Package 'worldmet'

November 9, 2026

Type Package

Title Import Surface Meteorological Data from NOAA

Version 1.2.0

Description Functions to import data from more than 30,000 surface meteorological sites around the world managed by the National Oceanic and Atmospheric Administration (NOAA) Global Historical Climate Network (GHCN) and Integrated Surface Database (ISD).

License MIT + file LICENSE

URL https://openair-project.github.io/worldmet/, https://github.com/openair-project/worldmet

BugReports https://github.com/openair-project/worldmet/issues

Depends R (>= 4.1.0)

Imports carrier, cli, dplyr, lifecycle, mirai, purrr (>= 1.1.0), readr, rlang, sf, tibble, tidyr

Suggests arrow, knitr, leaflet, rmarkdown, spelling

ByteCompile true

Config/Needs/website openair-project/openairpkgdown,openair

Config/roxygen2/version 8.1.1

Encoding UTF-8

Language en-GB

LazyData true

LazyLoad true

NeedsCompilation no

Author David Carslaw [aut, cre] (ORCID: <https://orcid.org/0000-0003-0991-950X>), Jack Davison [aut] (ORCID: <https://orcid.org/0000-0003-2653-6615>)

Maintainer David Carslaw <david.carslaw@york.ac.uk>

Repository CRAN

Date/Publication 2026-11-09 08:50:02 UTC

Contents

exportADMS . . . 2
getMeta . . . 3
import_ghcn_countries . . . 5
import_ghcn_daily . . . 6
import_ghcn_hourly . . . 10
import_ghcn_inventory . . . 13
import_ghcn_monthly_temp . . . 14
import_ghcn_stations . . . 15
import_isd_hourly . . . 17
import_isd_lite . . . 19
import_isd_stations . . . 21
import_isd_stations_live . . . 23
weatherCodes . . . 24
write_adms . . . 25
write_met . . . 26

Index **28**

`exportADMS` *Deprecated data functions*

Description

[Deprecated]

This function is part of an old `worldmet` API. Please use the following alternatives:

- `exportADMS()` -> `write_adms()`
- the `path` argument -> `write_met()`

Usage

```
exportADMS(dat, out = "./ADMS_met.MET", interp = FALSE, maxgap = 2)
```

Arguments

`dat`	A data frame imported by `importNOAA()`.
`out`	A file name for the ADMS file. The file is written to the working directory by default.
`interp`	Should interpolation of missing values be undertaken? If `TRUE` linear interpolation is carried out for gaps of up to and including `maxgap`.
`maxgap`	The maximum gap in hours that should be interpolated where there are missing data when `interp = TRUE`. Data with gaps more than `maxgap` are left as missing.

getMeta *Deprecated ISD access functions*

Description

[Deprecated]

This function is part of an old `worldmet` API. Please use the following alternatives:

- `getMeta()` -> `import_isd_stations()`
- `getMetaLive()` -> `import_isd_stations_live()`
- `importNOAA()` -> `import_isd_hourly()`
- `importNOAAlite()` -> `import_isd_lite()`
- the `path` argument -> `write_met()`

Usage

```
getMeta(
  site = NULL,
  lat = NULL,
  lon = NULL,
  crs = 4326,
  country = NULL,
  state = NULL,
  n = 10,
  end.year = "current",
  provider = c(`Street Map` = "OpenStreetMap", Satellite = "Esri.WorldImagery"),
  plot = TRUE,
  returnMap = FALSE
)

getMetaLive(...)

importNOAA(
  code = "037720-99999",
  year = 2014,
  hourly = TRUE,
  source = c("delim", "fwf"),
  quiet = FALSE,
  path = NA,
  n.cores = NULL
)

importNOAAlite(code = "037720-99999", year = 2025, quiet = FALSE, path = NA)
```

Arguments

site	A site name search string e.g. site = "heathrow". The search strings can be partial and can be upper or lower case e.g. site = "HEATHR".
lat, lon, n	Decimal latitude (lat) and longitude (lon) (or other Y/X coordinate if using a different crs). If provided, the n closest ISD stations to this coordinate will be returned.
crs	The coordinate reference system (CRS) of the data, passed to sf::st_crs(). By default this is EPSG:4326, the CRS associated with the commonly used latitude and longitude coordinates. Different coordinate systems can be specified using crs (e.g., crs = 27700 for the British National Grid). Note that non-lat/lng coordinate systems will be re-projected to EPSG:4326 for making comparisons with the NOAA metadata.
country	The country code. This is a two letter code. For a full listing see https://www.ncei.noaa.gov/pub/data/noaa/isd-history.csv.
state	The state code. This is a two letter code.
end.year	To help filter sites based on how recent the available data are. end.year can be "current", "all" or a numeric year such as 2016, or a range of years e.g. 1990:2016 (which would select any site that had an end date in that range. **By default only sites that have some data for the current year are returned**.
provider	When return = "map", by default a map will be created in which readers may toggle between a vector street map and a satellite/aerial image. provider allows users to override this default; see http://leaflet-extras.github.io/leaflet-providers/preview/ for a list of all base maps that can be used. Base maps can be toggled using a layer control menu; the labels will be taken from the name of the base map unless a named list is defined (see default value).
plot	If TRUE will plot sites on an interactive leaflet map.
returnMap	Should the leaflet map be returned instead of the meta data? Default is FALSE.
...	Currently unused.
code	The identifying code as a character string. The code is a combination of the USAF and the WBAN unique identifiers. The codes are separated by a "-" e.g. code = "037720-99999".
year	The year to import. This can be a vector of years e.g. year = 2000:2005.
hourly	Should hourly means be calculated? The default is TRUE. If FALSE then the raw data are returned.
source	The NOAA ISD service stores files in two formats; as delimited CSV files ("delim") and as fixed width files ("fwf"). import_isd_hourly() defaults to "delim" but, if the delimited data store is down, users may wish to try "fwf" instead. Both data sources should be identical to one another.
quiet	Print missing sites/years to the screen? Defaults to FALSE.
path	If a file path is provided, the data are saved as an rds file at the chosen location e.g. path = "C:/Users/David". Files are saved by year and site.
n.cores	No longer recommended; please set mirai::daemons() in your R session. This argument is provided for back compatibility, and is passed to the n argument of

mirai::daemons() on behalf of the user. Any set daemons will be reset once the function completes. Default is NULL, which means no parallelism. n.cores = 1L is equivalent to n.cores = NULL.

