Package 'vayuayan'

November 3, 2026

Type Package

Title Air Quality Data from Indian Monitoring Networks and Satellites

Version 0.3.0

Description Downloads and examines air quality data from three sources. The Central Pollution Control Board (CPCB) of India operates more than 600 monitoring stations. Open Air Quality (OAQ) <https://oaq.notf.in> collects data from low-cost sensor networks in India. The Atmospheric Composition Analysis Group at Washington University in St. Louis gives satellite estimates of fine particulate matter (PM2.5) for all areas of the Earth <https://www.satpm.org>. The package calculates the CPCB National Air Quality Index and the mean PM2.5 in an area.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.0.0)

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

Suggests ggplot2, knitr, ncdf4, patchwork, rmarkdown, scales, testthat (>= 3.1.7), tidyterra

VignetteBuilder knitr

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

BugReports https://github.com/saketlab/vayuayan/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

NeedsCompilation no

Author Saket Choudhary [aut, cre, cph], Mahesh Kumar Prajapat [aut], Darakhshan Nehal [ctb]

Maintainer Saket Choudhary <saketc@iitb.ac.in>

Repository CRAN

Date/Publication 2026-11-03 12:50:02 UTC

Contents

analyze_station_data . . . 4
aqi_health_impact . . . 4
aqi_pollutant_category . . . 5
aqi_pollutant_name . . . 6
aqi_sub_index . . . 6
clean_station_name . . . 7
collocated_precision . . . 7
compare_daily_records . . . 8
compute_aqi . . . 9
compute_us_aqi . . . 9
cpcb_add_coords . . . 10
cpcb_city_daily . . . 11
cpcb_download_city_data . . . 11
cpcb_download_hourly . . . 12
cpcb_download_raw_station_data . . . 13
cpcb_download_station_data . . . 14
cpcb_get_all_india . . . 14
cpcb_get_city_list . . . 15
cpcb_get_complete_list . . . 16
cpcb_get_file_path . . . 16
cpcb_get_live_aqi_data . . . 17
cpcb_get_live_aqi_data_for_station . . . 18
cpcb_get_nearest_station . . . 19
cpcb_get_nearest_stations . . . 19
cpcb_get_state_list . . . 20
cpcb_get_station_list . . . 21
cpcb_get_system_location . . . 21
cpcb_live_metrics . . . 22
cpcb_live_to_long . . . 22
cpcb_raw_readings . . . 23
cpcb_raw_to_long . . . 24
cpcb_station_daily . . . 24
cpcb_station_daily_aqi . . . 25
cpcb_tidy_daily . . . 26
daily_aqi . . . 26
daily_record_summary . . . 27
get_aqi_categories . . . 28
get_aqi_category . . . 28
haversine_distance . . . 29
india_boundary . . . 30
nearest_stations . . . 30

oaq_audit_daily_records . . . 31
oaq_check_daily_records . . . 32
oaq_city_tally . . . 33
oaq_daily_means . . . 33
oaq_daily_records . . . 34
oaq_fetch_monthly_stations . . . 35
oaq_get_annual . . . 36
oaq_get_city_latest . . . 36
oaq_get_daily . . . 37
oaq_get_hierarchy . . . 38
oaq_get_monthly . . . 38
oaq_get_stations_latest . . . 39
oaq_get_station_24h . . . 40
oaq_list_providers . . . 40
oaq_nearest_cpcb . . . 41
oaq_readings . . . 42
oaq_sensors . . . 42
oaq_session . . . 43
oaq_stations_table . . . 43
peer_ratio . . . 44
plot_daily_record_day . . . 45
plot_daily_record_difference . . . 46
plot_daily_record_scatter . . . 46
plot_daily_record_series . . . 47
plot_daily_record_share . . . 48
plot_daily_record_timeline . . . 49
plot_sensor_vs_reference . . . 49
pm25_clear_cache . . . 50
pm25_download_netcdf . . . 51
pm25_get_point . . . 52
pm25_get_stats . . . 53
pm25_get_stats_by_polygon . . . 54
pm25_list_cached_files . . . 55
pm25_load_netcdf_bbox . . . 56
pm25_map . . . 57
pm25_monthly . . . 57
pm25_read_netcdf_bbox . . . 58
pm25_v5gl04_path . . . 59
ratio_by_humidity . . . 60
readings_mean . . . 60
sensor_accuracy . . . 61
sensor_completeness . . . 62
sensor_plausibility . . . 63
stations_to_city_summary . . . 64
stations_to_dataframe . . . 65
theme_vayuayan . . . 66
us_aqi_category . . . 66
us_aqi_color . . . 67

us_aqi_pollutant_name 67
us_aqi_sub_index 68
vayuayan_cache_dir 69
vayuayan_welcome 69

Index **70**

`analyze_station_data` *Analyze station data*

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_health_impact` *Possible health impact of an AQI value*

Description

CPCB's "possible health impacts" text for the category of each AQI value.

Usage

```
aqi_health_impact(aqi_value)
```

Arguments

`aqi_value`	Numeric vector of AQI values.

Value

Character vector, "No Data" for missing values.

Examples

```
aqi_health_impact(c(250, NA))
```

`aqi_pollutant_category`

AQI category of a pollutant concentration

Description

The category of the sub-index of each concentration, for example of a PM2.5 mean from satellite data. The breakpoints are for 24-hour averages (8-hour for CO and O3), so for an annual or monthly mean the category is an indication only.

