# Package ‘rIACI’

July 23, 2025

**Type** Package

**Title** Iberian Actuarial Climate Index Calculations

**Version** 1.0.0

**Description** Calculates the Iberian Actuarial Climate Index and its components—including temperature, precipitation, wind power, and sea level data—to support climate change analysis and risk assessment. See ``Zhou et al.'' (2023) <doi:10.26360/2023_3> for further details.

**License** GPL-3

**Encoding** UTF-8

**Imports** Rcpp (>= 1.0.5), dplyr, tidyr, lubridate, magrittr, reticulate, ecmwfr, readr, stats, utils

**LinkingTo** Rcpp

**Suggests** roxygen2, devtools, knitr, rmarkdown, testthat (>= 3.0.0)

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Nan Zhou [aut, cre] (ORCID: <https://orcid.org/0009-0002-5166-6753>)

**Maintainer** Nan Zhou <zhounan@ucm.es>

**Repository** CRAN

**Date/Publication** 2025-03-17 16:10:02 UTC

## Contents

---

`cdd` *Calculate Consecutive Dry Days (CDD)*

---

### Description

Calculates the maximum length of consecutive dry days.

### Usage

```
cdd(ci, spells_can_span_years = TRUE, monthly = TRUE)
```

### Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `spells_can_span_years` | Logical. Whether spells can span across years (default is TRUE). |
| `monthly` | Logical. Whether to interpolate annual data to monthly data (default is TRUE). |

### Value

Data frame with dates and calculated CDD values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate annual CDD index
cdd_values <- cdd(ci, monthly = FALSE)
```

---

# cdd_std — *Calculate Standardized CDD Index*

---

## Description

Calculates the standardized consecutive dry days index.

## Usage

```
cdd_std(ci, freq = c("monthly", "seasonal"))
```

## Arguments

| | |
|---|---|
| ci | List. Climate input object. |
| freq | Character. Frequency of calculation, either "monthly" or "seasonal". |

## Value

Data frame with dates and standardized CDD values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate standardized CDD index on a monthly basis
cdd_std_values <- cdd_std(ci, freq = "monthly")
```

---

`climate_input` *Climate Input Function*

---

## Description

Processes climate data and calculates necessary statistics for climate index calculations.

## Usage

```
climate_input(
  tmax = NULL,
  tmin = NULL,
  prec = NULL,
  wind = NULL,
  dates = NULL,
  base.range = c(1961, 1990),
  n = 5,
  quantiles = NULL,
  temp.qtiles = c(0.1, 0.9),
```

```
  wind.qtile = 0.9,
  max.missing.days = c(annual = 15, monthly = 3),
  min.base.data.fraction.present = 0.1
)
```

## Arguments

| | |
|---|---|
| `tmax` | Numeric vector. Maximum temperature data. |
| `tmin` | Numeric vector. Minimum temperature data. |
| `prec` | Numeric vector. Precipitation data. |
| `wind` | Numeric vector. Wind speed data. |
| `dates` | Date vector. Dates corresponding to the data. |
| `base.range` | Numeric vector of length 2. Base range years for calculations (default is c(1961, 1990)). |
| `n` | Integer. Window size for running averages (default is 5). |
| `quantiles` | List. Pre-calculated quantiles (optional). |
| `temp.qtiles` | Numeric vector. Temperature quantiles to calculate (default is c(0.10, 0.90)). |
| `wind.qtile` | Numeric. Wind quantile to calculate (default is 0.90). |
| `max.missing.days` | Named numeric vector. Maximum allowed missing days for annual and monthly data (default is c(annual = 15, monthly = 3)). |
| `min.base.data.fraction.present` | Numeric. Minimum fraction of data required in base range (default is 0.1). |

## Value

A list containing processed data and related information for climate index calculations.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
```

```
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)

# 4. Examine the structure of ci
str(ci)
```

---

`csv_to_netcdf` *CSV to NetCDF Function*

---

## Description

Merges CSV files in a specified directory into a single NetCDF file, completing the grid by filling missing values.

## Usage

