Package 'epicsawrap'

December 3, 2024

Title What the Package Does (One Line, Title Case)

Version 0.0.1

Description What the package does (one paragraph).

License LGPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 2.10)

Imports DBI,
dplyr,
forcats,
googleCloudStorageR,
jsonlite,
lubridate,
magrittr,
purrr (>= 1.00),
rlang,
RMySQL,
rpicsa,
stringr,
testthat,
tidyr

Remotes IDEMSInternational/rpicsa

Contents

`add_data_to_bucket` *Add New Data to Google Cloud Storage Bucket*

Description

This function adds a new data file (RDS format) with a timestamp to a specified Google Cloud Storage (GCS) bucket.

Usage

```
add_data_to_bucket(country, station_id, data, timestamp = NULL)
```

Arguments

`country`	A character vector specifying the country that the data corresponds to. Options are `"mz"` and `"zm"`.
`station_id`	A character string specifying the ID of the station that the data corresponds to.
`data`	The data to upload.
`timestamp`	A character vector with a timestamp. By default this is `NULL` so is generated.

Details

The function creates a timestamp in the format "YYYYMMDDHHMMSS" and appends it to the station_id to form the filename. It then reads the provided file, creates a new file with the timestamped filename, and uploads it to the specified GCS bucket.

See Also

`get_bucket_name` for retrieving the GCS bucket name.

`add_definitions_to_bucket`
Add New Definitions to Google Cloud Storage Bucket

Description

This function adds a new definitions file with a timestamp to a specified Google Cloud Storage (GCS) bucket. The new definitions are sourced from a JSON file provided as input.

Usage

```
add_definitions_to_bucket(
  country,
  definitions_id,
  new_definitions,
  timestamp = NULL
)
```

Arguments

country	A character vector specifying the country or countries from which to get the definitions data. Options are "mz" and "zm".
definitions_id	A character string specifying the ID of the definitions data.
new_definitions	A character vector specifying the path to the JSON file containing the new definitions.
timestamp	A character vector with a timestamp. By default this is NULL so is generated.

Details

The function creates a timestamp in the format "YYYYMMDDHHMMSS" and appends it to the definitions_id to form the filename. It then reads the provided JSON file, creates a new JSON file with the timestamped filename, and uploads it to the specified GCS bucket.

See Also

get_bucket_name for retrieving the GCS bucket name.

---

add_summaries_to_bucket

Add New Summaries to Google Cloud Storage Bucket

Description

This function adds a new summary data file (RDS format) with a timestamp to a specified Google Cloud Storage (GCS) bucket.

Usage

```
add_summaries_to_bucket(country, station_id, data, summary, timestamp = NULL)
```

Arguments

country	A character vector specifying the country that the data corresponds to. Options are "mz" and "zm".
station_id	A character string specifying the ID of the station that the data corresponds to.
data	The data to upload.
summary	The summary function used to create the data.
timestamp	A character vector with a timestamp. By default this is NULL so is generated.

Details

The function creates a timestamp in the format "YYYYMMDDHHMMSS" and appends it to the station_id to form the filename. It then reads the provided file, creates a new file with the timestamped filename, and uploads it to the specified GCS bucket.

See Also

`get_bucket_name` for retrieving the GCS bucket name.

`annual_rainfall_annual_rain` *Calculate Annual Rainfall*

Description

This function calculates annual rainfall based on the provided definitions, daily rainfall data, and data variable names.

Usage

```
annual_rainfall_annual_rain(definitions, daily, data_names)
```

Arguments

`definitions`	A list containing definitions for the annual rainfall analysis, including options for calculating total rainfall, number of rainy days, and criteria for handling missing values.
`daily`	A data frame containing daily rainfall data.
`data_names`	A list of variable names used in the data frame.

Value

A numeric vector representing annual rainfall and related calculations.

Examples

```
# # Example usage:
# require(dplyr)
# library(cdms.products)
# data(daily_niger)
# definitions <- list(
#   annual_rain = list(
#     annual_rain = TRUE,
#     n_rain = FALSE,
#     na_rm = TRUE
#   )
# )
# data_names <- list(date = "date", station = "station_name", year = "year", rain = "rain")
# daily_data <- daily_niger %>% dplyr::filter(year > 1975) %>% dplyr::filter(station_name == "Zinder")
# annual_rainfall <- annual_rainfall_annual_rain(definitions, daily_data, data_names)
```

annual_rainfall_end_rains *Calculate End of Rains*

Description

This function calculates the end of the rainy season based on the provided definitions, daily rainfall data, and data variable names.

Usage

```
annual_rainfall_end_rains(definitions, daily, data_names)
```

Arguments

definitions	A list containing definitions for the end of the rainy season analysis, including start and end days, interval length, and minimum rainfall criteria.
daily	A data frame containing daily rainfall data.
data_names	A list of variable names used in the data frame.

Value

A numeric vector representing the end of the rainy season in terms of day of the year (DOY).