Usage

```
aqi_pollutant_category(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

Character vector of AQI categories.

Examples

```
aqi_pollutant_category("PM2.5", c(20, 75))
```

aqi_pollutant_name *NAQI name of a pollutant label*

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 *National AQI sub-index of concentrations*

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; above the last breakpoint the slope of the Severe band continues, so a sub-index can be more than 500. NA stays NA.

Examples

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

---

clean_station_name *Clean station name*

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")
```

---

collocated_precision *Precision of collocated sensors*

Description

Spread across collocated sensors on each day, as in the EPA protocol.

Usage

```
collocated_precision(daily, value = "pm25", min_sensors = 3, date_col = "date")
```

Arguments

daily	Daily means of sensors at one place (readings_mean()).
value	Column of the daily means.
min_sensors	Fewest sensors with a mean on a day.
date_col	Day column.

Value

Data frame with `date_col`, `sensors`, `mean`, `sd` (sample standard deviation across the sensors) and `cv_pct`, per day with enough sensors.

Examples

```
daily <- data.frame(
  sensor_id = rep(c("a", "b", "c"), 2), date = rep(as.Date("2026-01-01") + 0:1, each = 3),
  pm25 = c(10, 12, 14, 20, 20, 26)
)
collocated_precision(daily)
```

---

compare_daily_records *Compare OAQ daily records with the day's 15-minute readings*

Description

Each daily record against statistics of the same day's readings: the mean, last, maximum and median reading and the mean of the last 2 and last 4 readings.

Usage

```
compare_daily_records(readings, records, value = "pm25", tolerance = 0.005)
```

Arguments

`readings`	15-minute readings (`oaq_readings()`): `sensor_id`, `timestamp` and the value column.
`records`	Daily records (`oaq_daily_records()`): `sensor_id`, `date` and the value column.
`value`	Parameter to compare.
`tolerance`	Values within it count as equal (0.005: the daily records have 2 decimals).

Value

Data frame with `sensor_id`, `date`, `record`, `n`, the statistics, `record_is_<statistic>` for each, and `class`: `"mean"`, `"last"`, `"both"` (equal to the mean and to the last reading) or `"neither"`. One row per sensor-day with a record and readings; missing readings are left out.

Examples

```
readings <- data.frame(
  sensor_id = "s1",
  timestamp = as.POSIXct("2026-03-10", tz = "UTC") + 900 * 0:3,
  pm25 = c(10, 10, 10, 300)
)
records <- data.frame(sensor_id = "s1", date = as.Date("2026-03-10"), pm25 = 300)
compare_daily_records(readings, records)
```

compute_aqi *Overall National AQI of a set of concentrations*

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
```

compute_us_aqi *Overall US EPA AQI of a set of concentrations*

Description

The US AQI is the highest sub-index, rounded half up. The EPA has no minimum number of pollutants. For O3 and SO2 with both averaging times, the higher sub-index counts. If the 1-hour SO2 is 305 ppb or more and the 24-hour SO2 is below 305 ppb, the SO2 sub-index is 200.

Usage

```
compute_us_aqi(concentrations)
```

Arguments

concentrations	Named numeric vector or list: pollutant label to concentration (see `us_aqi_sub_index()`). Labels that are not US AQI pollutants and missing values are ignored.

Value

List with `aqi` (integer, `NA` when no pollutant has a sub-index), `category`, `dominant_pollutant` and the named `sub_indices`.

Examples

```
compute_us_aqi(c("PM2.5" = 35.4, O3 = 0.060))$aqi
```

`cpcb_add_coords` *Add coordinates to station list*

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 *Daily city-level AQI in long format*

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

`city`	Character vector of city names.
`year`	Integer year.
`cache_dir`	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 *Download city-level AQI data*

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

`city`	Character. City name (e.g. `"Delhi"`, `"Mumbai"`).
`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_city_data(
  city          = "Delhi",
  year          = 2022,
  save_location = tempfile(fileext = ".csv")
)
```

`cpcb_download_hourly` *Download hourly AQI data*

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` *Download raw CPCB station data*

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 *Get list of cities in a state*

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 as CPCB sends them, 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 of a station. Without station_id, the station nearest to coords or to the IP-based location is used.

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

`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"`).
`date_time`	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 the CPCB monitoring station nearest to given coordinates, or to the IP-based location when none are given.

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_live_metrics *Latest pollutant averages of live CPCB data*

Description

The `metrics` of live station data as a table with numeric columns.

Usage

```
cpcb_live_metrics(aqi_data)
```

Arguments

`aqi_data`	List from `cpcb_get_live_aqi_data()`.

Value

Data frame with columns `pollutant`, `avg`, `min`, `max` (numeric, `NA` where CPCB has none) and `period`, one row per pollutant.

Examples

```
cpcb_live_metrics(list(metrics = data.frame(
  name = "PM2.5", avg = "80", min = 60, max = "NA", avgDesc = "Over the last 24 hours"
)))
```

cpcb_live_to_long *Recent hours of live CPCB data in long format*

Description

Turn the `last_hours` series of live station data (one series of `date`/`val` points per pollutant) into one row per time and pollutant. The CPCB feed has no recent hours, so the result is empty for feed data.

