Package 'WES'

July 12, 2024

Type Package

Title Tools for Analyzing Wastewater and Environmental Sampling Data

Version 1.0.0

Date 2024-07-03

Description Provides reproducible functions for collating and analyzing data from environmental sampling studies. Environmental Sampling (ES) of infectious diseases involves collecting samples from various sources (such as sewage, water, air, soil, or surfaces) to monitor the presence of pathogens in the environment. Analysis of ES data often requires the calculation of Real-Time Quantitative PCR (qPCR) variables, normalizing ES observations, and analyzing sampling site characteristics. To help reduce the complexity of these analyses we have implemented tools that assist with establishing standardized ES data formats, absolute and relative quantification of qPCR data, estimation of qPCR amplification efficiency, and collating open-source spatial data for sampling sites.

URL https://www.r-wes.com/

License CC BY 4.0

RoxygenNote 7.3.1

Encoding UTF-8

LazyData true

Imports glue, foreach, data.table, dplyr, HDInterval, zoo, openmeteo, chirps, whitebox, exactextractr, elevatr, raster, sp, sf, rworldmap, stars, RCurl, XML, httr, jsonlite

Depends R (>= 4.1.1), stats, graphics, grDevices, utils

Suggests rmarkdown, rworldxtra, progress

NeedsCompilation no

Author John R Giles [aut, cre] (<https://orcid.org/0000-0002-0954-4093>)

Maintainer John R Giles <john.giles@gatesfoundation.org>

Repository CRAN

Date/Publication 2024-07-12 14:00:06 UTC

Contents

`apply_amplification_efficiency` *Apply PCR amplification efficiency estimation to a data.frame*

Description

This function applies the `est_amplification_efficiency()` function to a data.frame object which follows the standardized format shown in the `template_WES_standard_curve` data set.

Usage

```
apply_amplification_efficiency(standard_curves)
```

Arguments

`standard_curves`
: A data.frame giving the target name, serial diluted concentration of target nucleic acid, and Ct value from a standard curve assay. Must follow the `template_WES_standard_curve` standardized format.

Value

A data.frame containing the mean, and low and high of the 95% confidence interval of the percentile amplification efficiency for each target name.

Examples

```
apply_amplification_efficiency(template_WES_standard_curve)
```

`apply_delta_delta_ct` *Apply the delta delta Ct calculation to a data.frame*

Description

This function will calculate the delta delta Ct metric for all applicable observations in a data.frame by applying the `calc_delta_delta_ct` function. The data.frame must have the following columns: 'location_id', 'sample_date', 'target_name', and 'ct_value'. The relevant target_names and associated reference_names must be provided. The result is a data.frame containing a 'delta_delta_ct' column which can be merge into the source data.frame.

Usage

```
apply_delta_delta_ct(
  df,
  target_names,
  reference_names,
  pae_names = NULL,
  pae_values = NULL
)
```

Arguments

`df`	A data.frame containing the following columns: 'location_id', 'sample_date', 'target_name', and 'ct_value'.
`target_names`	Character vector giving the names of the target genes.
`reference_names`	Character vector giving the names of the reference genes associated with each target gene.
`pae_names`	Character vector giving the names of the target genes and reference genes for which the percentile amplification efficiency has been estimated. Default is NULL.
`pae_values`	A numeric scalar giving the estimated PCR amplification efficiency for each of the names in `pae_names`. Defaults is NULL, which assumes 100% efficiency.

Value

data.frame

Examples

```
pae <- apply_amplification_efficiency(template_WES_standard_curve)

ddct_standard <- apply_delta_delta_ct(df = template_WES_data,
                                      target_names = c('target_1', 'target_2', 'target_3'),
                                         reference_names = rep('target_0', 3))

ddct_adjusted <- apply_delta_delta_ct(df = template_WES_data,
                                      target_names = c('target_1', 'target_2', 'target_3'),
                                         reference_names = rep('target_0', 3),
                                         pae_names = pae$target_name,
                                         pae_values = pae$mean)

head(ddct_adjusted)
```

`calc_delta_delta_ct` *Calculate delta delta Ct*

Description

This function calculates relative gene expression using the delta delta Ct method described in Livak and Schmittgen (2001). Adjusted delta delta Ct values following Yuan et al. (2008) can be calculated by providing estimated values for the percentile amplification efficiency in `pae_*` arguments.

