Package: CDSimX (via r-universe)

October 9, 2026

Type Package

Title Simulating Climate Data for Research and Modelling

Version 1.1.0

Date 2026-04-19

Maintainer Isaac Osei <ikemillar65@gmail.com>

Description Advanced climate simulation, forecasting, visualization, export, and machine learning tools.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports copula, ggplot2, dplyr, tidyr, lubridate, randomForest, gbm, forecast, xgboost, nnet, ncdf4, rlang, readr, scales, stats

Suggests testthat (>= 3.0.0)

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

URL https://ikemillar.github.io/CDSimX/

BugReports https://github.com/ikemillar/CDSimX/issues

Config/pak/sysreqs make libgsl0-dev libicu-dev libnetcdf-dev libx11-dev

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

Date/Publication 2026-05-23 17:14:51 UTC

RemoteUrl https://github.com/ikemillar/cdsimx

RemoteRef HEAD

RemoteSha af31e381eec5a884e0cb21a8b3d9749ea08228cb

Contents

apply_physical_constraints . . . 2
bias_correction . . . 3
copula_dependence . . . 5
create_stations . . . 6
export_csv . . . 8
export_netcdf . . . 8
forecasting_ml . . . 9
generate_time_index . . . 11
plot_station_timeseries . . . 12
simulate_climate . . . 14
simulate_dewpoint . . . 16
simulate_evapotranspiration . . . 18
simulate_rainfall . . . 20
simulate_rh . . . 22
simulate_solar_radiation . . . 24
simulate_temperature . . . 27
simulate_wind_direction . . . 29
simulate_wind_speed . . . 32
validate_climate . . . 34
visualization . . . 35

Index **36**

`apply_physical_constraints`
Apply Physical Constraints to Climate Data

Description

Enforces physically realistic bounds and relationships across simulated climate variables.

Usage

```
apply_physical_constraints(climate_data, verbose = TRUE, tolerance = 0)
```

Arguments

`climate_data`	A dataframe generated from `simulate_climate()` or related simulation functions.
`verbose`	Logical. If TRUE, prints correction summaries. Default is TRUE.
`tolerance`	Numeric tolerance used when comparing floating-point values. Default is 0.

Details

This function is designed as a quality-control layer for synthetic climate simulations and ensures that impossible atmospheric or hydrological states are removed.

The function automatically checks and corrects:

- Tmin <= Tmax
- Avg.Temp lies between Tmin and Tmax
- Relative humidity remains within 0–100%
- Dew point does not exceed air temperature
- Rainfall and rain days remain non-negative
- Wind speed remains non-negative
- Solar radiation remains non-negative
- Evapotranspiration remains non-negative
- Vapor pressure deficit remains non-negative
- Sunshine fraction remains within 0–1
- Cloud factor remains within 0–1

Value

A corrected climate dataframe with physically realistic values enforced.

Examples

```
stations <- create_stations(n = 3)
time_index <- generate_time_index(
  start_date = "2010-01-01",
  end_date = "2012-12-31"
)

climate <- simulate_climate(
  stations = stations,
  time_index = time_index
)

climate <- apply_physical_constraints(climate)
```

---

bias_correction *Bias Correction for Simulated Climate Data*

Description

Applies statistical bias correction to simulated climate variables using observed reference data.

Usage

```
bias_correction(
  simulated_data,
  observed_data,
  variables = c("Tmin", "Tmax", "Avg.Temp", "Rainfall", "RH", "WindSpeed",
    "Solar_Radiation", "ET0"),
  method = "mean_scaling",
  digits = 2,
  return_factors = TRUE
)
```

Arguments

`simulated_data`	Simulated climate data frame.
`observed_data`	Observed/reference climate data frame.
`variables`	Character vector of variables to correct.
`method`	Bias-correction method. Options are: `"mean_scaling"`, `"additive"`, `"multiplicative"`.
`digits`	Number of decimal places.
`return_factors`	Logical. If TRUE, returns correction factors and bias statistics.

Details

Supported correction methods include:

- Mean scaling
- Additive correction
- Multiplicative correction

Physical constraints are automatically enforced after correction to ensure climatological realism.

Value

A list containing:

corrected_data Bias-corrected climate dataset.

correction_factors Bias-correction factors and bias statistics applied to each variable.

Examples

```
## Not run:

bc <- bias_correction(
  simulated_data = cd,
  observed_data = obs_data,
```

```
  variables = c("Rainfall", "Avg.Temp")
)

head(bc$corrected_data)
bc$correction_factors


## End(Not run)
```

`copula_dependence` *Apply Copula Dependence Structure to Climate Variables*

Description

Introduces multivariate dependence among simulated climate variables using Gaussian or t copulas.

