

Package: vayuayan (via r-universe)

October 18, 2026

Type Package

Title Air Quality Data from Indian Monitoring Networks and Satellites

Version 0.3.0

Description Download and analyze air pollution data from three sources. The Central Pollution Control Board (CPCB) of India publishes historical and real-time particulate matter and air quality index readings from more than 600 continuous ambient air quality monitoring stations
<<https://airquality.cpcb.gov.in/ccr/>>. Open Air Quality (OAQ) aggregates low-cost sensor networks in India. The Atmospheric Composition Analysis Group at Washington University in St. Louis provides satellite-derived fine particulate matter (PM2.5) estimates at 0.01 degree resolution for 1998-2024
<<https://www.satpm.org>>. Includes station lookup, nearest-station matching, the CPCB National Air Quality Index computed from pollutant concentrations, and regional PM2.5 summaries from 'NetCDF' rasters. An R port of the 'vayuayan' 'Python' package.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0)

Imports cli, dplyr, httr, jsonlite, readxl, sf, stats, terra, tools, utils

Suggests ggplot2, knitr, ncd4, rmarkdown, testthat (>= 3.1.7), tidyterra

VignetteBuilder knitr

URL <https://vayuayan.saketlab.org/R/>,
<https://github.com/saketlab/vayuayanR>

BugReports <https://github.com/saketlab/vayuayanR/issues>

Config/testthat/edition 3

Config/Needs/website exactextractr, ggrepel, here, kableExtra, khroma,
patchwork, purrr, scales, stringr, tidyverse, lwgeom

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-
dev libsqlite3-dev libudunits2-dev

Repository <https://saketlab.r-universe.dev>

Date/Publication 2026-10-18 09:39:57 UTC

RemoteUrl <https://github.com/saketlab/vayuayanR>

RemoteRef HEAD

RemoteSha 796f18b453967e8e1467f48ba1e810f4323aa613

Contents

analyze_station_data	3
aqi_pollutant_name	4
aqi_sub_index	4
clean_station_name	5
compute_aqi	5
cpcb_add_coords	6
cpcb_city_daily	7
cpcb_download_city_data	7
cpcb_download_hourly	8
cpcb_download_raw_station_data	9
cpcb_download_station_data	10
cpcb_get_all_india	10
cpcb_get_city_list	11
cpcb_get_complete_list	12
cpcb_get_file_path	12
cpcb_get_live_aqi_data	13
cpcb_get_live_aqi_data_for_station	14
cpcb_get_nearest_station	15
cpcb_get_nearest_stations	15
cpcb_get_state_list	16
cpcb_get_station_list	17
cpcb_get_system_location	17
cpcb_raw_to_long	18
cpcb_station_daily	18
cpcb_station_daily_aqi	19
cpcb_tidy_daily	20
daily_aqi	20
get_aqi_categories	21
get_aqi_category	22
haversine_distance	22
india_boundary	23

<i>analyze_station_data</i>	3
nearest_stations	24
oaq_fetch_monthly_stations	24
oaq_get_annual	25
oaq_get_city_latest	26
oaq_get_daily	27
oaq_get_hierarchy	27
oaq_get_monthly	28
oaq_get_station_24h	29
oaq_get_stations_latest	29
oaq_list_providers	30
oaq_session	31
oaq_stations_table	31
pm25_clear_cache	32
pm25_download_netcdf	32
pm25_get_point	34
pm25_get_stats	35
pm25_get_stats_by_polygon	36
pm25_list_cached_files	37
pm25_load_netcdf_bbox	38
pm25_map	39
pm25_monthly	39
pm25_read_netcdf_bbox	40
pm25_v5gl04_path	41
stations_to_city_summary	41
stations_to_dataframe	42
vayuayan_cache_dir	43
vayuayan_welcome	43
Index	44

<code>analyze_station_data</code>	<i>Analyze station data</i>
-----------------------------------	-----------------------------

Description

Comprehensive analysis of station data

Usage

`analyze_station_data(data)`

Arguments

data List of cities with nested stations

Value

List with analysis results

Examples

```
analyze_station_data(cpcb_get_all_india())
```

aqi_pollutant_name	<i>NAQI name of a pollutant label</i>
--------------------	---------------------------------------

Description

Match a pollutant label such as "PM2.5 (ug/m3)", "pm25" or "Ozone" to its National AQI name.

Usage

```
aqi_pollutant_name(label)
```

Arguments

label	Character vector of labels.
-------	-----------------------------

Value

Character vector of NAQI names ("PM10", "PM2.5", "NO2", "O3", "CO", "SO2", "NH3", "Pb"), NA for labels that are not NAQI pollutants.

Examples

```
aqi_pollutant_name(c("PM2.5 (ug/m3)", "Ozone", "Benzene"))
```

aqi_sub_index	<i>National AQI sub-index of concentrations</i>
---------------	---

Description

The CPCB National AQI sub-index of one pollutant, by linear interpolation within its breakpoint band.

Usage

```
aqi_sub_index(pollutant, concentration)
```

Arguments

pollutant	A pollutant label (see aqi_pollutant_name()).
concentration	Numeric vector of 24-hour averages (8-hour for CO and O3), in mg/m3 for CO and ug/m3 for the others.

Value

Numeric vector of sub-indices, capped at 500; NA stays NA.

Examples

```
aqi_sub_index("PM2.5", c(25, 45, 300))
```

clean_station_name	<i>Clean station name</i>
--------------------	---------------------------

Description

Convert station name to clean underscore-separated format

Usage

```
clean_station_name(station_name)
```

Arguments

station_name Original station name string

Value

Cleaned station name with underscores

Examples

```
clean_station_name("Bandra Kurla Complex, Mumbai - IITM")
```

compute_aqi	<i>Overall National AQI of a set of concentrations</i>
-------------	--

Description

The overall AQI is the highest sub-index. CPCB requires at least three pollutants, one of them PM2.5 or PM10.

Usage

```
compute_aqi(concentrations)
```

Arguments

concentrations Named numeric vector or list: pollutant label to averaged concentration. Labels that are not NAQI pollutants and missing values are ignored.

Value

List with aqi (integer, NA when the data are insufficient), category, dominant_pollutant and the named sub_indices.