Usage

```
calc_delta_delta_ct(
  ct_target_treatment,
  ct_target_control,
  ct_reference_treatment,
  ct_reference_control,
  pae_target_treatment = 1,
  pae_target_control = 1,
  pae_reference_treatment = 1,
  pae_reference_control = 1
)
```

Arguments

`ct_target_treatment`
: A numeric scalar providing the Ct value of the target gene for an observation in the treatment group

`ct_target_control`
: A numeric scalar providing the Ct value of the target gene for the reference observation in the control group

`ct_reference_treatment`
: A numeric scalar providing the Ct value of the reference gene for an observation in the treatment group

`ct_reference_control`
: A numeric scalar providing the Ct value of the reference gene for the reference observation in the control group

`pae_target_treatment`
: A numeric scalar providing the percentile amplification efficiency for the target gene and the treatment group. Defaults to 1.

`pae_target_control`
: A numeric scalar providing the percentile amplification efficiency for the target gene and the control group. Defaults to 1.

`pae_reference_treatment`
: A numeric scalar providing the percentile amplification efficiency for the reference gene and the treatment group. Defaults to 1.

`pae_reference_control`
: A numeric scalar providing the percentile amplification efficiency for the reference gene and the control group. Defaults to 1.

Value

Scalar

Examples

```
# Traditional method
calc_delta_delta_ct(ct_target_treatment = 32.5,
                    ct_reference_treatment = 25,
                    ct_target_control = 34,
                    ct_reference_control = 30)

# Adjusted calculation incorporating amplification efficiency
calc_delta_delta_ct(ct_target_treatment = 32.5,
                    ct_reference_treatment = 25,
                    ct_target_control = 34,
                    ct_reference_control = 30,
                    pae_target_treatment=0.97,
                    pae_target_control=0.98,
                    pae_reference_treatment=0.98,
                    pae_reference_control=0.99)
```

`calc_n_copies` *Calculate number of target copies*

Description

This function calculates the quantitative value of the qPCR Ct value. Cycle threshold here is converted into the estimated number of gene target copies (e.g. viral load for a viral pathogen) by fitting a log linear model to the standard curve data and then using that model to find a point estimate for the provided Ct values.

Usage

```
calc_n_copies(ct_values, target_names, standard_curves)
```

Arguments

`ct_values`	A numeric vector giving the Ct value for each observation.
`target_names`	A character vector giving the target names for each element in 'ct_values'.
`standard_curves`	A data.frame containing results from standard curve dilution experiment. Elements in 'target_names' must map to either 'target_name_unique' or 'target_name_concise'. See package data object `template_WES_standard_curves` for template.

Value

Vector

Examples

```
df <- template_WES_data[template_WES_data$target_name == 'target_1',]
sc <- template_WES_standard_curve[template_WES_standard_curve$target_name == 'target_1',]

tmp <- calc_n_copies(ct_values = df$ct_value,
                     target_names = df$target_name,
                     standard_curves = sc)

df$n_copies <- tmp
head(df)
```

---

calc_sample_sizes *Calculate sample sizes*

Description

This function takes a compiled data.frame following the format shown in the `template_WES_data` object and calculates basic sample sizes and detection rates for all gene targets.

Usage

```
calc_sample_sizes(df, cutoff = 40)
```

Arguments

df	A data.frame produced by the compile_tac_data function containing 'target_name' and 'ct_value' columns.
cutoff	Numeric scalar giving the cutoff Ct value over which a gene target is deemed absent from a sample. Default is 40.

Value

data.frame

Examples

```
calc_sample_sizes(template_WES_data)
```

---

coords_to_iso3 *Convert coordinates to ISO country code*

Description

This function takes a set of longitude and latitude coordinates and retrieves the administrative units that each point lies within. The administrative units are given in the ISO-3166 Alpha-3 country code standard (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3).

Usage

```
coords_to_iso3(lon, lat)
```

Arguments

lon	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
lat	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.

Value

data.frame

Examples

```
coords_to_iso3(lon = c(90.37, 90.38, 90.37),
               lat = c(23.80, 23.80, 23.81))
```

`download_admin_data` *Download administrative boundaries from geoBoundaries API*

Description

This function takes a single ISO country code and downloads the corresponding high resolution administrative boundary GeoJSON files from the www.geoBoundaries.org API hosted at GitHub HERE. If the desired administrative level is not available the next most detailed administrative level is returned.