Usage

```
copula_dependence(
  climate_data,
  variables = c("Tmin", "Tmax", "Avg.Temp", "Rainfall", "RH", "WindSpeed",
    "Solar_Radiation", "ET0"),
  copula_type = "gaussian",
  df = 4,
  seed = 123,
  digits = 2
)
```

Arguments

`climate_data`	Climate data frame.
`variables`	Variables used in dependence modeling.
`copula_type`	Type of copula. Options are: `"gaussian"` or `"t"`.
`df`	Degrees of freedom for t copula. Default is 4.
`seed`	Random seed for reproducibility.
`digits`	Number of decimal places.

Details

This function improves realism by preserving correlations and inter-variable dependency structures commonly observed in climate systems.

Supported variables include:

- Tmin

- Tmax
- Avg.Temp
- Rainfall
- RH
- WindSpeed
- Solar_Radiation
- ET0

Value

A list containing:

adjusted_data Climate dataset with copula-adjusted dependence structure.
correlation_matrix Empirical correlation matrix used in copula fitting.
copula_type Copula family used.

Examples

```
## Not run:

cp <- copula_dependence(cd)

head(cp$adjusted_data)


## End(Not run)
```

---

create_stations *Create or load station metadata*

Description

Create a station metadata table either by:

- loading from a CSV file,
- accepting an existing data.frame,
- or auto-generating synthetic stations within a bounding box.

Usage

```
create_stations(
  source = NULL,
  n = 10,
  bbox = c(-3.5, 1.5, 4.5, 11.5),
  derive_climate = TRUE,
  seed = NULL
)
```

Arguments

`source`	Path to CSV file OR a data.frame with Station/LON/LAT OR NULL (to generate synthetic stations).
`n`	Integer number of stations to generate when source = NULL. Default = 10.
`bbox`	Numeric vector: c(min_lon, max_lon, min_lat, max_lat). Default approximates Ghana's spatial extent.
`derive_climate`	Logical. If TRUE, additional climate metadata are derived for each station. Default = TRUE.
`seed`	Optional numeric seed for reproducibility.

Details

The function also supports optional derivation of climate-related station attributes used by the CDSimX simulation engine.

Value

A data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Synthetic elevation estimate (m)

CLIMATE_ZONE Derived climate zone

COASTAL_INDEX Relative coastal influence index

TEMP_BASE Baseline temperature estimate

RAIN_REGIME Derived rainfall regime

Examples

```
create_stations(n = 5, seed = 42)

create_stations(
  data.frame(
    Station = "Accra",
    LON = -0.18,
    LAT = 5.60
  )
)
```

export_csv *Export CDSimX Data to CSV*

Description

Exports any CDSimX dataframe to a CSV file.

Usage

```
export_csv(data, file = "CDSimX_output.csv", row_names = FALSE)
```

Arguments

data	A dataframe to export.
file	Output CSV filename.
row_names	Logical. Include row names?

Value

Invisibly returns the file path.

Examples

```
## Not run:
export_csv(climate_data)

## End(Not run)
```

export_netcdf *Export CDSimX Data to NetCDF*

Description

Exports climate simulation or forecasting data into NetCDF format.

Usage

```
export_netcdf(
  data,
  file = "CDSimX_output.nc",
  date_col = "DATE",
  station_col = "Station",
  lon_col = "LON",
  lat_col = "LAT",
  variables = NULL,
  fillvalue = -9999,
  overwrite = TRUE
)
```

Arguments

`data`	Climate dataframe.
`file`	Output NetCDF filename.
`date_col`	Date column name.
`station_col`	Station column name.
`lon_col`	Longitude column.
`lat_col`	Latitude column.
`variables`	Climate variables to export.
`fillvalue`	Missing value.
`overwrite`	Logical.

Value

Invisibly returns output filename.

Examples

```
## Not run:
export_netcdf(climate_data)

## End(Not run)
```

`forecasting_ml` *Machine Learning Forecasting for Climate Variables*

Description

Uses machine learning algorithms to forecast climate variables from simulated climate data.