Usage

```
cpcb_live_to_long(aqi_data, station_id = "")
```

Arguments

`aqi_data`	List from `cpcb_get_live_aqi_data()`.
`station_id`	Optional station ID for every row.

Value

Data frame with columns `station_id`, `timestamp` (POSIXct, UTC), `pollutant` and `value`, sorted by pollutant and time.

Examples

```
cpcb_live_to_long(list(station_id = "site_301", last_hours = list(
  list(name = "PM2.5", data = list(list(date = "2024-01-01 10:00:00", val = 80)))
)))
```

---

cpcb_raw_readings *Raw 15-minute data of several stations*

Description

`cpcb_download_raw_station_data(tidy = TRUE)` for several stations, as one long table. Stations without a raw file for the year are skipped with a warning.

Usage

```
cpcb_raw_readings(station_id, year, pollutants = NULL)
```

Arguments

`station_id`	Character vector of station IDs or names.
`year`	Integer year.
`pollutants`	Optional pollutant names to keep, such as `c("PM2.5", "RH")`.

Value

Data frame as `cpcb_raw_to_long()`.

Examples

```
raw <- cpcb_raw_readings(c("site_1554", "site_162"), 2025, "PM2.5")
```

cpcb_raw_to_long *Reshape raw CPCB station data to long format*

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 *Daily station-level AQI in long format*

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 cpcb_get_station_list(). 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 *Reshape a CPCB daily sheet to long format*

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 *Daily National AQI from tidy data*

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)
```

---

daily_record_summary *When OAQ daily records changed from the mean to the last reading*

Description

Per sensor and year, how many days the daily record equals the mean, the last reading or neither, counting only days whose mean and last reading differ by more than min_gap (so that a record cannot match both by chance), with the first day equal to only the last reading and the last day equal to only the mean before it.

Usage

```
daily_record_summary(audit, min_gap = 1)
```

Arguments

audit	Output of compare_daily_records() or oaq_audit_daily_records().
min_gap	Smallest difference between the mean and the last reading.

Value

Data frame with (provider,) sensor_id, year, days, clear_days, mean, last, neither, last_mean_day and first_last_day.

Examples

```
audit <- data.frame(
  sensor_id = "s", date = as.Date("2026-02-27") + 0:3,
  class = c("mean", "mean", "last", "last"), mean = 30, last = 90
)
daily_record_summary(audit)
```

`get_aqi_categories` *Vectorized AQI category*

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` *Get AQI category*

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` *Calculate haversine distance*

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 *Boundary of Indian districts or states*

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 *Find the nearest stations between two sets of points*

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

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

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)
```

---

oaq_audit_daily_records

Audit OAQ daily records of several sensors

Description

oaq_check_daily_records() for every day of some years of several sensors. Today (in India) is left out by default, as its values are still being written.

Usage

```
oaq_audit_daily_records(
  session,
  sensors,
  years,
  value = "pm25",
  tolerance = 0.005,
  until = NULL
)
```

Arguments

session	List returned by oaq_session.
sensors	Data frame with provider and sensor_id.
years	Integer years.
value	Parameter to compare.
tolerance	Values within it count as equal.
until	First day left out (default: today in India, IST).

Value

Data frame as `compare_daily_records()` with a `provider` column.

Examples

```
audit <- oaq_audit_daily_records(
  oaq_session(), data.frame(provider = "cpcb", sensor_id = "103"), 2026
)
daily_record_summary(audit)
```

`oaq_check_daily_records` *Check OAQ's daily records against the 15-minute readings*

Description

Fetches the daily records and the 15-minute readings of the days and calls `compare_daily_records()`.

Usage

```
oaq_check_daily_records(
  session,
  provider,
  sensor_id,
  dates,
  value = "pm25",
  tolerance = 0.005
)
```

Arguments

`session`	List returned by `oaq_session`.
`provider`	Character. Provider id (e.g. `"airnet"`).
`sensor_id`	Character. Sensor id, or several.
`dates`	Days, as `Date` or `"YYYY-MM-DD"`.
`value`	Parameter to check.
`tolerance`	Values within it count as equal (0.005: the daily records have 2 decimals).

Value

Data frame as `compare_daily_records()`. Days without a record or readings are left out.

Examples

```
oaq_check_daily_records(oaq_session(), "cpcb", "164", c("2026-05-29", "2026-05-30"))
```

oaq_city_tally *Tally of OAQ-only sensors by city*

Description

Count the sensors that are not OAQ's copy of CPCB, by city and provider, next to the CPCB stations of the city.

Usage

```
oaq_city_tally(sensors)
```

Arguments

sensors	Data frame from `oaq_sensors()`.

Value

Data frame with `city`, `provider`, `sensors`, `cpcb_stations` (CPCB stations of the city in the station directory, 0 where CPCB has none) and `median_km_to_cpcb` (median distance of the sensors to their nearest CPCB station), most sensors first.

Examples

```
sensors <- data.frame(
  id = c("s1", "s2", "s3"), name = c("A", "B", "C"),
  city = c("Bengaluru", "Bengaluru", "Delhi"),
  lat = c(12.99, 12.95, 28.61), lon = c(77.58, 77.6, 77.21),
  provider = c("airnet", "airnet", "cpcb")
)
oaq_city_tally(sensors)
```

oaq_daily_means *Daily means of an OAQ sensor's 15-minute readings*

Description

OAQ's own daily aggregates (`oaq_get_monthly()`, `oaq_get_annual()`) are often the last reading of the day rather than its mean, so this computes the means from the readings of `oaq_get_daily()` instead.