Examples

```
# Example usage:
# require(dplyr)
# library(cdms.products)
# data(daily_niger)
# definitions <- list(
#   end_rains = list(
#     start_day = 122,
#     end_day = 366,
#     interval_length = 7,
#     min_rainfall = 5
#   )
# )
# data_names <- list(date = "date", station = "station_name", year = "year", rain = "rain")
# daily_data <- daily_niger %>% dplyr::filter(year > 1975) %>% dplyr::filter(station_name == "Zinder")
# end_of_rains <- annual_rainfall_end_rains(definitions, daily_data, data_names)
```

annual_rainfall_end_season *Calculate End of Season*

Description

This function calculates the end of the rainy season based on the provided definitions, daily rainfall data, and data variable names.

Usage

```
annual_rainfall_end_season(definitions, daily, data_names)
```

Arguments

definitions	A list containing definitions for the end of the rainy season analysis, including start and end days, capacity, water balance, evaporation criteria, and related parameters.
daily	A data frame containing daily rainfall data.
data_names	A list of variable names used in the data frame.

Value

A numeric vector representing the end of the rainy season in terms of day of the year (DOY).

Examples

```
# Example usage:
# require(dplyr)
# library(cdms.products)
# data(daily_niger)
# definitions <- list(
#   end_season = list(
#     start_day = 150,
#     end_day = 366,
#     capacity = 100,
#     water_balance_max = 50,
#     evaporation = "value",
#     evaporation_value = 10
#   )
# )
# data_names <- list(date = "date", station = "station_name", year = "year", rain = "rain")
# daily_data <- daily_niger %>% dplyr::filter(year > 1975) %>% dplyr::filter(station_name == "Zinder")
# end_of_season <- annual_rainfall_end_season(definitions, daily_data, data_names)
```

annual_rainfall_seasonal_length *Calculate Seasonal Length*

Description

This function calculates the seasonal length of annual rainfall based on the provided definitions, daily rainfall data, summary data, data variable names, and summary types.

Usage

```
annual_rainfall_seasonal_length(
  definitions,
  daily,
  summary_data,
  data_names,
  summaries
)
```

Arguments

definitions	A list containing definitions for various aspects of the analysis.
daily	A data frame containing daily rainfall data.
summary_data	A data frame containing summary data.
data_names	A list of variable names used in the data frames.
summaries	A character vector specifying the summary types.

Value

A numeric vector representing the seasonal length of annual rainfall.

Examples

```
# Example usage:
definitions <- list(seasonal_length = list(end_type = "rains"))
data_names <- list(date = "date", station = "station", year = "year", rain = "rain")
summary_types <- c("start_rains", "end_rains", "end_season")
daily_data <- data.frame(date = seq(as.Date("2023-01-01"), as.Date("2023-12-31"), by = "days"),
                         station = "Station_A",
                         year = rep(2023, each = 365),
                         rain = runif(365, min = 0, max = 10))
summary_data <- data.frame(year = 2023, start_rains_date = as.Date("2023-04-01"),
                  end_rains_date = as.Date("2023-10-31"), end_season_date = as.Date("2023-12-31"))
seasonal_length <- annual_rainfall_seasonal_length(definitions, daily_data, summary_data, data_names, summary
```

annual_rainfall_seasonal_rain *Calculate Seasonal Rain*

Description

This function calculates the seasonal rainfall based on the provided definitions, daily rainfall data, summary data, data variable names, and summary types.

Usage

```
annual_rainfall_seasonal_rain(
  definitions,
  daily,
  summary_data,
  data_names,
  summaries
)
```

Arguments

definitions	A list containing definitions for various aspects of the analysis, including seasonal rainfall properties such as total rain, number of rainy days, and criteria for rainy days.
daily	A data frame containing daily rainfall data.
summary_data	A data frame containing summary data.
data_names	A list of variable names used in the data frames.
summaries	A character vector specifying the summary types.

Value

A numeric vector representing the seasonal rainfall based on the provided criteria.

Examples

```
# # Example usage:
# definitions <- list(
#   seasonal_rain = list(
#     total_rain = TRUE,
#     n_rain = TRUE,
#     rain_day = 10,
#     na_rm = TRUE,
#     end_type = "rains"
#   )
# )
# data_names <- list(date = "date", station = "station", year = "year", rain = "rain")
# summary_types <- c("start_rains", "end_rains", "end_season")
# daily_data <- data.frame(date = seq(as.Date("2023-01-01"), as.Date("2023-12-31"), by = "days"),
#                          station = "Station_A",
#                          year = rep(2023, each = 365),
#                          rain = runif(365, min = 0, max = 10))
# daily_data$doy <- lubridate::yday(daily_data$date)
```

```
# summary_data <- data.frame(year = 2023, start_rains = as.Date("2023-04-01"),
#                   end_rains = as.Date("2023-10-31"), end_season = as.Date("2023-12-31"))
# seasonal_rain <- annual_rainfall_seasonal_rain(definitions, daily_data, summary_data, data_names, summary_t
```

`annual_rainfall_start_rains` *Calculate Annual Rainfall Start of Rains*

Description

This function calculates the start of the rainy season based on the provided definitions, daily rainfall data, and data variable names.