Usage

```
forecasting_ml(
  climate_data,
  target = "Rainfall",
  predictors = c("Tmin", "Tmax", "Avg.Temp", "RH", "WindSpeed", "Solar_Radiation", "ET0"),
  forecast_horizon = 12,
  start_forecast = NULL,
  end_forecast = NULL,
  method = c("rf", "lm", "gbm", "arima", "xgboost", "nnet", "all"),
  frequency = c("month", "day", "year"),
  train_fraction = 0.8,
  include_lag = TRUE,
  lag_period = 1,
  ntree = 500,
  hidden_nodes = 5,
```

```
  digits = 2,
  seed = 123
)
```

Arguments

`climate_data`	Climate dataframe.
`target`	Variable to forecast.
`predictors`	Predictor variables.
`forecast_horizon`	Number of future periods.
`start_forecast`	Optional forecast start date. Must be coercible to Date.
`end_forecast`	Optional forecast end date. Must be coercible to Date. If supplied, forecast_horizon is automatically calculated.
`method`	Forecasting method.
`frequency`	Forecast interval. Options are: `"day"`, `"month"`, `"year"`.
`train_fraction`	Fraction of data for training.
`include_lag`	Logical. Include lag predictor.
`lag_period`	Lag size.
`ntree`	Number of trees for RF and GBM.
`hidden_nodes`	Number of hidden nodes for neural network.
`digits`	Decimal places.
`seed`	Random seed for reproducibility.

Details

Supported methods:

- Random Forest (”rf”)
- Linear Regression (”lm”)
- Gradient Boosting (”gbm”)
- ARIMA Time-Series (”arima”)
- Extreme Gradient Boosting (”xgboost”)
- Neural Network (”nnet”)
- Run All Models (”all”)

Value

A list containing:

forecast_data Forecasted future values.

model_performance RMSE, MAE, and correlation.

importance Variable importance table.

model Trained ML model.

all_results Returned only when method = "all".

`generate_time_index` *Generate Climate Simulation Time Index*

Description

Creates a standardized temporal framework for CDSimX simulations. Supports daily, monthly, and yearly temporal resolutions.

Usage

```
generate_time_index(
  start_date = "1990-01-01",
  end_date = "1999-12-31",
  frequency = "daily",
  calendar = "standard"
)
```

Arguments

`start_date`	Character or Date object. Simulation start date. Default = "1990-01-01"
`end_date`	Character or Date object. Simulation end date. Default = "1999-12-31"
`frequency`	Temporal resolution: • "daily" • "monthly" • "yearly"
`calendar`	Calendar type. Currently supports: • "standard" • "noleap"

Details

This function becomes the backbone of all climate simulations, ensuring that all variables share a synchronized temporal structure.

Value

A data.frame containing:

START_DATE Start date
END_DATE End date
DATE Date index
Year Calendar year
Month Month number
Day Day of month
DOY Day of year
Week Week number
Quarter Quarter
Season Climatological season
Frequency Simulation resolution

Examples

```
daily_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date   = "2000-12-31",
  frequency  = "daily"
)

monthly_index <- generate_time_index(
  start_date = "1990-01-01",
  end_date   = "2020-12-31",
  frequency  = "monthly"
)
```

---

plot_station_timeseries

Plot Station Climate Time Series

Description

Creates highly customizable climate time-series visualizations with automatic seasonal detection.

Supports:

- Custom line colors
- Seasonal coloring
- LOESS smoothing
- Trend lines
- Dark/light themes
- Flexible date handling
- Faceting

Usage

```
plot_station_timeseries(
  df,
  station,
  var = "Tmin",
  smooth = TRUE,
  smooth_span = 0.25,
  show_points = TRUE,
  point_size = 2,
  line_size = 1,
  line_color = NULL,
  smooth_color = NULL,
  seasonal_colors = NULL,
  use_season_colors = TRUE,
  alpha = 0.8,
  theme_style = c("minimal", "dark", "classic", "bw"),
  date_breaks = "2 years",
  date_labels = "%Y",
  facet = FALSE,
  show_trend = FALSE,
  trend_color = "black",
  title = NULL,
  subtitle = NULL
)
```

Arguments

`df`	Climate dataframe.
`station`	Station name.
`var`	Climate variable to plot.
`smooth`	Logical. Add LOESS smoothing line.
`smooth_span`	LOESS span parameter.
`show_points`	Logical. Show points on plot.
`point_size`	Point size.
`line_size`	Line width.
`line_color`	Main line color.
`smooth_color`	Smoothing line color.
`seasonal_colors`	Named vector of colors.
`use_season_colors`	Logical. Color points by season.
`alpha`	Transparency level.
`theme_style`	Plot theme. Options are: `"minimal"`, `"dark"`, `"classic"`, `"bw"`.
`date_breaks`	X-axis date interval.

date_labels	Date label format.
facet	Logical. Facet by season.
show_trend	Logical. Add linear trend line.
trend_color	Trend line color.
title	Plot title.
subtitle	Plot subtitle.