Value

A data frame is returned with all available meta data, mostly importantly including a code that can be supplied to importNOAA(). If latitude and longitude searches are made an approximate distance, dist in km is also returned.

---

import_ghcn_countries *Import FIPS country codes and State/Province/Territory codes used by the Global Historical Climatology Network*

Description

This function returns a two-column dataframe either of "Federal Information Processing Standards" (FIPS) codes and the countries to which they are associated, or state codes and their associated states. These may be a useful reference when examining GHCN site metadata.

Usage

```
import_ghcn_countries(
  table = c("countries", "states"),
  database = c("hourly", "daily", "monthly")
)
```

Arguments

table	One of "countries" or "states".
database	One of "hourly", "daily" or "monthly", which defines which of the NOAA databases to import the FIPS codes from. There is little difference between the data in the different sources, but this option may be useful if one of the services is not accessible.

Value

a tibble

Author(s)

Jack Davison

See Also

Other GHCN functions: import_ghcn_daily(), import_ghcn_hourly(), import_ghcn_inventory(), import_ghcn_monthly_temp(), import_ghcn_stations()

`import_ghcn_daily` *Import data from the Global Historical Climatology daily (GHCNd) database*

Description

This function flexibly accesses meteorological data from the GHCNd database. Users can provide any of stations, and control whether attribute codes are returned with the data.

Usage

```
import_ghcn_daily(
  station,
  year = NULL,
  source = c("csv"),
  extra = FALSE,
  append_codes = FALSE,
  codes = c("measurement_flag", "quality_flag", "source_flag"),
  progress = rlang::is_interactive()
)
```

Arguments

`station`	One or more site codes for the station(s) of interest, obtained using `import_ghcn_stations()`.
`year`	One or more years of interest. If `NULL`, the default, all years of data available for the chosen `stations` will be imported. Note that, in the GHCNd and GHCNm, files are split by station but not year, so setting a `year` will not speed up the download. Specifying fewer years will improve the speed of a GHCNh download, however.
`source`	The data format for the GHCNd. Currently only `"csv"` is supported. This argument is included for future use.
`extra`	Should additional columns be returned? The default, `FALSE`, returns an opinionated selection of elements that'll be of most interest to most users. `TRUE` will return everything available.
`append_codes`	Logical. Should various codes and flags be appended to the output dataframe?
`codes`	When `append_codes` is `TRUE`, which codes should be appended to the dataframe? Any combination of `"measurement_flag"`, `"quality_flag"`, and/or `"source_flag"`.
`progress`	Show a progress bar when importing many stations/years? Defaults to `TRUE` in interactive R sessions. Passed to `.progress` in `purrr::map()` and/or `purrr::pmap()`.

Value

a tibble

Data Definition

The core elements in the GHCNd are:

- **PRCP**: Precipitation (mm)
- **SNOW**: Snowfall (mm)
- **SNWD**: Snow depth (mm)
- **TMAX**: Maximum temperature (degrees C)
- **TMIN**: Minimum temperature (degrees C)

Other elements which may appear are:

- **ACMC**: Average cloudiness midnight to midnight from 30-second ceilometer data (percent)
- **ACMH**: Average cloudiness midnight to midnight from manual observations (percent)
- **ACSC**: Average cloudiness sunrise to sunset from 30-second ceilometer data (percent)
- **ACSH**: Average cloudiness sunrise to sunset from manual observations (percent)
- **ADPT**: Average Dew Point Temperature for the day (degrees C)
- **ASLP**: Average Sea Level Pressure for the day (hPa)
- **ASTP**: Average Station Level Pressure for the day (hPa)
- **AWBT**: Average Wet Bulb Temperature for the day (degrees C)
- **AWDR**: Average daily wind direction (degrees)
- **AWND**: Average daily wind speed (meters per second)
- **DAEV**: Number of days included in the multiday evaporation total (MDEV)
- **DAPR**: Number of days included in the multiday precipitation total (MDPR)
- **DASF**: Number of days included in the multiday snowfall total (MDSF)
- **DATN**: Number of days included in the multiday minimum temperature (MDTN)
- **DATX**: Number of days included in the multiday maximum temperature (MDTX)
- **DAWM**: Number of days included in the multiday wind movement (MDWM)
- **DWPR**: Number of days with non-zero precipitation included in multiday precipitation total (MDPR)
- **EVAP**: Evaporation of water from evaporation pan (mm)
- **FMTM**: Time of fastest mile or fastest 1-minute wind (hours and minutes, i.e., HHMM)
- **FRGB**: Base of frozen ground layer (cm)
- **FRGT**: Top of frozen ground layer (cm)
- **FRTH**: Thickness of frozen ground layer (cm)
- **GAHT**: Difference between river and gauge height (cm)
- **MDEV**: Multiday evaporation total (mm; use with **DAEV**)
- **MDPR**: Multiday precipitation total (mm; use with **DAPR** and **DWPR**, if available)
- **MDSF**: Multiday snowfall total
- **MDTN**: Multiday minimum temperature (degrees C; use with **DATN**)