Usage

```
oaq_daily_means(session, provider, sensor_id, dates, min_readings = 64)
```

Arguments

session	List returned by oaq_session.
provider	Character. Provider id (e.g. "airnet").
sensor_id	Character. Sensor id, or several.
dates	Days, as Date or "YYYY-MM-DD".
min_readings	Fewest readings of a parameter for a daily mean; the default, 64, is 16 hours, as for the National AQI.

Value

Data frame with sensor_id, date (Date) and, for each parameter, its mean and its number of readings (n_<parameter>). Means with fewer than min_readings readings are NA. Days that OAQ has no file for are left out.

Examples

```
oaq_daily_means(oaq_session(), "airnet", "12032", c("2026-09-14", "2026-09-15"))
```

---

oaq_daily_records *OAQ's own daily records of a sensor*

Description

The daily values of oaq_get_annual() for several years. These are OAQ's values, which since about March 2026 are often the last reading of the day rather than its mean; see oaq_check_daily_records() and oaq_daily_means().

Usage

```
oaq_daily_records(session, provider, sensor_id, years)
```

Arguments

session	List returned by oaq_session.
provider	Character. Provider id (e.g. "airnet").
sensor_id	Character. Sensor id, or several.
years	Integer years.

Value

Data frame with sensor_id, date (Date) and one numeric column per parameter. Years that OAQ has no file for are left out.

Examples

```
oaq_daily_records(oaq_session(), "cpcb", "164", 2026)
```

`oaq_fetch_monthly_stations` *Fetch monthly station data for a provider across one or more months*

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. 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.

Details

Since about March 2026, OAQ's daily values are often the last 15-minute reading of the day, not its mean; use `oaq_daily_means()` or `readings_mean()` on `oaq_readings()` for daily means.

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 *Get annual aggregated data for a sensor*

Description

Retrieve annual aggregated air quality data for a specific sensor and year. Since about March 2026, OAQ's daily values are often the last 15-minute reading of the day, not its mean; use `oaq_daily_means()` or `readings_mean()` on `oaq_readings()` for daily means.

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, "airnet", "12036", 2026)
```

oaq_get_city_latest *Get latest AQI for a city*

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 in the provider hierarchy (any case).

Value

Parsed list of sensor readings for the city.

Examples

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

`oaq_get_daily` *Get daily historical data for a sensor*

Description

Retrieve daily aggregated air quality data for a sensor and day.

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, "airnet", "12036", 2026, 9, 14)
```

oaq_get_hierarchy *Get provider geographic hierarchy*

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 *Get monthly aggregated data for a sensor*

Description

Retrieve monthly aggregated air quality data for a specific sensor and month. Since about March 2026, OAQ's daily values are often the last 15-minute reading of the day, not its mean; use oaq_daily_means() or readings_mean() on oaq_readings() for daily means.

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, "airnet", "12036", 2026, "09")
```

`oaq_get_stations_latest` *Get latest readings from all stations*

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_get_station_24h *Get last 24 hours of data for a sensor*

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, "airnet", "12036")
```

oaq_list_providers *List all OAQ data providers*

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)
providers$providers$id
```

`oaq_nearest_cpcb` *Nearest CPCB stations of OAQ sensors*

Description

The `k` nearest CPCB stations of each sensor, from the bundled CPCB station directory, so the result does not depend on which stations are online.

Usage

```
oaq_nearest_cpcb(sensors, k = 1)
```

Arguments

`sensors`	Data frame from `oaq_sensors()` (`id`, `name`, `city`, `lat`, `lon`, `provider`).
`k`	Number of CPCB stations per sensor.

Value

Data frame with `provider`, `sensor_id`, `sensor_name`, `city`, `rank`, `cpcb_station_id`, `cpcb_station_name`, `cpcb_city` and `distance_km`, nearest first for each sensor.

Examples

```
sensors <- data.frame(
  id = "s1", name = "Kumara Park", city = "Bengaluru",
  lat = 12.99, lon = 77.58, provider = "airnet"
)
oaq_nearest_cpcb(sensors, k = 2)
```

oaq_readings *15-minute readings of an OAQ sensor*

Description

The readings of `oaq_get_daily()` for several days, as one table.

Usage

```
oaq_readings(session, provider, sensor_id, dates)
```

Arguments

`session`	List returned by `oaq_session`.
`provider`	Character. Provider id (e.g. `"airnet"`).
`sensor_id`	Character. Sensor id, or several.
`dates`	Days, as `Date` or `"YYYY-MM-DD"`.

Value

Data frame with `sensor_id`, `timestamp` (POSIXct of the local wall time, labeled UTC as CPCB data are) and one numeric column per parameter. Days that OAQ has no file for, and readings without a valid time, are left out.

Examples

```
oaq_readings(oaq_session(), "airnet", "12036", c("2026-09-14", "2026-09-15"))
```

oaq_sensors *List OAQ sensors*

Description

Every sensor of the providers, all providers by default. `type` is `"Reference Station"` for the CPCB stations that OAQ mirrors and `"Sensor"` for low-cost sensors.

Usage

```
oaq_sensors(session, provider = NULL)
```

Arguments

`session`	List returned by `oaq_session`.
`provider`	Optional character vector of provider ids; `NULL` lists every provider of `oaq_list_providers()`.

Value

Data frame as `oaq_stations_table()`.

Examples

```
sensors <- oaq_sensors(oaq_session())
table(sensors$provider, sensors$type)
```

`oaq_session` *Establish an OAQ API session*

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.

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` *Latest OAQ readings for one or more providers*

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"))
```

---

peer_ratio *Departure of each collocated sensor from the others*

Description

On days with at least min_sensors sensors, each sensor's daily mean over the median of the other sensors.

