sus_mod_spacetime_predict(
  fit     = st_fit,
  horizon = 6L,
  lang    = "en"
)
pred_future$predictions

# 3. Counterfactual: +2 degrees C scenario
```

```
pred_counter <- sus_mod_spacetime_predict(
  fit            = st_fit,
  newdata        = new_grid,
  covariates_new = list(temp_mean = new_grid$temp_mean + 2),
  lang           = "pt"
)

## End(Not run)
```

`sus_mod_spatial_bayes` *Bayesian CAR / BYM Disease Mapping with Climate Covariates*

Description

Fits a Bayesian hierarchical spatial model for disease mapping using Conditional Autoregressive (CAR) priors implemented in the **CARBayes** package, or the reparameterised BYM2 model via **INLA**. Four model specifications are available: the Besag-York-Mollie (BYM) model with structured and unstructured random effects, the Leroux model with a mixing parameter for spatial dependence, the independent random effects model (all via CARBayes/MCMC), and the BYM2 model (via INLA with penalised complexity priors). Climate and environmental covariates stored in a `climasus_df` can be passed directly as fixed effects.

Usage

```
sus_mod_spatial_bayes(
  df,
  outcome,
  W,
  covariates = NULL,
  offset = NULL,
  family = c("poisson", "binomial", "gaussian"),
  model = c("bym", "leroux", "independent", "bym2"),
  n_iter = 10000L,
  burnin = 2000L,
  thin = 10L,
  prior_tau2 = c(1, 0.01),
  seed = 42L,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` or `climasus_df` at the aggregate pipeline stage. Must contain `code_muni` (7-digit IBGE code), the `outcome` column, all columns

named in `covariates`, and optionally an `offset` column. Rows are sorted by `code_muni` internally to align with `W`.

`outcome` Character. Name of the response variable column in `df` (e.g. `"deaths"`, `"hospitalizations"`).

`W` A `climasus_weights` object produced by `sus_mod_spatial_weights()`. Must contain either a `$W` dense matrix or a `$nb` neighbour list from which a binary adjacency matrix is built.

`covariates` Character vector of covariate column names in `df` to include as fixed effects. Default `NULL` (intercept-only model).

`offset` Character. Name of an **expected counts** column in `df` (on the natural scale, not log-transformed). Used as `log(offset)` in the linear predictor for Poisson models. Default `NULL`.

`family` Character. Response distribution. One of `"poisson"` (default), `"binomial"`, or `"gaussian"`. Note: `"gaussian"` is not supported for `model = "bym2"`.

`model` Character. Spatial model structure. One of:

- `"bym"` (default) – Besag-York-Mollie model via CARBayes/MCMC.
- `"leroux"` – Leroux CAR model via CARBayes/MCMC.
- `"independent"` – Independent random effects via CARBayes/MCMC.
- `"bym2"` – Reparameterised BYM2 model via INLA with penalised complexity priors. Requires the **INLA** package (not on CRAN; see https://www.r-inla.org/download-install).

`n_iter` Positive integer. Total number of MCMC iterations (including burn-in). Used only for CARBayes models (`"bym"`, `"leroux"`, `"independent"`). Default `10000`. Ignored for `model = "bym2"`.

`burnin` Positive integer. Number of initial MCMC iterations to discard as burn-in. Must be less than `n_iter`. Used only for CARBayes models. Default `2000`. Ignored for `model = "bym2"`.

`thin` Positive integer. Thinning interval. Used only for CARBayes models. Default `10`. Ignored for `model = "bym2"`.

`prior_tau2` Numeric vector of length 2. Shape and rate parameters of the Inverse-Gamma prior for the variance of spatial random effects. Used only for CARBayes models. Default `c(1, 0.01)` (weakly informative). Ignored for `model = "bym2"` (which uses PC priors instead).

`seed` Integer. Random seed passed to `base::set.seed()` for reproducible MCMC (CARBayes models) or to `INLA::inla.set.control.compute` via `inla.seed` (BYM2). Default `42`.

`lang` Character. Language for CLI messages: `"pt"` (default), `"en"`, `"es"`.

`verbose` Logical. If `TRUE` (default), print progress messages via **cli**.

Value

A named list of class `c("climasus_spatial_bayes", "list")` with slots `$fixed` (posterior fixed effects), `$random` (spatial random effects per municipality), `$rr` (posterior relative risk with 95\ credible intervals), `$fitted` (posterior mean fitted values), `$dic` (Deviance Information Criterion; for BYM2 a named numeric with both DIC and WAIC), `$model`, `$family`, `$n_iter_effective`, and `$call`.

Model specifications

Smoothed relative risks are computed as:

$$RR_i = \frac{\hat{Y}_i}{E_i}$$

where $\hat{Y}_i$ are the posterior mean fitted counts and E_i are the expected counts (offset on the original scale). When no offset is supplied, E_i is set to the overall mean of the fitted values, which is equivalent to a standardised mortality/morbidity ratio. Credible intervals for the RR are derived from the 2.5th and 97.5th percentiles of the MCMC samples for fitted values divided by the expected (CARBayes models), or from the INLA marginal posterior quantiles (BYM2 model).

BYM2 model (INLA)

The BYM2 model of Riebler et al. (2016) reparameterises the original Besag-York-Mollie model into a single precision parameter τ and a mixing parameter $\phi \in [0, 1]$ that quantifies the proportion of variance attributable to structured spatial variation. The latent field is:

$$u_i = \frac{1}{\sqrt{\tau}} \left(\sqrt{\phi} v_i^* + \sqrt{1 - \phi} w_i \right)$$

where v_i^* is the scaled ICAR component and $w_i \sim N(0, 1)$. Penalised complexity (PC) priors are placed on ϕ and τ: $P(\phi > 0.5) = 0.5$ and $P(1/\sqrt{\tau} > 1) = 0.01$.

Prior specification (CARBayes models)

`prior_tau2` specifies the shape and rate parameters of an Inverse-Gamma prior for the variance component τ^2: $\tau^2 \sim$ Inv-Gamma(shape, rate). The default `c(1, 0.01)` is weakly informative. Fixed-effect coefficients receive flat (improper) priors.

References

Besag, J., York, J., & Mollie, A. (1991). Bayesian image restoration, with two applications in spatial statistics. *Annals of the Institute of Statistical Mathematics*, 43(1), 1-20.

Leroux, B. G., Lei, X., & Breslow, N. (2000). Estimation of disease rates in small areas: a new mixed model for spatial dependence. In *Statistical Models in Epidemiology, the Environment, and Clinical Trials* (pp. 179-191). Springer.

Lee, D. (2013). CARBayes: an R package for Bayesian spatial modelling with conditional autoregressive priors. *Journal of Statistical Software*, 55(13), 1-24.

Riebler, A., Sorbye, S. H., Simpson, D., & Rue, H. (2016). An intuitive Bayesian spatial model for disease mapping that accounts for scaling. *Statistical Methods in Medical Research*, 25(4), 1145-1165.

Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2), 319-392.

See Also

sus_mod_spatial_weights() to build the required adjacency object, sus_mod_spatial_moran() for exploratory spatial autocorrelation, sus_mod_spatial_scan() for cluster detection.

Examples

```
## Not run:
library(climasus4r)
library(geobr)

# Build spatial weights for Nordeste
shp <- geobr::read_municipality(code_muni = "all", year = 2022)
shp_ne <- shp[shp$abbrev_state %in% c("CE", "RN", "PB", "PE"), ]
W <- sus_mod_spatial_weights(shp_ne, return_matrix = TRUE)

# Prepare aggregated health-climate data (must have code_muni)
# agg <- sus_data_aggregate(cleaned_df) |> sus_climate_aggregate()

# BYM Poisson model with temperature as covariate (CARBayes/MCMC)
result_bym <- sus_mod_spatial_bayes(
  df         = agg,
  outcome    = "deaths",
  W          = W,
  covariates = c("temp_mean", "prec_total"),
  offset     = "expected_deaths",
  family     = "poisson",
  model      = "bym",
  n_iter     = 10000,
  burnin     = 2000,
  thin       = 10,
  seed       = 42,
  lang       = "pt"
)
print(result_bym)

# BYM2 Poisson model with PC priors (INLA) - requires INLA package
result_bym2 <- sus_mod_spatial_bayes(
  df         = agg,
  outcome    = "deaths",
  W          = W,
  covariates = c("temp_mean", "prec_total"),
  offset     = "expected_deaths",
  family     = "poisson",
  model      = "bym2",
  seed       = 42,
  lang       = "pt"
)
print(result_bym2)
result_bym2$rr
result_bym2$fixed

## End(Not run)
```

`sus_mod_spatial_moran` *Global Moran's I and LISA Local Autocorrelation for Health Outcomes*

Description

Computes Moran's I global spatial autocorrelation statistic and Local Indicators of Spatial Association (LISA) for a numeric health outcome measured at municipality level. The global statistic tests whether the outcome is spatially clustered across the study region; the local statistics identify significant **High-High** (hotspots), **Low-Low** (coldspots), **High-Low**, and **Low-High** spatial outlier clusters.