Usage

```
annual_rainfall_start_rains(definitions, daily, data_names)
```

Arguments

`definitions`	A list containing definitions for the start of the rainy season analysis, including threshold, start and end days, and various options for criteria and calculations.
`daily`	A data frame containing daily rainfall data.
`data_names`	A list of variable names used in the data frame.

Value

A numeric vector representing the start of the rainy season in terms of day of the year (DOY).

Examples

```
# require(dplyr)
# library(cdms.products)
# data(daily_niger)
# # Example usage:
# definitions <- list(
#   start_rains = list(
#     threshold = 0.5,
#     start_day = 90,
#     end_day = 200,
#     total_rainfall = TRUE,
#     over_days = 7,
#     amount_rain = 10,
#     proportion = FALSE,
#     prob_rain_day = 0.6,
#     number_rain_days = TRUE,
#     min_rain_days = 3,
#     rain_day_interval = 5,
#     dry_spell = TRUE,
#     spell_interval = 14,
#     spell_max_dry_days = 10,
#     dry_period = FALSE
#   )
```

```
# )
# data_names <- list(date = "date", station = "station_name", year = "year", rain = "rain")
# daily_data <- daily_niger %>% dplyr::filter(year > 1975) %>% dplyr::filter(station_name == "Zinder")
# start_of_rains <- annual_rainfall_start_rains(definitions, daily_data, data_names)
```

`annual_rainfall_summaries`

Annual Rainfall Summaries

Description

A table containing all the annual rainfall summaries for PICSA e.g. start of rain, total rainfall, number of rain days, end of season. One row per year/station and one column per summary.

Usage

```
annual_rainfall_summaries(
  country,
  station_id,
  call = c("climsoft", "googlebuckets"),
 summaries = c("annual_rain", "start_rains", "end_rains", "end_season", "seasonal_rain",
    "seasonal_length"),
  override = FALSE
)
```

Arguments

`country`	`character(1)` The country code of the data.
`station_id`	`character` The id's of the stations to analyse. Either a single value or a vector.
`call`	A character vector specifying where to call the raw data from if calling raw data.
`summaries`	`character` The names of the summaries to produce.
`override`	A logical argument default `FALSE` indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A data frame with yearly summaries.

Examples

```
#annual_rainfall_summaries(country = "zm", station_id = "01122", summaries = "annual_rain")
#annual_rainfall_summaries(country = "zm", station_id = "16", summaries = c("start_rains", "end_rains", "annua
```

annual_temperature_summaries
Annual Temperature Summaries

Description

Annual Temperature Summaries

Usage

```
annual_temperature_summaries(
  country,
  station_id,
  call = c("climsoft", "googlebuckets"),
 summaries = c("mean_tmin", "mean_tmax", "min_tmin", "min_tmax", "max_tmin", "max_tmax"),
  override = FALSE
)
```

Arguments

country	character(1) The country code of the data.
station_id	character The id's of the stations to analyse. Either a single value or a vector.
call	A character vector specifying where to call the raw data from if calling raw data.
summaries	character The names of the summaries to produce.
override	A logical argument default FALSE indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A data frame with yearly summaries.

Examples

```
# annual_temperature_summaries(country = "zm", station_id = "16") # made a fake "16" json definitions data
# because it contains temperature data.
```

build_annual_summaries_definitions
Get annual summaries definitions

Description

Retrieves annual summaries definitions including start of rains, end of rains, end of season, seasonal length, and annual rainfall summaries.

Usage

```
build_annual_summaries_definitions(
  data_name,
  data_by_year,
 rain_name = data_book$get_climatic_column_name(data_name = data_name, col_name =
    "rain"),
  start_rains_column,
  start_rains_status_column,
  end_rains_column,
  end_rains_status_column,
  end_season_column,
  end_season_status_column,
  seasonal_length_column
)
```

Arguments

data_name	The name of the data.
data_by_year	A list containing definitions for start of rains, end of rains, end of season, and seasonal length.
rain_name	The name of the rainfall column in the data.
start_rains_column	The name of the start of rains column in the data
start_rains_status_column	The name of the start of rains status column in the data
end_rains_column	The name of the end of rains column in the data
end_rains_status_column	The name of the end of rains status column in the data.
end_season_column	The name of the end of season column in the data.
end_season_status_column	The name of the end of seasons status column in the data.
seasonal_length_column	The name of the seasonal length column in the data

Value

A list of annual summaries definitions.

Examples

```
# Example usage:
#get_annual_summaries_definitions("data_name", data_by_year, data)
```

build_crop_definitions *Build Crop Definitions from File*

Description

This function reads crop definition data from a provided file structure generated in R-Instat. It then extracts information about water requirements, planting dates, and planting length for different crops. The extracted values are then split into lists..

Usage

```
build_crop_definitions(definition_file = NULL)
```

Arguments

definition_file	A list containing file data and attributes generated in R-Instat with named vectors `Var1`, `Var2`, and `Var3` for water requirements, planting dates, and planting length respectively.

Value

A list representing the structured crop definition data, including water requirements, planting dates, and planting length.