- **MDTX**: Multiday maximum temperature (degrees C; use with **DATX**)
- **MDWM**: Multiday wind movement (km)
- **MNPN**: Daily minimum temperature of water in an evaporation pan (degrees C)
- **MXPN**: Daily maximum temperature of water in an evaporation pan (degrees C)
- **PGTM**: Peak gust time (hours and minutes, i.e., HHMM)
- **PSUN**: Daily percent of possible sunshine (percent)
- **RHAV**: Average relative humidity for the day (percent)
- **RHMN**: Minimum relative humidity for the day (percent)
- **RHMX**: Maximum relative humidity for the day (percent)
- **SN$#**: Minimum soil temperature (degrees C) where $ corresponds to a code for ground cover and # corresponds to a code for soil depth. Ground cover codes: 0=unknown, 1=grass, 2=fallow, 3=bare ground, 4=brome grass, 5=sod, 6=straw mulch, 7=grass muck, 8=bare muck. Depth codes: 1=5cm, 2=10cm, 3=20cm, 4=50cm, 5=100cm, 6=150cm, 7=180cm
- **SX$#**: Maximum soil temperature (degrees C) where $ corresponds to a code for ground cover and # corresponds to a code for soil depth. See **SN$#** for ground cover and depth codes
- **TAXN**: Average daily temperature computed as `(TMAX+TMIN)/2.0` (degrees C)
- **TAVG**: Average daily temperature (degrees C). Note that TAVG from source 'S' corresponds to an average of hourly readings for the period ending at 2400 UTC rather than local midnight or other Local Standard Time according to a specific Met Service's protocol. For sources other than 'S' TAVG is computed in a variety of ways including traditional fixed hours of the day whereas TAXN is solely computed as (TMAX+TMIN)/2.0
- **THIC**: Thickness of ice on water (mm)
- **TOBS**: Temperature at the time of observation (degrees C)
- **TSUN**: Daily total sunshine (minutes)
- **WDF1**: Direction of fastest 1-minute wind (degrees)
- **WDF2**: Direction of fastest 2-minute wind (degrees)
- **WDF5**: Direction of fastest 5-second wind (degrees)
- **WDFG**: Direction of peak wind gust (degrees)
- **WDFI**: Direction of highest instantaneous wind (degrees)
- **WDFM**: Fastest mile wind direction (degrees)
- **WDMV**: 24-hour wind movement (km)
- **WESD**: Water equivalent of snow on the ground (mm)
- **WESF**: Water equivalent of snowfall (mm)
- **WSF1**: Fastest 1-minute wind speed (meters per second)
- **WSF2**: Fastest 2-minute wind speed (meters per second)
- **WSF5**: Fastest 5-second wind speed (meters per second)
- **WSFG**: Peak gust wind speed (meters per second)
- **WSFI**: Highest instantaneous wind speed (meters per second)
- **WSFM**: Fastest mile wind speed (meters per second)

There can be any number of weather types (**WT$$**)

- **WT01**: Fog, ice fog, or freezing fog (may include heavy fog)
- **WT02**: Heavy fog or heaving freezing fog (not always distinguished from fog)
- **WT03**: Thunder
- **WT04**: Ice pellets, sleet, snow pellets, or small hail
- **WT05**: Hail (may include small hail)
- **WT06**: Glaze or rime
- **WT07**: Dust, volcanic ash, blowing dust, blowing sand, or blowing obstruction
- **WT08**: Smoke or haze
- **WT09**: Blowing or drifting snow
- **WT10**: Tornado, waterspout, or funnel cloud
- **WT11**: High or damaging winds
- **WT12**: Blowing spray
- **WT13**: Mist
- **WT14**: Drizzle
- **WT15**: Freezing drizzle
- **WT16**: Rain (may include freezing rain, drizzle, and freezing drizzle)
- **WT17**: Freezing rain
- **WT18**: Snow, snow pellets, snow grains, or ice crystals
- **WT19**: Unknown source of precipitation
- **WT21**: Ground fog
- **WT22**: Ice fog or freezing fog

There can also be any number of 'weather in the vicinity' columns (**WV$$**)

- **WV01**: Fog, ice fog, or freezing fog (may include heavy fog)
- **WV03**: Thunder
- **WV07**: Ash, dust, sand, or other blowing obstruction
- **WV18**: Snow or ice crystals
- **WV20**: Rain or snow shower

Parallel Processing

If you are importing a lot of meteorological data, this can take a long while. This is because each combination of year and station requires downloading a separate data file from NOAA's online data directory, and the time each download takes can quickly add up. Many data import functions in `{worldmet}` can use parallel processing to speed downloading up, powered by the capable `{mirai}` package. If users have any `{mirai}` "daemons" set, these functions will download files in parallel. The greatest benefits will be seen if you spawn as many daemons as you have cores on your machine, although one fewer than the available cores is often a good rule of thumb. Your mileage may vary, however, and naturally spawning more daemons than station-year combinations will lead to diminishing returns.

```
# set workers - once per session
mirai::daemons(4)

# import lots of data - NB: no change in the import function!
big_met <- import_ghcn_hourly(station = "UKI0000EGLL", year = 2010:2025)
```

Author(s)

Jack Davison

See Also

Other GHCN functions: `import_ghcn_countries()`, `import_ghcn_hourly()`, `import_ghcn_inventory()`, `import_ghcn_monthly_temp()`, `import_ghcn_stations()`

`import_ghcn_hourly` *Import data from the Global Historical Climatology hourly (GHCNh) database*

Description

This function flexibly accesses meteorological data from the GHCNh database. Users can provide any number of years and stations, and fully control the sorts of data flag codes that are returned with the data. By default, column names are shortened for easier use in R, but longer, more descriptive names can be requested.