Examples

```
compute_aqi(c("PM2.5" = 45, PM10 = 80, NO2 = 30))$aqi
```

cpcb_add_coords	<i>Add coordinates to station list</i>
-----------------	--

Description

Add lat/lon coordinates to a station data frame. Coordinates come from the bundled CPCB station directory, then from the CPCB live feed for stations newer than the directory, then from Nominatim geocoding of the station name as a last resort.

Usage

```
cpcb_add_coords(stations)
```

Arguments

stations	Data frame from cpcb_get_station_list()
----------	---

Value

Same data frame with lat and lon columns added

Examples

```
stations <- cpcb_get_station_list("Mumbai")
stations_with_coords <- cpcb_add_coords(stations)
head(stations_with_coords)
```

cpcb_city_daily	<i>Daily city-level AQI in long format</i>
-----------------	--

Description

Download (or reuse) the CPCB daily city file for each city and year and return it in long format. CPCB files of past years are cached in `vayuayan_cache_dir()`. With `cache_dir`, each sheet is also saved as `<city>_aqi_<year>.csv` there and read back on later calls.

Usage

```
cpcb_city_daily(city, year, cache_dir = NULL)
```

Arguments

<code>city</code>	Character vector of city names.
<code>year</code>	Integer year.
<code>cache_dir</code>	Optional directory for CSV copies of the sheets.

Value

Data frame with columns `city`, `day`, `month` and `value`.

Examples

```
daily <- cpcb_city_daily(c("Delhi", "Mumbai"), 2022)
aggregate(value ~ city + month, daily, mean)
```

cpcb_download_city_data	<i>Download city-level AQI data</i>
-------------------------	-------------------------------------

Description

Download daily PM2.5 AQI data for a city from the CPCB data repository for a given year. Data is saved as a CSV file with one row per day and one column per month.

Usage

```
cpcb_download_city_data(city, year, save_location)
```

Arguments

<code>city</code>	Character. City name (e.g. "Delhi", "Mumbai").
<code>year</code>	Integer. Year for data (e.g. 2022).
<code>save_location</code>	Character. Full file path to save the downloaded CSV.

Value

Data frame preview (first 6 rows) of the downloaded data. The full data is saved to save_location.

Examples

```
cpcb_download_city_data(
  city      = "Delhi",
  year      = 2022,
  save_location = tempfile(fileext = ".csv")
)
```

cpcb_download_hourly	<i>Download hourly AQI data</i>
----------------------	---------------------------------

Description

Download CPCB’s hourly AQI workbooks for a city or a station and combine the months of a year.

Usage

```
cpcb_download_hourly_city_data(city, year, save_location = "", month = NULL)

cpcb_download_hourly_station_data(
  station_id,
  year,
  save_location = "",
  month = NULL
)
```

Arguments

city	Character. City name.
year	Integer year.
save_location	Character. CSV path to save the result to; "" saves nothing.
month	Optional month name (e.g. "January"). If NULL, all months of the year are combined.
station_id	Character. Station ID (e.g. "site_301") or name. Part of a name is enough when it picks one station (e.g. "Anand Vihar, Delhi").

Value

Data frame of hourly AQI with Year and Month columns.

Examples

```
delhi_jan <- cpcb_download_hourly_city_data("Delhi", 2024, month = "January")
```

cpcb_download_raw_station_data
<i>Download raw CPCB station data</i>

Description

Download a station’s raw CPCB file for a year, for example 15-minute observations of every pollutant and weather variable.

Usage

```
cpcb_download_raw_station_data(  
  station_id,  
  year,  
  save_location = "",  
  frequency = "15_minute",  
  tidy = FALSE  
)
```

Arguments

station_id	Character. Station ID (e.g. "site_301") or name. Part of a name is enough when it picks one station (e.g. "Anand Vihar, Delhi").
year	Integer year.
save_location	Character. CSV path to save the result to; "" saves nothing.
frequency	Character. Raw interval: "15_minute" (default), "hourly" or "daily".
tidy	Logical. If TRUE, return (and save) long rows from cpcb_raw_to_long() instead of CPCB’s wide table.

Value

Data frame of observations.

Examples

```
raw <- cpcb_download_raw_station_data("site_301", 2024, tidy = TRUE)
```

cpcb_download_station_data

Download station-level AQI data

Description

Download daily PM2.5 AQI data for a single CPCB monitoring station for a given year. Data is saved as a CSV file with one row per day and one column per month.

Usage

```
cpcb_download_station_data(station_id, year, save_location)
```

Arguments

station_id	Character. Station ID (e.g. "site_301") or name. Part of a name is enough when it picks one station (e.g. "Anand Vihar, Delhi").
year	Integer. Year for data (e.g. 2022).
save_location	Character. Full file path to save the downloaded CSV.

Value

Data frame preview (first 6 rows) of the downloaded data. The full data is saved to save_location.

Examples

```
cpcb_download_station_data(
  station_id = "Bandra Kurla Complex, Mumbai - MPCB",
  year       = 2022,
  save_location = tempfile(fileext = ".csv")
)
```

cpcb_get_all_india

Get all India monitoring stations

Description

Get all air quality monitoring stations in India with their latest readings, from the public CPCB station feed.

Usage

```
cpcb_get_all_india()
```

Value

Data frame of cities (cityName, cityID, stateID) with a stationsInCity list-column of station data frames (id, name, latitude, longitude, live, avg, lastUpdate, pollutants).

Examples

```
stations <- cpcb_get_all_india()
df <- stations_to_dataframe(stations)
head(df)
```

cpcb_get_city_list	<i>Get list of cities in a state</i>
--------------------	--------------------------------------

Description

Get the cities of a state from the bundled CPCB station directory. Matching is case-insensitive and also accepts CPCB state codes.

Usage

```
cpcb_get_city_list(state)
```

Arguments

state	Character. State name (e.g. "Maharashtra", "Delhi").
-------	--

Value

Character vector of city names

Examples

```
cities <- cpcb_get_city_list("Maharashtra")
head(cities)
```

`cpcb_get_complete_list`*Fetch complete station and city list from CPCB*

Description

List every state, city and station in the CPCB station directory bundled with the package. Needs no internet connection.

Usage

```
cpcb_get_complete_list()
```

Value

List with cities (named by state; data frames of label/value) and stations (named by city; data frames of station label and ID value).