Examples

```
# Assuming definition_file is a correctly structured list:
#get_crop_definitions(definition_file)
```

build_season_start_probabilities *Build Season Start Probabilities from File*

Description

This function processes a file structure to extract information about the specified day for season start probabilities. The information is split into lists

Usage

```
build_season_start_probabilities(definition_file = NULL)
```

Arguments

definition_file	A list containing file data and attributes generated in R-Instat.

Value

A list representing the season start probabilities with the specified days.

Examples

```
#get_season_start_probabilities(definition_file)
```

`build_total_temperature_summaries` *Calculate total temperature summaries*

Description

Calculates total temperature summaries based on provided parameters.

Usage

```
build_total_temperature_summaries(
  year = data_book$get_climatic_column_name(data_name, "year"),
  month = data_book$get_climatic_column_name(data_name, "month"),
  data_by_year,
  data_by_year_month,
  min_tmin_column,
  mean_tmin_column,
  max_tmin_column,
  min_tmax_column,
  mean_tmax_column,
  max_tmax_column
)
```

Arguments

`year`	Character vector specifying the year.
`month`	Character vector specifying the month.
`data_by_year`	A list of temperature summaries by definition (e.g., year).
`data_by_year_month`	An optional second list of temperature summaries by definition (e.g., year and month).
`min_tmin_column`	The name of the minimum of minimum temperature column in the data.
`mean_tmin_column`	The name of the mean of minimum temperature column in the data.
`max_tmin_column`	The name of the maximum of minimum temperature column in the data.
`min_tmax_column`	The name of the minimum of maximum temperature column in the data.
`mean_tmax_column`	The name of the mean of maximum temperature column in the data.
`max_tmax_column`	The name of the maximum of maximum temperature column in the data.

Value

A list containing total temperature summaries.

Examples

```
# Example usage:
#total_temperature_summaries(tmin = "tmin", tmax = "tmax", year = "year", month = "month",
#                    data_by_year = my_definition_list, data_by_year_month = my_definition_list_2)
```

---

check_and_rename_variables

Check and Rename Variables in a Dataset

Description

This function checks the variable names in a dataset and renames them based on the provided `data_names` vector.

Usage

```
check_and_rename_variables(data, data_names)
```

Arguments

`data`	A data frame or data table containing the dataset.
`data_names`	A named character vector where the names are the standard variable names, and the values are the corresponding variable names in the dataset.

Value

A data frame with variable names renamed according to `data_names`.

Examples

```
# Example: Check and rename variables in a dataset
data <- data.frame(station_name = c("A", "B", "C"), date = c("2022-01-01", "2022-01-02", "2022-01-03"),
                   tmax = c(25, 26, 24), tmin = c(15, 16, 14))
data_names <- c(station = "station_name", date = "date", tmax = "tmax", tmin = "tmin")
renamed_data <- check_and_rename_variables(data, data_names)
```

collate_definitions_data *Collate Definitions Data for Climatic Analysis from R-Instat*

Description

This function aggregates various climatic data definitions, including annual summaries, temperature summaries, crop data, and probabilities of season starts. It is designed to work within a specific context that involves climatic data processing and analysis, particularly focusing on data related to Ghana's climate. The function uses multiple sources of data and calculations to generate a comprehensive list-formatted summary.

Usage

```
collate_definitions_data(
  data_by_year = "ghana_by_station_year",
  data_by_year_month = NULL,
  crop_data = "crop_def",
  rain = data_book$get_climatic_column_name(data_name = "ghana", "rain"),
  year = data_book$get_climatic_column_name("ghana", "year"),
  month = data_book$get_climatic_column_name("ghana", "month"),
  summaries = c("annual_rainfall", "annual_temperature", "monthly_temperature",
    "extremes", "crop_success", "start_season"),
  start_rains_column = "start_rain",
  start_rains_status_column = "start_rain_status",
  end_rains_column = "end_rains",
  end_rains_status_column = "end_rain_status",
  end_season_column = "end_season",
  end_season_status_column = "end_season_status",
  seasonal_length_column = "seasonal_length",
  min_tmin_column = "min_tmin",
  mean_tmin_column = "mean_tmin",
  max_tmin_column = "max_tmin",
  min_tmax_column = "min_tmax",
  mean_tmax_column = "mean_tmax",
  max_tmax_column = "max_tmax"
)
```

Arguments

data_by_year	The name of the data set that contains data aggregated by year, default is "ghana_by_station_year".
data_by_year_month	The name of the data set that contains data aggregated by year and month, default is NULL.
crop_data	The name of the crop data set, default is "crop_def".
rain	The name of the column containing rainfall data.
year	The name of the column containing year data.
month	The name of the column containing month data.
summaries	The name of the summaries to show. Options are "annual_rainfall", "annual_temperature", "monthly_temperature", "extremes", "crop_success", "start_season".

start_rains_column
: The name of the start of rains column in the data.

start_rains_status_column
: The name of the start of rains status column in the data.

end_rains_column
: The name of the end of rains column in the data.

end_rains_status_column
: The name of the end of rains status column in the data.

end_season_column
: The name of the end of season column in the data.

end_season_status_column
: The name of the end of seasons status column in the data.

seasonal_length_column
: The name of the seasonal length column in the data.

min_tmin_column
: The name of the minimum of minimum temperature column in the data.

mean_tmin_column
: The name of the mean of minimum temperature column in the data.

max_tmin_column
: The name of the maximum of minimum temperature column in the data.

min_tmax_column
: The name of the minimum of maximum temperature column in the data.

mean_tmax_column
: The name of the mean of maximum temperature column in the data.

max_tmax_column
: The name of the maximum of maximum temperature column in the data.