Usage

```
sus_mod_spatial_moran(
  df,
  outcome,
  W,
  municipalities = NULL,
  permutations = 999L,
  alpha = 0.05,
  adjust_p = c("fdr", "bonferroni", "none"),
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` or `climasus_df` containing at least the columns `code_muni` (7-digit IBGE municipality code) and the outcome column named by `outcome`. The row order must correspond to the spatial units encoded in `W` unless `municipalities` is supplied to subset and reorder.
`outcome`	Character. Name of the numeric outcome column in `df` (e.g. `"deaths"`, `"rate_100k"`, `"hospitalization_rate"`).
`W`	A `climasus_weights` object produced by `sus_mod_spatial_weights()`. Must contain a `listw` element (a `listw`-class object from **spdep**) whose length matches `nrow(df)` after any filtering by `municipalities`.
`municipalities`	Optional character or integer vector of `code_muni` values to include. When supplied the function subsets and reorders `df` to match this vector before computation. Default `NULL` (use all rows).
`permutations`	Positive integer. Number of Monte Carlo permutations for both global and local inference. Default `999`.

`alpha`	Numeric in (0, 1). Significance threshold applied to p-adjusted local Moran p-values when assigning quadrant labels. Default `0.05`.
`adjust_p`	Character. Method passed to `stats::p.adjust()` for multiple-testing correction of local Moran p-values. One of `"fdr"` (default), `"bonferroni"`, or `"none"`.
`lang`	Character. Output language for messages: `"pt"` (default), `"en"`, or `"es"`.
`verbose`	Logical. If `TRUE` (default), print progress messages via **cli**.

Value

An object of class `c("climasus_spatial_moran", "list")` with the following named elements:

`$global` One-row `data.frame` with columns `I`, `E.I`, `Var.I`, `Z.I`, `p_value` (analytical), and `p_simulated` (permutation-based).

`$local` `data.frame` with one row per spatial unit and columns `code_muni`, `Ii` (local Moran statistic), `Z.Ii` (z-score), `p_raw` (unit-level permutation p), `p_adj` (adjusted p), and `quadrant` (factor: `"HH"`, `"LL"`, `"HL"`, `"LH"`, `"NS"`).

`$n_HH` Integer. Number of significant HH (hotspot) units.

`$n_LL` Integer. Number of significant LL (coldspot) units.

`$n_HL` Integer. Number of significant HL spatial outlier units.

`$n_LH` Integer. Number of significant LH spatial outlier units.

`$outcome_name` Character. The `outcome` argument value.

`$call` The matched function call.

Global statistic

Moran's I is computed via permutation inference using `spdep::moran.mc()`. The observed statistic is compared against the empirical null distribution obtained by randomly permuting the outcome values across spatial units. The reported `p_simulated` is the proportion of permuted values as extreme as or more extreme than the observed I.

Local statistic (LISA)

Local Moran statistics are computed via `spdep::localmoran_perm()`, which uses conditional permutation to generate unit-specific null distributions. Each observation is assigned a LISA quadrant based on its standardised value and the standardised spatial lag:

`HH` High value surrounded by high neighbours (hotspot).

`LL` Low value surrounded by low neighbours (coldspot).

`HL` High value surrounded by low neighbours (spatial outlier).

`LH` Low value surrounded by high neighbours (spatial outlier).

`NS` Not significant at the specified `alpha` level.

References

Moran, P. A. P. (1950). Notes on continuous stochastic phenomena. *Biometrika*, 37(1-2), 17-23.

Anselin, L. (1995). Local indicators of spatial association—LISA. *Geographical Analysis*, 27(2), 93-115.

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied Spatial Data Analysis with R* (2nd ed.). Springer.

See Also

`sus_mod_spatial_weights()`, `sus_spatial_join()`

Examples

```
## Not run:
# Requires spdep and a climasus_weights object W
library(climasus4r)

# W <- sus_mod_spatial_weights(shp, style = "W")
result <- sus_mod_spatial_moran(
  df           = my_df,
  outcome      = "deaths",
  W            = W,
  permutations = 999,
  alpha        = 0.05,
  adjust_p     = "fdr",
  lang         = "pt"
)
print(result)

# Access local clusters
result$local[result$local$quadrant == "HH", ]

## End(Not run)
```

---

sus_mod_spatial_reg *Spatial Regression (SAR / SEM / SDM / SAC) for Climate-Health Associations*

Description

Fits maximum-likelihood spatial regression models that account for spatial spillovers between geographic units (e.g. municipalities) in climate-health studies. Four model families are supported: the Spatial Autoregressive Model (SAR / spatial lag), the Spatial Error Model (SEM), the Spatial Durbin Model (SDM), and the Spatial Autocorrelation Model (SAC).

Usage

```
sus_mod_spatial_reg(
  df,
  formula,
  W,
  model = c("lag", "error", "durbin", "sac"),
  method = c("eigen", "LU", "Chebyshev", "MC"),
  zero_policy = TRUE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` or `climasus_df` containing all variables in `formula`. If an `sf` object, geometry is dropped automatically.
`formula`	A `formula` specifying the outcome and predictors, e.g. `deaths ~ mean_temp + precip + pib_pc`.
`W`	A `climasus_weights` object produced by `sus_mod_spatial_weights()`. Must contain a `listw` element whose length matches `nrow(df)`.
`model`	Character. Spatial model family. One of `"lag"` (SAR, default), `"error"` (SEM), `"durbin"` (SDM), or `"sac"` (SAC).
`method`	Character. Estimation method passed to the underlying **spatialreg** function. One of `"eigen"` (default), `"LU"`, `"Chebyshev"`, or `"MC"`. `"eigen"` is exact; the others are approximations suitable for large datasets.
`zero_policy`	Logical. If `TRUE` (default), spatial units with no neighbours (islands) are allowed and receive a zero spatial lag/error. Passed to all **spatialreg** and **spdep** calls.
`lang`	Character. Output language for messages: `"pt"` (default), `"en"`, or `"es"`.
`verbose`	Logical. If `TRUE` (default), print progress messages via **cli**.

Value

An object of class `c("climasus_spatial_reg", "list")` with the following named elements:

`$model` Character. The model argument value.

`$model_label` Character. Human-readable model name.

`$coefficients` `data.frame` with columns `term`, `estimate`, `std_error`, `z_value`, `p_value` for all covariates (excluding the spatial parameters `rho`/`lambda`).

`$rho` Numeric spatial lag parameter (`NULL` for the `error` model).

`$lambda` Numeric spatial error parameter (`NULL` for the `lag` and `durbin` models).

`$impacts` `data.frame` with columns `term`, `direct`, `indirect`, `total` (from `spatialreg::impacts()`), or `NULL` for the `error` model or when impact computation fails.

`$aic` Numeric. AIC of the spatial model.

`$lm_aic` Numeric. AIC of the OLS baseline (same formula, same data).

`$moran_residuals` Named list with elements `I` (Moran statistic), `p_value` (analytical p-value), and `z` (z-score).

`$fitted` Numeric vector of fitted values.

`$residuals` Numeric vector of model residuals.

`$call` The matched function call.

Model families

`"lag"` **(SAR)** The outcome in unit i depends on a weighted average of outcomes in neighbouring units (spatial lag). Estimated via `spatialreg::lagsarlm()`. Contains a spatial lag parameter `rho`.

`"error"` **(SEM)** Spatial dependence enters through a spatially autocorrelated error term. Estimated via `spatialreg::errorsarlm()`. Contains a spatial error parameter `lambda`.

`"durbin"` **(SDM)** Extends the SAR model by including spatially lagged covariates (Durbin terms). Estimated via `spatialreg::lagsarlm()` with `Durbin = TRUE`. Contains both `rho` and lagged-covariate coefficients.

`"sac"` **(SAC)** Contains both a spatial lag of the outcome (`rho`) and a spatial error term (`lambda`). Estimated via `spatialreg::sacsarlm()`.

Impacts

For models with a spatial lag (`lag`, `durbin`, `sac`), the total effect of a covariate is decomposed into *direct* (own-unit), *indirect* (spillover), and *total* effects using `spatialreg::impacts()` with `R` simulations for inference. For the `error` model impacts equal OLS slopes and no decomposition is meaningful; the `$impacts` element will be `NULL`.

Residual diagnostics

After fitting, `spdep::moran.test()` is applied to the model residuals to verify that spatial autocorrelation has been adequately absorbed. A non-significant Moran p-value indicates a well-specified spatial model.

References

Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Kluwer Academic Publishers.

LeSage, J., & Pace, R. K. (2009). *Introduction to Spatial Econometrics*. CRC Press / Taylor & Francis.

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied Spatial Data Analysis with R* (2nd ed.). Springer.

Elhorst, J. P. (2014). *Spatial Econometrics: From Cross-Sectional Data to Panels*. Springer.