Examples

```
all_data <- cpcb_get_complete_list()
names(all_data)
```

`cpcb_get_file_path`*Get file path for CPCB data download*

Description

Get file path for downloading data from CPCB. Used internally by `cpcb_download_city_data()` and `cpcb_download_station_data()`.

Usage

```
cpcb_get_file_path(
  station_id = "",
  station_name = "",
  state = "",
  city = "",
  year = "",
  frequency = "daily",
  data_type = "cityLevel"
)
```

Arguments

station_id	Character. Station ID (e.g. "site_103").
station_name	Character. Station name.
state	Character. State name.
city	Character. City name.
year	Integer or character. Year for data (e.g. 2022).
frequency	Character. Data frequency: "daily", "hourly", "monthly" or "15_minute".
data_type	Character. Type of data: "cityLevel", "stationLevel" or "raw".

Value

List of file entries, each with filePath, year and month (NA when CPCB gives none).

Examples

```
paths <- cpcb_get_file_path(
  city = "Delhi", frequency = "daily",
  data_type = "cityLevel"
)
str(paths[[1]])
```

cpcb_get_live_aqi_data

Get live AQI data

Description

Get live AQI data with flexible parameters. Automatically resolves station from coordinates or IP location if not provided.

Usage

```
cpcb_get_live_aqi_data(
  station_id = NULL,
  coords = NULL,
  date = NULL,
  hour = NULL
)
```

Arguments

station_id	Optional character. Station ID or name. If NULL, nearest station is found automatically using coords or IP geolocation.
coords	Optional numeric vector c(lat, lon). Used to find nearest station if station_id is not provided.
date, hour	Ignored; the feed holds only the latest reading.

Value

List as returned by `cpcb_get_live_aqi_data_for_station()`.

Examples

```
# Get live AQI at the current location
aqi <- cpcb_get_live_aqi_data()

# Get live AQI for a specific station
aqi <- cpcb_get_live_aqi_data(station_id = "site_103")
```

```
cpcb_get_live_aqi_data_for_station
```

Get live AQI data for a station

Description

Get the latest AQI reading for a station from the public CPCB station feed.

The feed holds only the latest hour, so `date_time` is ignored.

Usage

```
cpcb_get_live_aqi_data_for_station(station_id, date_time = NULL)
```

Arguments

<code>station_id</code>	Character. Station ID (e.g. "site_301") or name. Part of a name is enough when it picks one station (e.g. "Anand Vihar, Delhi").
<code>date_time</code>	Ignored; the feed holds only the latest reading.

Value

List with `station_id`, `station_name`, `lastUpdate`, `airQualityIndex` and a metrics data frame of pollutant readings (`name`, `avg`, `min`, `max`, `avgDesc`).

Examples

```
aqi <- cpcb_get_live_aqi_data_for_station(station_id = "site_103")
```

`cpcb_get_nearest_station`*Find nearest monitoring station*

Description

Find nearest CPCB monitoring station to given coordinates. If no coordinates provided, uses IP-based geolocation automatically. Returns station ID, name, and distance in kilometers.

Usage

```
cpcb_get_nearest_station(coords = NULL)
```

Arguments

`coords` Optional numeric vector `c(lat, lon)`. If `NULL`, uses IP location.

Value

List with `station_id`, `station_name`, and `distance_km`

Examples

```
# Auto-detect location
nearest <- cpcb_get_nearest_station()
cat(
  "Nearest station:", nearest$station_name,
  "(", round(nearest$distance_km, 1), "km)"
)

# Provide coordinates manually (Mumbai)
nearest <- cpcb_get_nearest_station(coords = c(19.076, 72.877))
```

`cpcb_get_nearest_stations`*Find the nearest monitoring stations*

Description

The `k` CPCB stations in the live feed nearest to a location, optionally within a radius.

Usage

```
cpcb_get_nearest_stations(coords = NULL, k = 5, max_distance_km = Inf)
```

Arguments

`coords` Optional numeric vector `c(lat, lon)`. If `NULL`, uses IP location.

`k` Integer. Number of stations to return.

`max_distance_km` Numeric. Only stations within this distance.

Value

Data frame sorted by distance with `station_id`, `station_name`, `city`, `state`, `latitude`, `longitude`, `live`, `avg` and `distance_km`. It has no rows when no station is in range.

Examples

```
# 5 nearest stations to central Delhi
cpcb_get_nearest_stations(c(28.6139, 77.2090), k = 5)

# nearest station within 10 km, if any
cpcb_get_nearest_stations(c(28.6139, 77.2090), k = 1, max_distance_km = 10)
```

`cpcb_get_state_list` *Get list of states with AQI data*

Description

Get the states in the CPCB station directory bundled with the package. Needs no internet connection.

Usage

```
cpcb_get_state_list()
```

Value

Character vector of state names

Examples

```
states <- cpcb_get_state_list()
head(states)
```

cpcb_get_station_list *Get station list for a city*

Description

Get the monitoring stations of a city from the bundled CPCB station directory.

Usage

```
cpcb_get_station_list(city)
```

Arguments

city Character. City name (e.g. "Delhi", "Mumbai").

Value

Data frame with columns label (station name) and value (station ID)

Examples

```
stations <- cpcb_get_station_list("Delhi")
head(stations)
```

cpcb_get_system_location

Get system location via IP geolocation

Description

Get approximate system location using IP-based geolocation. Returns city-level coordinates based on the machine's public IP address.

Usage

```
cpcb_get_system_location()
```

Value

Numeric vector c(lat, lon)

Examples

```
loc <- cpcb_get_system_location()
cat("Latitude:", loc["lat"], "Longitude:", loc["lon"])
```

cpcb_raw_to_long	<i>Reshape raw CPCB station data to long format</i>
------------------	---

Description

Turn a raw CPCB station file, with a Timestamp column and one "<name> (<unit>)" column per pollutant, into one row per time and pollutant.

Usage

```
cpcb_raw_to_long(raw, station = NULL, start_date = NULL, end_date = NULL)
```

Arguments

- raw Data frame as returned by cpcb_download_raw_station_data().
- station Optional named list with station_id, station_name, city and state for every row.
- start_date, end_date Optional first and last dates to keep (inclusive).