Value

A list that contains the aggregated data definitions.

Examples

```
#data_book <- list(get_climatic_column_name = function(data_name, col_name) { return(col_name) },
#                  get_calculations = function(data_name) { list() },
#                  get_data_frame_metadata = function(data_name) { list() })
#collate_definitions_data(data_book = data_book)
```

---

crop_success_probabilities

Probability Crop Tables

Description

The probabilities of crop success for given planting maturity lengths, seasonal total rainfall requirements, and planting dates.

Usage

```
crop_success_probabilities(
  country,
  station_id,
  call = c("climsoft", "googlebuckets"),
  planting_dates = NULL,
  water_requirements = NULL,
  planting_length = NULL,
  start_before_season = NULL,
  override = FALSE
)
```

Arguments

country	character(1) The country code of the data.
station_id	character The id's of the stations to analyse. Either a single value or a vector.
call	A character vector specifying where to call the raw data from if calling raw data.
planting_dates	numeric Vector containing planting dates requirements.
water_requirements	numeric Vector containing water requirements requirements.
planting_length	numeric Vector containing seasonal crop length requirements.
start_before_season	logical A logical value indicating whether to check the start day condition (default is TRUE).
override	A logical argument default FALSE indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A list containing the definitions and a data frame with probability summaries.

Examples

```
#
#library(epicsawrap)
#library(tidyverse)
#epicsawrap::setup(dir = getwd())
#epicsawrap::gcs_auth_file(file = "C:/Users/lclem/Downloads/e-picsa-e630400792e7.json")
#crop_success_probabilities(country = "zm", station_id = "16")

# or some can be defined in the dialog
#x <- crop_success_probabilities(country = "zm", station_id = "16", water_requirements = c(100, 300, 800))

# or all can be defined in the dialog
#crop_success_probabilities(country = "zm", station_id = "16", water_requirements = c(100, 300, 800),
#                        planting_length = c(100, 150), planting_dates = c(90, 100, 110))
```

data_definitions *Get Data Column Names*

Description

This function takes a vector of variable names and standardises them by mapping various variations of variable names to their corresponding standard names.

Usage

```
data_definitions(data_names, rename_vars = FALSE, exact_match = TRUE)
```

Arguments

data_names	A character vector containing the variable names in the dataset.
rename_vars	Logical value indicating whether to rename variables to their corresponding standard names. If TRUE, the function returns a list of renamed variables; if FALSE, it returns a list of mappings from variations to standard names.
exact_match	Logical value indicating whether to perform an exact match for variable name variations or not. If TRUE, it matches only exact variable names; if FALSE, it performs a partial match.

Value

A list of standard variable names or a list of mappings from variations to standard names, depending on the value of rename_vars.

Examples

```
# Example 1: Redefine variables without renaming
data_definitions(c("station_id", "station_", "date", "year", "month",
                       "day", "doy", "rain", "tmin", "tmax"),
                       rename_vars = FALSE, exact_match = FALSE)

# Example 2: Redefine variables with renaming
data_definitions(c("date", "year", "month",
                       "day", "doy", "rain", "minimum_temperature", "max_temp"),
                       rename_vars = TRUE, exact_match = FALSE)
```

definitions *Definitions*

Description

Definitions

Usage

```
definitions(
  country,
  station_id = NULL,
  definitions_id = NULL,
  summaries,
  file = NULL
)
```

Arguments

country	character(1) The country code of the data.
station_id	character(1) The definitions code in the data.
definitions_id	character(1) The definitions code in the data.
summaries	character Vector of summaries to display
file	Default NULL meaning that the most recent definitions file will be found and imported. Otherwise specify as a string the file to import. In format: "STATIONNAME.TIMESTAMP" e.g. "1.20240311152831"

Value

TODO

Examples

```
# e.g. definitions("zm", "16", "annual_rain")
# error: definitions("zm", "1", c("annual_rain", "hi", "end_season"))
```

---

export_r_instat_to_bucket

Export R-Instat Data to Google Cloud Storage Bucket

Description

This function exports R-Instat data to a specified Google Cloud Storage bucket. It collates the data and saves it as a JSON file on the local machine, then uploads the file to the specified bucket.