Value

A ggplot object.

Examples

```
## Not run:

plot_station_timeseries(
  climate_data,
  station = "Station_1",
  var = "Tmin"
)


## End(Not run)
```

---

simulate_climate *Simulate Integrated Climate Dataset*

Description

Generates a complete synthetic climate dataset by internally calling all climate simulation modules and merging their outputs into a single dataframe.

Usage

```
simulate_climate(stations, time_index, seed = NULL)
```

Arguments

stations	Output from create_stations().
time_index	Output from generate_time_index().
seed	Optional random seed for reproducibility.

Details

The function currently simulates:

- temperature
- rainfall
- relative humidity
- dew point
- wind speed
- wind direction
- solar radiation
- evapotranspiration

Value

A merged dataframe containing all simulated climate variables.

Examples

```
## Not run:

stations <- create_stations(
  n = 10
)

tindex <- generate_time_index(
  start_date = "2020-01-01",
  end_date   = "2020-12-31",
  frequency  = "monthly"
)

climate <- simulate_climate(
  stations,
  tindex
)

head(climate)


## End(Not run)
```

`simulate_dewpoint` *Simulate Dew Point Temperature*

Description

Generates synthetic dew point temperature series for multiple stations using simulated air temperature and relative humidity.

Usage

```
simulate_dewpoint(
  temperature,
  rh,
  min_dewpoint = -5,
  max_dewpoint = 35,
  noise_sd = 0.5,
  seed = NULL
)
```

Arguments

`temperature`	data.frame from simulate_temperature()
`rh`	data.frame from simulate_rh()
`min_dewpoint`	Numeric. Minimum allowable dew point (°C). Default = -5
`max_dewpoint`	Numeric. Maximum allowable dew point (°C). Default = 35
`noise_sd`	Numeric. Stochastic variability in dew point. Default = 0.5
`seed`	Optional numeric seed.

Details

Dew point is physically linked to:

- air temperature,
- atmospheric moisture,
- rainfall regimes,
- coastal humidity effects,
- elevation controls.

The simulation uses a Magnus-type approximation for realistic atmospheric thermodynamics.

Value

data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Elevation

DATE Simulation timestamp

Year Calendar year

Month Calendar month

Season Climatological season

Tmean Mean air temperature (°C)

RH Relative humidity (%)

DewPoint Simulated dew point temperature (°C)

Dewpoint_Depression Tmean − DewPoint

Examples

```
stations <- create_stations(
  n = 3,
  seed = 123
)

tindex <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)

temp <- simulate_temperature(
  stations,
  tindex
)

rh <- simulate_rh(
  stations,
  tindex
)

dew <- simulate_dewpoint(
  temperature = temp,
  rh = rh
)

head(dew)
```

`simulate_evapotranspiration` *Simulate Evapotranspiration*

Description

Simulates reference evapotranspiration (ET0) using temperature, relative humidity, solar radiation, and wind speed.

Usage

```
simulate_evapotranspiration(
  temperature,
  rh,
  solar_radiation,
  wind_speed,
  method = "FAO56",
  crop_factor = 1,
  humidity_sensitivity = 0.35,
  wind_sensitivity = 0.08,
  radiation_sensitivity = 0.12,
  noise_sd = 0.25,
  min_et = 0,
  max_et = 15,
  seed = NULL
)
```

Arguments

`temperature`	Output from simulate_temperature().
`rh`	Output from simulate_rh().
`solar_radiation`	Output from simulate_solar_radiation().
`wind_speed`	Output from simulate_wind_speed().
`method`	Method used for evapotranspiration estimation. Currently supports: **FAO56** Simplified FAO-56 Penman-Monteith inspired approach Default is "FAO56".
`crop_factor`	Numeric scaling factor for evapotranspiration. Default is 1.
`humidity_sensitivity`	Numeric humidity reduction factor. Default is 0.35.
`wind_sensitivity`	Numeric aerodynamic enhancement factor. Default is 0.08.
`radiation_sensitivity`	Numeric solar radiation enhancement factor. Default is 0.12.

`noise_sd`	Standard deviation for stochastic variability. Default is 0.25.
`min_et`	Minimum ET0 value. Default is 0.
`max_et`	Maximum ET0 value. Default is 15.
`seed`	Optional random seed.