Usage

```
peer_ratio(
  daily,
  value = "pm25",
  min_sensors = 3,
  tolerance = 0.3,
  id_col = "sensor_id",
  date_col = "date"
)
```

Arguments

daily	Daily means of sensors at one place (readings_mean()).
value	Column of the daily means.
min_sensors	Fewest sensors with a mean on a day.
tolerance	Ratios within 1 +/- tolerance count as agreeing.
id_col	Sensor column.
date_col	Day column.

Value

Data frame with id_col, days, median_ratio and within_pct, sorted by median_ratio.

Examples

```
daily <- data.frame(
  sensor_id = rep(c("a", "b", "c"), 2), date = rep(as.Date("2026-01-01") + 0:1, each = 3),
  pm25 = c(10, 12, 24, 20, 20, 40)
)
peer_ratio(daily)
```

---

plot_daily_record_day *Plot one day's readings with their mean and OAQ's daily record*

Description

Plot one day's readings with their mean and OAQ's daily record

Usage

```
plot_daily_record_day(readings, check, value = "pm25", title = NULL)
```

Arguments

readings	Readings of one sensor on one day (oaq_readings()).
check	The day's row of compare_daily_records() or oaq_check_daily_records().
value	Parameter plotted.
title	Plot title (default: sensor and date).

Value

A ggplot object.

Examples

```
readings <- data.frame(
  sensor_id = "s1",
  timestamp = as.POSIXct("2026-03-10", tz = "UTC") + 900 * 0:95,
  pm25 = c(rep(40, 95), 120)
)
records <- data.frame(sensor_id = "s1", date = as.Date("2026-03-10"), pm25 = 120)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_daily_record_day(readings, compare_daily_records(readings, records))
}
```

plot_daily_record_difference

Plot how far OAQ's daily record is from the daily mean

Description

Record minus mean per day and sensor, before and from the change date, with the median absolute difference of each under the sensor's label.

Usage

```
plot_daily_record_difference(audit, change, labels = NULL)
```

Arguments

audit	Output of `oaq_audit_daily_records()` or `compare_daily_records()`.
change	Date that splits the days.
labels	Optional named character vector of row labels by sensor key (`"provider sensor_id"`).

Value

A ggplot object.

Examples

```
audit <- data.frame(
  provider = "cpcb", sensor_id = "103", date = as.Date("2026-02-25") + 0:9,
  class = rep(c("mean", "last"), each = 5), mean = 50, last = 80,
  record = rep(c(50, 80), each = 5)
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_daily_record_difference(audit, "2026-03-01")
}
```

plot_daily_record_scatter

Plot OAQ's daily record against the daily mean and the last reading

Description

Plot OAQ's daily record against the daily mean and the last reading

Usage

```
plot_daily_record_scatter(audit)
```

Arguments

audit	Output of `oaq_audit_daily_records()` or `compare_daily_records()`.

Value

A ggplot object with two panels.

Examples

```
audit <- data.frame(
  sensor_id = "103", date = as.Date("2026-02-25") + 0:9,
  class = rep(c("mean", "last"), each = 5), mean = 50, last = 80,
  record = rep(c(50, 80), each = 5)
)
if (requireNamespace("ggplot2", quietly = TRUE)) plot_daily_record_scatter(audit)
```

`plot_daily_record_series` *Plot a sensor's OAQ daily record and daily mean over time*

Description

Plot a sensor's OAQ daily record and daily mean over time

Usage

```
plot_daily_record_series(
  audit,
  sensor_id,
  change = NULL,
  title = NULL,
  value = "pm25"
)
```

Arguments

audit	Output of `oaq_audit_daily_records()` or `compare_daily_records()`.
sensor_id	Sensor to plot.
change	Optional date to mark with a vertical line.
title	Plot title.
value	Parameter name for the axis label.

Value

A ggplot object.

Examples

```
audit <- data.frame(
  sensor_id = "103", date = as.Date("2026-02-25") + 0:9,
  class = "last", mean = 50 + 0:9, last = 80, record = 80 - 0:9
)
if (requireNamespace("ggplot2", quietly = TRUE)) plot_daily_record_series(audit, "103")
```

`plot_daily_record_share` *Plot the share of days by month that OAQ's record equals each statistic*

Description

Plot the share of days by month that OAQ's record equals each statistic

Usage

```
plot_daily_record_share(audit, change = NULL)
```

Arguments

audit	Output of `oaq_audit_daily_records()` or `compare_daily_records()`.
change	Optional date to mark with a vertical line.

Value

A ggplot object.