See Also

`sus_mod_spatial_weights()`, `sus_mod_spatial_moran()`, `sus_mod_spatial_scan()`

Examples

```
## Not run:
library(climasus4r)

# W <- sus_mod_spatial_weights(shp, style = "W")
result <- sus_mod_spatial_reg(
  df      = my_df,
  formula = deaths ~ mean_temp + precip + pib_pc,
  W       = W,
  model   = "lag",
  lang    = "pt"
)
print(result)
summary(result)

# SDM with spillovers
result_sdm <- sus_mod_spatial_reg(
  df      = my_df,
  formula = deaths ~ mean_temp + precip,
  W       = W,
  model   = "durbin"
)
result_sdm$impacts

## End(Not run)
```

---

sus_mod_spatial_scan *Kulldorff Circular Scan Statistic for Spatial Cluster Detection*

Description

Applies the Kulldorff circular spatial scan statistic to detect geographic clusters of disease excess. The most-likely cluster and any significant secondary clusters are identified by Monte Carlo hypothesis testing. The analysis requires a tabular dataset with case counts and population denominators at the municipality level, plus an `sf` polygon layer providing municipality boundaries.

Usage

```
sus_mod_spatial_scan(
  df,
  cases,
  population,
  municipalities,
  expected = NULL,
  max_pop_frac = 0.5,
  n_simulations = 999L,
  alpha = 0.05,
```

```
  lang = c("pt", "en", "es"),
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` (or `climasus_df`) containing at minimum: a `code_muni` column (character or integer) and the columns named by `cases`, `population`, and optionally `expected`.
`cases`	Character. Name of the column in `df` with case counts (non-negative integer or numeric).
`population`	Character. Name of the column in `df` with the at-risk population denominator.
`municipalities`	An `sf` object with POLYGON or MULTIPOLYGON geometry and a `code_muni` column used to join to `df`.
`expected`	Character or `NULL`. Name of the column in `df` with the expected number of cases under the null model. When `NULL` (default), `SpatialEpi::kulldorff()` computes expected counts internally from the overall rate.
`max_pop_frac`	Numeric in (0, 1]. Maximum fraction of the total population that a single cluster window may contain. Default `0.5`.
`n_simulations`	Positive integer. Number of Monte Carlo simulations for hypothesis testing. Default `999`. Increase to `9999` for publication-quality p-values.
`alpha`	Numeric in (0, 1). Significance level for secondary cluster filtering. Default `0.05`.
`lang`	Character. Output language: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_spatial_scan` object (named list) with:

`$most_likely_cluster` Named list: `location_ids` (character vector of `code_muni` in the cluster), `observed` (numeric), `expected` (numeric), `RR` (relative risk), `log_lik` (log-likelihood ratio), `p_value` (Monte Carlo p-value).

`$secondary_clusters` List of lists with the same structure as `most_likely_cluster`, one per significant secondary cluster. Empty list when no secondary cluster is significant.

`$n_clusters` Integer. Total number of significant clusters (1 + length of `secondary_clusters`), or 0 if the most-likely cluster itself is not significant.

`$data` Tibble with one row per municipality: `code_muni`, `cases`, `population`, `expected`, and a logical column `in_mlc` indicating membership in the most-likely cluster.

`$meta` Named list of analysis parameters.

`$call` The matched call.

Methodology

The scan statistic imposes circular windows of variable radius over the map. For each window the likelihood ratio (LR) under a Poisson model (observed vs. expected cases inside vs. outside the window) is computed. The window with the maximum LR is the most-likely cluster. Its p-value is obtained by comparing to the empirical distribution of max-LR under `n_simulations` Monte Carlo permutations. Secondary clusters are non-overlapping windows with LR > the `alpha`-level critical value from the same Monte Carlo distribution.

Computations are delegated to `SpatialEpi::kulldorff()`. Municipality centroids are derived from the polygon geometry using `sf::st_centroid()` after reprojecting to WGS 84 (EPSG 4326) so that longitude/latitude coordinates are passed to `kulldorff()`.

References

Kulldorff, M. & Nagarwalla, N. (1995). Spatial disease clusters: detection and inference. *Statistics in Medicine*, 14(8), 799–810. doi:10.1002/sim.4780140809

Kulldorff, M. (1997). A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6), 1481–1496. doi:10.1080/03610929708831995

Kim, A. Y. & Wakefield, J. (2010). R data and methods for spatial epidemiology: the SpatialEpi package. University of Washington.

See Also

`sus_spatial_join()`, `sus_mod_dlnm()`, `sus_mod_af()`

Examples

```
## Not run:
library(geobr)
library(dplyr)

# Download municipality polygons for Minas Gerais (state code 31)
muni_sf <- geobr::read_municipality(code_muni = 31, year = 2020)
muni_sf <- dplyr::rename(muni_sf, code_muni = code_muni)

# Build a synthetic data.frame
set.seed(42)
df_cases <- tibble::tibble(
  code_muni  = as.character(muni_sf$code_muni),
  cases      = rpois(nrow(muni_sf), lambda = 5),
  population = sample(5000:200000, nrow(muni_sf), replace = TRUE)
)

result <- sus_mod_spatial_scan(
  df            = df_cases,
  cases         = "cases",
  population    = "population",
  municipalities = muni_sf,
  n_simulations = 199,
  lang          = "pt"
```

```
)

print(result)
result$most_likely_cluster
result$secondary_clusters

## End(Not run)
```

`sus_mod_spatial_weights` *Build Spatial Weights from Municipality Polygons*

Description

Constructs a `spdep` spatial weights object from an `sf` data frame of municipality (or any polygon) boundaries. The result — a `climasus_weights` object — is the required input for all other `sus_mod_spatial_*` functions in the climasus4r pipeline.

Usage

```
sus_mod_spatial_weights(
  municipalities,
  style = "W",
  queen = TRUE,
  snap = NULL,
  zero_policy = TRUE,
  return_matrix = FALSE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`municipalities`
: An `sf` object with **polygon** or **multipolygon** geometry. Typically the result of `geobr::read_municipality()` or any shape file read with `sf::st_read()`. A column named `code_muni` is used as row labels when present; otherwise integer indices are used.

`style`
: Character. Weight normalisation style passed to `spdep::nb2listw()`. One of `"W"` (default), `"B"`, `"C"`, `"U"`, `"S"`, `"minmax"`. See **Weight styles** section.

`queen`
: Logical. If `TRUE` (default) Queen contiguity is used (shared edge or vertex). If `FALSE`, Rook contiguity (shared edge only).

`snap`
: Numeric or `NULL`. Snap distance tolerance passed to `spdep::poly2nb()`. `NULL` (default) uses `1e-3`. See **Snap distance** section.

`zero_policy`	Logical. If `TRUE` (default) municipalities with no neighbours ("islands") are allowed — their spatial lag is set to `NA`. If `FALSE` an error is raised when islands are found.
`return_matrix`	Logical. If `TRUE`, a dense $n \times n$ spatial weights matrix `W` is included in the output (slot `$W`). This can be memory-intensive for large datasets. Default `FALSE`.
`lang`	Character. Language for CLI messages: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Details

Contiguity neighbours are detected with `spdep::poly2nb()`, which supports both Queen (default, shared edge **or** vertex) and Rook (shared edge only) contiguity. Municipalities that share no boundary ("spatial islands") are automatically detected and reported. The resulting neighbour list is normalised to a `listw` object via `spdep::nb2listw()`.

Value

A `climasus_weights` object (named list with class `c("climasus_weights", "list")`) containing:

`$listw` `spdep` `listw` object ready for spatial modelling.

`$nb` `spdep` `nb` neighbour list (before weight normalisation).

`$n_regions` Integer. Total number of regions / municipalities.

`$n_islands` Integer. Number of regions with zero neighbours.

`$island_ids` Character or integer vector of island identifiers (empty if none).

`$W` Dense numeric $n \times n$ weight matrix, or `NULL` when `return_matrix = FALSE`.

`$style` Character. Weight normalisation style used.

`$call` The matched `call()`.

Snap distance

Municipal boundaries digitised at different scales often have small gaps or overlaps at shared borders. The `snap` argument passes a distance tolerance (in the units of the CRS) to `spdep::poly2nb()`: any two polygon boundaries within `snap` distance are treated as touching. The default (`NULL`) uses `1e-3` (appropriate for degree-based CRS such as EPSG:4674 / SIRGAS2000). For projected CRS (metres) consider values around `1` to `100`.

Weight styles

- `"W"` (default) — row-standardised: each neighbour weight = 1/n_neighbours
- `"B"` — binary: 1 if neighbour, 0 otherwise
- `"C"` — globally standardised
- `"U"` — equal to `"C"` divided by number of neighbours
- `"S"` — variance-stabilising (Tiefelsdorf et al. 1999)
- `"minmax"` — min/max normalisation

References

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied Spatial Data Analysis with R* (2nd ed.). Springer.

Cliff, A. D., & Ord, J. K. (1981). *Spatial Processes: Models and Applications*. Pion.

Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Kluwer Academic.

See Also

`sus_mod_spatial_reg()` for spatial lag model (SAR/SLM), `sus_mod_spatial_reg()` for spatial error model (SEM), `sus_mod_spatial_bayes()` for Bayesian hierarchical spatial models, `sus_mod_spatial_scan()` for spatial scan statistics.