Value

Data frame with columns station_id, station_name, city, state, timestamp (POSIXct, UTC), pollutant, unit and value, sorted by time and pollutant. Rows without a time or a numeric value are dropped.

Examples

```
raw <- data.frame(
  Timestamp = c("2024-01-01 00:00:00", "2024-01-01 00:15:00"),
  "PM2.5 (ug/m3)" = c(158, 160), "CO (mg/m3)" = c(1.6, NA),
  check.names = FALSE
)
cpcb_raw_to_long(raw, station = list(station_id = "site_301"))
```

cpcb_station_daily	<i>Daily station-level AQI in long format</i>
--------------------	---

Description

Download (or reuse) the CPCB daily file for each station and year and return them in one long table. CPCB files of past years are cached in vayuayan_cache_dir(). With cache_dir, each sheet is also saved as <station_id>_<year>.csv there. Stations that are not found or have no data for the year are skipped with a warning.

Usage

```
cpcb_station_daily(station_id, year, cache_dir = NULL)
```

Arguments

station_id	Character vector of station IDs or names, for example the value or label column of <code>cpcb_get_station_list()</code> . Part of a name is enough when it picks one station.
year	Integer year.
cache_dir	Optional directory for CSV copies of the sheets.

Value

Data frame with columns `station_id`, `station_name`, `day`, `month` and `value`.

Examples

```
daily <- cpcb_station_daily(c("Anand Vihar, Delhi", "ITO, Delhi"), 2024)
```

`cpcb_station_daily_aqi`

Daily National AQI of a station from its raw data

Description

Download a station's raw 15-minute data for a year and compute its daily NAQI with `daily_aqi()`.

Usage

```
cpcb_station_daily_aqi(station_id, year, day_ends_at = 24)
```

Arguments

station_id	Character. Station ID (e.g. "site_301") or name. Part of a name is enough when it picks one station (e.g. "Anand Vihar, Delhi").
year	Integer year.
day_ends_at	Hour at which each day's 24 hours end; 16 matches CPCB's 4 PM bulletin.

Value

Data frame as returned by `daily_aqi()`, limited to dates in year.

Examples

```
anand_vihar <- cpcb_station_daily_aqi("Anand Vihar, Delhi", 2024, day_ends_at = 16)
```

cpcb_tidy_daily	<i>Reshape a CPCB daily sheet to long format</i>
-----------------	--

Description

CPCB daily files have one row per day of the month and one column per month. This turns that sheet into one row per day.

Usage

```
cpcb_tidy_daily(x, year = NULL)
```

Arguments

- x Data frame from a CPCB daily file, or the path to a CSV written by `cpcb_download_city_data()` or `cpcb_download_station_data()`.
- year Optional year of the sheet. When given, a date column is added and impossible dates (31 February) are dropped.

Value

Data frame with integer columns `day` and `month`, numeric value, and date when year is given. Days without a reading are dropped.

Examples

```
sheet <- data.frame(Day = 1:2, January = c(180, NA), February = c(150, 140))
cpcb_tidy_daily(sheet)
cpcb_tidy_daily(sheet, year = 2024)
```

daily_aqi	<i>Daily National AQI from tidy data</i>
-----------	--

Description

Daily NAQI of each station from long data, such as 15-minute raw data from `cpcb_raw_to_long()`. Values are first averaged by hour. PM10, PM2.5, NO2, SO2, NH3 and Pb use the daily mean of the hourly values; CO and O3 use the highest 8-hour running mean (of at least 6 hourly values). A sub-index needs data in at least `min_hours` hours of the day.

CPCB’s daily bulletin reports the AQI at 4 PM from the 24 hours before it; `day_ends_at = 16` computes the same window. CPCB validates its data before publishing, so results can differ from the published AQI.

Usage

```
daily_aqi(long, min_hours = .aqi_min_hours, day_ends_at = 24)
```

Arguments

long	Data frame with columns timestamp, pollutant, value, and optionally station_id and unit. Values in ug/m3 or mg/m3 are converted to the NAQI unit of each pollutant.
min_hours	Minimum hours with data for a sub-index (CPCB: 16).
day_ends_at	Hour (1–24) at which each day’s 24 hours end. 24 means calendar days. A row is dated by the day its window ends.

Value

One row per station and date: station_id, date, aqi, category, dominant_pollutant, n_pollutants and one <pollutant>_sub_index column per pollutant with data. aqi, category and dominant_pollutant are NA when the data are insufficient.

Examples

```
hours <- seq(as.POSIXct("2024-01-01", tz = "UTC"), by = "hour", length.out = 24)
long <- data.frame(
  timestamp = rep(hours, 3),
  pollutant = rep(c("PM2.5", "PM10", "NO2"), each = 24),
  value = rep(c(45, 80, 30), each = 24)
)
daily_aqi(long)
```

get_aqi_categories	<i>Vectorized AQI category</i>
--------------------	--------------------------------

Description

Convert vector of AQI values to categories

Usage

```
get_aqi_categories(aqi_values)
```

Arguments

aqi_values	Numeric vector of AQI values
------------	------------------------------

Value

Character vector of AQI categories

Examples

```
get_aqi_categories(c(42, 180, NA, 450))
```

get_aqi_category	<i>Get AQI category</i>
------------------	-------------------------

Description

Convert AQI numeric value to category

Usage

```
get_aqi_category(aqi_value)
```

Arguments

aqi_value	Numeric AQI value
-----------	-------------------

Value

AQI category string

Examples

```
get_aqi_category(150)
```

haversine_distance	<i>Calculate haversine distance</i>
--------------------	-------------------------------------

Description

Calculate great circle distance between two points

Usage

```
haversine_distance(lat1, lon1, lat2, lon2)
```

Arguments

lat1	Latitude of first point
lon1	Longitude of first point
lat2	Latitude of second point
lon2	Longitude of second point

Value

Distance in kilometers

Examples

```
# Delhi to Mumbai
haversine_distance(28.61, 77.21, 19.08, 72.88)
```

india_boundary	<i>Boundary of Indian districts or states</i>
----------------	---

Description

Union of BharatViz LGD district polygons, selected by district or state name. The district layer is downloaded once per session.