Examples

```
audit <- data.frame(
  sensor_id = "103", date = as.Date("2026-02-20") + 0:19,
  class = rep(c("mean", "last"), each = 10), mean = 50, last = 80, record = 50
)
if (requireNamespace("ggplot2", quietly = TRUE)) plot_daily_record_share(audit, "2026-03-01")
```

plot_daily_record_timeline

Plot which statistic each day's OAQ record equals

Description

Plot which statistic each day's OAQ record equals

Usage

```
plot_daily_record_timeline(audit, change = NULL, labels = NULL)
```

Arguments

audit	Output of `oaq_audit_daily_records()` or `compare_daily_records()`.
change	Optional date to mark with a vertical line.
labels	Optional named character vector of row labels by sensor key (`"provider sensor_id"`).

Value

A ggplot object.

Examples

```
audit <- data.frame(
  provider = "cpcb", sensor_id = "103", date = as.Date("2026-02-25") + 0:9,
  class = rep(c("mean", "last"), each = 5), mean = 50, last = 80, record = 50
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_daily_record_timeline(audit, change = "2026-03-01")
}
```

plot_sensor_vs_reference

Daily sensor values against their reference station

Description

The left panel plots each sensor-day against the reference, with the 1:1 line. The right panel plots the median of the sensors and of the reference over time. With `change`, the days from that date are orange in the left panel and a line marks the date in the right panel.

Usage

```
plot_sensor_vs_reference(
  paired,
  value = "pm25",
  reference = "reference",
  change = NULL,
  sensor_label = "Sensors",
  reference_label = "Reference",
  unit = "PM2.5 (µg/m³)"
)
```

Arguments

paired	Data frame with one row per sensor and day: columns `date`, `value` and `reference` (for example the output of `readings_mean()` merged with the daily means of the nearest reference station).
value	Column of the sensor values.
reference	Column of the reference values.
change	Optional date to split the days at, such as `"2026-03-01"`.
sensor_label	Name of the sensors in the legends.
reference_label	Name of the reference in the axis label and legend.
unit	Quantity and unit, for the axis labels.

Value

A patchwork of two ggplot2 panels.

Examples

```
paired <- data.frame(
  date = rep(seq(as.Date("2026-01-01"), by = "day", length.out = 90), 2),
  pm25 = c(runif(90, 20, 60), runif(90, 20, 60)), reference = runif(180, 20, 60)
)
plot_sensor_vs_reference(paired, change = "2026-03-01")
```

---

pm25_clear_cache *Clear PM2.5 cache*

Description

Delete all cached NetCDF files from the cache directory.

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 *Download PM2.5 NetCDF file*

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

year	Integer. Year for data, within the range of version.
month	Optional integer (1–12). If NULL, downloads the annual file.
cache_dir	Character. Directory to cache NetCDF files. NULL uses the pm25 folder of vayuayan_cache_dir() and also reads the Python package's cache.
force_download	Logical. Download again even if the file is cached.
region	Character. Spatial subset: "GL" (global, default), "AS" (Asia), "EU", "NA", or "AF" (V6GL03 only).
version	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 *PM2.5 at points*

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 pm25_download_netcdf().

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 *Get PM2.5 statistics for a region*

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 http(s):// URLs are read directly).

year
: Integer. Year of the NetCDF data (e.g. 2023). Ignored when nc_file is supplied.

month
: Optional integer (1–12). If NULL, uses annual data. Ignored when nc_file is supplied.

cache_dir
: Character. Directory to cache NetCDF files. NULL uses the pm25 folder of vayuayan_cache_dir() and also reads 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 year/ month are ignored. Useful for offline data (e.g. WUSTL V5GL04 files on a local mount).
region	Character. Spatial subset to download, passed to pm25_download_netcdf(). "AS" covers India in a smaller file.
version	Character. Dataset version, passed to pm25_download_netcdf().

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

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

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")
```

ratio_by_humidity *Sensor-to-reference ratio by relative humidity*

Description

Low-cost optical sensors can read high in humid air. Hours with a reference value below `min_reference` are left out, as the ratio is unstable there.

Usage

```
ratio_by_humidity(
  hourly,
  sensor = "sensor",
  reference = "reference",
  rh = "rh",
  breaks = c(0, 40, 50, 60, 70, 80, 90, 100),
  min_reference = 5
)
```

Arguments

`hourly`	Data frame with hourly sensor, reference and humidity values.
`sensor`, `reference`, `rh`	Their columns.
`breaks`	Humidity band limits (bands are `(lower, upper]`).
`min_reference`	Smallest reference value used.

Value

Data frame with `band` (`"40-50"` for 40 < RH <= 50), `hours`, `median_ratio`, `q25` and `q75`.

Examples

```
hourly <- data.frame(sensor = c(20, 30, 45), reference = c(20, 20, 30), rh = c(35, 65, 95))
ratio_by_humidity(hourly)
```

readings_mean *Daily or hourly means of sensor readings*

Description

Daily or hourly means of sensor readings

Usage

```
readings_mean(
  readings,
  value = "pm25",
  by = "day",
  min_readings = NULL,
  id_col = "sensor_id",
  time_col = "timestamp"
)
```

Arguments

readings	Data frame with one row per reading, such as `oaq_readings()` or `cpcb_raw_to_long()` (with `id_col = "station_id"`, `value = "value"`).
value	Column to average.
by	`"day"` or `"hour"`.
min_readings	Fewest readings for a mean. `NULL` uses 75 \ the 15-minute readings of the period (72 a day, 3 an hour), the EPA completeness rule.
id_col, time_col	Sensor and time columns.

Value

Data frame with `id_col`, `date` (`Date`) or `hour` (POSIXct), `value` and `n`, for the periods with enough readings.

Examples