Examples

```
## Not run:
library(geobr)
library(climasus4r)

# Download Nordeste municipalities (SIRGAS2000 / EPSG:4674)
muni <- geobr::read_municipality(code_muni = "all", year = 2022)
muni_ne <- muni[muni$abbrev_state %in% c("CE","RN","PB","PE","AL",
                                         "SE","BA","PI","MA"), ]

# Row-standardised Queen contiguity (default)
w <- sus_mod_spatial_weights(muni_ne)
print(w)

# Binary Rook weights, returning the dense W matrix
w_rook <- sus_mod_spatial_weights(
  municipalities = muni_ne,
  style          = "B",
  queen          = FALSE,
  return_matrix  = TRUE
)
dim(w_rook$W)

## End(Not run)
```

`sus_mod_swot` *SWOT Analysis for Climate-Health Surveillance*

Description

Synthesizes pre-computed climasus model objects into a Strengths, Weaknesses, Opportunities, and Threats (SWOT) framework for climate-health risk communication and territorial planning. Each quadrant aggregates normalized indicators (0–100 scale) extracted from the supplied model objects; quadrant scores are returned both as continuous values and as user-defined categorical labels.

Usage

```
sus_mod_swot(
  vulnerability = NULL,
  af = NULL,
  burden = NULL,
  dlnm = NULL,
  sensitivity = NULL,
  score_type = c("both", "numeric", "categorical"),
  breaks = c(33, 66),
  labels = NULL,
  city_col = "city",
  lang = c("pt", "en", "es"),
  verbose = TRUE
)
```

Arguments

`vulnerability`	A `climasus_vi` object from `sus_mod_vulnerability_index()`, or `NULL`. Provides per-city VI scores, exposure, sensitivity, and adaptive capacity indicators.
`af`	A `climasus_af` object from `sus_mod_af()`, or `NULL`. Provides total, heat, and cold attributable fraction percentages.
`burden`	A `climasus_burden` object from `sus_mod_burden()`, or `NULL`. Provides city-level ranked attributable numbers and fractions.
`dlnm`	A `climasus_dlnm` object from `sus_mod_dlnm()`, or `NULL`. Provides the heat exposure-response RR at the 95th percentile of temperature.
`sensitivity`	A `climasus_sensitivity` object from `sus_mod_sensitivity()`, or `NULL`. Provides stratum-level RR inequality and the maximum hot RR across strata.
`score_type`	Character. Which score types to compute: `"numeric"` (0–100), `"categorical"` (cut-point labels), or `"both"` (default).
`breaks`	Numeric vector. Strictly increasing cut-points in (0, 100) used to convert numeric scores into categories. Default `c(33, 66)` produces three categories. Generates `length(breaks) + 1` categories.
`labels`	Character vector or `NULL`. Category labels with length `length(breaks) + 1`. `NULL` (default) uses language-appropriate defaults: Portuguese `c("Baixo", "Médio", "Alto")`, English `c("Low", "Medium", "High")`, Spanish `c("Bajo", "Medio", "Alto")`.
`city_col`	Character. Column name of the city/entity identifier in the `vulnerability$vi_table`. Defaults to `"city"`. Overridden by `vulnerability$meta$city_col` when available.
`lang`	Character. Language for labels: `"pt"` (default), `"en"`, `"es"`.
`verbose`	Logical. Print progress messages. Default `TRUE`.

Value

A `climasus_swot` list with:

`$scores` Tibble. One row per entity with: `entity`, `S_score`, `W_score`, `O_score`, `T_score` (numeric 0–100, `NA` when no indicators are available for a quadrant), `n_S`, `n_W`, `n_O`, `n_T` (indicator counts). When `score_type` ∈ `c("categorical","both")`: `S_cat`, `W_cat`, `O_cat`, `T_cat`.

`$indicators` Tibble (long). One row per (entity × quadrant × indicator): `entity`, `quadrant`, `ind_code`, `indicator`, `raw_value`, `norm_score`, `direction` (`"positive"` or `"negative"`).

`$meta` List: `n_entities`, `n_indicators`, `inputs_used`, `score_type`, `breaks`, `labels`, `lang`, `call_time`.

SWOT Quadrant Indicators

Quadrant	Meaning	Indicators from each input
S Strengths	Current protective factors	`adaptive_capacity_score` (VI), low total AF% (af/burden), low b
W Weaknesses	Current vulnerabilities	`sensitivity_score` (VI), heat/cold AF% (af), stratum RR inequa
O Opportunities	Intervention windows	Low VI percentile (VI), low current exposure (VI)
T Threats	Climate-health risks	`exposure_score` + `vi_score` (VI), heat RR at P95 (dlnm), attribu

All indicators are normalized to 0–100 within the supplied data. For **Strength** and **Opportunity** indicators the raw value is already oriented so that higher = better; for **Weakness** and **Threat** indicators, higher means worse. The quadrant score is the mean of available indicator scores.

When `vulnerability` and/or `burden` provide multi-city data, the SWOT is computed per city. Single-city inputs (`af`, `dlnm`, `sensitivity`) are broadcast to all detected entities and treated as shared context.

See Also

`sus_mod_plot_swot()`, `sus_mod_vulnerability_index()`, `sus_mod_af()`, `sus_mod_burden()`, `sus_mod_dlnm()`, `sus_mod_sensitivity()`

Examples

```
## Not run:
swot <- sus_mod_swot(
  vulnerability = vi_result,
  af            = af_result,
  burden        = burden_result,
  dlnm          = dlnm_result,
  score_type    = "both",
  lang          = "pt"
)

swot$scores
sus_mod_plot_swot(swot, type = "matrix", lang = "pt")
```

```
sus_mod_plot_swot(swot, type = "radar",  lang = "en")

## End(Not run)
```

`sus_mod_vulnerability_index` *IPCC 3-Pillar Composite Vulnerability Index for Climate-Health Analysis*

Description

Builds a city-level composite vulnerability index following the IPCC AR6 vulnerability framework: Vulnerability = f(**Exposure** + **Sensitivity** - **Adaptive Capacity**). Each pillar is constructed from a user-supplied data frame of city-level numeric indicators. Indicators are normalized (min-max or z-score), weighted within pillars, and combined into a final score [0, 1] where **1 = most vulnerable**.

Usage

```
sus_mod_vulnerability_index(
  exposure_df = NULL,
  sensitivity_df = NULL,
  adaptive_capacity_df = NULL,
  fits = NULL,
  city_col = "city",
  normalize = "minmax",
  weights = NULL,
  pillar_weights = c(exposure = 1, sensitivity = 1, adaptive_capacity = 1),
  hot_percentile = 0.99,
  cold_percentile = 0.01,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`exposure_df` A data frame with one row per city and numeric columns representing **exposure** indicators (higher value = more hazard). Typically derived by summarising a `climasus_df` produced by `sus_climate_aggregate()`. Must contain the column named `city_col`.

`sensitivity_df` A data frame with numeric **sensitivity** indicators (higher value = more sensitive population). Can include demographic columns (% elderly, % children), health-outcome rates, or any other stratum-level vulnerability marker. If `fits` is also provided, the DLNM-derived metrics are merged into this data frame.

`adaptive_capacity_df`
: A data frame with numeric **adaptive capacity** indicators (higher value = *more* adaptive capacity = *less* vulnerable). This pillar is inverted internally before computing VI. Typical columns come from `sus_census_join()` (e.g., HDI, literacy rate, piped-water coverage) or `sus_spatial_join()` (e.g., hospital density per capita).

`fits`
: Optional named list of `climasus_dlnm` objects (one per city, names = city identifiers). When provided, the function extracts `dlnm_hot_rr` (cumulative RR at `hot_percentile`), `dlnm_cold_rr` (cumulative RR at `cold_percentile`), and `dlnm_sensitivity_index` for each city and merges them into the sensitivity pillar.

`city_col`
: Character. Name of the city identifier column present in every supplied data frame. Default `"city"`.

`normalize`
: Character. Normalization method: `"minmax"` (default, maps each indicator to [0, 1]) or `"zscore"` (standardizes by mean and SD — useful for heterogeneous indicator scales but requires back- transformation for interpretation).

`weights`
: Named numeric vector. Within-pillar indicator weights. Names must match indicator column names in the respective data frame. Indicators not listed receive weight 1. Weights are normalized to sum to 1 within each pillar. Example: `c(pct_elderly = 2, pct_children = 1)` doubles the influence of elderly share in the sensitivity pillar.

`pillar_weights`
: Named numeric vector with elements `exposure`, `sensitivity`, and `adaptive_capacity`. Controls the relative contribution of each pillar to the final VI. Default `c(1, 1, 1)` (equal weights). Absent pillars automatically receive weight 0.

`hot_percentile`
: Numeric in (0, 1). Quantile used as the "hot" threshold for DLNM extraction. Default `0.99`. Only used when `fits` is provided.

`cold_percentile`
: Numeric in (0, 1). Quantile used as the "cold" threshold for DLNM extraction. Default `0.01`. Only used when `fits` is provided.

`lang`
: Character. Language for messages: `"pt"` (default), `"en"`, `"es"`.

`verbose`
: Logical. Print progress messages. Default `TRUE`.

Details

Exposure quantifies how much climate hazard a city experiences (e.g., mean maximum temperature, number of heat-wave days, threshold exceedances from `sus_climate_aggregate()`). **Sensitivity** captures how responsive local health outcomes are to the hazard (e.g., percentage elderly, chronic-disease rates, DLNM-derived cumulative RRs from `sus_mod_dlnm()`). **Adaptive Capacity** reflects ability to cope (e.g., HDI, piped-water coverage, healthcare density from `sus_census_join()` or `sus_spatial_join()`). Providing `fits` — a named list of `climasus_dlnm` objects — automatically extracts heat and cold RRs plus a sensitivity index and merges them into the sensitivity pillar.

Value

A `climasus_vi` object (named list) with:

`$vi_table` Tibble sorted by `vi_rank` (1 = most vulnerable). Columns: city identifier, `vi_score` [0, 1], `exposure_score`, `sensitivity_score`, `adaptive_capacity_score` (raw, before inversion), `vi_rank` (integer), `vi_percentile` (0–100, higher = more vulnerable relative to peers).

`$pillar_scores` Long-format tibble with columns city identifier, `pillar` (`"exposure"`, `"sensitivity"`, `"adaptive_capacity"`), and `score` (raw pillar score before AC inversion).

`$normalized_indicators` Wide tibble: city identifier + all normalized indicator values, prefixed by pillar (`exp_*`, `sen_*`, `ac_*`).

`$raw_indicators` Wide tibble: city identifier + all original (un-normalized) indicator values, prefixed by pillar.

`$weights` Named list with within-pillar normalized weights for `exposure`, `sensitivity`, and `adaptive_capacity`, plus the normalized `pillar` weights vector.

`$gini_coefficient` Gini coefficient of the VI distribution across cities (0 = perfect equality, 1 = all burden on one city).

`$meta` Named list of all parameters used in this call.

Formula

Each indicator column x_{ij} is normalized to $\tilde{x}_{ij}$ [0, 1] (min-max) or $\in \mathbb{R}$ (z-score). The pillar score for city i is the within-pillar weighted mean of its normalized indicators. The adaptive-capacity pillar is inverted before combining (higher capacity lower vulnerability):

$$VI_i = \frac{w_E \cdot E_i + w_S \cdot S_i + w_{AC} \cdot (1 - AC_i)}{w_E + w_S + w_{AC}}$$

Missing pillar scores for a city are omitted and remaining weights re-normalized, so partial data does not force full exclusion.

References

IPCC (2022). *Sixth Assessment Report — Working Group II*. Chapter 16: Key Risks across Sectors and Regions. Cambridge University Press.

Turner, B.L., et al. (2003). A framework for vulnerability analysis in sustainability science. *PNAS*, 100(14), 8074–8079. doi:10.1073/pnas.1231335100

Cutter, S.L., et al. (2003). Social vulnerability to environmental hazards. *Social Science & Medicine*, 56(11), 2301–2322. doi:10.1016/S02779536(02)00605X

See Also

`sus_mod_sensitivity()`, `sus_mod_dlnm()`, `sus_mod_burden()`, `sus_census_join()`, `sus_climate_aggregate()`

Examples