Usage

```
india_boundary(district = NULL, state = NULL)
```

Arguments

district	Character vector of district names (column district_name).
state	Character vector of state names (column state_name). Used when district is NULL.

Value

An sfc polygon in EPSG:4326.

Source

https://geo.bharatviz.org/geojsons/districts/India_LGD_districts.geojson

Examples

```
mumbai <- india_boundary(district = c("Mumbai", "Mumbai Suburban"))
delhi <- india_boundary(state = "Delhi")
```

nearest_stations	<i>Find the nearest stations between two sets of points</i>
------------------	---

Description

For every row in `from` (e.g. low-cost sensors), find the `k` closest rows in `to` (e.g. reference stations) by haversine distance. Useful for pairing sensor networks for calibration or cross-validation.

Usage

```
nearest_stations(from, to, k = 1, from_id = "id", to_id = "id")
```

Arguments

<code>from</code>	Data frame of source points. Must contain lat/lon.
<code>to</code>	Data frame of candidate target points. Must contain lat/lon.
<code>k</code>	Integer number of nearest targets to return per source point.
<code>from_id, to_id</code>	Column names identifying each point, carried into the output as <code>from_id/to_id</code> .

Value

A data frame with one row per (source, nearest target) pair: `from_id`, `to_id`, `rank` (1 = closest), and `dist_km`.

Examples

```
sensors <- data.frame(id = c("a", "b"), lat = c(28.61, 19.08), lon = c(77.21, 72.88))
stations <- data.frame(
  id = c("delhi", "mumbai", "pune"),
  lat = c(28.63, 19.07, 18.52), lon = c(77.22, 72.87, 73.86)
)
nearest_stations(sensors, stations, k = 2)
```

oag_fetch_monthly_stations	<i>Fetch monthly station data for a provider across one or more months</i>
----------------------------	--

Description

Pulls the station list for provider, optionally restricts it to one or more cities, then fetches monthly aggregates for every station across the requested months. Returns a single tidy data frame with station metadata (`id`, `name`, `lat`, `lon`) joined to each daily record. Stations that return no data for a month are skipped.

Usage

```
oaq_fetch_monthly_stations(
  session,
  provider,
  year,
  months = sprintf("%02d", 1:12),
  city = NULL
)
```

Arguments

session	List returned by oaq_session .
provider	Character vector of provider ids (e.g. "cpcb", "airnet").
year	Integer year (e.g. 2026).
months	Character vector of zero-padded months (e.g. c("03", "04")). Defaults to all twelve months.
city	Optional character vector of city names to keep (matched against the station list's city column). NULL keeps every city.

Value

A data frame of daily records with provider, station_id, date (class Date), the pollutant columns returned by the API as numbers, and lat/lon/ name from the station list.

Examples

```
session <- oaq_session()
blr <- oaq_fetch_monthly_stations(
  session, c("cpcb", "airnet"),
  year = 2026, months = c("03", "04"), city = "Bengaluru"
)
```

oaq_get_annual	<i>Get annual aggregated data for a sensor</i>
----------------	--

Description

Retrieve annual aggregated air quality data for a specific sensor and year.

Usage

```
oaq_get_annual(session, provider, sensor_id, year)
```

Arguments

session	List returned by oaq_session().
provider	Character. Provider name (e.g. "AURASSURE").
sensor_id	Character. Sensor station ID.
year	Integer or character. Year (e.g. 2024).

Value

Parsed annual aggregate record.

Examples

```
session <- oaq_session()
annual <- oaq_get_annual(session, "AURASSURE", "sensor_001", 2024)
```

oaq_get_city_latest	<i>Get latest AQI for a city</i>
---------------------	----------------------------------

Description

Get the latest air quality map readings for all sensors in a city.

Usage

```
oaq_get_city_latest(session, provider, city)
```

Arguments

session	List returned by oaq_session().
provider	Character. Provider name (e.g. "AURASSURE").
city	Character. City name as it appears in the provider hierarchy.

Value

Parsed list of sensor readings for the city.

Examples

```
session <- oaq_session()
latest <- oaq_get_city_latest(session, "AURASSURE", "Bengaluru")
```

oaq_get_daily	<i>Get daily historical data for a sensor</i>
---------------	---

Description

Retrieve daily aggregated air quality data for a specific sensor, date.

Usage

```
oaq_get_daily(session, provider, sensor_id, year, month, day)
```

Arguments

- session List returned by oaq_session().
- provider Character. Provider name (e.g. "AURASSURE").
- sensor_id Character. Sensor station ID.
- year Integer or character. Year (e.g. 2024).
- month Integer or character. Month (1–12).
- day Integer or character. Day of month (1–31).

Value

Parsed daily data record.

Examples

```
session <- oaq_session()
daily <- oaq_get_daily(session, "AURASSURE", "sensor_001", 2024, 1, 15)
```

oaq_get_hierarchy	<i>Get provider geographic hierarchy</i>
-------------------	--

Description

Get the Country > City > Sensor geographical tree for a data provider.

Usage

```
oaq_get_hierarchy(session, provider)
```

Arguments

- session List returned by oaq_session().
- provider Character. Provider name (e.g. "AURASSURE").

Value

Parsed hierarchy list.

Examples

```
session <- oaq_session()
hierarchy <- oaq_get_hierarchy(session, "AURASSURE")
```

oaq_get_monthly	<i>Get monthly aggregated data for a sensor</i>
-----------------	---

Description

Retrieve monthly aggregated air quality data for a specific sensor and month.

Usage

```
oaq_get_monthly(session, provider, sensor_id, year, month)
```

Arguments

- session List returned by oaq_session().
- provider Character. Provider name (e.g. "AURASSURE").
- sensor_id Character. Sensor station ID.
- year Integer or character. Year (e.g. 2024).
- month Integer or character. Month (1–12).

Value

Parsed monthly aggregate record.

Examples

```
session <- oaq_session()
monthly <- oaq_get_monthly(session, "AURASSURE", "sensor_001", 2024, 1)
```

oaq_get_station_24h	<i>Get last 24 hours of data for a sensor</i>
---------------------	---

Description

Retrieve the last 24 hours of readings for a specific sensor station.

Usage

```
oaq_get_station_24h(session, provider, sensor_id)
```

Arguments

session	List returned by oaq_session().
provider	Character. Provider name (e.g. "AURASSURE").
sensor_id	Character. Sensor station ID.