Usage

```
import_ghcn_hourly(
  station = "UKI0000EGLL",
  year = NULL,
  source = c("psv", "parquet"),
  hourly = TRUE,
  extra = FALSE,
  abbr_names = TRUE,
  append_codes = FALSE,
  codes = c("measurement_code", "quality_code", "report_type", "source_code",
    "source_id"),
  progress = rlang::is_interactive()
)
```

Arguments

`station`	One or more site codes for the station(s) of interest, obtained using `import_ghcn_stations()`.
`year`	One or more years of interest. If `NULL`, the default, all years of data available for the chosen `stations` will be imported. Note that, in the GHCNd and GHCNm, files are split by station but not year, so setting a `year` will not speed up the download. Specifying fewer years will improve the speed of a GHCNh download, however.

source	There are two identical data formats to read from - "psv" (flat, pipe-delimited files) and "parquet" (a newer, faster, columnar format). The latter is typically faster, but requires the arrow package as an additional dependency. Note that this only applies when year is not NULL; all by-site files are psv files.
hourly	Should hourly means be calculated? The default is TRUE. If FALSE then the raw data are returned, which can be sub-hourly.
extra	Should additional columns be returned? The default, FALSE, returns an opinionated selection of elements that'll be of most interest to most users. TRUE will return everything available.
abbr_names	Should column names be abbreviated? When TRUE, the default, columns like "wind_direction" are shortened to "wd". When FALSE, names will match the raw data, albeit in lower case.
append_codes	Logical. Should various codes and flags be appended to the output dataframe?
codes	When append_codes is TRUE, which codes should be appended to the dataframe? Any combination of "measurement_code", "quality_code", "report_type", "source_code", and/or "source_id".
progress	Show a progress bar when importing many stations/years? Defaults to TRUE in interactive R sessions. Passed to .progress in purrr::map() and/or purrr::pmap().

Value

a tibble

Data Definition

The following core elements are in the GHCNh:

- **wind_direction:** (wd) Wind direction from true north (degrees). 360 = north, 180 = south, 270 = west; 000 indicates calm winds.
- **wind_speed:** (ws) Average wind speed (m/s).
- **temperature:** (air_temp) Air (dry bulb) temperature at approximately 2 meters above ground level, in degrees Celsius (to tenths).
- **station_level_pressure:** (atmos_pres) Pressure observed at station elevation; true barometric pressure at the location (hPa).
- **visibility:** (visibility) Horizontal visibility distance (km).
- **dew_point_temperature:** (dew_point) Dew point temperature in degrees Celsius (to tenths).
- **relative_humidity:** (rh) Relative humidity in whole percent, measured or calculated from temperature and dew point.
- **sky_cover:** (cl) Maximum of all **sky_cover_X** elements (oktas).
- **sky_cover_baseht:** (cl_baseht) The height above ground level (AGL) of the lowest cloud or obscuring phenomena layer aloft with 5/8 or more summation total sky cover, which may be predominantly opaque, or the vertical visibility into a surface-based obstruction.
- **sky_cover_X:** (cl_X) Fraction of sky covered by clouds (oktas). Up to 3 layers reported.
- **sky_cover_baseht_X:** (cl_baseht_X) Cloud base height for lowest layer (m). Up to 3 layers reported.

- **precipitation:** (precip) Total liquid precipitation (rain or melted snow) for the hour (mm). "T" indicates trace amounts.
- **precipitation_XX_hour:** (precip_XX) 3-hour total liquid precipitation (mm). "T" indicates trace amounts.

When extra = TRUE, the following additional columns are included:

- **sea_level_pressure:** (sea_pres) Estimated pressure at sea level directly below the station using actual temperature profile (hPa).
- **wind_gust:** (wg) Peak short-duration wind speed (m/s) exceeding the average wind speed.
- **wet_bulb_temperature:** (wet_bulb) Wet bulb temperature in degrees Celsius (to tenths), measured or calculated from temperature, dew point, and station pressure.
- **snow_depth:** (snow_depth) Depth of snowpack on the ground (mm).
- **altimeter:** (altimeter) Pressure reduced to mean sea level using standard atmosphere profile (hPa).
- **pressure_3hr_change:** (pres_03) Change in atmospheric pressure over a 3-hour period (hPa), with tendency code.

If hourly = FALSE, the following character columns may also be present.

- **pres_wx_MWX:** (wx_mwX) Present weather observation from manual reports (code). Up to 3 observations per report.
- **pres_wx_AUX:** (wx_auX) Present weather observation from automated ASOS/AWOS sensors (code). Up to 3 observations per report.
- **pres_wx_AWX:** (wx_aqX) Present weather observation from automated sensors (code). Up to 3 observations per report.
- **remarks:** (remarks) Raw observation remarks encoded in ICAO-standard METAR/SYNOP format.

Parallel Processing

If you are importing a lot of meteorological data, this can take a long while. This is because each combination of year and station requires downloading a separate data file from NOAA's online data directory, and the time each download takes can quickly add up. Many data import functions in {worldmet} can use parallel processing to speed downloading up, powered by the capable {mirai} package. If users have any {mirai} "daemons" set, these functions will download files in parallel. The greatest benefits will be seen if you spawn as many daemons as you have cores on your machine, although one fewer than the available cores is often a good rule of thumb. Your mileage may vary, however, and naturally spawning more daemons than station-year combinations will lead to diminishing returns.

```
# set workers - once per session
mirai::daemons(4)

# import lots of data - NB: no change in the import function!
big_met <- import_ghcn_hourly(station = "UKI0000EGLL", year = 2010:2025)
```

Author(s)

Jack Davison

See Also

Other GHCN functions: `import_ghcn_countries()`, `import_ghcn_daily()`, `import_ghcn_inventory()`, `import_ghcn_monthly_temp()`, `import_ghcn_stations()`

---

import_ghcn_inventory *Import station inventory for the Global Historical Climatology Network*

Description

This function accesses a data inventory of GHCN stations available through either the GHCNh or GHCNd. The returned `data.frame` contains data which reveals the earliest and latest years of data available for each station from the NOAA database.