Usage

```
export_r_instat_to_bucket(
  data = NULL,
  data_by_year,
  data_by_year_month = NULL,
  crop_data_name = NULL,
  rain = NULL,
  station = NULL,
  year = NULL,
  month = NULL,
  summaries = c("annual_rainfall", "annual_temperature", "monthly_temperature",
    "extremes", "crop_success", "start_season"),
```

```
  station_id = NULL,
  definitions_id,
  country,
  include_summary_data = FALSE,
  annual_rainfall_data = NULL,
  annual_temperature_data = NULL,
  monthly_temperature_data = NULL,
  crop_success_data = NULL,
  season_start_data = NULL,
  start_rains_column = "start_rain",
  start_rains_status_column = "start_rain_status",
  end_rains_column = "end_rains",
  end_rains_status_column = "end_rains_status",
  end_season_column = "end_season",
  end_season_status_column = "end_season_status",
  seasonal_length_column = "seasonal_length",
  min_tmin_column = "min_tmin",
  mean_tmin_column = "mean_tmin",
  max_tmin_column = "max_tmin",
  min_tmax_column = "min_tmax",
  mean_tmax_column = "mean_tmax",
  max_tmax_column = "max_tmax"
)
```

Arguments

`data`	The main dataset. Depreciated. can be removed after updates to R-Instat dialog.
`data_by_year`	The dataset grouped by year.
`data_by_year_month`	The dataset grouped by year and month.
`crop_data_name`	Name of the crop data used for definitions when `summaries = "crop_success"`.
`rain`	The rainfall data.
`station`	The station variable.
`year`	The year data.
`month`	The month data.
`summaries`	A character vector specifying the types of summaries to include.
`station_id`	`character` The id's of the stations to analyse. Either a single value or a vector.
`definitions_id`	`character` The ID to give to the definition file.
`country`	`character(1)` The country code of the data.
`include_summary_data`	Logical indicating whether to include summary data in the export.
`annual_rainfall_data`	Annual rainfall summary data.
`annual_temperature_data`	Annual temperature summary data.
`monthly_temperature_data`	Monthly temperature summary data.

`crop_success_data`
: The proportion crop data output. Used if `summaries = "crop_success"` when `include_summary_data = TRUE`. This is called `crop_prop` by default in R-Instat.

`season_start_data`
: The crop data to be read into `"season_start_probabilities"`. This is called `crop_def` by default in R-Instat.

`start_rains_column`
: The name of the start of rains column in the data.

`start_rains_status_column`
: The name of the start of rains status column in the data.

`end_rains_column`
: The name of the end of rains column in the data.

`end_rains_status_column`
: The name of the end of rains status column in the data.

`end_season_column`
: The name of the end of season column in the data.

`end_season_status_column`
: The name of the end of season status column in the data.

`seasonal_length_column`
: The name of the seasonal length column in the data.

`min_tmin_column`
: The name of the minimum of minimum temperature column in the data.

`mean_tmin_column`
: The name of the mean of minimum temperature column in the data.

`max_tmin_column`
: The name of the maximum of minimum temperature column in the data.

`min_tmax_column`
: The name of the minimum of maximum temperature column in the data.

`mean_tmax_column`
: The name of the mean of maximum temperature column in the data.

`max_tmax_column`
: The name of the maximum of maximum temperature column in the data.

Details

This function collates the specified data into a JSON format and saves it to the local machine. Then it uploads the JSON file to the specified Google Cloud Storage bucket. If `include_summary_data` is TRUE, it also uploads additional summary data to the bucket.

Value

A message confirming that the data has been uploaded to the bucket.

Examples

```
# Provide examples here if needed
```

extract_most_recent_json *Extract the most recent JSON file from a list of filenames*

Description

This function takes a list of filenames as input and returns the filename of the most recent JSON file. The filenames should be in the format "definitions/1.YYYYMMDDHHMMSS.json".

Usage

```
extract_most_recent_json(files)
```

Arguments

files	A character vector containing filenames.

Value

A character string representing the filename of the most recent JSON file.

Examples

```
files <- c(
  "definitions/1.20240116155433.json",
  "definitions/1.20240304125111.json",
  "definitions/1.json"
)
```

extract_value *Extract Value*

Description

Extracts a specific value from a string using a regular expression.

Usage

```
extract_value(string, value_expr, as_numeric = TRUE)
```

Arguments

string	The input string.
value_expr	The regular expression pattern to extract the value.
as_numeric	Logical indicating whether the extracted value should be converted to numeric.

Value

The extracted value.

Examples

```
# Example usage:
extract_value("Example string with value: 123.45", "\\d+(\\.\\d+)?")
```

`extremes_summaries` *Get the Extreme Data*

Description

This function identifies extreme values in a specified element (column) of a data frame. It can operate in two modes: percentile-based and threshold-based.

Usage

```
extremes_summaries(
  country,
  station_id,
  summaries = c("extremes_rain", "extremes_tmin", "extremes_tmax"),
  override = FALSE
)
```

Arguments

`country`	A character string specifying the country code of the data.
`station_id`	A character vector specifying the ID(s) of the station(s) to analyse.
`summaries`	A character vector specifying the names of the summaries to produce.
`override`	A logical argument default `FALSE` indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A data frame containing the extreme data.

Examples

```
# Generate annual temperature summaries for station 16 in Zambia
#extremes_summaries(country, station_id, c("extremes_rain"))
```

gcs_auth_file *GCS Auth with json file*

Description

This function authenticates the user for Google Cloud Storage (GCS) using a JSON key file.

Usage

```
gcs_auth_file(filename)
```

Arguments

filename	The path to the JSON key file for GCS authentication.