```
csv_to_netcdf(csv_dir, output_file, freq)
```

## Arguments

| | |
|---|---|
| `csv_dir` | Character. Directory containing CSV files, each file representing a single latitude-longitude point. The filename format should be 'lat_lon.csv'. |
| `output_file` | Character. Path to the output NetCDF file. |
| `freq` | Character. Frequency of the data, either ''monthly'' or ''seasonal''. - ''monthly'' data uses date format ''YYYY-MM''. - ''seasonal'' data uses date format like ''YYYY-SSS'' (e.g., ''1961-DJF''). |

## Value

None. The NetCDF file is saved to the specified location.

## Examples

```
## Not run:
# Example usage of csv_to_netcdf
csv_directory <- "/path/to/csv_files"
output_netcdf_file <- "/path/to/output_file.nc"
csv_to_netcdf(csv_dir = csv_directory, output_file = output_netcdf_file, freq = "monthly")

## End(Not run)
```

download_data *Download ERA5-Land Data*

### Description

Downloads ERA5-Land data from the ECMWF Climate Data Store for the specified time range and variables. Implements a retry mechanism to handle transient errors during data download.

### Usage

```
download_data(
  start_year,
  end_year,
  start_month = 1,
  end_month = 12,
 variables = c("10m_u_component_of_wind", "10m_v_component_of_wind", "2m_temperature",
    "total_precipitation"),
  dataset = "reanalysis-era5-land",
  area = c(-90, -180, 90, 180),
  output_dir = "cds_data",
  user_id,
  user_key,
  max_retries = 3,
  retry_delay = 5,
  timeout = 7200
)
```

### Arguments

| | |
|---|---|
| `start_year` | Integer. The starting year for data download. |
| `end_year` | Integer. The ending year for data download. |
| `start_month` | Integer. The starting month (default is 1). |
| `end_month` | Integer. The ending month (default is 12). |
| `variables` | Character vector. Variables to download. Default includes common variables: `c("10m_u_component_of_wind", "10m_v_component_of_wind","2m_temperature", "total_precipitation")`. |
| `dataset` | Character. Dataset short name (default is "reanalysis-era5-land"). |
| `area` | Numeric vector. Geographical area specified as `c(North, West, South, East)`. |
| `output_dir` | Character. Directory to save the downloaded data (default is "cds_data"). |
| `user_id` | Character. Your ECMWF user ID. |
| `user_key` | Character. Your ECMWF API key. |
| `max_retries` | Integer. Maximum number of retry attempts in case of download failure (default is 3). |
| `retry_delay` | Numeric. Delay between retry attempts in seconds (default is 5). |
| `timeout` | Numeric. Timeout duration for each request in seconds (default is 7200, i.e., 2 hours). |

### Value

None. Data is downloaded to the specified output directory.

### Examples

```
## Not run:
# Set your ECMWF user ID and key
user_id <- "your_user_id"
user_key <- "your_api_key"

# Define the geographical area (North, West, South, East)
area <- c(90, -180, -90, 180) # Global

# Download data for 2020
download_data(
  start_year = 2020,
  end_year = 2020,
  variables = c("2m_temperature", "total_precipitation"),
  area = area,
  user_id = user_id,
  user_key = user_key
)

## End(Not run)
```

---

## `export_data_to_csv` *Export Data to CSV Function*

---

### Description

Exports data from a NetCDF file to CSV files, one for each latitude and longitude point, including only points where data is present. This function utilizes a Python script to perform the data processing.

### Usage

```
export_data_to_csv(nc_file, output_dir)
```

### Arguments

| | |
|---|---|
| `nc_file` | Character. Path to the NetCDF file. |
| `output_dir` | Character. Output directory where CSV files will be saved. |

### Details

The function calls a Python script using the ‘reticulate‘ package to process the NetCDF file. The Python script ‘data_processing.py‘ should be located in the ‘python‘ directory of the ‘rIACI‘ package. Only grid points with available data are exported to CSV files. Each CSV file corresponds to a specific latitude and longitude point.

## Value

None. CSV files are saved to the specified output directory.

## Examples