Details

The function incorporates:

- temperature-driven evaporation
- humidity suppression
- solar radiation forcing
- aerodynamic wind enhancement
- elevation-based pressure adjustment
- stochastic environmental variability

Value

A data frame containing:

Station Station identifier

LON Longitude

LAT Latitude

ELEV Elevation (m)

DATE Date

Year Year

Month Month

Season Season category

Avg.Temp Average temperature (°C)

RH Relative humidity (%)

Solar_Radiation Solar radiation (MJ/m²/day)

WindSpeed Wind speed (m/s)

Atmospheric_Pressure Estimated atmospheric pressure (kPa)

VPD Vapour pressure deficit (kPa)

ET0 Reference evapotranspiration (mm/day)

Dryness_Index Normalized dryness indicator

Dryness_Class Categorical atmospheric moisture condition derived from the dryness index.Classes include: Humid, Moderate, Dry, and Very Dry

ET_Anomaly Evapotranspiration anomaly

Examples

```
## Not run:
temp <- simulate_temperature(stations, tindex_month)
rh   <- simulate_rh(stations, tindex_month)
sr   <- simulate_solar_radiation(stations, tindex_month)
ws   <- simulate_wind_speed(stations, tindex_month)

et <- simulate_evapotranspiration(
  temperature = temp,
  rh = rh,
  solar_radiation = sr,
  wind_speed = ws
)

## End(Not run)
```

---

simulate_rainfall *Simulate Rainfall Time Series*

Description

Generates synthetic rainfall series for multiple stations using stochastic climate dynamics and climate-zone controls.

Usage

```
simulate_rainfall(
  stations,
  time_index,
  wetday_prob = 0.35,
  gamma_shape = 2,
  gamma_scale = 8,
  ar_coeff = 0.4,
  seasonal_strength = 1,
  extreme_event_prob = 0.01,
  extreme_multiplier = 3,
  max_rainfall = 800,
  seed = NULL
)
```

Arguments

stations	data.frame from create_stations()
time_index	data.frame from generate_time_index()
wetday_prob	Numeric. Base wet-day probability. Used mainly for daily simulations. Default = 0.35

`gamma_shape` Numeric. Shape parameter for Gamma rainfall generation. Default = 2

`gamma_scale` Numeric. Scale parameter for Gamma rainfall generation. Default = 8

`ar_coeff` Numeric. Temporal persistence coefficient. Default = 0.4

`seasonal_strength` Numeric. Controls rainfall seasonality intensity. Default = 1

`extreme_event_prob` Numeric. Probability of extreme rainfall occurrence. Default = 0.01

`extreme_multiplier` Numeric. Multiplier applied during extreme events. Default = 3

`max_rainfall` Numeric. Maximum allowable rainfall amount. Default = 500

`seed` Optional numeric seed.

Details

The simulation incorporates:

- wet/dry occurrence processes,
- seasonal rainfall regimes,
- spatial climate variability,
- coastal moisture effects,
- elevation enhancement,
- temporal persistence,
- extreme rainfall events,
- Gamma-distributed rainfall amounts.

Supports:

- daily simulations,
- monthly simulations,
- yearly simulations.

Value

data.frame containing:

Station Station name
LON Longitude
LAT Latitude
ELEV Elevation
DATE Simulation timestamp
Year Calendar year
Month Calendar month
Season Climatological season
Rainfall Simulated rainfall amount (mm)
Wet_Day Wet occurrence indicator
Extreme_Event Extreme rainfall indicator
Rain_Anomaly Rainfall anomaly

Examples

```
stations <- create_stations(
  n = 5,
  seed = 123
)

time_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)

rain <- simulate_rainfall(
  stations = stations,
  time_index = time_index,
  seed = 123
)

head(rain)
```

---

simulate_rh *Simulate Relative Humidity Time Series*

Description

Generates synthetic Relative Humidity (RH) series for multiple climate stations using stochastic hydro-climatic relationships.

Usage

```
simulate_rh(
  stations,
  time_index,
  rainfall = NULL,
  temperature = NULL,
  ar_coeff = 0.7,
  seasonal_strength = 8,
  rain_sensitivity = 0.04,
  temp_sensitivity = 0.6,
  coastal_moisture = 12,
  noise_sd = 3,
  min_rh = 15,
  max_rh = 100,
  seed = NULL
)
```

Arguments

`stations`	data.frame from create_stations()
`time_index`	data.frame from generate_time_index()
`rainfall`	Optional rainfall data.frame from simulate_rainfall().
`temperature`	Optional temperature data.frame from simulate_temperature().
`ar_coeff`	Numeric. AR(1) persistence coefficient. Default = 0.7
`seasonal_strength`	Numeric. Controls seasonal RH variability. Default = 8
`rain_sensitivity`	Numeric. RH increase per rainfall unit. Default = 0.04
`temp_sensitivity`	Numeric. RH decrease per temperature unit. Default = 0.6
`coastal_moisture`	Numeric. Coastal humidity enhancement factor. Default = 12
`noise_sd`	Numeric. Standard deviation of stochastic noise. Default = 3
`min_rh`	Numeric. Minimum allowable RH (%). Default = 15
`max_rh`	Numeric. Maximum allowable RH (%). Default = 100
`seed`	Optional numeric seed.