Usage

```
download_admin_data(
  iso3,
  release,
  path_output,
  simplified = FALSE,
  keep_geojson = FALSE
)
```

Arguments

`iso3`	A three-letter capitalized character string. Must follow the ISO-3166 Alpha-3 country code
`release`	A character string specifying the release type on the geoBoundaries API. It should be one of 'gbOpen', 'gbHumanitarian', or 'gbAuthoritative'. Release types are described at `https://www.geoboundaries.org/api.html#api`.
`path_output`	A character string giving the file path of an output directory to save downloaded data.
`simplified`	Logical indicating whether to download simplified administrative boundaries instead of high resolution. Default is FALSE.
`keep_geojson`	Logical indicating whether to keep the raw geojson files downloaded from geoBoundaries API. Default is FALSE.

Value

Character string giving path to downloaded data.

Examples

```
download_admin_data(iso3 = 'MCO',
                    release = 'gbOpen',
                    path_output = tempdir())
```

`download_elevation_data` *Download DEM from AWS Terrain Tiles*

Description

This function takes the coordinates of sampling sites (longitude and latitude) and downloads a Digital Elevation Model (DEM) for the surrounding area. The DEM has an approximate spatial resolution of 100 meters. These data are derived from the Shuttle Radar Topography Mission (SRTM) DEM, which is accessible through the Amazon Web Services (AWS) API and the `elevatr` R package.

Usage

```
download_elevation_data(lon, lat, path_output)
```

Arguments

`lon`	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.Can accept a vector of multiple ISO codes.
`lat`	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
`path_output`	A character string giving the file path of an output directory to save downloaded data.

Value

Character string giving path to downloaded data.

Examples

```
download_elevation_data(lon = template_WES_data$lon,
                        lat = template_WES_data$lat,
                        path_output = tempdir())
```

`download_worldpop_data` *Download WorldPop population raster data*

Description

This function takes a single ISO country code and downloads the appropriate population count raster data (100m grid cell resolution) from the WorldPop FTP data server. Note that these data are spatial disaggregations of census data using random forest models described in Lloyd et al. 2019 and available for manual download at https://hub.worldpop.org/geodata/listing?id=29. Downloaded data sets are saved to the path_output directory in .tif format.

Usage

```
download_worldpop_data(
  iso3,
  year,
  constrained = FALSE,
  UN_adjusted = FALSE,
  path_output
)
```

Arguments

iso3	A three-letter capitalized character string. Must follow the ISO-3166 Alpha-3 country code standard (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3). Can accept a vector of multiple ISO codes.
year	A numeric or integer scalar giving the year of WorldPop data to download (as of 2024-05-15, years 2000-2020 are available)
constrained	Logical indicating whether to get population counts estimated using constrained models (details HERE). Default is FALSE.
UN_adjusted	Logical indicating whether to get population counts that are adjusted to match United Nations national population estimates (details HERE). Default is FALSE.
path_output	A character string giving the file path of an output directory to save downloaded data.

Value

Character string giving path to downloaded data.

Examples

```
download_worldpop_data(iso3 = 'MCO',
                       year = 2020,
                       constrained = TRUE,
                       UN_adjusted = FALSE,
                       path_output = tempdir())
```

est_amplification_efficiency *Estimate PCR amplification efficiency*

Description

This function takes a set of serial diluted concentrations of target nucleic acid from a standard curve assay and their associated Ct values and estimates the percentile amplification efficiency using a linear model as described in Yuan et al. (2008). Note that the model uses a log base 2 transform which assumes that serial dilutions double with each increase in concentration. The function also requires a minimum of 5 observations.

Usage

```
est_amplification_efficiency(n_copies, ct_value)
```

Arguments

n_copies	A numeric vector giving the serial diluted concentration of target nucleic acid
ct_value	A numeric vector giving the measured Ct value for each serial dilution in the standard curve design

Value

List containing the mean, and low and high of the 95% confidence interval for the percentile amplification efficiency.

