Package 'imdR'

July 17, 2026

Title Download, Process and Visualize IMD Gridded Meteorological Data

Version 0.4.0

Description Interface to India Meteorological Department (IMD) gridded daily rainfall (0.25 degree, 1901-present) and temperature (1.0 degree, 1951-present) binary data. Provides functions to download, read, extract by point or boundary, compute climate indices, perform trend analysis, and produce publication-quality maps with Survey of India approved boundaries.

License MIT + file LICENSE

URL https://github.com/Subhradip25/imdR

BugReports https://github.com/Subhradip25/imdR/issues

Encoding UTF-8

Language en-US

RoxygenNote 8.0.0

Depends R (>= 4.1.0)

Imports ggplot2, httr2, Kendall, ncdf4, rlang, sf, terra, tidyterra, zoo

LazyData true

Suggests future, future.apply, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Subhradip Bhattacharjee [aut, cre],
Amitava Panja [aut],
Ankita Chakraborty [aut],
Basavareddy [aut]

Maintainer Subhradip Bhattacharjee <subhradip25@gmail.com>

Repository CRAN

Date/Publication 2026-07-17 17:10:02 UTC

Contents

`compute_rainfall_indices`

Compute rainfall climate indices

Description

Computes 11 indices per grid cell per year. Handles single-year SpatRasters and multi-year named lists from get_data().

Usage

```
compute_rainfall_indices(
  rain_raster,
  level = NULL,
  name = NULL,
  file_dir,
  save_csv = TRUE
)
```

Arguments

rain_raster	A SpatRaster or named list from get_data("rain",...).
level	NULL, "state", or "district".
name	State or district name.
file_dir	Output directory for CSV.
save_csv	Save results as CSV? Default TRUE.

Value

Invisible data frame with columns year, cell, dr, d64, d115, rx1day, rx5day, rtwd, sdii, total, cwd, cdd, pci.

Examples

```
# Full India rainfall indices for 2020
r   <- get_data("rain", 2020, 2020, tempdir())
idx <- compute_rainfall_indices(r, file_dir = tempdir())

# State level indices
goa_idx <- compute_rainfall_indices(r,
  level    = "state",
  name     = "Goa",
  file_dir = tempdir())

# Multi-year indices for trend analysis
r_3yr   <- get_data("rain", 2018, 2020, tempdir())
idx_3yr <- compute_rainfall_indices(r_3yr, file_dir = tempdir())
```

---

compute_temp_indices *Compute temperature climate indices*

Description

Computes 13 indices per grid cell per year from daily tmax and tmin.

Usage

```
compute_temp_indices(
  tmax_raster,
  tmin_raster,
  level = NULL,
  name = NULL,
  file_dir,
  save_csv = TRUE
)
```

Arguments

tmax_raster	A SpatRaster or named list for tmax.
tmin_raster	A SpatRaster or named list for tmin.
level	NULL, "state", or "district".
name	State or district name.
file_dir	Output directory for CSV.
save_csv	Save results as CSV? Default TRUE.

Value

Invisible data frame with columns year, cell, mean_tmax, mean_tmin, mean_dtr, txx, txn, tnx, tnn, su35, su40, tr10, tr25, wsdi, csdi.

Examples

```
# Full India temperature indices for 2020
tx  <- get_data("tmax", 2020, 2020, tempdir())
tn  <- get_data("tmin", 2020, 2020, tempdir())
idx <- compute_temp_indices(tx, tn, file_dir = tempdir())

# State level indices
goa_idx <- compute_temp_indices(tx, tn,
  level    = "state",
  name     = "Goa",
  file_dir = tempdir())

# Multi-year temperature indices
tx_3yr  <- get_data("tmax", 2018, 2020, tempdir())
tn_3yr  <- get_data("tmin", 2018, 2020, tempdir())
idx_3yr <- compute_temp_indices(tx_3yr, tn_3yr,
                                file_dir = tempdir())
```

---

extract_by_boundary *Extract IMD raster masked to a state or district boundary*

Description

Crops and masks an IMD SpatRaster to any named state or district using bundled SOI-approved boundaries. Supports three output formats:

- "netcdf" – CF-1.7 compliant NetCDF
- "geotiff" – Multi-band GeoTIFF, opens in QGIS/ArcGIS
- "csv" – Long-format table: date, lat, lon, value

Usage

```
extract_by_boundary(
  imd_raster,
  level = "state",
  name = NULL,
  variable = "rain",
  save = FALSE,
  format = "netcdf",
  file_dir
)
```

Arguments

imd_raster	A SpatRaster or named list from get_data().
level	"state" (default) or "district".
name	State or district name (partial match allowed).
variable	Variable name for output column and filename.
save	Save output to disk? Default FALSE.
format	"netcdf" (default), "geotiff", or "csv".
file_dir	Output directory.