Value

Parsed list of hourly readings.

Examples

```
session <- oaq_session()
data <- oaq_get_station_24h(session, "AURASSURE", "sensor_001")
```

oaq_get_stations_latest	<i>Get latest readings from all stations</i>
-------------------------	--

Description

Get the latest sensor readings from all monitoring stations under a provider.

Usage

```
oaq_get_stations_latest(session, provider)
```

Arguments

session	List returned by oaq_session().
provider	Character. Provider name (e.g. "AURASSURE").

Value

Parsed list of station-level latest readings.

Examples

```
session <- oaq_session()
stations <- oaq_get_stations_latest(session, "AURASSURE")
```

oaq_list_providers	<i>List all OAQ data providers</i>
--------------------	------------------------------------

Description

Retrieve the list of all available data providers on the OAQ platform, including their IDs, display names, and data attribution.

Usage

```
oaq_list_providers(session)
```

Arguments

session	List returned by oaq_session().
---------	---------------------------------

Value

List with a providers element containing id, name, and attribution for each provider.

Examples

```
session <- oaq_session()
providers <- oaq_list_providers(session)
sapply(providers$providers, `[`, "id")
```

oaq_session	<i>Establish an OAQ API session</i>
-------------	-------------------------------------

Description

Performs the OAQ 2-step handshake: calls the Data Broker with your API key and returns the signed-URL parameters needed for all subsequent data requests. Sessions are valid for 24 hours.

The API key is read from the OAQ_API_KEY environment variable if api_key is not supplied.

Usage

```
oaq_session(api_key = Sys.getenv("OAQ_API_KEY"))
```

Arguments

api_key Character. Your OAQ API key. Defaults to Sys.getenv("OAQ_API_KEY").

Value

Named list with base_url and sig_string (the full query-string to append to every data URL).

Examples

```
session <- oaq_session()
```

oaq_stations_table	<i>Latest OAQ readings for one or more providers</i>
--------------------	--

Description

Combine oaq_get_stations_latest() across providers into one table with numeric columns, dropping stations without coordinates.

Usage

```
oaq_stations_table(session, provider)
```

Arguments

session List returned by [oaq_session](#).

provider Character vector of provider ids.

Value

Data frame of stations with a provider column and numeric lat, lon and pollutant columns.

Examples

```
session <- oaq_session()
stations <- oaq_stations_table(session, c("cpcb", "airnet"))
```

pm25_clear_cache	<i>Clear PM2.5 cache</i>
------------------	--------------------------

Description

Delete all cached NetCDF files from the cache directory to free disk space.

Usage

```
pm25_clear_cache(cache_dir = NULL)
```

Arguments

cache_dir	Character. Directory containing cached files. NULL uses the pm25 folder of vayuayan_cache_dir().
-----------	--

Value

Invisibly returns NULL.

Examples

```
pm25_clear_cache()
```

pm25_download_netcdf	<i>Download PM2.5 NetCDF file</i>
----------------------	-----------------------------------

Description

Download a WUSTL ACAG SatPM2.5 NetCDF file from the AWS Open Data registry and cache it locally. A cached file larger than 1 MB is reused without downloading it again.

Use region = "AS" for work within Asia, including India; its files are much smaller than the global ones.

Usage

```
pm25_download_netcdf(
  year,
  month = NULL,
  cache_dir = NULL,
  force_download = FALSE,
  region = "GL",
  version = "V6GL03"
)
```

Arguments

<code>year</code>	Integer. Year for data, within the range of version.
<code>month</code>	Optional integer (1–12). If NULL, downloads the annual file.
<code>cache_dir</code>	Character. Directory to cache NetCDF files. NULL uses the pm25 folder of <code>vayuayan_cache_dir()</code> , and also reads files from the Python package's cache.
<code>force_download</code>	Logical. Download again even if the file is cached.
<code>region</code>	Character. Spatial subset: "GL" (global, default), "AS" (Asia), "EU", "NA", or "AF" (V6GL03 only).
<code>version</code>	Character. Dataset version: "V6GL03" V6.GL.03, CNN, 1998–2024 (default). "V6" V6.GL.02.04, CNN, 1998–2023. "V5" V5.GL.05.02, GWR, 1998–2023. "V5GL04" V5.GL.04, GWR, 1998–2022.

Value

Character. Path to the downloaded (or cached) NetCDF file.

Source

<https://www.satpm.org/v6-gl-03>

Examples

```
# Monthly Asia subset
nc_path <- pm25_download_netcdf(year = 2023, month = 11, region = "AS")

# The older GWR product
nc_path <- pm25_download_netcdf(2022, 11, region = "AS", version = "V5GL04")
```

pm25_get_point	<i>PM2.5 at points</i>
----------------	------------------------

Description

Satellite PM2.5 of the grid cell that contains each point.

Usage

```
pm25_get_point(
  latitude,
  longitude,
  year = NULL,
  month = NULL,
  version = "V6GL03",
  region = "GL",
  cache_dir = NULL,
  nc_file = NULL
)
```

Arguments

latitude, longitude	Numeric vectors of coordinates in decimal degrees.
year	Integer year.
month	Optional integer (1–12). If NULL, uses the annual file.
version, region, cache_dir	Passed to <code>pm25_download_netcdf()</code> .
nc_file	Optional path to a local NetCDF file to use instead of a download.

Value

Numeric vector of PM2.5 in ug/m3, NA where the grid has no value.

Examples

```
pm25_get_point(28.6139, 77.2090, year = 2023, month = 11, region = "AS")
```

pm25_get_stats	<i>Get PM2.5 statistics for a region</i>
----------------	--

Description

Compute mean, SD, min, max, and pixel count of satellite PM2.5 for a region defined by a GeoJSON file. Optionally group by a column in the GeoJSON (e.g. district name) to return per-group statistics.