Details

The simulation incorporates:

- seasonal humidity cycles,
- rainfall-humidity coupling,
- temperature-humidity interaction,
- coastal moisture effects,
- elevation drying effects,
- temporal persistence,
- stochastic atmospheric variability,
- physically realistic RH bounds.

Higher rainfall generally increases RH, while higher temperature lowers RH.

Value

data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Elevation

DATE Simulation timestamp

Year Calendar year

Month Calendar month

Season Climatological season

RH Relative humidity (%)

Humidity_Anomaly Humidity anomaly

Examples

```
stations <- create_stations(
  n = 5,
  seed = 123
)

time_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)

rain <- simulate_rainfall(
  stations,
  time_index
)

temp <- simulate_temperature(
  stations,
  time_index
)

rh <- simulate_rh(
  stations,
  time_index,
  rainfall = rain,
  temperature = temp
)

head(rh)
```

`simulate_solar_radiation`

Simulate Solar Radiation Time Series

Description

Generates synthetic incoming solar radiation series for multiple stations using physically consistent astronomical and atmospheric controls.

Usage

```
simulate_solar_radiation(
  stations,
  time_index,
  rainfall = NULL,
  rh = NULL,
  atmospheric_transmissivity = 0.65,
  cloud_attenuation = 0.12,
  humidity_attenuation = 0.08,
  elevation_factor = 0.00012,
  seasonal_strength = 1,
  noise_sd = 1.5,
  min_radiation = 0,
  max_radiation = 35,
  seed = NULL
)
```

Arguments

`stations`	data.frame from create_stations()
`time_index`	data.frame from generate_time_index()
`rainfall`	Optional numeric vector. Rainfall series from simulate_rainfall(). Used for cloud attenuation effects.
`rh`	Optional numeric vector. Relative humidity series from simulate_rh(). Used for atmospheric moisture attenuation.
`atmospheric_transmissivity`	Numeric. Baseline atmospheric transmissivity coefficient. Default = 0.65
`cloud_attenuation`	Numeric. Radiation reduction factor from rainfall/cloudiness. Default = 0.12
`humidity_attenuation`	Numeric. Radiation reduction factor from RH. Default = 0.08
`elevation_factor`	Numeric. Radiation increase per meter elevation. Default = 0.00012
`seasonal_strength`	Numeric. Controls annual radiation seasonality. Default = 1
`noise_sd`	Numeric. Standard deviation of stochastic variability. Default = 1.5
`min_radiation`	Numeric. Minimum allowable solar radiation. Default = 0
`max_radiation`	Numeric. Maximum allowable solar radiation. Default = 35
`seed`	Optional numeric seed.

Details

The simulation incorporates:

- annual solar cycle,

- latitude-dependent extraterrestrial radiation,
- cloud/rainfall attenuation,
- humidity effects,
- elevation enhancement,
- seasonal variability,
- stochastic atmospheric variability,
- physically bounded radiation values.

Solar radiation is simulated as:

- daily total radiation (MJ/m²/day) for daily simulations
- monthly mean radiation for monthly simulations
- yearly mean radiation for yearly simulations

Value

data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Elevation

DATE Simulation timestamp

Year Calendar year

Month Calendar month

Season Climatological season

Solar_Radiation Simulated solar radiation (MJ/m²/day)

Clear_Sky_Radiation Potential clear-sky radiation

Cloud_Factor Cloud attenuation factor

Sunshine_Fraction Fraction of available sunshine reaching the surface

Radiation_Anomaly Solar radiation anomaly

Examples

```
stations <- create_stations(
  n = 3,
  seed = 123
)

time_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)
```

```
rain <- simulate_rainfall(
  stations,
  time_index
)

rh <- simulate_rh(
  stations,
  time_index
)

solar <- simulate_solar_radiation(
  stations = stations,
  time_index = time_index,
  rainfall = rain$Rainfall,
  rh = rh$RH,
  seed = 123
)

head(solar)
```

`simulate_temperature` *Simulate Temperature Time Series*

Description

Generates synthetic Tmin, Tmax, and Tmean climate series for multiple stations using stochastic climate dynamics.