Usage

```
import_ghcn_inventory(
  database = c("hourly", "daily"),
  pivot = c("wide", "long"),
  progress = rlang::is_interactive()
)
```

Arguments

`database`	One of `"hourly"` or `"daily"`, which defines whether to import the GHCNh or GHCNd inventory. The way in which these files is formatted is different.
`pivot`	One of `"wide"` or `"long"`. The GHCNh inventory can be returned in a `"wide"` format (with `id`, `year` and twelve month columns) or a `"long"` format (with `id`, `year`, `month`, and `count` columns). Does not apply to the GHCNd inventory.
`progress`	The inventory file is large and can be slow to download. Show a progress indicator when accessing the inventory? Defaults to `TRUE` in interactive R sessions. Passed to `progress` in `readr::read_fwf()` and/or `purrr::pmap()`.

Value

a tibble

Author(s)

Jack Davison

See Also

Other GHCN functions: `import_ghcn_countries()`, `import_ghcn_daily()`, `import_ghcn_hourly()`, `import_ghcn_monthly_temp()`, `import_ghcn_stations()`

`import_ghcn_monthly_temp` *Import data from the Global Historical Climatology monthly (GHCNm) database*

Description

This function is a convenient way to access the monthly summaries of the GHCN. Monthly average temperature is available via `import_ghcn_monthly_temp()` and monthly precipitation via `import_ghcn_monthly_prcp()`. Note that these functions can take a few minutes to run, and parallelism is only enabled for precipitation data.

Usage

```
import_ghcn_monthly_temp(
  table = c("inventory", "data"),
  dataset = c("qcu", "qcf", "qfe")
)

import_ghcn_monthly_prcp(
  station = NULL,
  year = NULL,
  table = c("inventory", "data"),
  progress = rlang::is_interactive()
)
```

Arguments

`table`	Either `"inventory"`, `"data"`, or both. The tables to read and return in the output list.
`dataset`	For `import_ghcn_monthly_temp()`. One of the below options. More information is available at https://www.ncei.noaa.gov/pub/data/ghcn/v4/readme.txt. • `"qcu"`: Quality Control, Unadjusted • `"qcf"`: Quality Control, Adjusted, using the Pairwise Homogeneity Algorithm. • `"qfe"`: Quality Control, Adjusted, Estimated using the Pairwise Homogeneity Algorithm. Only the years 1961-2010 are provided. This is to help maximize station coverage when calculating normals.
`station`	For `import_ghcn_monthly_prcp()`. The specific stations to import monthly precipitation data for.

year	One or more years of interest. If NULL, the default, all years of data available for the chosen stations will be imported. Note that, in the GHCNd and GHCNm, files are split by station but not year, so setting a year will not speed up the download. Specifying fewer years will improve the speed of a GHCNh download, however.
progress	For import_ghcn_monthly_prcp(). Show a progress bar when importing many stations? Defaults to TRUE in interactive R sessions. Passed to .progress in purrr::map().

Value

a list of tibbles

Parallel Processing

If you are importing a lot of meteorological data, this can take a long while. This is because each combination of year and station requires downloading a separate data file from NOAA's online data directory, and the time each download takes can quickly add up. Many data import functions in {worldmet} can use parallel processing to speed downloading up, powered by the capable {mirai} package. If users have any {mirai} "daemons" set, these functions will download files in parallel. The greatest benefits will be seen if you spawn as many daemons as you have cores on your machine, although one fewer than the available cores is often a good rule of thumb. Your mileage may vary, however, and naturally spawning more daemons than station-year combinations will lead to diminishing returns.

```
# set workers - once per session
mirai::daemons(4)

# import lots of data - NB: no change in the import function!
big_met <- import_ghcn_hourly(station = "UKI0000EGLL", year = 2010:2025)
```

Author(s)

Jack Davison

See Also

Other GHCN functions: import_ghcn_countries(), import_ghcn_daily(), import_ghcn_hourly(), import_ghcn_inventory(), import_ghcn_stations()

---

import_ghcn_stations *Import station metadata for the Global Historical Climatology Network*

Description

This function accesses a full list of GHCN stations available through either the GHCNh or GHCNd. As well as the station id, needed for importing measurement data, useful geographic and network metadata is also returned.

Usage

```
import_ghcn_stations(
  name = NULL,
  country = NULL,
  state = NULL,
  lat = NULL,
  lng = NULL,
  crs = 4326,
  n_max = 10L,
  provider = c(`Street Map` = "OpenStreetMap", Satellite = "Esri.WorldImagery"),
  database = c("hourly", "daily"),
  return = c("table", "sf", "map")
)
```

Arguments

`name`, `country`, `state`
: String values to use to filter the metadata for specific site names, countries and states. `country` and `state` are matched exactly to codes accessed using `import_ghcn_countries()`. `name` is searched as a sub-string case insensitively.

`lat`, `lng`, `n_max`
: Decimal latitude (`lat`) and longitude (`lng`) (or other Y/X coordinate if using a different `crs`). If provided, the `n_max` closest GHCN stations to this coordinate will be returned.

`crs`
: The coordinate reference system (CRS) of the data, passed to `sf::st_crs()`. By default this is EPSG:4326, the CRS associated with the commonly used latitude and longitude coordinates. Different coordinate systems can be specified using `crs` (e.g., `crs = 27700` for the British National Grid). Note that non-lat/lng coordinate systems will be re-projected to `EPSG:4326` for making comparisons with the NOAA metadata.

`provider`
: When `return = "map"`, by default a map will be created in which readers may toggle between a vector street map and a satellite/aerial image. `provider` allows users to override this default; see `http://leaflet-extras.github.io/leaflet-providers/preview/` for a list of all base maps that can be used. Base maps can be toggled using a layer control menu; the labels will be taken from the name of the base map unless a named list is defined (see default value).

`database`
: One of `"hourly"` or `"daily"`, which defines whether to import stations available in the GHCNh or GHCNd. Note that there is overlap between the two, but some stations may only be available in one or the other.

`return`
: The type of R object to import the data as. One of the following:

  - `"table"`, which returns an R `data.frame`.
  - `"sf"`, which returns a spatial `data.frame` from the `sf` package.
  - `"map"`, which returns an interactive `leaflet` map.