```
## Not run:
# Example usage of export_data_to_csv
netcdf_file <- "/path/to/processed_data.nc"
csv_output_directory <- "/path/to/csv_output"
export_data_to_csv(nc_file = netcdf_file, output_dir = csv_output_directory)

## End(Not run)
```

---

`iaci_output` *Calculate Integrated Iberian Actuarial Climate Index (IACI)*

---

## Description

Integrates various standardized indices to compute the IACI.

## Usage

```
iaci_output(ci, si, freq = c("monthly", "seasonal"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object. |
| `si` | Data frame. Sea level input data. |
| `freq` | Character. Frequency of calculation, either "monthly" or "seasonal". |

## Value

Data frame with dates and IACI values.

## Examples

```
# 1. Generate a climate_input object
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
```

```
wind <- runif(n, min = 0, max = 10)
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# 2. Create a sea level data frame with sea_input()
monthly_seq <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-01"),
  by   = "month"
)
sea_dates <- format(monthly_seq, "%Y-%m")
n <- length(sea_dates)
linear_trend <- seq(0, 10, length.out = n)
random_noise <- rnorm(n, mean = 0, sd = 0.3)
sea_values   <- 10 + linear_trend + random_noise  # starts ~10, ends ~20
si <- sea_input(Date = sea_dates, Value = sea_values)
# Then:
# 3. Calculate the IACI with monthly frequency
result <- iaci_output(ci, si, freq = "monthly")
```

---

`monthly_to_seasonal` *Convert Monthly Data to Seasonal Data*

---

## Description

Aggregates monthly data into seasonal averages.

## Usage

```
monthly_to_seasonal(data)
```

## Arguments

| | |
|---|---|
| `data` | Data frame. Monthly data with Date and Value columns. |

## Value

Data frame with seasonal data.

## Examples

```
## Not run:
# Assuming you have monthly data in a data frame 'monthly_data'
# with columns 'Date' (in 'YYYY-MM' format) and 'Value'
seasonal_data <- monthly_to_seasonal(monthly_data)
```

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

output_all *Output IACI of all grids*

## Description

Processes all CSV files in the input directory and outputs the results to the output directory.

## Usage

```
output_all(
  si,
  input_dir,
  output_dir,
  freq = c("monthly", "seasonal"),
  base.range = c(1961, 1990),
  time.span = c(1961, 2022)
)
```

## Arguments

| | |
|---|---|
| si | Data frame. Sea level input data. |
| input_dir | Character. Directory containing input CSV files. |
| output_dir | Character. Directory to save output files. |
| freq | Character. Frequency of calculation, either "monthly" or "seasonal". |
| base.range | Numeric vector. Base range years (default is c(1961, 1990)). |
| time.span | Numeric vector. Time span for output data (default is c(1961, 2022)). |

## Value

None. Results are saved to the output directory.

## Examples

```
## Not run:
# Assume we have sea level data 'si' and input/output directories
input_dir <- "path/to/input/csv/files"
output_dir <- "path/to/save/output/files"
# Run the output_all function with monthly frequency
output_all(si, input_dir, output_dir, freq = "monthly")

## End(Not run)
```

## `process_data` *Process Data Function*

### Description

Processes NetCDF files in the input directory and saves merged and processed data to the output directory.

### Usage

```
process_data(input_dir, output_dir, save_merged = FALSE)
```

### Arguments

| | |
|---|---|
| `input_dir` | Character. Directory containing input NetCDF files. |
| `output_dir` | Character. Directory to save output files. |
| `save_merged` | Logical. If TRUE, saves the merged NetCDF file. Default is FALSE. |

### Value

None. Outputs are saved to the specified directory.

### Examples

```
## Not run:
# Example usage of process_data
input_directory <- "/path/to/input/netcdf_files"
output_directory <- "/path/to/output"
process_data(input_dir = input_directory, output_dir = output_directory, save_merged = TRUE)

## End(Not run)
```

## `rx5day` *Calculate Rx5day Index*

### Description

Calculates the maximum consecutive 5-day precipitation amount.