Value

None

References

For more information on GCS authentication, refer to the official documentation: https://cloud.google.com/storage/docs/r
libraries-usage-r

Examples

```
# gcs_auth_file("path/to/key.json")
```

get_binary_file *Update PDF/JPEG Data*

Description

This function updates the PDF/JPEG data for a specific station in the specified country. It retrieves the data from Google Cloud Storage using the `get_data` function.

Usage

```
get_binary_file(country = c("mz", "zm"), station_id, type = c("pdf", "jpeg"))
```

Arguments

country	A character vector specifying the country or countries from which to update the data. Options are "mz" and "zm".
station_id	A character string specifying the ID of the station for which to update the daily data.
type	A character string specifying whether the data to retrieve is JPEG or PDF.

Details

The country argument is a character vector that allows specifying one or more countries from which to get the PDF/JPEG data. The data will be updated for Mozambique ("mz") and Zambia ("zm"). You can modify this argument to update data for different countries. The station_id argument is a character string that specifies the ID of the station for which to update the PDF/JPEG data. The function will construct the filename by concatenating the "pdf/" or "jpeg"/ directory, the station_id, and the file extension. The filename will be passed to the get_data function to retrieve the data. The function uses the invisible function to suppress the output of the get_data function, ensuring that the data retrieval process is not visible in the console.

Value

This function does not return any value explicitly. It gets the PDF/JPEG data for the specified station in the specified country.

Examples

```
# get_binary_file("zm", "16", "pdf")
```

---

get_bucket_name *Get Bucket Name*

Description

Get Bucket Name

Usage

```
get_bucket_name(
  country = c("mw", "zm", "zm_test", "ml_test", "mw_test", "ke_test", "internal_tests",
    "zm_workshops", "mw_workshops")
)
```

Arguments

country	A character vector specifying the country or countries from which to update the data. Options are "mz", "zm", "zm_test", "ml_test", "ke_test".

Value

Returns the name of the bucket for the Malawi or Zambia data.

Examples

```
#get_bucket_name("mw")
```

get_climsoft_conn *Get Climsoft Connection*

Description

Retrieves the stored Climsoft database connection from the package environment.

Usage

```
get_climsoft_conn()
```

Value

The database connection object.

Examples

```
#con <- get_climsoft_conn()
```

get_daily_data *Get Daily Data*

Description

Get Daily Data

Usage

```
get_daily_data(country, station_id, call_from = c("climsoft", "googlebuckets"))
```

Arguments

country	A character vector specifying the country or countries from which to get the data. Common options are "mz", "zm", and "zm_test". Any defined in get_bucket_name().
station_id	A character string specifying the ID of the station for which to get the daily data.
call_from	A character string specifying the location of the raw data.

Value

A data frame containing the daily data for the specified station and country.

Examples

```
#
```

get_data *Get data from Google Cloud Storage*

Description

This function retrieves data from Google Cloud Storage for a specified country and file. The data can either be parsed and returned as an R object or saved to a local disk.

Usage

```
get_data(country, filename, save_to = NULL)
```

Arguments

country	A character string specifying the country.
filename	A character string specifying the name of the file in Google Cloud Storage.
save_to	(Optional) A character string specifying the local path where the file should be saved. If not provided, the data will be parsed and returned as an R object.

Details

The `get_data` function retrieves data from a specified country and file stored in Google Cloud Storage. It uses the `googleCloudStorageR` package If the `save_to` argument is not provided or set to `NULL`, the function will parse the data and return it as an R object using the `googleCloudStorageR::gcs_parse_rds` function. This is useful when you want to directly work with the data in R, and can be more efficient. If the `save_to` argument is provided with a valid local path, the function will save the file to the specified location on the disk using the `saveToDisk` parameter. This is useful when you want to download the file for further processing or analysis outside of R.

Value

The function returns the retrieved data as an R object if `save_to` is `NULL`. If `save_to` is provided, the function saves the data locally.

Examples

```
# TODO
```

get_definitions_data *Get Daily Definitions Data*

Description

This function retrieves definitions data for weather stations from a Google Cloud Storage (GCS) bucket. It includes timestamp handling to ensure that the most recent definitions file is imported.

Usage

```
get_definitions_data(country, station_id, definitions_id = NULL, file = NULL)
```

Arguments

country	A character vector specifying the country or countries from which to get the definitions data. Options are any defined in get_bucket_name(). Common options are "mz" and "zm".
station_id	A character string specifying the ID of the station for which to get the definitions data.
definitions_id	A character string specifying the ID of the definitions for which to get the definitions data. If NULL this is found from the metadata.
file	Default NULL meaning that the most recent definitions file will be found and imported. Otherwise specify as a string the file to import. In format: "STATIONNAME.TIMESTAMP" e.g. "1.20240311152831"

Value

A data frame containing daily data based on the station ID.

See Also

update_definitions_data for updating definitions files.

Examples

```
# todo
```

---

get_definitions_id_from_metadata

Get Definitions ID from Metadata

Description

This function retrieves the definitions ID from station metadata for a given country and station ID.

Usage

```
get_definitions_id_from_metadata(country, station_id)
```

Arguments

country	A character string representing the country code.
station_id	A character string representing the station ID.