Value

One of:

- a tibble
- an sf object
- an interactive `leaflet` map

Author(s)

Jack Davison

See Also

Other GHCN functions: `import_ghcn_countries()`, `import_ghcn_daily()`, `import_ghcn_hourly()`, `import_ghcn_inventory()`, `import_ghcn_monthly_temp()`

`import_isd_hourly` *Import Meteorological data from the NOAA Integrated Surface Database (ISD)*

Description

This is the main function to import data from the NOAA Integrated Surface Database (ISD). The ISD contains detailed surface meteorological data from around the world for over 30,000 locations.

Usage

```
import_isd_hourly(
  code = "037720-99999",
  year = 2014,
  hourly = TRUE,
  source = c("delim", "fwf"),
  progress = rlang::is_interactive(),
  quiet = FALSE
)
```

Arguments

`code`	The identifying code as a character string. The code is a combination of the USAF and the WBAN unique identifiers. The codes are separated by a "-" e.g. `code = "037720-99999"`.
`year`	The year to import. This can be a vector of years e.g. `year = 2000:2005`.
`hourly`	Should hourly means be calculated? The default is `TRUE`. If `FALSE` then the raw data are returned.
`source`	The NOAA ISD service stores files in two formats; as delimited CSV files (`"delim"`) and as fixed width files (`"fwf"`). `import_isd_hourly()` defaults to `"delim"` but, if the delimited data store is down, users may wish to try `"fwf"` instead. Both data sources should be identical to one another.
`progress`	Show a progress bar when importing many stations/years? Defaults to `TRUE` in interactive R sessions. Passed to `.progress` in `purrr::map()` and/or `purrr::pmap()`.
`quiet`	Print missing sites/years to the screen? Defaults to `FALSE`.

Details

Note the following units for the main variables:

date Date/time in POSIXct format. **Note the time zone is GMT (UTC) and may need to be adjusted to merge with other local data. See details below.**

latitude Latitude in decimal degrees (-90 to 90).

longitude Longitude in decimal degrees (-180 to 180). Negative numbers are west of the Greenwich Meridian.

elevation Elevation of site in metres.

wd Wind direction in degrees. 90 is from the east.

ws Wind speed in m/s.

ceil_hgt The height above ground level (AGL) of the lowest cloud or obscuring phenomena layer aloft with 5/8 or more summation total sky cover, which may be predominantly opaque, or the vertical visibility into a surface-based obstruction.

visibility The visibility in metres.

air_temp Air temperature in degrees Celsius.

dew_point The dew point temperature in degrees Celsius.

atmos_pres The sea level pressure in millibars.

RH The relative humidity (%).

cl_1, ..., cl_3 Cloud cover for different layers in Oktas (1-8).

cl Maximum of cl_1 to cl_3 cloud cover in Oktas (1-8).

cl_1_height, ..., cl_3_height Height of the cloud base for each layer in metres.

precip_12 12-hour precipitation in mm. The sum of this column should give the annual precipitation.

precip_6 6-hour precipitation in mm.

precip This value of precipitation spreads the 12-hour total across the previous 12 hours.

pwc The description of the present weather description (if available).

The data are returned in GMT (UTC). It may be necessary to adjust the time zone when combining with other data. For example, if air quality data were available for Beijing with time zone set to "Etc/GMT-8" (note the negative offset even though Beijing is ahead of GMT. See the `openair` package and manual for more details), then the time zone of the met data can be changed to be the same. One way of doing this would be `attr(met$date, "tzone") <- "Etc/GMT-8"` for a meteorological data frame called `met`. The two data sets could then be merged based on `date`.

Value

Returns a data frame of surface observations. The data frame is consistent for use with the `openair` package. Note that the data are returned in GMT (UTC) time zone format. Users may wish to express the data in other time zones, e.g., to merge with air pollution data.

Parallel Processing

If you are importing a lot of meteorological data, this can take a long while. This is because each combination of year and station requires downloading a separate data file from NOAA's online data directory, and the time each download takes can quickly add up. Many data import functions in `{worldmet}` can use parallel processing to speed downloading up, powered by the capable `{mirai}` package. If users have any `{mirai}` "daemons" set, these functions will download files in parallel. The greatest benefits will be seen if you spawn as many daemons as you have cores on your machine, although one fewer than the available cores is often a good rule of thumb. Your mileage may vary, however, and naturally spawning more daemons than station-year combinations will lead to diminishing returns.

```
# set workers - once per session
mirai::daemons(4)

# import lots of data - NB: no change in the import function!
big_met <- import_ghcn_hourly(station = "UKI0000EGLL", year = 2010:2025)
```

Author(s)

David Carslaw

See Also

Other NOAA ISD functions: `import_isd_lite()`, `import_isd_stations()`, `import_isd_stations_live()`

Examples

```
## Not run:
# import some data
beijing_met <- import_isd_hourly(code = "545110-99999", year = 2010:2011)

# importing lots of data? use mirai for parallel processing
mirai::daemons(4)
beijing_met2 <- import_isd_hourly(code = "545110-99999", year = 2010:2025)

## End(Not run)
```

`import_isd_lite` *Import "Lite" Meteorological data from the NOAA Integrated Surface Database (ISD)*

Description

This function is an alternative to `import_isd_hourly()`, and provides access to the "Lite" format of the data. This is a subset of the larger `import_isd_hourly()` dataset featuring eight common climatological variables. As it assigns the nearest measurement to the "top of the hour" to the data, specific values are likely similar but different to those returned by `import_isd_hourly()`. Read the technical document for more information.

Usage

```
import_isd_lite(
  code = "037720-99999",
  year = 2025,
  progress = rlang::is_interactive(),
  quiet = FALSE
)
```

Arguments

code	The identifying code as a character string. The code is a combination of the USAF and the WBAN unique identifiers. The codes are separated by a "-" e.g. code = "037720-99999".
year	The year to import. This can be a vector of years e.g. year = 2000:2005.
progress	Show a progress bar when importing many stations/years? Defaults to TRUE in interactive R sessions. Passed to .progress in purrr::map() and/or purrr::pmap().
quiet	Print missing sites/years to the screen? Defaults to FALSE.