```
r <- data.frame(
  sensor_id = "s1",
  timestamp = as.POSIXct("2026-01-01", tz = "UTC") + 900 * 0:95,
  pm25 = 1:96
)
readings_mean(r)
```

---

sensor_accuracy *Accuracy of sensors against a reference*

Description

Per sensor: least-squares slope and intercept of the sensor on the reference, R squared, RMSE and NRMSE, and which EPA targets are met.

Usage

```
sensor_accuracy(
  paired,
  sensor = "pm25",
  reference = "reference",
  min_days = 30,
  id_col = "sensor_id"
)
```

Arguments

paired	Data frame with one row per sensor and day, with the sensor and the reference daily means.
sensor, reference	Columns of the sensor and reference values.
min_days	Fewest paired days for a sensor to be reported.
id_col	Sensor column.

Value

Data frame with id_col, days, slope, intercept, r2, rmse, nrmse_pct, slope_ok, intercept_ok, r2_ok, error_ok (RMSE or NRMSE) and targets_met (0 to 4).

Examples

```
paired <- data.frame(sensor_id = "s1", reference = 1:40, pm25 = 2 + 1.1 * (1:40))
sensor_accuracy(paired)
```

---

sensor_completeness *Completeness of sensor readings*

Description

Share of the expected 15-minute readings, and of days with a valid 24-hour average (75 \

Usage

```
sensor_completeness(
  readings,
  start,
  end,
  value = "pm25",
  ids = NULL,
  id_col = "sensor_id",
  time_col = "timestamp"
)
```

Arguments

readings	Data frame with one row per reading, such as oaq_readings() or cpcb_raw_to_long() (with id_col = "station_id", value = "value").
start, end	First and last day of the period (inclusive).
value	Column to average.
ids	Sensors to report, also those without readings. NULL: the sensors in readings.
id_col, time_col	Sensor and time columns.

Value

Data frame with id_col, readings_pct, valid_days and valid_days_pct, sorted by id_col.

Examples

```
r <- data.frame(
  sensor_id = "s1",
  timestamp = as.POSIXct("2026-01-01", tz = "UTC") + 900 * 0:95,
  pm25 = 10
)
sensor_completeness(r, "2026-01-01", "2026-01-02")
```

---

sensor_plausibility *Plausibility flags of sensor readings*

Description

Screening flags per sensor: negative, zero and extreme readings, stuck runs and spikes. These are screening rules, not EPA targets: a flag marks readings to examine, not proof of a fault.

Usage

```
sensor_plausibility(
  readings,
  value = "pm25",
  max_value = 1000,
  stuck_run = 8,
  spike_factor = 10,
  spike_min = 100,
  spike_window = 9,
  id_col = "sensor_id",
  time_col = "timestamp"
)
```

Arguments

readings	Data frame with one row per reading, such as oaq_readings() or cpcb_raw_to_long() (with id_col = "station_id", value = "value").
value	Column to check, such as "pm25" or "temp".
max_value	Readings above it are flagged.
stuck_run	Runs of at least this many equal consecutive readings are flagged as stuck (8 readings are 2 hours).
spike_factor, spike_min	A reading above spike_factor times the median of the spike_window readings around it, and above spike_min, is a spike.
spike_window	Readings in the centered window (odd); the median needs at least half of them.
id_col, time_col	Sensor and time columns.

Value

Data frame with id_col, readings, negative_pct, zero_pct, above_max_pct, stuck_pct and spikes_pct.

Examples

```
r <- data.frame(
  sensor_id = "s1",
  timestamp = as.POSIXct("2026-01-01", tz = "UTC") + 900 * 0:19,
  pm25 = c(rep(5, 10), 900, 6:14)
)
sensor_plausibility(r)
```

---

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)
```

theme_vayuayan *The vayuayan ggplot2 theme*

Description

A ggplot2 theme with a light surface, recessive grid and axes, a left-aligned bold title and the legend below the panel. The `plot_*` functions use it. Same look as the Python package's `set_theme()`.

Usage

```
theme_vayuayan(base_size = 10.5)
```

Arguments

`base_size`	Base font size in points.

Value

A ggplot2 theme, to add to a plot with +.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_vayuayan()
```

us_aqi_category *US EPA AQI category of AQI values*

Description

US EPA AQI category of AQI values

Usage

```
us_aqi_category(aqi_value)
```

Arguments

`aqi_value`	Numeric vector of AQI values.

Value

Character vector of categories (`"Good"` to `"Hazardous"`, which includes all values above 300), `"No Data"` for missing values.

Examples

```
us_aqi_category(c(42, 120, NA))
```

`us_aqi_color` *US EPA AQI color of AQI values*

Description

The EPA color of each AQI value as a hex code: the standard colors (EPA Table 2), or the ColorVision Assist colors (Table 3), which have more contrast for people with color vision deficiency.

Usage

```
us_aqi_color(aqi_value, color_vision_assist = FALSE)
```

Arguments

`aqi_value`	Numeric vector of AQI values.
`color_vision_assist`	Logical. Use the ColorVision Assist colors.

Value

Character vector of hex colors, `NA` for missing values.

Examples