Usage

```
simulate_temperature(
  stations,
  time_index,
  ar_coeff = 0.7,
  seasonal_amplitude = 3,
  warming_trend = 0.02,
  noise_sd = 1,
  mean_dtr = 6,
  rainfall = NULL,
  cooling_factor = 0.15,
  min_tmin = 10,
  max_tmin = 35,
  min_tmax = 15,
  max_tmax = 45,
  seed = NULL
)
```

Arguments

`stations`	data.frame from create_stations()
`time_index`	data.frame from generate_time_index()
`ar_coeff`	Numeric. AR(1) persistence coefficient. Default = 0.7
`seasonal_amplitude`	Numeric. Baseline annual temperature cycle amplitude. Default = 3
`warming_trend`	Numeric. Annual warming trend (°C/year). Default = 0.02
`noise_sd`	Numeric. Standard deviation of stochastic variability. Default = 1
`mean_dtr`	Numeric. Mean diurnal temperature range (Tmax − Tmin). Default = 6
`rainfall`	Optional numeric vector. Rainfall values used for rainfall-temperature coupling.
`cooling_factor`	Numeric. Controls rainfall cooling strength on Tmax. Default = 0.15
`min_tmin`	Numeric. Minimum allowable Tmin. Default = 10
`max_tmin`	Numeric. Maximum allowable Tmin. Default = 35
`min_tmax`	Numeric. Minimum allowable Tmax. Default = 15
`max_tmax`	Numeric. Maximum allowable Tmax. Default = 45
`seed`	Optional numeric seed.

Details

The simulation incorporates:

- seasonal variability,
- autoregressive temporal persistence,
- spatial station effects,
- climate-zone variability,
- coastal moderation,
- elevation lapse-rate adjustment,
- long-term warming trends,
- stochastic climate variability,
- physically consistent Tmax > Tmin relationships,
- optional rainfall-temperature coupling.

Value

data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Elevation

DATE Simulation timestamp

Year Calendar year

Month Calendar month

Season Climatological season

Tmin Simulated minimum temperature (°C)

Tmax Simulated maximum temperature (°C)

Avg.Temp Mean temperature (°C)

DTR Diurnal temperature range (°C)

Examples

```
stations <- create_stations(
  n = 3,
  seed = 123
)

time_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)

temp <- simulate_temperature(
  stations = stations,
  time_index = time_index,
  seed = 123
)

head(temp)
```

`simulate_wind_direction`

Simulate Wind Direction Time Series

Description

Generates synthetic wind direction fields for multiple stations using stochastic atmospheric circulation dynamics.

Usage

```
simulate_wind_direction(
  stations,
  time_index,
  wind_speed = NULL,
```

```
  base_direction = 225,
  seasonal_shift = 30,
  noise_sd = 30,
  extreme_shift_prob = 0.01,
  max_shift = 90,
  seed = NULL
)
```

Arguments

`stations`	data.frame from create_stations()
`time_index`	data.frame from generate_time_index()
`wind_speed`	Optional numeric vector. Wind speed values used to dynamically adjust directional variability. If NULL, synthetic wind speed is generated internally.
`base_direction`	Numeric. Default prevailing wind direction (degrees). Default = 225
`seasonal_shift`	Numeric. Controls seasonal directional oscillation. Default = 30
`noise_sd`	Numeric. Base directional variability. Default = 30
`extreme_shift_prob`	Numeric. Probability of abrupt directional shifts. Default = 0.01
`max_shift`	Numeric. Maximum extreme directional deviation. Default = 90
`seed`	Optional numeric seed.

Details

The simulation incorporates:

- prevailing regional wind regimes,
- seasonal directional shifts,
- coastal circulation effects,
- temporal persistence,
- wind-speed-dependent directional variability,
- stochastic directional turbulence,
- extreme wind-direction shifts,
- optional wind-speed coupling,
- vector wind components (u and v).