Examples

```
sel <- template_WES_standard_curve$target_name == 'target_1'
tmp_n_copies <- template_WES_standard_curve$n_copies[sel]
tmp_ct_value <- template_WES_standard_curve$ct_value[sel]

est_amplification_efficiency(n_copies = tmp_n_copies,
                             ct_value = tmp_ct_value)
```

get_admin_data *Get administrative data for a set of points*

Description

This function takes a set of longitude and latitude coordinates and retrieves the administrative units that each point lies within.

Usage

```
get_admin_data(lon, lat, path_admin_data)
```

Arguments

lon	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
lat	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
path_admin_data	The file path to the admin data. Note that the function expects .shp format output from the download_admin_data function or from another user supplied source.

Value

data.frame

Examples

```
download_admin_data(iso3 = "MCO",
                    release = 'gbOpen',
                    path_output = tempdir())

get_admin_data(lon = c(7.416, 7.434),
               lat = c(43.734, 43.747),
               path_admin_data = file.path(tempdir(), 'MCO_admin_levels.shp'))
```

---

get_elevation_data *Get elevation data*

Description

This function takes information of where and when a set of environmental samples were collected and retrieves the elevation (in meters) for those locations at an approximate 100m spatial resolution. Data come from the SRTM DEM which are accessed through the Amazon Web Services (AWS) API and the elevatr R package.

Usage

```
get_elevation_data(lon, lat)
```

Arguments

lon	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
lat	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.

Value

data.frame

Examples

```
get_elevation_data(lon = template_WES_data$lon,
                   lat = template_WES_data$lat)
```

`get_esi_data` *Get Evaporative Stress Index (ESI) data*

Description

This function takes information of where and when a set of environmental samples were collected and retrieves the Evaporative Stress Index (ESI) for those locations and times. For more information about ESI, see description HERE. Data come from the Climate Hazards Center InfraRed Precipitation with Station data (CHIRPS) via the `chirps` R package. Additionally, the optional `intervals` argument specifies a set of intervals over which the function will calculate the average ESI for the previous X number of days for each location.

Usage

```
get_esi_data(lon, lat, dates, intervals = NULL)
```

Arguments

`lon`	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
`dates`	A character or date vector of dates giving the date when each sample was collected (format is YYYY-MM-DD)
`intervals`	An integer vector giving a set of time intervals over which to calculate the average ESI. Default is NULL where the interval is 0 (returns the ESI value at time t). If `intervals=3` then the average ESI over the preceding 3 days is returned.

Value

data.frame

Examples

```
tmp <- get_esi_data(lon = c(-54.9857, -52.9857),
                    lat = c(-5.9094, -25.8756),
                    dates = c("2020-06-01", "2020-10-31"),
                    intervals = c(5,10,20))

head(tmp)
```

`get_geoboundaries_api_data`

Request metadata from GeoBoundaries API

Description

This function retrieves GeoBoundaries data from the API based on the specified release, ISO3 country code, and administrative level. If data is not found at the specified administrative level, it attempts to retrieve data from a lower administrative level until data is found or the lowest level is reached.

Usage

```
get_geoboundaries_api_data(iso3, admin_level, release = "gbOpen")
```

Arguments

`iso3`	A three-letter capitalized character string. Must follow the ISO-3166 Alpha-3 country code standard (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3).
`admin_level`	An integer specifying the administrative level. It should be between 0 and 5.
`release`	A character string specifying the release type. It should be one of 'gbOpen', 'gbHumanitarian', or 'gbAuthoritative'. Default is 'gbOpen'.

Value

A list containing the GeoBoundaries API data and file paths to admin boundaries in .geojson format.

Examples

```
tmp <- get_geoboundaries_api_data(iso3 = 'MCO', admin_level = 2, release = 'gbOpen')
head(tmp)
```

`get_hydro_data` *Get hydrological data*

Description

This function takes information of where and when a set of environmental samples were collected and retrieves a suite of topographical and hydrological variables for each unique location. The variables include: elevation, slope, aspect, Topographical Wetness Index (TWI), flow accumulation, total flow accumulation within 500m, and distance to the nearest stream. If a DEM is not provided, then a DEM is acquired via `elevatr::get_elev_raster` and the suite of variables are calculated using functions from the 'WhiteboxTools' R frontend.