```
## Not run:
#   Minimal example with two pillars
exposure_df <- data.frame(
  city = c("fortaleza", "recife", "salvador"),
  mean_tmax = c(32.1, 30.5, 31.2),
  heat_wave_days = c(45, 30, 38)
)
adaptive_capacity_df <- data.frame(
  city = c("fortaleza", "recife", "salvador"),
  hdi  = c(0.754, 0.772, 0.759),
  pct_piped_water = c(72.1, 68.4, 74.3)
)
vi <- sus_mod_vulnerability_index(
  exposure_df          = exposure_df,
  adaptive_capacity_df = adaptive_capacity_df,
  lang = "pt"
)
print(vi)
tidy(vi)

#   Full 3-pillar with DLNM auto-extraction
dlnm_fits <- list(
  fortaleza = sus_mod_dlnm(df_fortaleza, ...),
  recife    = sus_mod_dlnm(df_recife, ...)
)
vi_full <- sus_mod_vulnerability_index(
  exposure_df          = exposure_df,
  fits                 = dlnm_fits,          # auto-extracted to sensitivity
  adaptive_capacity_df = adaptive_capacity_df,
  weights              = c(mean_tmax = 2),   # double-weight temperature
  pillar_weights       = c(exposure = 1.5, sensitivity = 1, adaptive_capacity = 1),
  lang = "en"
)

## End(Not run)
```

`sus_rap_export` *Export a climasus4r Pipeline as a Reproducible Analytical Pipeline (RAP)*

Description

Converts a `climasus4r` pipeline expression into a portable, self-contained artefact - an R script (`.R`), RMarkdown document (`.Rmd`), Quarto document (`.qmd`), or an encapsulated R function - suitable for sharing, archiving, and re-execution. The exported file embeds session metadata (R version, package versions, timestamp, platform), an editable parameters block, and optional validation checks, following Reproducible Analytical Pipeline principles.

Usage

```
sus_rap_export(
  pipeline = NULL,
  file_path = NULL,
  format = c("script", "rmarkdown", "quarto", "function"),
  include_metadata = TRUE,
  include_validation = TRUE,
  include_documentation = c("standard", "minimal", "comprehensive"),
  output_type = c("analysis", "dashboard", "report"),
  lang = "pt",
  overwrite = FALSE
)
```

Arguments

`pipeline`	A pipeline expression (created with `quote()` or `rlang::quo()`) containing `sus_*` function calls connected by `|>` or `%>%`. Pass `NULL` to capture from the calling context.
`file_path`	`character(1)` - Output file path. Extension informs format detection. Pass `NULL` to return content without writing to disk.
`format`	`character(1)` - One of `"script"`, `"rmarkdown"`, `"quarto"`, or `"function"`. Inferred from `file_path` extension when possible.
`include_metadata`	`logical(1)` - Embed R version, package versions, timestamp, and platform. Default `TRUE`.
`include_validation`	`logical(1)` - Add a validation block (`stopifnot` • NA-rate check). Default `TRUE`.
`include_documentation`	`character(1)` - Comment density in the exported code: `"minimal"`, `"standard"` (default), or `"comprehensive"`.
`output_type`	`character(1)` - Purpose of the document: `"analysis"`, `"dashboard"`, or `"report"`. Affects visualisation sections in Rmd/Quarto outputs.
`lang`	`character(1)` - Language for comments and messages: `"pt"` (default), `"en"`, or `"es"`.
`overwrite`	`logical(1)` - Overwrite an existing file. Default `FALSE`.

Value

Invisibly, a `character` vector with the exported file content. Side-effect: writes to `file_path` when provided.

Formats

`"script"` Standalone `.R` script with header, params block, pipeline execution, validation, and `sessionInfo()`.

`"rmarkdown"` `.Rmd` document with YAML params, setup chunk, pipeline, validation, and optional visualisation section.

`"quarto"` `.qmd` document using Quarto YAML front matter.

`"function"` Parametrised `.R` function with Roxygen2 header.

Examples

```
## Not run:
pipeline <- quote(
  sus_data_import(uf = "SP", years = 2020:2022, system = "SIM-DO") |>
    sus_data_clean_encoding() |>
    sus_data_filter_cid(cid_group = "respiratory") |>
    sus_data_aggregate(by = "month")
)

# Export as R script
sus_rap_export(pipeline, "pipeline_respiratorio.R", lang = "pt")

# Export as Quarto document
sus_rap_export(pipeline, "analise.qmd", format = "quarto",
               include_documentation = "comprehensive")

## End(Not run)
```

`sus_rap_from_recipe` *Import and Optionally Run a RAP Recipe*

Description

Reads a YAML recipe exported by `sus_rap_recipe()` and reconstructs a `rap_object`. Parameter overrides can be supplied at import time. Passing `execute = TRUE` immediately re-runs the pipeline using `sus_rap_run()`.

Usage

```
sus_rap_from_recipe(
  recipe_path,
  override_params = list(),
  execute = FALSE,
  lang = "pt",
  ...
)
```

Arguments

`recipe_path` `character(1)` - Path to the `.yaml` recipe file.

`override_params`
Named list of parameters to override from the recipe (e.g. `list(uf = "RJ", years = 2022:2024)`).

`execute`	`logical(1)` - Run the pipeline immediately after loading. Default `FALSE`.
`lang`	`character(1)` - Language for messages.
`...`	Additional arguments forwarded to `sus_rap_run()` when `execute = TRUE`.

Value

A `rap_object` (or the pipeline result when `execute = TRUE`).

Examples

```
## Not run:
rap <- sus_rap_from_recipe("pipeline_sp.yaml")
print(rap)

# Import with parameter overrides
rap_rj <- sus_rap_from_recipe("pipeline_sp.yaml",
                              override_params = list(uf = "RJ"))

## End(Not run)
```

`sus_rap_gui` *Launch an Interactive GUI for Editing and Running RAPs*

Description

Opens a Shiny application that lets users visually configure pipeline parameters (UF, years, system, aggregation), preview the generated code, and execute or export the RAP - all without writing R code directly.

Usage

```
sus_rap_gui(rap = NULL, launch.browser = TRUE, ...)
```

Arguments

`rap`	A `rap_object` (from `sus_rap_read()`) to pre-populate the interface. Pass `NULL` (default) to start from a blank template.
`launch.browser`	`logical(1)` - Open in the system browser. Default `TRUE`.
`...`	Additional arguments forwarded to `shiny::runApp()`.

Value

Does not return; runs the Shiny application interactively.

Interface panels

Left Parameter editors: UF checkboxes, year range slider, system dropdown, aggregation unit selector, language selector.

Center Step-by-step pipeline card view showing each function and its arguments.

Right Live log console. Buttons: **Preview**, **Run**, **Export**, **Save Recipe**.

Examples