Value

data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Elevation

CLIMATE_ZONE Climate classification

DATE Simulation timestamp

Year Calendar year

Month Calendar month

Season Climatological season

WindSpeed Wind speed (m/s)

WindDirection Wind direction (degrees)

WindSector Compass sector

Prevailing_Direction Mean prevailing direction

Direction_Variability Directional variability

Extreme_Shift Extreme directional event flag

Wind_u Zonal wind component

Wind_v Meridional wind component

Examples

```
stations <- create_stations(
  n = 3,
  seed = 123
)

time_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)

ws <- simulate_wind_speed(
  stations,
  time_index
)

wd <- simulate_wind_direction(
  stations,
  time_index,
  wind_speed = ws$WindSpeed
)

head(wd)
```

`simulate_wind_speed` *Simulate Wind Speed Time Series*

Description

Generates synthetic wind speed climate series for multiple stations using stochastic atmospheric dynamics.

Usage

```
simulate_wind_speed(
  stations,
  time_index,
  ar_coeff = 0.6,
  seasonal_strength = 1,
  noise_sd = 1.5,
  extreme_event_prob = 0.01,
  extreme_multiplier = 2,
  min_ws = 0,
  max_ws = 40,
  seed = NULL
)
```

Arguments

`stations`	data.frame from create_stations()
`time_index`	data.frame from generate_time_index()
`ar_coeff`	Numeric. AR(1) persistence coefficient. Default = 0.6
`seasonal_strength`	Numeric. Controls seasonal wind variability. Default = 1
`noise_sd`	Numeric. Standard deviation of stochastic turbulence. Default = 1.5
`extreme_event_prob`	Numeric. Probability of extreme wind events. Default = 0.01
`extreme_multiplier`	Numeric. Multiplier applied during extreme events. Default = 2
`min_ws`	Numeric. Minimum allowable wind speed. Default = 0
`max_ws`	Numeric. Maximum allowable wind speed. Default = 40
`seed`	Optional numeric seed.

Details

The simulation incorporates:

- seasonal circulation variability,
- coastal enhancement,

- elevation acceleration,
- temporal persistence,
- stochastic turbulence,
- extreme wind events,
- climate-zone effects.

Wind speed is simulated in m/s.

Value

data.frame containing:

Station Station name

LON Longitude

LAT Latitude

ELEV Elevation

DATE Simulation timestamp

Year Calendar year

Month Calendar month

Season Climatological season

WindSpeed Simulated wind speed (m/s)

Extreme_Wind Extreme wind event indicator

Wind_Anomaly Wind speed anomaly

Examples

```
stations <- create_stations(
  n = 5,
  seed = 123
)

time_index <- generate_time_index(
  start_date = "2000-01-01",
  end_date = "2005-12-31",
  frequency = "monthly"
)

wind <- simulate_wind_speed(
  stations,
  time_index
)

head(wind)
```

`validate_climate` *Validate Simulated Climate Dataset*

Description

Performs statistical and physical validation checks on a simulated climate dataset generated using `simulate_climate()`.

Usage

```
validate_climate(climate_data, digits = 2, return_data = TRUE)
```

Arguments

`climate_data`	Data frame generated from `simulate_climate()`.
`digits`	Number of decimal places for summaries. Default is 2.
`return_data`	Logical. If TRUE, returns all validation outputs as a list. Default is TRUE.

Details

The function computes:

- Descriptive statistics
- Missing value diagnostics
- Physical consistency checks
- Correlation structure
- Seasonal summaries
- Station summaries
- Extreme-event frequencies
- Validation summary metrics

Value

A list containing:

summary_statistics Descriptive statistics for all numeric climate variables.

missing_values Count and percentage of missing values.

physical_checks Number of physical inconsistencies detected.

correlation_matrix Correlation matrix among major climate variables.

seasonal_summary Mean climate conditions by season.

station_summary Mean climate conditions by station.

extreme_summary Frequency of extreme rainfall and wind events.

validation_summary Overall validation metrics for the climate dataset.

Examples

```
## Not run:

cd <- simulate_climate(stations, tindex_month)

vc <- validate_climate(cd)

vc$summary_statistics
vc$correlation_matrix
vc$physical_checks
vc$validation_summary


## End(Not run)
```

`visualization` *Visualization Functions for Climate Data*

Description

Flexible visualization tools for CDSimX climate datasets.

Index

apply_physical_constraints, 2

bias_correction, 3

copula_dependence, 5
create_stations, 6

export_csv, 8
export_netcdf, 8

forecasting_ml, 9

generate_time_index, 11

plot_station_timeseries, 12

simulate_climate, 14
simulate_dewpoint, 16
simulate_evapotranspiration, 18
simulate_rainfall, 20
simulate_rh, 22
simulate_solar_radiation, 24
simulate_temperature, 27
simulate_wind_direction, 29
simulate_wind_speed, 32

validate_climate, 34
visualization, 35