Usage

```
get_hydro_data(lon, lat, path_dem_raster = NULL, path_output)
```

Arguments

`lon`	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
`path_dem_raster`	The file path to a Digital Elevation Model (DEM) raster. See `download_elevation_data` for methods to download DEM raster data. If NULL, a DEM is downloaded automatically using this function.
`path_output`	The file path of an output directory where spatial data will be saved.

Value

data.frame

Examples

```
MCO_lon <- c(7.416, 7.434)
MCO_lat <- c(43.734, 43.747)

download_elevation_data(lon = MCO_lon,
                        lat = MCO_lat,
                        path_output = tempdir())

get_hydro_data(lon = MCO_lon,
               lat = MCO_lat,
               path_dem_raster = file.path(tempdir(), 'dem.tif'),
               path_output = tempdir())
```

`get_population_catchment` *Get population counts within catchments of sampling sites*

Description

This function takes vectors of sampling site longitude and latitude and calculates the total population residing within the drainage catchment of each coordinate pair. Raster data giving population counts per grid cell and a Digital Elevation Model (DEM) are required. By default, the function delineates streams based on the provided DEM. However, an optional shapefile (such as an urban sewer network) can be specified using the `path_stream_shp` argument and will be used instead of the natural stream network calculated from the DEM. Note that the delineation of catchments along streams (or sewer networks) still depends on the directional flow from the provided DEM. Intermediate spatial variables are written to the directory specified in `path_output`.

Usage

```
get_population_catchment(
  lon,
  lat,
  path_pop_raster,
  path_dem_raster,
  path_stream_shp = NULL,
  path_output
)
```

Arguments

`lon`	A numeric vector giving the longitudes of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitudes of the sampling sites in Decimal Degrees.
`path_pop_raster`	The file path to a raster object providing population counts in each grid cell. See `download_worldpop_data` for methods to download population raster data.
`path_dem_raster`	The file path to a Digital Elevation Model (DEM) raster. See `download_elevation_data` for methods to download DEM raster data.
`path_stream_shp`	An optional file path to a stream or sewer network shapefile. If NULL (the default), streams are delineated based on flow accumulation in the provided DEM.
`path_output`	The file path of an output directory where spatial data will be saved.

Value

A `data.frame` containing the catchment area and population counts for each sampling site.

Examples

```
MCO_lon <- c(7.416, 7.434)
MCO_lat <- c(43.734, 43.747)

download_worldpop_data(iso3 = 'MCO',
                       year = 2020,
                       constrained = TRUE,
                       UN_adjusted = FALSE,
                       path_output = tempdir())

download_elevation_data(lon = MCO_lon,
                        lat = MCO_lat,
                        path_output = tempdir())

get_population_catchment(lon = MCO_lon,
                         lat = MCO_lat,
                 path_pop_raster = file.path(tempdir(), 'mco_ppp_2020_constrained.tif'),
                        path_dem_raster = file.path(tempdir(), 'dem.tif'),
                        path_output = tempdir())
```

---

get_population_radius *Get population counts within a radius of sampling sites*

Description

This function takes vectors of sampling site longitude and latitude and calculates the total population residing within a given radius around each sampling site. Intermediate spatial variables are written to the directory specified in `path_output`.

Usage

```
get_population_radius(lon, lat, radius, path_pop_raster, path_output)
```

Arguments

`lon`	A numeric vector giving the longitudes of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitudes of the sampling sites in Decimal Degrees.
`radius`	Numeric giving the radius (in meters) around each point to calculate total population
`path_pop_raster`	The file path to a raster object providing population counts in each grid cell. See `download_worldpop_data` for methods to download population raster data.
`path_output`	The file path of an output directory where spatial data will be saved.

Value

A `data.frame` containing the total population counts for the given radius around each sampling site.

Examples

```
download_worldpop_data(iso3 = 'MCO',
                       year = 2020,
                       constrained = TRUE,
                       UN_adjusted = FALSE,
                       path_output = tempdir())

get_population_radius(lon = c(7.416, 7.434),
                      lat = c(43.734, 43.747),
                      radius = 100,
                  path_pop_raster = file.path(tempdir(), 'mco_ppp_2020_constrained.tif'),
                      path_output = tempdir())
```

`get_precip_data` *Get precipitation data*

Description

This function takes information of where and when a set of environmental samples were collected and retrieves precipitation data (in millimeters) for those locations and times. Data come from the Open-Meteo Historical Weather API (https://open-meteo.com/en/docs/historical-weather-api) via the openmeteo R package. Additionally, the optional `intervals` argument specifies a set of intervals over which the function will calculate the cumulative sum of precipitation in millimeters (mm) for the previous X number of days for each location.