Details

Note the following units for the main variables:

date Date/time in POSIXct format. **Note the time zone is UTC and may need to be adjusted to merge with other local data.**

latitude Latitude in decimal degrees (-90 to 90).

longitude Longitude in decimal degrees (-180 to 180). Negative numbers are west of the Greenwich Meridian.

elev Elevation of site in metres.

ws Wind speed in m/s.

wd Wind direction in degrees. 90 is from the east.

air_temp Air temperature in degrees Celsius.

atmos_pres The sea level pressure in millibars.

dew_point The dew point temperature in degrees Celsius.

precip_6 6-hour precipitation in mm.

precip_1 1-hour precipitation in mm.

sky Sky Condition Total Coverage Code.

The data are returned in GMT (UTC). It may be necessary to adjust the time zone when combining with other data. For example, if air quality data were available for Beijing with time zone set to "Etc/GMT-8" (note the negative offset even though Beijing is ahead of GMT. See the openair package and manual for more details), then the time zone of the met data can be changed to be the same. One way of doing this would be attr(met$date, "tzone") <- "Etc/GMT-8" for a meteorological data frame called met. The two data sets could then be merged based on date.

Value

Returns a data frame of surface observations. The data frame is consistent for use with the openair package. Note that the data are returned in GMT (UTC) time zone format. Users may wish to express the data in other time zones, e.g., to merge with air pollution data.

Parallel Processing

If you are importing a lot of meteorological data, this can take a long while. This is because each combination of year and station requires downloading a separate data file from NOAA's online data directory, and the time each download takes can quickly add up. Many data import functions in {worldmet} can use parallel processing to speed downloading up, powered by the capable {mirai} package. If users have any {mirai} "daemons" set, these functions will download files in parallel. The greatest benefits will be seen if you spawn as many daemons as you have cores on your machine, although one fewer than the available cores is often a good rule of thumb. Your mileage may vary, however, and naturally spawning more daemons than station-year combinations will lead to diminishing returns.

```
# set workers - once per session
mirai::daemons(4)

# import lots of data - NB: no change in the import function!
big_met <- import_ghcn_hourly(station = "UKI0000EGLL", year = 2010:2025)
```

Author(s)

Jack Davison

See Also

import_isd_stations() to obtain the codes based on various site search approaches.

Other NOAA ISD functions: import_isd_hourly(), import_isd_stations(), import_isd_stations_live()

Examples

```
## Not run:
heathrow_lite <- import_isd_lite(code = "037720-99999", year = 2025)

## End(Not run)
```

---

import_isd_stations *Import station metadata for the Integrated Surface Database*

Description

This function is primarily used to find a site code that can be used to access data using import_isd_hourly(). Sites searches of approximately 30,000 sites can be carried out based on the site name and based on the nearest locations based on user-supplied latitude and longitude.

Usage

```
import_isd_stations(
  site = NULL,
  country = NULL,
  state = NULL,
  lat = NULL,
  lng = NULL,
  crs = 4326,
  n_max = 10,
  end_year = "current",
  provider = c(`Street Map` = "OpenStreetMap", Satellite = "Esri.WorldImagery"),
  return = c("table", "sf", "map")
)
```

Arguments

site	A site name search string e.g. site = "heathrow". The search strings can be partial and can be upper or lower case e.g. site = "HEATHR".
country	The country code. This is a two letter code. For a full listing see https://www.ncei.noaa.gov/pub/data/noaa/isd-history.csv.
state	The state code. This is a two letter code.
lat, lng, n_max	Decimal latitude (lat) and longitude (lng) (or other Y/X coordinate if using a different crs). If provided, the n_max closest GHCN stations to this coordinate will be returned.
crs	The coordinate reference system (CRS) of the data, passed to sf::st_crs(). By default this is EPSG:4326, the CRS associated with the commonly used latitude and longitude coordinates. Different coordinate systems can be specified using crs (e.g., crs = 27700 for the British National Grid). Note that non-lat/lng coordinate systems will be re-projected to EPSG:4326 for making comparisons with the NOAA metadata.
end_year	To help filter sites based on how recent the available data are. end_year can be "current", "all" or a numeric year such as 2016, or a range of years e.g. 1990:2016 (which would select any site that had an end date in that range. **By default only sites that have some data for the current year are returned**.
provider	When return = "map", by default a map will be created in which readers may toggle between a vector street map and a satellite/aerial image. provider allows users to override this default; see http://leaflet-extras.github.io/leaflet-providers/preview/ for a list of all base maps that can be used. Base maps can be toggled using a layer control menu; the labels will be taken from the name of the base map unless a named list is defined (see default value).
return	The type of R object to import the data as. One of the following: • "table", which returns an R data.frame. • "sf", which returns a spatial data.frame from the sf package. • "map", which returns an interactive leaflet map.

Value

A data frame is returned with all available meta data, mostly importantly including a code that can be supplied to `import_isd_hourly()`. If latitude and longitude searches are made an approximate distance, `dist` in km is also returned.

Author(s)

David Carslaw

See Also

Other NOAA ISD functions: `import_isd_hourly()`, `import_isd_lite()`, `import_isd_stations_live()`

Examples

```
## Not run:
## search for sites with name beijing
import_isd_stations(site = "beijing")

## End(Not run)

## Not run:
## search for near a specified lat/lng - near Beijing airport
## returns 'n_max' nearest by default
import_isd_stations(lat = 40, lng = 116.9)

## End(Not run)
```

---

import_isd_stations_live

Obtain site meta data from NOAA server

Description

Download all NOAA meta data, allowing for re-use and direct querying.

Usage

```
import_isd_stations_live(...)
```

Arguments

...	Currently unused.