```
## Not run:
sus_rap_gui()

rap <- sus_rap_read("pipeline_sp.R")
sus_rap_gui(rap)

## End(Not run)
```

---

sus_rap_inspect *Inspect or Diff rap_object(s)*

Description

Prints a structured summary of a `rap_object`. When two objects are supplied, performs a side-by-side diff showing changed parameters, added or removed pipeline steps, and package version drift.

Usage

```
sus_rap_inspect(rap, rap2 = NULL, verbose = TRUE, lang = "pt")
```

Arguments

`rap`	A `rap_object` returned by `sus_rap_read()`.
`rap2`	A second `rap_object` for comparison, or `NULL` (default) for single-object summary only.
`verbose`	`logical(1)` - Show detailed step list. Default `TRUE`.
`lang`	`character(1)` - Language for messages.

Value

Invisibly, a list with `$summary` (and `$diff` when `rap2` is given).

Examples

```
## Not run:
rap <- sus_rap_read("pipeline_sp.R")
sus_rap_inspect(rap)

rap_v2 <- sus_rap_read("pipeline_sp_v2.R")
sus_rap_inspect(rap, rap_v2)

## End(Not run)
```

---

sus_rap_make *Execute a targets Pipeline from a RAP*

Description

Runs a `_targets.R` pipeline (generated by `sus_rap_targets()`) using `targets::tar_make()`, with optional parallelism via `workers`.

Usage

```
sus_rap_make(
  rap,
  workers = 1L,
  reporter = "verbose",
  branching = "none",
  lang = "pt",
  ...
)
```

Arguments

`rap`	A `rap_object`, a `tar_rap_object`, or a `character(1)` path to a `_targets.R` file. When a `rap_object` is passed, a temporary `_targets.R` is generated automatically.
`workers`	`integer(1)` - Parallel workers. Default `1`.
`reporter`	`character(1)` - Progress reporter for `tar_make()`. Default `"verbose"`.
`branching`	`character(1)` - Branching strategy when `rap` is a `rap_object` (ignored when a file path is passed). Default `"none"`.
`lang`	`character(1)` - Language for messages.
`...`	Additional arguments forwarded to `targets::tar_make()`.

Value

A list with `$results`, `$rap`, and `$store`.

Examples

```
## Not run:
rap <- sus_rap_read("pipeline_sp.R")
result <- sus_rap_make(rap, workers = 2, branching = "uf")
head(result$results)

## End(Not run)
```

`sus_rap_read` *Read an Exported RAP File*

Description

Reads a file produced by `sus_rap_export()` and reconstructs a live `rap_object` containing session metadata, configurable parameters, and the pipeline step list. The result can be inspected with `sus_rap_inspect()`, re-executed with `sus_rap_run()`, or patched with `sus_rap_update()`.

Usage

```
sus_rap_read(file_path, lang = "pt", validate = TRUE)
```

Arguments

`file_path`	`character(1)` - Path to an `.R`, `.Rmd`, or `.qmd` file exported by `sus_rap_export()`.
`lang`	`character(1)` - Language for messages: `"pt"` (default), `"en"`, or `"es"`.
`validate`	`logical(1)` - Check structural integrity. Default `TRUE`.

Value

A `rap_object` list with fields `$metadata`, `$params`, `$steps`, `$structure`, `$source`, `$format`, and `$raw`.

Examples

```
## Not run:
rap <- sus_rap_read("pipeline_sp.R")
print(rap)

## End(Not run)
```

`sus_rap_recipe` *Export a RAP as a Compact YAML Recipe*

Description

Serialises the essential information of a `rap_object` - parameters, pipeline steps, and package versions - into a compact YAML file (typically under 50 lines). The recipe can be shared with colleagues and re-imported with `sus_rap_from_recipe()`.

Usage

```
sus_rap_recipe(
  rap,
  file_path = NULL,
  include_data_hash = TRUE,
  lang = "pt",
  overwrite = FALSE
)
```

Arguments

`rap`	A `rap_object` returned by `sus_rap_read()` or `sus_rap_export()`.
`file_path`	`character(1)` - Output path for the `.yaml` file. If `NULL`, a timestamped file is created in the working directory.
`include_data_hash`	`logical(1)` - Compute and embed a hash of the input parameters for traceability. Default `TRUE`.
`lang`	`character(1)` - Language for messages.
`overwrite`	`logical(1)` - Overwrite existing file. Default `FALSE`.

Value

Invisibly, the path of the created `.yaml` file.

Examples

```
## Not run:
rap <- sus_rap_read("pipeline_sp.R")
sus_rap_recipe(rap, "pipeline_sp.yaml")

## End(Not run)
```

sus_rap_run *Re-execute an Exported RAP*

Description

Reconstructs and executes the pipeline contained in a `rap_object`. Any parameters passed via `...` override the originals before execution, making it easy to replay an analysis for a different state or year range.

Usage

```
sus_rap_run(rap, ..., envir = parent.frame(), dry_run = FALSE, lang = "pt")
```

Arguments

`rap`	A `rap_object` returned by `sus_rap_read()`.
`...`	Name-value pairs overriding fields in `rap$params` (e.g. `uf = "RJ"`, `years = 2021:2023`).
`envir`	`environment` - Execution environment. Default `parent.frame()`.
`dry_run`	`logical(1)` - Print the reconstructed code without executing. Default `FALSE`.
`lang`	`character(1)` - Language for messages.

Value

Invisibly, the `data.frame` produced by the pipeline.

Examples

```
## Not run:
rap <- sus_rap_read("pipeline_sp.R")
df  <- sus_rap_run(rap, uf = "RJ", years = 2021:2023)
sus_rap_run(rap, dry_run = TRUE)

## End(Not run)
```

sus_rap_targets *Generate a targets Pipeline from a climasus4r RAP*

Description

Converts a `rap_object` or a pipeline expression into a ready-to-use `_targets.R` file. The generated script uses `targets::tar_target()` with optional dynamic branching over UFs, years, or their Cartesian product.

Usage

```
sus_rap_targets(
  pipeline,
  file_path = "_targets.R",
  branching = c("none", "uf", "year", "cross"),
  workers = 1L,
  include_qc = TRUE,
  lang = "pt",
  overwrite = FALSE
)
```

Arguments

`pipeline`	A `rap_object` (from `sus_rap_read()`) or a quoted pipeline expression (from `quote()`).
`file_path`	`character(1)` - Output path. Default `"_targets.R"`.
`branching`	`character(1)` - Branching strategy: `"none"` (default), `"uf"`, `"year"`, or `"cross"`.
`workers`	`integer(1)` - Number of parallel workers. Default `1`.
`include_qc`	`logical(1)` - Add a quality-control target. Default `TRUE`.
`lang`	`character(1)` - Language for messages.
`overwrite`	`logical(1)` - Overwrite existing file. Default `FALSE`.

Value

Invisibly, the path of the generated `_targets.R` file.

Branching strategies

`"none"` Single analysis, no branching.

`"uf"` One branch per UF - ideal for multi-state comparisons.

`"year"` One branch per year.

`"cross"` All UF-year combinations via `tidyr::crossing()`.

Examples

```
## Not run:
rap <- sus_rap_read("pipeline_sp.R")
sus_rap_targets(rap, "_targets.R", branching = "uf", workers = 4)
# then run: targets::tar_make()

## End(Not run)
```

`sus_rap_template` *Scaffold a Reproducible Analytical Pipeline Project*

Description

Creates a ready-to-use project directory with a `_targets.R` pipeline, a parameterised Quarto report, optional `renv` dependency snapshot, optional GitHub Actions CI workflow, and a structured file layout following RAP best practices.

Usage

```
sus_rap_template(
  path,
  name,
  system = "SIM-DO",
  include_renv = TRUE,
  include_targets = TRUE,
  include_quarto = TRUE,
  include_github_actions = FALSE,
  lang = "pt"
)
```

Arguments

`path`	`character(1)` - Parent directory where `<name>/` will be created (e.g. `"~/projetos"`).
`name`	`character(1)` - Project name; becomes the directory name.
`system`	`character(1)` - DATASUS system for the template pipeline (e.g. `"SIM-DO"`, `"SINAN"`, `"SIH"`). Default `"SIM-DO"`.
`include_renv`	`logical(1)` - Initialise `renv` for dependency management. Requires the `renv` package. Default `TRUE`.
`include_targets`	`logical(1)` - Create `_targets.R` and `run.R`. Default `TRUE`.
`include_quarto`	`logical(1)` - Create `analysis.qmd`. Default `TRUE`.
`include_github_actions`	`logical(1)` - Create `.github/workflows/rap.yml`. Default `FALSE`.
`lang`	`character(1)` - Language for generated comments and messages: `"pt"` (default), `"en"`, or `"es"`.

Value

Invisibly, the absolute path of the created project directory.

Generated structure