Value

Invisible masked SpatRaster.

Examples

```
r <- get_data("rain", 2020, 2020, tempdir())

# Return masked raster without saving
nagaland_rain <- extract_by_boundary(r, "state", "Nagaland", "rain")

# State — NetCDF
extract_by_boundary(r, "state", "Nagaland", "rain",
  save = TRUE, format = "netcdf", file_dir = tempdir())

# State — GeoTIFF (QGIS/ArcGIS)
extract_by_boundary(r, "state", "Nagaland", "rain",
  save = TRUE, format = "geotiff", file_dir = tempdir())

# State — CSV (all grid points x all days)
extract_by_boundary(r, "state", "Nagaland", "rain",
  save = TRUE, format = "csv", file_dir = tempdir())

# District — all formats work the same way
extract_by_boundary(r, "district", "North Goa", "rain",
  save = TRUE, format = "csv", file_dir = tempdir())
```

get_bbox *Extract IMD data within a bounding box*

Description

Crops IMD raster data to a user-defined latitude/longitude bounding box. Useful for custom regions such as the Indo-Gangetic Plains, Western Ghats, or any area not matching a state or district boundary. Supports three output formats: NetCDF, GeoTIFF, and long-format CSV.

Usage

```
get_bbox(
  lat_min,
  lat_max,
  lon_min,
  lon_max,
  variable,
  start_yr,
  end_yr,
  file_dir,
  format = "netcdf",
  save = TRUE
)
```

Arguments

lat_min	Numeric. Minimum latitude.
lat_max	Numeric. Maximum latitude.
lon_min	Numeric. Minimum longitude.
lon_max	Numeric. Maximum longitude.
variable	One of "rain", "tmax", "tmin".
start_yr	Integer. Start year.
end_yr	Integer. End year.
file_dir	Character. Directory for files.
format	"netcdf" (default), "geotiff", or "csv".
save	Logical. Save output? Default TRUE.

Value

Invisible SpatRaster of the cropped region.

Examples

```
# Indo-Gangetic Plains -- NetCDF
get_bbox(lat_min = 24, lat_max = 30,
         lon_min = 73, lon_max = 88,
         variable = "rain",
         start_yr = 2020, end_yr = 2020,
         file_dir = tempdir(),
         format   = "netcdf")

# Western Ghats -- GeoTIFF
get_bbox(lat_min = 8,  lat_max = 21,
         lon_min = 73, lon_max = 78,
         variable = "rain",
         start_yr = 2020, end_yr = 2020,
         file_dir = tempdir(),
         format   = "geotiff")

# Northeast India -- CSV (all grid points x all days)
get_bbox(lat_min = 22, lat_max = 29,
         lon_min = 89, lon_max = 97,
         variable = "rain",
         start_yr = 2020, end_yr = 2020,
         file_dir = tempdir(),
         format   = "csv")
```

---

get_boundary *Get the sf boundary for a named state or district*

Description

Get the sf boundary for a named state or district

Usage

```
get_boundary(level = "state", name)
```

Arguments

level	"state" (default) or "district".
name	State or district name (partial match allowed).

Value

An sf object with the matching boundary.

Examples

```
goa       <- get_boundary("state", "Goa")
north_goa <- get_boundary("district", "North Goa")
```

`get_data` *Download and read IMD gridded data*

Description

Downloads binary .grd files from IMD Pune and converts them to terra SpatRaster objects. Single year returns a SpatRaster directly. Multi-year returns a named list of SpatRasters (one per year) because leap and non-leap years have different layer counts.

Usage

```
get_data(
  variable,
  start_yr,
  end_yr,
  file_dir,
  overwrite = FALSE,
  parallel = FALSE,
  workers = 4
)
```

Arguments

`variable`	One of "rain", "tmax", "tmin".
`start_yr`	Start year (rain: 1901+, temp: 1951+).
`end_yr`	End year.
`file_dir`	Directory to save downloaded .grd files.
`overwrite`	Re-download even if file exists? Default FALSE.
`parallel`	Download years in parallel? Default FALSE.
`workers`	Number of parallel workers when parallel = TRUE. Default 4.