Value

a tibble

See Also

Other NOAA ISD functions: `import_isd_hourly()`, `import_isd_lite()`, `import_isd_stations()`

Examples

```
## Not run:
meta <- import_isd_stations_live()
head(meta)

## End(Not run)
```

`weatherCodes` *Codes for weather types*

Description

This data frame consists of the weather description codes used in the ISD. It is not of general use to most users.

Usage

```
weatherCodes
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 100 rows and 2 columns.

Details

pwc Weather code, which can be merged with the `pwc` column in `import_isd_hourly()` datasets.

description Description associated with the weather codes.

Examples

```
weatherCodes
```

write_adms *Export a meteorological data frame in ADMS format*

Description

Writes a text file in the ADMS format to a location of the user's choosing, with optional interpolation of missing values. This function works with data from both import_ghcn_hourly() and import_isd_hourly().

Usage

```
write_adms(x, file = "./ADMS_met.MET", interp = FALSE, max_gap = 2)
```

Arguments

x	A data frame imported by import_ghcn_hourly() or import_isd_hourly(). Note that this function only works for hourly GHCN data when abbr_names = TRUE.
file	A file name for the ADMS file. The file is written to the working directory by default.
interp	Should interpolation of missing values be undertaken? If TRUE linear interpolation is carried out for gaps of up to and including max_gap.
max_gap	The maximum gap in hours that should be interpolated where there are missing data when interp = TRUE. Data with gaps more than max_gap are left as missing.

Value

write_adms() returns the input dat invisibly.

See Also

Other Met writing functions: write_met()

Examples

```
## Not run:
# import some data then export it
dat <- import_isd_hourly(year = 2012)
write_adms(dat, file = "~/adms_met.MET")

## End(Not run)
```

write_met *Export a meteorological data frame in files, chunked by site and year*

Description

Writes data returned by any of `import_isd_hourly()`, `import_ghcn_hourly()`, or `import_ghcn_daily()` to a file. Each station and year in the data is written to a separate file.

Usage

```
write_met(
  x,
  path = ".",
  ext = c("rds", "delim", "parquet"),
  delim = ",",
  suffix = "",
  progress = rlang::is_interactive()
)
```

Arguments

x	A data frame imported by `import_isd_hourly()`, `import_ghcn_hourly()`, or `import_ghcn_daily()`.
path	The path to a directory to save each file. By default, this is the working directory.
ext	The file type to use when saving the data. Can be `"rds"`, `"delim"` or `"parquet"`. Note that `"parquet"` requires the `arrow` package.
delim	Delimiter used to separate values when `ext = "delim"`. Must be a single character. Defaults to being comma-delimited (`","`).
suffix	An additional suffix to append to file names. Useful examples could be `"_isd"`, `"_hourly"`, `"_lite"`, and so on.
progress	Show a progress bar when writing many stations/years? Defaults to `TRUE` in interactive R sessions. Passed to `.progress` in `purrr::walk()`.

Value

`write_met()` returns `path` invisibly.

See Also

Other Met writing functions: `write_adms()`

Examples

```
## Not run:
# import some data then export it
dat <- import_isd_hourly(year = 2012)
write_met(dat)

## End(Not run)
```

Index

∗ **ADMS functions**
  write_adms, 25
∗ **GHCN functions**
  import_ghcn_countries, 5
  import_ghcn_daily, 6
  import_ghcn_hourly, 10
  import_ghcn_inventory, 13
  import_ghcn_monthly_temp, 14
  import_ghcn_stations, 15
∗ **Met writing functions**
  write_adms, 25
  write_met, 26
∗ **NOAA ISD functions**
  import_isd_hourly, 17
  import_isd_lite, 19
  import_isd_stations, 21
  import_isd_stations_live, 23
∗ **datasets**
  weatherCodes, 24

exportADMS, 2
exportADMS(), *2*

getMeta, 3
getMeta(), *3*
getMetaLive *(*getMeta*)*, 3
getMetaLive(), *3*

import_ghcn_countries, 5
import_ghcn_countries(), *10*, *13–17*
import_ghcn_daily, 6
import_ghcn_daily(), *5*, *13–15*, *17*, *26*
import_ghcn_hourly, 10
import_ghcn_hourly(), *5*, *10*, *14, 15*, *17*, *25, 26*
import_ghcn_inventory, 13
import_ghcn_inventory(), *5*, *10*, *13*, *15*, *17*
import_ghcn_monthly_prcp *(*import_ghcn_monthly_temp*)*, 14
import_ghcn_monthly_prcp(), *14, 15*
import_ghcn_monthly_temp, 14
import_ghcn_monthly_temp(), *5*, *10*, *13, 14*, *17*
import_ghcn_stations, 15
import_ghcn_stations(), *5, 6*, *10*, *13–15*
import_isd_hourly, 17
import_isd_hourly(), *3, 4*, *17*, *19*, *21*, *23–26*
import_isd_lite, 19
import_isd_lite(), *3*, *19*, *23, 24*
import_isd_stations, 21
import_isd_stations(), *3*, *19*, *21*, *24*
import_isd_stations_live, 23
import_isd_stations_live(), *3*, *19*, *21*, *23*
importNOAA *(*getMeta*)*, 3
importNOAA(), *2, 3*, *5*
importNOAAlite *(*getMeta*)*, 3
importNOAAlite(), *3*

mirai::daemons(), *4, 5*

purrr::map(), *6*, *11*, *15*, *17*, *20*
purrr::pmap(), *6*, *11*, *13*, *17*, *20*
purrr::walk(), *26*

readr::read_fwf(), *13*

sf, *17*
sf::st_crs(), *4*, *16*, *22*

tibble, *5, 6*, *11*, *13*, *17*, *23*
tibbles, *15*

weatherCodes, 24
write_adms, 25
write_adms(), *2*, *26*
write_met, 26
write_met(), *2, 3*, *25*