```
<name>/
|-- _targets.R          # targets pipeline
|-- run.R               # convenience runner: source + tar_make()
|-- analysis.qmd        # parameterised Quarto report
|-- R/
|   `-- functions.R     # custom analysis functions
|-- data/               # raw data (git-ignored)
|-- output/             # results (git-ignored)
|-- renv/               # renv library (when include_renv = TRUE)
|-- renv.lock
|-- .gitignore
`-- README.md
```

Examples

```
## Not run:
sus_rap_template(
  path   = "~/projetos",
  name   = "dengue_am",
  system = "SINAN",
  lang   = "pt"
)

## End(Not run)
```

---

sus_rap_update *Update Parameters in an Exported RAP File*

Description

Applies targeted patches to the editable `params <- list(...)` block of an exported RAP file without re-exporting from scratch. A `.bak` backup is created before any modification.

Usage

```
sus_rap_update(rap, ..., backup = TRUE, lang = "pt")
```

Arguments

`rap`	A `rap_object` returned by `sus_rap_read()`.
`...`	Named parameters to update (e.g. `uf = "\"RJ\""`, `years = "2021:2024"`). Values are treated as raw R code strings.
`backup`	`logical(1)` - Create a `.bak` backup before writing. Default `TRUE`.
`lang`	`character(1)` - Language for messages.

Value

Invisibly, the updated `rap_object` (re-read from the patched file).

Examples

```
## Not run:
rap    <- sus_rap_read("pipeline_sp.R")
rap_rj <- sus_rap_update(rap, uf = '"RJ"', years = "2021:2024")

## End(Not run)
```

`sus_socio_compute_indicators` *Compute Socioeconomic and Epidemiological Indicators*

Description

Computes a set of standardised indicators from the columns already present in a `climasus_df` object (typically the output of `sus_census_join()`). Indicators span demographics, socioeconomic vulnerability, mortality, morbidity, maternal-child health, and health-resource availability. Column names from the census aggregation step (e.g. `V003_sum`) can be mapped to the formula-space names expected by each indicator via `col_mapping`.

Usage

```
sus_socio_compute_indicators(
  df,
  indicators = NULL,
  col_mapping = list(),
  confidence_level = 0.95,
  add_ci = TRUE,
  lang = "pt",
  verbose = TRUE
)
```

Arguments

`df`	A `climasus_df` object at stage `>= "spatial"`. Typically the output of `sus_census_join()`.
`indicators`	Character vector of indicator IDs to compute. If `NULL` (default), all indicators whose required columns are present in `df` are computed automatically. Use `sus_socio_list_indicators()` to inspect the full catalogue.
`col_mapping`	Named list mapping formula-space variable names to actual column names in `df`. For example: `list(pop_young = "pop_0_14_sum", total_hh = "V003_sum")` Only mappings that override the defaults need to be specified.

`confidence_level`
: Numeric. Confidence level for Poisson / Binomial intervals. Default `0.95`.

`add_ci`
: Logical. If `TRUE` (default), adds `*_low` and `*_high` columns for indicators that support uncertainty quantification (Poisson for rates, Wilson score for proportions).

`lang`
: Language for messages: `"pt"` (default), `"en"`, `"es"`.

`verbose`
: Logical. Print progress messages. Default `TRUE`.

Details

Formula evaluation: each indicator formula is evaluated using `rlang::eval_tidy()` against a named-list environment built from the resolved column values, so no package-level NSE column names are needed.

Missing columns: if a requested indicator lacks required columns (after applying `col_mapping`), a warning is emitted and the indicator is skipped. The function never aborts due to missing columns.

Confidence intervals:

- *Poisson (Garwood exact)*: for event-count / population-at-risk rates (mortality, incidence, hospitalisation).
- *Binomial (Wilson score)*: for proportions (%, coverage indices).
- *None*: for ratios with complex numerators or direct index values.

Value

The input `climasus_df` with additional columns prefixed `ind_`:

- `ind_<id>` — computed value
- `ind_<id>_low` / `ind_<id>_high` — confidence-interval bounds (only for Poisson/Binomial indicators when `add_ci = TRUE`)

`sus_meta` is updated to `type = "indicators"` with a history entry.

See Also

`sus_census_join()`, `sus_socio_list_indicators()`

Examples

```
## Not run:
df_ind <- sus_socio_compute_indicators(
  df_census,
  indicators = c("dependency_ratio", "water_connection_rate"),
  col_mapping = list(
    pop_young   = "pop_0_14_sum",
    pop_elderly = "pop_65_plus_sum",
    pop_working = "pop_15_64_sum",
    hh_water    = "V111_sum",
    total_hh    = "V003_sum"
  )
```

```
)

## End(Not run)
```

`sus_socio_list_indicators` *List Available Indicators in the Catalogue*

Description

Returns a data frame describing all indicators available in `sus_socio_compute_indicators()`, including their IDs, names, categories, required columns, and uncertainty method.

Usage

```
sus_socio_list_indicators(lang = "pt", category = NULL)
```

Arguments

`lang`	Language for indicator names: `"pt"` (default), `"en"`, `"es"`.
`category`	Optional character vector to filter by category. Use `NULL` (default) to return all.

Value

A `tibble` with columns `id`, `name`, `category`, `required_cols`, `unit`, `uncertainty_method`, `source`.

Examples

```
sus_socio_list_indicators(lang = "pt")
sus_socio_list_indicators(lang = "en", category = "mortality")
```

`sus_spatial_join` *Spatially Link SUS Data to Brazilian Geographic Units*

Description

Performs a spatial join between a data.frame containing SUS data and official Brazilian geographic boundaries (state, municipality, census tract) or geocodes postal codes (CEP). This function is the core spatial engine for the entire climasus4r package and serves as the foundation for all spatial analyses.

Usage

```
sus_spatial_join(
  df,
  level = "munic",
  join_col = NULL,
  lang = "pt",
  use_cache = TRUE,
  cache_dir = "~/.climasus4r_cache/spatial",
  verbose = TRUE
)
```

Arguments

`df`	A `data.frame` containing health data with a geographic identifier column. The data frame should have a `system` column created by `detect_health_system()`.
`level`	Character string specifying the geographic aggregation level. Default is `"munic"`. Options: • **Administrative:** ”munic” (municipality). Default. • **Health:** ”health_region”, ”health_facilities”, ”school” • **Environmental:** ”amazon”, ”biomes”, ”conservation_units”, ”disaster_risk_area”, ”semiarid”, ”indigenous_land” • **Urban:** ”neighborhood”, ”urban_area”, ”metro_area”, ”urban_concentrations”, ”pop_arrangements”, • **Miscellaneous:** ”cep” (postal code) **Note**: `"cep"` level is only available for SIH and CNES systems.
`join_col`	Character string with the name of the column in `df` containing the geographic identifier. If `NULL` (default), the function will automatically detect the appropriate column based on the `level` and common SUS column patterns.
`lang`	Character string specifying the language for messages. Options: `"pt"` (Portuguese, default), `"en"` (English), `"es"` (Spanish).
`use_cache`	Logical. If `TRUE` (default), uses cached spatial data to avoid re-downloads and improve performance.
`cache_dir`	Character string specifying the directory to store cached files. Default is `"~/.climasus4r_cache/spatial"`.
`verbose`	Logical. If `TRUE` (default), prints detailed progress information including cache status, download progress, and join statistics.

Details

Caching Strategy: Spatial data is cached as Parquet files in `~/.climasus4r_cache/spatial` to:

- Avoid repeated downloads
- Improve performance (10-100x faster)

- Enable offline reuse

Code Normalization:

- **Municipality codes**: Accepts 6 or 7 digits; 7-digit codes have the verification digit removed automatically
- **CEP**: Always zero-padded to 8 characters
- All join columns are coerced to character type

Value

An `sf` object (Simple Features data.frame) with all original columns from `df` plus a `geometry` column containing the spatial geometries:

- **/munic/neighborhood**: POLYGON or MULTIPOLYGON geometries
- **/cnes/cep**: POINT geometries (geocoded locations)

References

Pereira, R.H.M.; Goncalves, C.N.; et. all (2019) geobr: Loads Shapefiles of Official Spatial Data Sets of Brazil. GitHub repository - https://github.com/ipeaGIT/geobr. Pereira, Rafael H. M.; Barbosa, Rogerio J. (2023) censobr: Download Data from Brazil's Population Census. R package version v0.4.0, https://CRAN.R-project.org/package=censobr. DOI: 10.32614/CRAN.package.censobr.

Examples

```
## Not run:
library(climasus4r)

# Example 1: Link mortality data to municipalities
df_sim <- sus_data_import(uf = "SP", year = 2023, system = "SIM-DO") %>%
  sus_data_standardize(lang = "pt")

sf_sim <- sus_spatial_join(
  df = df_sim,
  level = "munic",
  lang = "pt"
)

# Example 3: Geocode CEP for hospitalization data (SIH only)
df_sih <- sus_data_import(uf = "RJ", year = 2023, system = "SIH-RD") %>%
  sus_data_standardize(lang = "pt")

sf_cep <- sus_spatial_join(
  df = df_sih,
  level = "cep",
  lang = "pt"
)

# Example 4: Census tract level analysis
sf_census <- sus_spatial_join(
```

```
  df = df_sim,
  level = "schools",
  lang = "pt"
)