Details

Set `parallel = TRUE` to download multiple years concurrently (requires the **future** and **future.apply** packages). Only the network download is parallelized; reading/rasterizing stays sequential because terra pointers are not safe across processes.

Value

A SpatRaster (single year) or named list of SpatRasters (multi-year).

Examples

```
# Download single year rainfall
rain2020 <- get_data("rain", 2020, 2020, tempdir())

# Download multiple years (returns named list)
rain_3yr <- get_data("rain", 2018, 2020, tempdir())

# Parallel download of a long range
rain_hist <- get_data("rain", 2000, 2020, tempdir(),
                      parallel = TRUE, workers = 4)
```

`get_district_series` *Extract a district or state spatial-mean daily time series*

Description

Returns a tidy daily time series of the spatial mean across all grid cells within a named state or district. This is the correct way to summarise gridded data over an area: values are averaged across space (cells sample the same region), giving one value per day. To obtain a seasonal or annual total for rainfall, sum the returned daily values.

Usage

```
get_district_series(
  variable,
  name,
  start,
  end,
  file_dir,
  level = "district",
  touches = FALSE,
  save_csv = TRUE,
  plot = FALSE,
  realtime = FALSE
)
```

Arguments

`variable`	One of "rain", "tmax", "tmin".
`name`	State or district name (partial match allowed).
`start`	Start of period. For archive data, a year (e.g. 2018) or date string. For real-time data, a date string "YYYY-MM-DD".
`end`	End of period. Same format as start.
`file_dir`	Directory for downloaded data and output.
`level`	"district" (default) or "state".

touches	Logical. If TRUE, include every grid cell the boundary overlaps, not only those whose centre falls inside. Useful for small regions and for the coarse 1.0 degree temperature grid. Default FALSE.
save_csv	Save the series as CSV? Default TRUE.
plot	Produce a publication-quality line plot? Default FALSE.
realtime	Use provisional real-time daily data instead of the archive? Default FALSE. When TRUE, start and end must be date strings "YYYY-MM-DD".

Details

Works with both the quality-controlled archive (default) and the provisional real-time daily data (realtime = TRUE).

Note on resolution: archive temperature is 1.0 degree (~111 km cells), so small states or districts may contain no grid-cell centre. In that case the result is all NA; set touches = TRUE to capture every cell the boundary overlaps.

Value

Invisible tidy data frame with columns date, region, and the variable (daily spatial mean).

Examples

```
# Archive: Karnal district rainfall, multi-year daily series
df <- get_district_series("rain", "Karnal",
                          start = 2018, end = 2020,
                          file_dir = tempdir())
head(df)

# Seasonal total for rainfall = sum of daily means
sum(df$rain, na.rm = TRUE)

# State-level temperature series with a plot.
# Goa is smaller than a 1.0 degree temperature cell, so touches = TRUE
# is needed to capture overlapping cells.
get_district_series("tmax", "Goa",
                    start = 2020, end = 2020,
                    level = "state",
                    file_dir = tempdir(),
                    touches = TRUE,
                    plot = TRUE)

# Real-time: North Tripura rainfall, this month to date
get_district_series("rain", "North Tripura",
                    start = "2026-06-01", end = "2026-06-25",
                    file_dir = tempdir(),
                    realtime = TRUE, touches = TRUE)
```

`get_point` *Extract daily time series for a single variable at a point*

Description

Extract daily time series for a single variable at a point

Usage

```
get_point(lat, lon, variable, start_yr, end_yr, file_dir, save_csv = TRUE)
```

Arguments

`lat`	Latitude in decimal degrees.
`lon`	Longitude in decimal degrees.
`variable`	One of "rain", "tmax", "tmin".
`start_yr`	Start year.
`end_yr`	End year.
`file_dir`	Directory for .grd files.
`save_csv`	Save output as CSV? Default TRUE.

Value

Invisible data frame with columns date, lat, lon, variable.

Examples

```
# Extract daily rainfall at Panaji, Goa
df <- get_point(lat = 15.5, lon = 73.8,
                variable = "rain",
                start_yr = 2020, end_yr = 2020,
                file_dir = tempdir())
head(df)

# Extract temperature
df_tmax <- get_point(lat = 15.5, lon = 73.8,
                     variable = "tmax",
                     start_yr = 2020, end_yr = 2020,
                     file_dir = tempdir())
```

get_points *Extract daily time series for one variable at many points*

Description

Reads the raster once and extracts all locations in a single pass with grid-cell deduplication. Far faster than looping get_point() for many coordinates.