Usage

```
get_precip_data(lon, lat, dates, intervals = NULL)
```

Arguments

`lon`	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
`dates`	A character or date vector of dates giving the date when each sample was collected (format is YYYY-MM-DD)
`intervals`	An integer vector giving a set of time intervals over which to sum the precipitation data. Default is NULL where the interval is 0 (returns the precipitation value at time t). If `intervals=3` then the cumulative precipitation over the preceding 3 days is returned.

Value

data.frame

Examples

```
tmp <- get_precip_data(lon = c(-56.0281, -54.9857),
                       lat = c(-2.9094, -2.8756),
                       dates = c("2017-12-01", "2017-12-31"),
                       intervals = c(1,3,7))

head(tmp)
```

`get_river_discharge_data` *Get river discharge data*

Description

This function takes information of where and when a set of environmental samples were collected and retrieves daily river discharge data from the nearest river (m^3/s) for those locations and times. Data come from the Open-Meteo Global Flood API (https://open-meteo.com/en/docs/flood-api) via the openmeteo R package.

Usage

```
get_river_discharge_data(lon, lat, dates)
```

Arguments

`lon`	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
`dates`	A character or date vector of dates giving the date when each sample was collected (format is YYYY-MM-DD)

Value

data.frame

Examples

```
tmp <- get_river_discharge_data(lon = c(-54.9857, -52.9857),
                                lat = c(-10.9094, -25.8756),
                                dates = c("2020-06-01", "2020-10-31"))

head(tmp)
```

`get_temp_data` *Get temperature data*

Description

This function takes information of where and when a set of environmental samples were collected and retrieves temperature data (measured in accumulated degree-days) for those locations and times. Data come from the Open-Meteo Historical Weather API (https://open-meteo.com/en/docs/historical-weather-api) via the openmeteo R package. The optional `intervals` argument specifies a set of intervals over which the function will calculate the accumulated temperature in the form of Accumulated Thermal Units (ATUs) for each interval.

Usage

```
get_temp_data(lon, lat, dates, intervals = NULL)
```

Arguments

`lon`	A numeric vector giving the longitude of the sampling sites in Decimal Degrees.
`lat`	A numeric vector giving the latitude of the sampling sites in Decimal Degrees.
`dates`	A character or date vector of dates giving the date when each sample was collected (format is YYYY-MM-DD)
`intervals`	An integer vector giving a set of time intervals over to calculate accumulated degree-days. Default is NULL where the interval is 0 (returns the daily temperature in degrees Celsius at time t). If `intervals=3` then the accumulated degree-days for the preceding 3 days is returned.

Value

data.frame

Examples

```
tmp <- get_temp_data(lon = c(30.0281, -52.9857),
                     lat = c(15.9094, -25.8756),
                     dates = c("2020-08-01", "2020-12-31"),
                     intervals = c(1,5,10))

head(tmp)
```

`template_WES_data` *Template environmental sampling data*

Description

The `template_WES_data` object provides a template of the data format required by the 'WES' package.

Usage

```
template_WES_data
```

Format

`template_WES_data`**:**
A data frame with 6 columns:

date The date each sample was collected. Formate is "YYYY-MM-DD".

location_id A unique identifier for each of the sampling locations.

lat The lattitude of the sampling location in decimal degrees.

lon The longitude of the sampling location in decimal degrees.

target_name The unique name of the gene target for which the Ct values correspond.

ct_value The Cycle Threshold (Ct) of the qPCR assay.

`template_WES_standard_curve`
Template standard curve data

Description

The `template_WES_standard_curve` object provides a template of the data format required by the 'WES' package for standard curve values. These data are only required when calculating the number of gene copies using the `calc_n_copies` function.

Usage

```
template_WES_standard_curve
```

Format

`template_WES_standard_curve`**:**

A data frame with 3 columns:

target_name The unique name of the gene target for which the Ct values correspond.

n_copies The number of gene copies represented in the particular dilution.

ct_value The Cycle Threshold (Ct) of the qPCR assay.

Index