### Usage

```
rx5day(ci, freq = c("monthly", "annual"), center_mean_on_last_day = FALSE)
```

### Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |
| `center_mean_on_last_day` | Logical. Whether to center the mean on the last day (default is FALSE). |

### Value

Data frame with dates and calculated Rx5day values.

### Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate monthly Rx5day index
rx5day_values <- rx5day(ci, freq = "monthly")
```

---

## rx5day_std — *Calculate Standardized Rx5day Index*

---

### Description

Calculates the standardized Rx5day index.

## Usage

```
rx5day_std(ci, freq = c("monthly", "seasonal"))
```

## Arguments

| | |
|---|---|
| ci | List. Climate input object. |
| freq | Character. Frequency of calculation, either "monthly" or "seasonal". |

## Value

Data frame with dates and standardized Rx5day values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate standardized Rx5day index on a monthly basis
rx5day_std_values <- rx5day_std(ci, freq = "monthly")
```

---

`sea_input` *Sea Level Input Function*

---

## Description

Creates a data frame for sea level data input.

## Usage

```
sea_input(Date = NULL, Value = NA)
```

## Arguments

| | |
|---|---|
| Date | Character vector. Dates in "YYYY-MM" format. |
| Value | Numeric vector. Sea level values (default is NA). |

## Value

Data frame with Date and Value columns.

## Examples

```
# 1. Create a monthly sequence from 1960-01 through 2020-12
monthly_seq <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-01"),
  by   = "month"
)

# Convert to "YYYY-MM" format, which sea_input() expects
sea_dates <- format(monthly_seq, "%Y-%m")

# 2. Generate random data with an upward linear trend plus some noise
n <- length(sea_dates)
linear_trend <- seq(0, 10, length.out = n)
random_noise <- rnorm(n, mean = 0, sd = 0.3)
sea_values   <- 10 + linear_trend + random_noise  # starts ~10, ends ~20

# 3. Create a sea level data frame with sea_input()
si <- sea_input(Date = sea_dates, Value = sea_values)

# 4. Inspect the first few rows
head(si)
```

---

sea_std *Calculate Standardized Sea Level Index*

---

## Description

Calculates the standardized sea level index.

## Usage

```
sea_std(si, ci, freq = c("monthly", "seasonal"))
```

### Arguments

| | |
|---|---|
| `si` | Data frame. Sea level input data created by `sea_input`. |
| `ci` | List. Climate input object containing the base range (e.g., created by `climate_input`). |
| `freq` | Character. Frequency of calculation, either "monthly" or "seasonal". |

### Value

Data frame with dates and standardized sea level values.

---

## t10p *Calculate T10p Index*

### Description

Calculates the combined percentage of days when temperature is below the 10th percentile.

### Usage

```
t10p(ci, freq = c("monthly", "annual"))
```

### Arguments

| | |
|---|---|
| `ci` | List. Climate input object. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

### Value

Data frame with dates and calculated T10p values.

### Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
```

```
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate monthly T10p index
t10p_values <- t10p(ci, freq = "monthly")
```

---

t10p_std *Calculate Standardized T10p Index*

---

## Description

Calculates the standardized T10p index.

## Usage

```
t10p_std(ci, freq = c("monthly", "seasonal"))
```

## Arguments

| | |
|---|---|
| ci | List. Climate input object. |
| freq | Character. Frequency of calculation, either "monthly" or "seasonal". |

## Value

Data frame with dates and standardized T10p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)
```

```
# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate standardized T10p index on a monthly basis
t10p_std_values <- t10p_std(ci, freq = "monthly")
```

---

`t90p` *Calculate T90p Index*

---

## Description

Calculates the combined percentage of days when temperature is above the 90th percentile.