Usage

```
get_points(
  points,
  variable,
  start_yr,
  end_yr,
  file_dir,
  format = "long",
  download = TRUE,
  dedup = TRUE,
  save_csv = TRUE
)
```

Arguments

points	Data frame with columns lat, lon, optionally name.
variable	One of "rain", "tmax", "tmin".
start_yr	Start year.
end_yr	End year.
file_dir	Directory for .grd files.
format	"long" or "wide". Default "long".
download	Use get_data() if TRUE, open_data() if FALSE. Default TRUE.
dedup	Collapse points sharing a grid cell? Default TRUE.
save_csv	Save output as CSV? Default TRUE.

Value

Invisible data frame in the requested format.

Examples

```
pts <- data.frame(name = c("Panaji", "Margao"),
                  lat = c(15.49, 15.28), lon = c(73.83, 73.99))
df  <- get_points(pts, "rain", 2020, 2020,
                  file_dir = tempdir(), format = "wide")
```

`get_points_all` *Extract all variables (rain, tmax, tmin, DTR) at many points*

Description

Reads each variable's raster once, extracts all points, merges to a single long data frame with diurnal temperature range.

Usage

```
get_points_all(
  points,
  start_yr,
  end_yr,
  file_dir,
  download = TRUE,
  dedup = TRUE,
  save_csv = TRUE
)
```

Arguments

`points`	Data frame with columns lat, lon, optionally name.
`start_yr`	Start year.
`end_yr`	End year.
`file_dir`	Directory for .grd files.
`download`	Passed to get_points(). Default TRUE.
`dedup`	Passed to get_points(). Default TRUE.
`save_csv`	Save merged CSV? Default TRUE.

Value

Invisible data frame: date, name, lat, lon, rain, tmax, tmin, dtr.

Examples

```
pts <- data.frame(name = c("Panaji", "Margao"),
                  lat = c(15.49, 15.28), lon = c(73.83, 73.99))
df  <- get_points_all(pts, 2020, 2020, file_dir = tempdir())
```

`get_point_all` *Extract daily time series for all variables at a point*

Description

Downloads or reads rain, tmax, and tmin at a location and merges them into a single data frame that also includes diurnal temperature range (DTR). Extraction is done year by year to avoid memory issues with long time series on Windows.

Usage

```
get_point_all(lat, lon, start_yr, end_yr, file_dir, save_csv = TRUE)
```

Arguments

`lat`	Latitude in decimal degrees.
`lon`	Longitude in decimal degrees.
`start_yr`	Start year.
`end_yr`	End year.
`file_dir`	Directory for .grd files.
`save_csv`	Save merged output as CSV? Default TRUE.

Value

Invisible data frame with columns date, lat, lon, rain, tmax, tmin, dtr.

Examples

```
# Extract rain, tmax, tmin and DTR at Panaji, Goa
df <- get_point_all(lat = 15.5, lon = 73.8,
                    start_yr = 2020, end_yr = 2020,
                    file_dir = tempdir())
head(df)

# Long time series -- works on Windows without memory errors
df <- get_point_all(lat = 15.5, lon = 73.8,
                    start_yr = 1985, end_yr = 2020,
                    file_dir = tempdir())
nrow(df)
```

`get_realtime` *Download IMD real-time (provisional) daily gridded data*

Description

Downloads provisional daily grids from IMD's real-time endpoints, which are updated with roughly a one-day lag. Unlike get_data() (quality-controlled yearly archive), real-time data is PROVISIONAL and values may be revised later by IMD. Use get_data() for research requiring the final quality-controlled dataset.

Usage

```
get_realtime(
  variable,
  start_date,
  end_date = NULL,
  file_dir,
  overwrite = FALSE
)
```

Arguments

`variable`	One of "rain", "tmax", "tmin".
`start_date`	Start date "YYYY-MM-DD".
`end_date`	End date "YYYY-MM-DD". Defaults to start_date.
`file_dir`	Directory to save downloaded .grd files.
`overwrite`	Re-download existing files? Default FALSE.

Details

Note: real-time rainfall is 0.25 degree (135 x 129), but real-time temperature is 0.5 degree (61 x 61) – different from the 1.0 degree archive temperature returned by get_data().

Value

Invisible SpatRaster with one layer per day.