Usage

```
pm25_get_stats(
    geojson_file,
    year = NULL,
    month = NULL,
    cache_dir = NULL,
    group_by = NULL,
    nc_file = NULL,
    region = "GL",
    version = "V6GL03"
)
```

Arguments

geojson_file	Character. Path or URL to a GeoJSON file defining the region (remote <code>http(s)://</code> URLs are read directly).
year	Integer. Year of the NetCDF data (e.g. 2023). Ignored when <code>nc_file</code> is supplied.
month	Optional integer (1–12). If <code>NULL</code> , uses annual data. Ignored when <code>nc_file</code> is supplied.
cache_dir	Character. Directory to cache NetCDF files. <code>NULL</code> uses the <code>pm25</code> folder of <code>vayuayan_cache_dir()</code> , and also reads files from the Python package's cache.
group_by	Optional character. Column name(s) in the GeoJSON to group polygons by (comma-separated for multiple columns).
nc_file	Optional character. Path to a local NetCDF file to use directly. When supplied, no download is attempted and <code>year/ month</code> are ignored. Useful for offline data (e.g. WUSTL V5GL04 files on a local mount).
region	Character. Spatial subset to download, passed to <code>pm25_download_netcdf()</code> . "AS" covers India in a smaller file.
version	Character. Dataset version, passed to <code>pm25_download_netcdf()</code> .

Value

List (or data frame if `group_by` is specified) with PM2.5 statistics: mean, std, min, max, count.

Examples

```
# Single region (remote BharatViz boundary)
stats <- pm25_get_stats(
  "https://geo.bharatviz.org/geojsons/cities/delhi.geojson",
  year = 2023, month = 11
)
cat("Mean PM2.5:", round(stats$mean, 1), "ug/m3")

# Grouped by district
stats_by_district <- pm25_get_stats(
  "https://geo.bharatviz.org/geojsons/districts/India_LGD_districts.geojson",
  year      = 2023,
  month     = 11,
  group_by  = "district_name"
)

# Use a local (offline) NetCDF file directly
stats <- pm25_get_stats(
  "https://geo.bharatviz.org/geojsons/cities/delhi.geojson",
  nc_file = "V5GL04.HybridPM25.Global.202211-202211.nc"
)
```

```
pm25_get_stats_by_polygon
```

PM2.5 statistics for each polygon

Description

Mean and standard deviation of satellite PM2.5 within each feature of a GeoJSON file, over the cells whose centers fall inside it.

Usage

```
pm25_get_stats_by_polygon(
  geojson_file,
  year = NULL,
  month = NULL,
  id_field = NULL,
  version = "V6GL03",
  region = "GL",
  cache_dir = NULL,
  nc_file = NULL
)
```

Arguments

`geojson_file` Path or URL of a GeoJSON file, or an sf object.

year	Integer year.
month	Optional integer (1–12). If NULL, uses the annual file.
id_field	Optional column that names each feature. Defaults to NAME_1, then name, then the row number.
version, region, cache_dir	Passed to pm25_download_netcdf().
nc_file	Optional path to a local NetCDF file to use instead of a download.

Value

Data frame with the identifier column, mean and std.

Examples

```
pm25_get_stats_by_polygon(
  "https://geo.bharatviz.org/geojsons/districts/India_LGD_districts.geojson",
  year = 2023, month = 11, id_field = "district_name", region = "AS"
)
```

pm25_list_cached_files

List cached PM2.5 NetCDF files

Description

List all cached NetCDF files in the cache directory with file sizes.

Usage

```
pm25_list_cached_files(cache_dir = NULL)
```

Arguments

cache_dir	Character. Directory containing cached files. NULL uses the pm25 folder of vayuayan_cache_dir().
-----------	--

Value

Character vector of cached file paths.

Examples

```
files <- pm25_list_cached_files()
```

pm25_load_netcdf_bbox *Load a gridded PM2.5 raster cropped to a region's bounding box*

Description

Reads the pixels covering shape's bounding box and returns them as a terra raster in EPSG:4326, ready for masking or zonal extraction.

Usage

```
pm25_load_netcdf_bbox(
  nc_path,
  shape,
  var = "GWRPM25",
  lon_name = "lon",
  lat_name = "lat",
  mask = FALSE
)
```

Arguments

nc_path	Path to a gridded PM2.5 NetCDF file.
shape	An sf object defining the region.
var	NetCDF variable name. Defaults to "GWRPM25" (WUSTL V5GL04).
lon_name, lat_name	Longitude/latitude dimension names.
mask	Logical. If TRUE, set pixels outside shape to NA.

Value

A terra SpatRaster cropped to the bounding box.

Examples

```
shape <- sf::st_read("https://geo.bharatviz.org/geojsons/cities/delhi.geojson")
r <- pm25_load_netcdf_bbox(pm25_v5gl04_path(2022, 11, "."), shape)
```

pm25_map

*Map a PM2.5 raster***Description**

Plot a PM2.5 raster with an optional boundary outline. The color scale spans the 2nd to 98th percentile of the raster so that outliers do not wash out the map.

Usage

```
pm25_map(r, boundary = NULL, title = NULL)
```

Arguments

<code>r</code>	A terra SpatRaster, for example from <code>pm25_load_netcdf_bbox()</code> .
<code>boundary</code>	Optional sf or sfc outline to draw on top.
<code>title</code>	Optional plot title.

Value

A ggplot object. The subtitle gives the raster mean.

Examples

```
shape <- india_boundary(state = "Delhi")
r <- pm25_load_netcdf_bbox(pm25_v5gl04_path(2022, 11, "."), shape, mask = TRUE)
pm25_map(r, shape, title = "Delhi")
```

pm25_monthly

*Monthly mean PM2.5 over a region***Description**

Mean satellite PM2.5 over the bounding box of shape for each month of a year, downloading the monthly files when they are not in `cache_dir`. A directory that already holds the files under their WUSTL names, such as a local copy of V5GL04, is read without downloading.

Usage

```
pm25_monthly(
  shape,
  year,
  months = 1:12,
  version = "V6GL03",
  region = "GL",
  cache_dir = NULL
)
```

Arguments

shape An sf or sfc object defining the region.
 year Integer year.
 months Integer vector of months.
 version, region, cache_dir
 Passed to pm25_download_netcdf().

Value

Data frame with columns month and pm25.

Examples

```
delhi <- india_boundary(state = "Delhi")
pm25_monthly(delhi, 2022, version = "V5GL04", region = "AS")
```

pm25_read_netcdf_bbox *Mean gridded PM2.5 over a region's bounding box*

Description

Reads only the pixels covering shape's bounding box and returns the mean value. Useful for a quick monthly figure over a city or district.