## Usage

```
t90p(ci, freq = c("monthly", "annual"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

## Value

Data frame with dates and calculated T90p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
```

```
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate monthly T90p index
t90p_values <- t90p(ci, freq = "monthly")
```

---

`t90p_std` *Calculate Standardized T90p Index*

---

## Description

Calculates the standardized T90p index.

## Usage

```
t90p_std(ci, freq = c("monthly", "seasonal"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object. |
| `freq` | Character. Frequency of calculation, either "monthly" or "seasonal". |

## Value

Data frame with dates and standardized T90p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
```

```
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate standardized T90p index on a monthly basis
t90p_std_values <- t90p_std(ci, freq = "monthly")
```

---

`tn10p` *Calculate TN10p Index*

---

## Description

Calculates the percentage of days when minimum temperature is below the 10th percentile.

## Usage

```
tn10p(ci, freq = c("monthly", "annual"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

## Value

Data frame with dates and calculated TN10p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
```

```
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate monthly TN10p index
tn10p_values <- tn10p(ci, freq = "monthly")
```

---

`tn90p` *Calculate TN90p Index*

---

## Description

Calculates the percentage of days when minimum temperature is above the 90th percentile.

## Usage

```
tn90p(ci, freq = c("monthly", "annual"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

## Value

Data frame with dates and calculated TN90p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)
```

```
# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 5. Calculate monthly TN90p index
tn90p_values <- tn90p(ci, freq = "monthly")
```

---

`tx10p` *Calculate TX10p Index*

---

## Description

Calculates the percentage of days when maximum temperature is below the 10th percentile.

## Usage

```
tx10p(ci, freq = c("monthly", "annual"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

## Value

Data frame with dates and calculated TX10p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
```

```
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# Calculate monthly TX10p index
tx10p_values <- tx10p(ci, freq = "monthly")
```

---

`tx90p` *Calculate TX90p Index*

---

## Description

Calculates the percentage of days when maximum temperature is above the 90th percentile.

## Usage

```
tx90p(ci, freq = c("monthly", "annual"))
```

## Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

## Value

Data frame with dates and calculated TX90p values.

### Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 5. Calculate monthly TX90p index
tx90p_values <- tx90p(ci, freq = "monthly")
```

---

## w90p — *Calculate W90p Index*

---

### Description

Calculates the percentage of days when wind speed is above the 90th percentile.

### Usage

```
w90p(ci, freq = c("monthly", "annual"))
```

### Arguments

| | |
|---|---|
| `ci` | List. Climate input object created by `climate_input`. |
| `freq` | Character. Frequency of calculation, either "monthly" or "annual". |

### Value

Data frame with dates and calculated W90p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate monthly W90p index
w90p_values <- w90p(ci, freq = "monthly")
```

---

`w90p_std` *Calculate Standardized W90p Index*

---

## Description

Calculates the standardized W90p index.

## Usage

```
w90p_std(ci, freq = c("monthly", "seasonal"))
```

## Arguments

| | |
|---|---|
| ci | List. Climate input object. |
| freq | Character. Frequency of calculation, either "monthly", "seasonal". |

## Value

Data frame with dates and standardized W90p values.

## Examples

```
# 1. Generate a daily date sequence from 1960-01-01 to 2020-12-31
dates <- seq.Date(
  from = as.Date("1960-01-01"),
  to   = as.Date("2020-12-31"),
  by   = "day"
)

# 2. Create random weather data for each date
n <- length(dates)
tmax <- runif(n, min = 5, max = 40)
tmin <- runif(n, min = -10,  max = 5)
# Example: use a Poisson distribution to simulate precipitation
prec <- rpois(n, lambda = 2)
# Random wind speeds, e.g., 0 to 10 m/s
wind <- runif(n, min = 0, max = 10)

# 3. Construct the climate_input object
ci <- climate_input(
  tmax  = tmax,
  tmin  = tmin,
  prec  = prec,
  wind  = wind,
  dates = dates
)
# Then:
# 4. Calculate standardized W90p index on a monthly basis
w90p_std_values <- w90p_std(ci, freq = "monthly")
```

# Index