Examples

```
# Yesterday's rainfall
y <- Sys.Date() - 1
r <- get_realtime("rain", as.character(y), file_dir = tempdir())

# This month to date
first <- format(Sys.Date(), "%Y-%m-01")
r <- get_realtime("rain", first, as.character(Sys.Date() - 1),
                  file_dir = tempdir())
```

india_districts *India district boundaries (SOI-approved)*

Description

An sf object with boundaries for 808 Indian districts, sourced from Survey of India (SOI) shapefiles, reprojected to WGS84.

Usage

```
india_districts
```

Format

An sf data frame with 808 rows and columns state_name, district_name, and geometry.

india_states *India state boundaries (SOI-approved)*

Description

An sf object with boundaries for all 36 Indian states and union territories, sourced from Survey of India (SOI) shapefiles, reprojected to WGS84.

Usage

```
india_states
```

Format

An sf data frame with 36 rows and columns state_name and geometry.

list_districts *List district names, optionally filtered by state*

Description

List district names, optionally filtered by state

Usage

```
list_districts(state = NULL)
```

Arguments

state	Character or NULL. Partial match, case-insensitive.

Value

A sorted character vector of district names.

Examples

```
list_districts()
list_districts("Goa")
```

list_states *List all state names in the bundled SOI shapefile*

Description

List all state names in the bundled SOI shapefile

Usage

```
list_states()
```

Value

A sorted character vector of 36 state/UT names.

Examples

```
list_states()
```

open_data *Read cached IMD .grd files from disk*

Description

Read cached IMD .grd files from disk

Usage

```
open_data(variable, start_yr, end_yr, file_dir)
```

Arguments

variable	One of "rain", "tmax", "tmin".
start_yr	Start year.
end_yr	End year.
file_dir	Directory containing the variable sub-folder.

Value

A SpatRaster (single year) or named list (multi-year).

Examples

```
rain2020 <- open_data("rain", 2020, 2020, tempdir())
rain_3yr <- open_data("rain", 2018, 2020, tempdir())
```

plot_imd *Plot a single day of IMD gridded data*

Description

Publication-quality map with SOI boundaries. Supports full-India, state-level, and district-level zoom.

Usage

```
plot_imd(
  imd_raster,
  date,
  variable = "rain",
  level = NULL,
  name = NULL,
  title = NULL,
```

```
  save_path = NULL,
  width = 7,
  height = 8
)
```

Arguments

imd_raster	A SpatRaster or named list from get_data().
date	Date to plot (must match a layer name).
variable	One of "rain", "tmax", "tmin".
level	NULL, "state", or "district" for zoom.
name	State or district name for zoom.
title	Custom title. Auto-generated if NULL.
save_path	File path to save PNG/PDF. NULL = no save.
width	Plot width in inches. Default 7.
height	Plot height in inches. Default 8.

Value

Invisible ggplot2 object.

Examples

```
r <- get_data("rain", 2020, 2020, tempdir())

# Full India map
plot_imd(r, "2020-06-28", "rain")

# Zoom to Kerala
plot_imd(r, "2020-06-28", "rain",
         level = "state", name = "Kerala")

# Zoom to North Goa district
plot_imd(r, "2020-06-28", "rain",
         level = "district", name = "North Goa")

# Save to file
plot_imd(r, "2020-06-28", "rain",
         save_path = file.path(tempdir(), "rain_20200628.png"))
```

`plot_timeseries` *Plot a daily time series with 30-day rolling mean*

Description

Plot a daily time series with 30-day rolling mean

Usage

```
plot_timeseries(
  df,
  variable = "rain",
  title = NULL,
  save_path = NULL,
  width = 10,
  height = 5
)
```

Arguments

`df`	Data frame with columns date and the variable.
`variable`	Column name to plot.
`title`	Plot title. Auto-generated if NULL.
`save_path`	File path to save PNG. NULL = no save.
`width`	Width in inches. Default 10.
`height`	Height in inches. Default 5.

Value

Invisible ggplot2 object.

Examples

```
# Extract point data and plot
df <- get_point(lat = 15.5, lon = 73.8,
                variable = "rain",
                start_yr = 2020, end_yr = 2020,
                file_dir = tempdir(),
                save_csv = FALSE)
plot_timeseries(df, variable = "rain")

# Plot temperature with custom title
df_tmax <- get_point(lat = 15.5, lon = 73.8,
                     variable = "tmax",
                     start_yr = 2020, end_yr = 2020,
                     file_dir = tempdir(),
                     save_csv = FALSE)
```

```
plot_timeseries(df_tmax, variable = "tmax",
                title = "Goa Maximum Temperature 2020")
```

`to_csv` *Extract a daily time series at a point location*

Description

Extracts daily values from an IMD SpatRaster at the nearest grid cell to the specified latitude/longitude and returns a data frame.