```
us_aqi_color(c(42, 120, NA))
us_aqi_color(120, color_vision_assist = TRUE)
```

`us_aqi_pollutant_name` *US AQI name of a pollutant label*

Description

Match a pollutant label such as `"PM2.5 (ug/m3)"`, `"Ozone"`, `"O3_1h"` or `"SO2 24h"` to its US EPA AQI name.

Usage

```
us_aqi_pollutant_name(label)
```

Arguments

`label`	Character vector of labels.

Value

Character vector of US AQI names ("PM2.5", "PM10", "O3", "O3_1h", "CO", "SO2", "SO2_24h", "NO2"), NA for other labels.

Examples

```
us_aqi_pollutant_name(c("PM2.5 (ug/m3)", "Ozone", "NH3"))
```

---

us_aqi_sub_index *US EPA AQI sub-index of concentrations*

Description

The US EPA AQI sub-index of one pollutant (May 2024 tables). Each concentration is truncated to the precision of the table, then the sub-index is a linear function of the concentration within its band.

Usage

```
us_aqi_sub_index(pollutant, concentration)
```

Arguments

pollutant	A pollutant label (see us_aqi_pollutant_name()). A label with a mass unit, such as "O3 (ug/m3)" or "CO (mg/m3)", is converted to ppm or ppb at 25 degrees C. Without a unit, PM2.5 and PM10 are in ug/m3, O3 and CO in ppm, and SO2 and NO2 in ppb.
concentration	Numeric vector of averages: 24-hour for PM2.5, PM10 and SO2_24h, 8-hour for O3 and CO, and 1-hour for O3_1h, SO2 and NO2.

Value

Numeric vector of sub-indices. Above the highest breakpoint the sub-index continues on the slope of the 301-500 band, so it can be more than 500. NA for missing values, for 8-hour O3 above 0.200 ppm, 1-hour O3 below 0.125 ppm, 1-hour SO2 above 304 ppb and 24-hour SO2 below 305 ppb, where the EPA uses the other averaging time.

Examples

```
us_aqi_sub_index("PM2.5", c(9, 35.4, 400))
```

vayuayan_cache_dir *Cache directory of vayuayan*

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 *Print vayuayan welcome message*

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, 4
aqi_health_impact, 4
aqi_pollutant_category, 5
aqi_pollutant_name, 6
aqi_sub_index, 6

clean_station_name, 7
collocated_precision, 7
compare_daily_records, 8
compute_aqi, 9
compute_us_aqi, 9
cpcb_add_coords, 10
cpcb_city_daily, 11
cpcb_download_city_data, 11
cpcb_download_hourly, 12
cpcb_download_hourly_city_data
(cpcb_download_hourly), 12
cpcb_download_hourly_station_data
(cpcb_download_hourly), 12
cpcb_download_raw_station_data, 13
cpcb_download_station_data, 14
cpcb_get_all_india, 14
cpcb_get_city_list, 15
cpcb_get_complete_list, 16
cpcb_get_file_path, 16
cpcb_get_live_aqi_data, 17
cpcb_get_live_aqi_data_for_station, 18
cpcb_get_nearest_station, 19
cpcb_get_nearest_stations, 19
cpcb_get_state_list, 20
cpcb_get_station_list, 21
cpcb_get_system_location, 21
cpcb_live_metrics, 22
cpcb_live_to_long, 22
cpcb_raw_readings, 23
cpcb_raw_to_long, 24
cpcb_station_daily, 24
cpcb_station_daily_aqi, 25
cpcb_tidy_daily, 26
daily_aqi, 26
daily_record_summary, 27

get_aqi_categories, 28
get_aqi_category, 28

haversine_distance, 29

india_boundary, 30

nearest_stations, 30

oaq_audit_daily_records, 31
oaq_check_daily_records, 32
oaq_city_tally, 33
oaq_daily_means, 33
oaq_daily_records, 34
oaq_fetch_monthly_stations, 35
oaq_get_annual, 36
oaq_get_city_latest, 36
oaq_get_daily, 37
oaq_get_hierarchy, 38
oaq_get_monthly, 38
oaq_get_station_24h, 40
oaq_get_stations_latest, 39
oaq_list_providers, 40
oaq_nearest_cpcb, 41
oaq_readings, 42
oaq_sensors, 42
oaq_session, *31*, *32*, *34*, *35*, *42*, 43, *44*
oaq_stations_table, 43

peer_ratio, 44
plot_daily_record_day, 45
plot_daily_record_difference, 46
plot_daily_record_scatter, 46
plot_daily_record_series, 47
plot_daily_record_share, 48
plot_daily_record_timeline, 49
plot_sensor_vs_reference, 49
pm25_clear_cache, 50

pm25_download_netcdf, 51
pm25_get_point, 52
pm25_get_stats, 53
pm25_get_stats_by_polygon, 54
pm25_list_cached_files, 55
pm25_load_netcdf_bbox, 56
pm25_map, 57
pm25_monthly, 57
pm25_read_netcdf_bbox, 58
pm25_v5gl04_path, 59

ratio_by_humidity, 60
readings_mean, 60

sensor_accuracy, 61
sensor_completeness, 62
sensor_plausibility, 63
stations_to_city_summary, 64
stations_to_dataframe, 65

theme_vayuayan, 66

us_aqi_category, 66
us_aqi_color, 67
us_aqi_pollutant_name, 67
us_aqi_sub_index, 68

vayuayan_cache_dir, 69
vayuayan_welcome, 69