## End(Not run)
```

`sus_welcome` *Display the climasus4r Pipeline Overview*

Description

Prints a colour-coded pipeline overview to the console and, optionally, opens a self-contained HTML page in the browser (or RStudio viewer) showing all ten stages of the climasus4r health-climate-environment workflow plus utility functions.

Usage

```
sus_welcome(lang = "pt", output = c("console", "html"), open = TRUE)
```

Arguments

`lang`	Character. Language: `"pt"` (default), `"en"`, or `"es"`.
`output`	Character vector. Outputs to produce: `"console"`, `"html"`, or both (default). Multiple values are accepted.
`open`	Logical. If `TRUE` (default) and the session is interactive, opens the HTML file automatically.

Value

Invisibly returns the path to the HTML file (or `NULL` when `"html"` is not in `output`). Called for its side effects.

Examples

```
## Not run:
sus_welcome()
sus_welcome(lang = "en")
sus_welcome(output = "html", open = TRUE)
sus_welcome(lang = "es", output = c("console", "html"))

## End(Not run)
```

tidy.climasus_af *Reduce climasus_af to a tidy one-row tibble*

Description

Returns a single-row tibble suitable for `dplyr::bind_rows()` pooling across multiple cities or time periods.

Usage

```
## S3 method for class 'climasus_af'
tidy(x, ...)
```

Arguments

`x`	A `climasus_af` object.
`...`	Unused.

Value

A one-row tibble with key attributable fraction statistics.

tidy.climasus_burden *Tidy a climasus_burden object into a flat tibble*

Description

Returns the full ranked burden table with the cumulative concentration percentage joined. Each row is one city (one row per component when `component = "all"` was used). Suitable for `dplyr::bind_rows()` across multiple analyses or for plotting.

Usage

```
## S3 method for class 'climasus_burden'
tidy(x, ...)
```

Arguments

`x`	A `climasus_burden` object from `sus_mod_burden()`.
`...`	Unused.

Value

A tibble with all burden-table columns plus `cumulative_pct`.

`tidy.climasus_casecrossover` *Tidy a climasus_casecrossover object*

Description

Returns the OR table as a tibble, with analysis metadata columns prepended. Suitable for row-binding across multiple analyses with `dplyr::bind_rows()`.

Usage

```
## S3 method for class 'climasus_casecrossover'
tidy(x, ...)
```

Arguments

`x`	A `climasus_casecrossover` object from `sus_mod_casecrossover()`.
`...`	Unused.

Value

A tibble with one row per exposure term.

`tidy.climasus_dlnm` *Reduce climasus_dlnm to a tidy one-row tibble*

Description

Convenience extractor compatible with `dplyr::bind_rows()` for pooling results across multiple model runs.

Usage

```
## S3 method for class 'climasus_dlnm'
tidy(x, ...)
```

Arguments

`x`	A `climasus_dlnm` object.
`...`	Unused.

Value

A one-row tibble with key model statistics.

tidy.climasus_excess *Tidy a climasus_excess object into a one-row summary tibble*

Description

Returns a one-row tibble summarising the excess result, suitable for combining across multiple analyses with `dplyr::bind_rows()`.

Usage

```
## S3 method for class 'climasus_excess'
tidy(x, ...)
```

Arguments

`x`	A `climasus_excess` object from `sus_mod_excess()`.
`...`	Unused.

Value

A one-row tibble.

tidy.climasus_its *Tidy a climasus_its object*

Description

Returns the effects table with analysis metadata columns prepended. Suitable for combining results across series with `dplyr::bind_rows()`.

Usage

```
## S3 method for class 'climasus_its'
tidy(x, ...)
```

Arguments

`x`	A `climasus_its` object from `sus_mod_its()`.
`...`	Unused.

Value

A tibble with one row per interruption.

`tidy.climasus_metaregression` *Tidy a climasus_metaregression object into a summary tibble*

Description

Returns a one-row tibble summarising the meta-regression result plus a `covariate_tests` column containing the nested covariate test table. Suitable for combining results across multiple analyses.

Usage

```
## S3 method for class 'climasus_metaregression'
tidy(x, ...)
```

Arguments

`x`	A `climasus_metaregression` object from `sus_mod_metaregression()`.
`...`	Unused.

Value

A one-row tibble.

`tidy.climasus_ml` *Tidy a climasus_ml object into a flat predictions tibble*

Description

Returns the predictions table with `observed`, `fitted`, `cv_predicted`, and `residual` columns. Suitable for plotting, residual diagnostics, or combining across analyses.

Usage

```
## S3 method for class 'climasus_ml'
tidy(x, ...)
```

Arguments

`x`	A `climasus_ml` object from `sus_mod_ml()`.
`...`	Unused.

Value

A tibble with one row per observation.

tidy.climasus_pool *Tidy a climasus_pool object into a one-row summary tibble*

Description

Returns a one-row tibble summarising the pooled result, suitable for combining across multiple pooling runs with `dplyr::bind_rows()`.

Usage

```
## S3 method for class 'climasus_pool'
tidy(x, ...)
```

Arguments

`x`	A `climasus_pool` object from `sus_mod_pool()`.
`...`	Unused.

Value

A one-row tibble.

tidy.climasus_sensitivity *Tidy a climasus_sensitivity object*

Description

Returns the comparison table with metadata columns prepended, suitable for combining results across analyses with `dplyr::bind_rows()`.

Usage

```
## S3 method for class 'climasus_sensitivity'
tidy(x, ...)
```

Arguments

`x`	A `climasus_sensitivity` object from `sus_mod_sensitivity()`.
`...`	Unused.

Value

A tibble with one row per stratum.

tidy.climasus_swot *Tidy a climasus_swot object into a flat scores tibble*

Description

Tidy a climasus_swot object into a flat scores tibble

Usage

```
## S3 method for class 'climasus_swot'
tidy(x, ...)
```

Arguments

x	A `climasus_swot` object from `sus_mod_swot()`.
...	Ignored.

Value

A tibble with one row per entity.

tidy.climasus_vi *Tidy a climasus_vi object into a flat tibble*

Description

Returns the full VI table with normalized indicator columns appended, one row per city. Suitable for plotting, further modelling, or export.

Usage

```
## S3 method for class 'climasus_vi'
tidy(x, ...)
```

Arguments

x	A `climasus_vi` object from `sus_mod_vulnerability_index()`.
...	Unused.

Value

A tibble with city-level VI scores, pillar scores, indicator-level normalized values, and raw values — all in wide format.

`vcov.climasus_metaregression` *Extract variance-covariance from a climasus_metaregression object*

Description

Extract variance-covariance from a climasus_metaregression object

Usage

```
## S3 method for class 'climasus_metaregression'
vcov(object, ...)
```

Arguments

`object`	A `climasus_metaregression` object.
`...`	Passed to `stats::vcov()`.

Value

Variance-covariance matrix of meta-regression coefficients.

`vcov.climasus_pool` *Extract pooled variance-covariance from a climasus_pool object*

Description

Extract pooled variance-covariance from a climasus_pool object

Usage

```
## S3 method for class 'climasus_pool'
vcov(object, ...)
```

Arguments

`object`	A `climasus_pool` object.
`...`	Passed to `stats::vcov()`.

Value

Pooled variance-covariance matrix.

`write_duckdb_climasus` *Write a climasus_df to a persistent DuckDB file*

Description

Convenience wrapper that opens a persistent DuckDB connection to `path`, registers the data and metadata, then closes the connection.

Usage

```
write_duckdb_climasus(x, path, view_name = "climasus_data", overwrite = TRUE)
```

Arguments

`x`	A `climasus_df` object.
`path`	Character. File path for the DuckDB database (e.g. `"data/sim.duckdb"`).
`view_name`	Character. Name of the table inside the DuckDB file (default: `"climasus_data"`).
`overwrite`	Logical. Whether to overwrite an existing table (default: `TRUE`).

Value

`invisible(x)` — the original `climasus_df`, updated with `backend = "duckdb"` in its metadata.

Examples

```
## Not run:
df <- write_duckdb_climasus(df, "data/sim_respiratory.duckdb")
sus_meta(df, "backend")  # "duckdb"

# Read back later
con <- duckdb::dbConnect(duckdb::duckdb(), "data/sim_respiratory.duckdb")
df2 <- from_duckdb_climasus(con, "climasus_data")
duckdb::dbDisconnect(con)

## End(Not run)
```

`write_parquet_climasus` *Write a climasus_df directly to a Parquet file*

Description

Convenience wrapper around `as_arrow_climasus()` and `arrow::write_parquet()`. Metadata is preserved in the Parquet file's schema.

Usage

```
write_parquet_climasus(x, path, ...)
```

Arguments

`x`	A `climasus_df` object.
`path`	Character. File path for the Parquet file (e.g. `"data/sim_2020.parquet"`).
`...`	Additional arguments passed to `arrow::write_parquet()`.

Value

`invisible(x)` — the original `climasus_df`, updated with `backend = "parquet"` in its metadata.

Examples

```
## Not run:
df <- write_parquet_climasus(df, "data/sim_respiratory.parquet")
sus_meta(df, "backend")  # "parquet"

## End(Not run)
```

Index