Usage

```
pm25_read_netcdf_bbox(
  nc_path,
  shape,
  var = "GWRPM25",
  lon_name = "lon",
  lat_name = "lat"
)
```

Arguments

nc_path Path to a gridded PM2.5 NetCDF file.
 shape An sf object defining the region.
 var NetCDF variable name. Defaults to "GWRPM25" (WUSTL V5GL04).
 lon_name, lat_name
 Longitude/latitude dimension names.

Value

Numeric mean value over the bounding box.

Examples

```
shape <- sf::st_read("https://geo.bharatviz.org/geojsons/cities/delhi.geojson")
pm25_read_netcdf_bbox(pm25_v5gl04_path(2022, 11, "."), shape)
```

pm25_v5gl04_path

Build the file path for a WUSTL V5GL04 monthly NetCDF

Description

Build the file path for a WUSTL V5GL04 monthly NetCDF

Usage

```
pm25_v5gl04_path(year, month, base_dir)
```

Arguments

year	Integer year (1998–2022).
month	Integer month (1–12).
base_dir	Directory holding the monthly NetCDF files.

Value

Character path to the monthly NetCDF file.

Examples

```
pm25_v5gl04_path(2022, 11, base_dir = "~/data/V5GL04/Monthly")
```

stations_to_city_summary

Summarize station data by city

Description

One row per city with station counts and the AQI of its live stations.

Usage

```
stations_to_city_summary(data)
```

Arguments

data	Cities with nested stations, from cpcb_get_all_india().
------	---

Value

Data frame with city_name, city_id, state_id, total_stations, live_stations, offline_stations, live_percentage, avg_aqi, min_aqi, max_aqi and stations_with_data.

Examples

```
summary <- stations_to_city_summary(cpcb_get_all_india())  
head(summary[order(-summary$avg_aqi), ])
```

stations_to_dataframe *Convert station data to data frame*

Description

Convert nested station data to a flat data frame with geographic information, AQI values, and categories. This is the primary way to get station coordinates (latitude/longitude) from live CPCB data.

Usage

```
stations_to_dataframe(data)
```

Arguments

data List of cities with nested stations (from cpcb_get_all_india())

Value

Data frame with columns: station_id, station_name, city_name, city_id, state_id, longitude, latitude, live, avg_aqi, status, aqi_category

Examples

```
df <- stations_to_dataframe(cpcb_get_all_india())  
table(df$aqi_category)
```

vayuayan_cache_dir	<i>Cache directory of vayuayan</i>
--------------------	------------------------------------

Description

Downloads are cached under one root, laid out like the Python package's cache: CPCB repository files under cpcb/ by their repository path, and PM2.5 NetCDF files under pm25/ by their WUSTL file names.

The root is the `vayuayan.cache_dir` option, else the `VAYUAYAN_CACHE_DIR` environment variable, else `tools::R_user_dir("vayuayan", "cache")`. Setting `VAYUAYAN_CACHE_DIR` makes R and Python share one cache. Files already in the Python package's default cache (`~/.cache/vayuayan`, or its `pm25_data` folder in the working directory) are read without downloading them again.

Usage

```
vayuayan_cache_dir()
```

Value

Character path of the cache root. It is not created.

Examples

```
vayuayan_cache_dir()
```

vayuayan_welcome	<i>Print vayuayan welcome message</i>
------------------	---------------------------------------

Description

Display package information and usage

Usage

```
vayuayan_welcome()
```

Value

NULL, invisibly. Called for its printed side effect.

Examples

```
vayuayan_welcome()
```

Index

analyze_station_data, [3](#)
aqi_pollutant_name, [4](#)
aqi_sub_index, [4](#)

clean_station_name, [5](#)
compute_aqi, [5](#)
cpcb_add_coords, [6](#)
cpcb_city_daily, [7](#)
cpcb_download_city_data, [7](#)
cpcb_download_hourly, [8](#)
cpcb_download_hourly_city_data
 (cpcb_download_hourly), [8](#)
cpcb_download_hourly_station_data
 (cpcb_download_hourly), [8](#)
cpcb_download_raw_station_data, [9](#)
cpcb_download_station_data, [10](#)
cpcb_get_all_india, [10](#)
cpcb_get_city_list, [11](#)
cpcb_get_complete_list, [12](#)
cpcb_get_file_path, [12](#)
cpcb_get_live_aqi_data, [13](#)
cpcb_get_live_aqi_data_for_station, [14](#)
cpcb_get_nearest_station, [15](#)
cpcb_get_nearest_stations, [15](#)
cpcb_get_state_list, [16](#)
cpcb_get_station_list, [17](#)
cpcb_get_system_location, [17](#)
cpcb_raw_to_long, [18](#)
cpcb_station_daily, [18](#)
cpcb_station_daily_aqi, [19](#)
cpcb_tidy_daily, [20](#)

daily_aqi, [20](#)

get_aqi_categories, [21](#)
get_aqi_category, [22](#)

haversine_distance, [22](#)

india_boundary, [23](#)

nearest_stations, [24](#)

oaq_fetch_monthly_stations, [24](#)
oaq_get_annual, [25](#)
oaq_get_city_latest, [26](#)
oaq_get_daily, [27](#)
oaq_get_hierarchy, [27](#)
oaq_get_monthly, [28](#)
oaq_get_station_24h, [29](#)
oaq_get_stations_latest, [29](#)
oaq_list_providers, [30](#)
oaq_session, [25](#), [31](#), [31](#)
oaq_stations_table, [31](#)

pm25_clear_cache, [32](#)
pm25_download_netcdf, [32](#)
pm25_get_point, [34](#)
pm25_get_stats, [35](#)
pm25_get_stats_by_polygon, [36](#)
pm25_list_cached_files, [37](#)
pm25_load_netcdf_bbox, [38](#)
pm25_map, [39](#)
pm25_monthly, [39](#)
pm25_read_netcdf_bbox, [40](#)
pm25_v5gl04_path, [41](#)

stations_to_city_summary, [41](#)
stations_to_dataframe, [42](#)

vayuayan_cache_dir, [43](#)
vayuayan_welcome, [43](#)