Usage

```
to_csv(imd_raster, lat, lon, file_path = NULL)
```

Arguments

`imd_raster`	A terra SpatRaster from get_data().
`lat`	Latitude in decimal degrees (WGS84).
`lon`	Longitude in decimal degrees (WGS84).
`file_path`	Character or NULL. If provided, saves output as CSV.

Value

An invisible data frame with columns date and value.

Examples

```
r  <- get_data("rain", 2020, 2020, tempdir())
df <- to_csv(r, lat = 15.5, lon = 73.8)
head(df)

# Save directly to file
to_csv(r, lat = 15.5, lon = 73.8,
       file_path = file.path(tempdir(), "panaji_rain_2020.csv"))
```

to_geotiff *Save an IMD SpatRaster as a compressed GeoTIFF*

Description

Writes a multi-layer terra SpatRaster to a DEFLATE-compressed, tiled GeoTIFF suitable for use in QGIS, ArcGIS, Python (rasterio), and other spatial software.

Usage

```
to_geotiff(imd_raster, file_path)
```

Arguments

imd_raster	A terra SpatRaster.
file_path	Character. Output .tif file path.

Value

Invisible character: the file path written.

Examples

```
r <- get_data("rain", 2020, 2020, tempdir())
to_geotiff(r, file.path(tempdir(), "rain_2020.tif"))

# Save a boundary-extracted region
goa <- extract_by_boundary(r, "state", "Goa", "rain")
to_geotiff(goa, file.path(tempdir(), "rain_Goa_2020.tif"))
```

to_netcdf *Save an IMD SpatRaster as a CF-1.7 compliant NetCDF file*

Description

Writes a multi-layer terra SpatRaster to a CF-1.7 compliant NetCDF file with correct time, latitude, and longitude dimensions and standard metadata attributes.

Usage

```
to_netcdf(imd_raster, file_path, variable = "rain")
```

Arguments

imd_raster	A terra SpatRaster.
file_path	Character. Output .nc file path.
variable	One of "rain", "tmax", "tmin".

Value

Invisible character: the file path written.

Examples

```
r <- get_data("rain", 2020, 2020, tempdir())
to_netcdf(r, file.path(tempdir(), "rain_2020.nc"), "rain")

# Save a boundary-extracted region
goa <- extract_by_boundary(r, "state", "Goa", "rain")
to_netcdf(goa, file.path(tempdir(), "rain_Goa_2020.nc"), "rain")
```

---

trend_analysis *Mann-Kendall trend analysis with Sen's slope*

Description

Aggregates multi-cell index data to spatial means per year, then performs Mann-Kendall test and Sen's slope estimation.

Usage

```
trend_analysis(
  index_df,
  index_col,
  level = NULL,
  name = NULL,
  file_dir,
  save_csv = TRUE,
  plot = TRUE
)
```

Arguments

index_df	Data frame from compute_rainfall_indices() or compute_temp_indices().
index_col	Column name to analyse (e.g. "total", "dr").
level	Not used in computation; passed to filename.
name	Region name for output filename.
file_dir	Output directory.
save_csv	Save results table as CSV? Default TRUE.
plot	Produce and save a trend plot? Default TRUE.

Value

Invisible data frame with tau, S, pvalue, significance, sens_slope, trend_direction, total_change.

Examples

```
# Download 10 years of rainfall
r   <- get_data("rain", 2011, 2020, tempdir())
idx <- compute_rainfall_indices(r, file_dir = tempdir())

# Trend in annual total rainfall
trend_analysis(idx, index_col = "total",
               file_dir = tempdir())

# Trend in rainy days
trend_analysis(idx, index_col = "dr",
               file_dir = tempdir())

# Region-specific trend
goa_idx <- compute_rainfall_indices(r,
  level = "state", name = "Goa",
  file_dir = tempdir())
trend_analysis(goa_idx, index_col = "total",
               name = "Goa", file_dir = tempdir())

# Temperature trend
tx   <- get_data("tmax", 2011, 2020, tempdir())
tn   <- get_data("tmin", 2011, 2020, tempdir())
tidx <- compute_temp_indices(tx, tn, file_dir = tempdir())
trend_analysis(tidx, index_col = "mean_tmax",
               file_dir = tempdir())
```

Index