Value

A character string representing the definitions ID from the station metadata.

get_end_rains_definitions *Get end rains definitions*

Description

Retrieves end rains definitions.

Usage

```
get_end_rains_definitions(end_rains = NULL)
```

Arguments

end_rains	The end rains data.

Value

A list representation of end rains definitions.

Examples

```
# Example usage:
#get_end_rains_definitions(end_rains)
```

get_end_season_definitions *Get end of season definitions*

Description

Retrieves end season definitions.

Usage

```
get_end_season_definitions(end_season = NULL)
```

Arguments

end_season	The end season data.

Value

A list representation of end season definitions.

Examples

```
# Example usage:
#get_end_season_definitions(end_season)
```

get_forecast_data *Get Forecast Data*

Description

Get Forecast Data

Usage

```
get_forecast_data(country, station_id)
```

Arguments

country	A character vector specifying the country or countries from which to get the forecast data. Options are defined in `get_bucket_name()`.
station_id	A character string specifying the ID of the station for which to get the forecast data.

Value

todo

Examples

```
# todo
```

get_objects_in_bucket *Import Summary Definitions*

Description

This function imports summary definitions based on the country, station ID, summaries, and get_summaries. It checks if there are any files corresponding to the provided station and summary in the Google Cloud Storage bucket. If files are found, it imports definitions from the file; otherwise, it generates new definitions.

Usage

```
get_objects_in_bucket(country, station_id, timestamp)
```

Arguments

country	A character string specifying the country.
station_id	A character string specifying the station ID.
timestamp	The timestamp on the object file name to import. Default `NULL`

Value

A list containing imported or generated summary definitions.

Examples

```
# Import summary definitions
#import_summary_definitions("USA", "station123", list("info1", "info2"))
```

`get_offset_term` *Get Offset Term*

Description

This function retrieves the start day of the year (DOY) from the metadata of the given data. If there are multiple start DOYs, it issues a warning and selects the first one.

Usage

```
get_offset_term(data_by_year)
```

Arguments

`data_by_year`	A data object from which to retrieve the start DOY. This data object is expected to have an associated metadata containing a `doy_start` field.

Value

Returns a single start DOY value if found in the metadata; otherwise, returns `NULL`.

Examples

```
#data <- some_function_to_get_data()
#offset_term <- get_offset_term(data)
#print(offset_term)
```

`get_r_instat_definitions` *Get R-Instat definitions*

Description

Retrieves R-Instat definitions based on given calculations.

Usage

```
get_r_instat_definitions(calculation)
```

Arguments

`calculation`	A list of calculations.

Value

A list of R-Instat definitions.

Examples

```
# Example usage:
#get_r_instat_definitions()
```

`get_season_length_definitions` *Get season length definitions*

Description

Retrieves season length definitions.

Usage

```
get_season_length_definitions(length = NULL)
```

Arguments

`length`	The season length data.

Value

A list representation of season length definitions.

Examples

```
# Example usage:
#get_season_length_definitions(length)
```

`get_start_rains_definitions` *Get start of rains definitions*

Description

Retrieves start rains definitions.

Usage

```
get_start_rains_definitions(start_rains = NULL)
```

Arguments

`start_rains`	The start rains data.

Value

A list of start of rains definitions

Examples

```
# Example usage:
#get_start_rains_definitions(start_rains)
```

get_summaries_data *Get Summaries Data*

Description

Get Summaries Data

Usage

```
get_summaries_data(country, station_id, summary)
```

Arguments

country	A character vector specifying the country or countries from which to get the data. Options are defined in `get_bucket_name()` (e.g., `"zm"`, `"mw"`).
station_id	A character string specifying the ID of the station for which to get the summary data.
summary	A character string specifying the summary to retrieve.

Value

A list of data frames containing the summary data for the specified stations and country.

Examples

```
#
```

get_temperature_summary_definitions *Get temperature summary definitions*

Description

Retrieves temperature summary definitions based on provided parameters.

Usage

```
get_temperature_summary_definitions(
  year = data_book$get_climatic_column_name(data_name, "year"),
  month = data_book$get_climatic_column_name(data_name, "month"),
  data_by_year,
  data_by_year_month = NULL,
  min_tmin_column,
  mean_tmin_column,
  max_tmin_column,
  min_tmax_column,
  mean_tmax_column,
  max_tmax_column
)
```

Arguments

year	Character vector specifying the year.
month	Character vector specifying the month.
data_by_year	A list of temperature summaries by definition (e.g., year).
data_by_year_month	An optional second list of temperature summaries by definition (e.g., year and month).
min_tmin_column	The name of the minimum of minimum temperature column in the data.
mean_tmin_column	The name of the mean of minimum temperature column in the data.
max_tmin_column	The name of the maximum of minimum temperature column in the data.
min_tmax_column	The name of the minimum of maximum temperature column in the data.
mean_tmax_column	The name of the mean of maximum temperature column in the data.
max_tmax_column	The name of the maximum of maximum temperature column in the data.

Value

A list containing temperature summary definitions.

Examples

```
# Example usage:
#get_temperature_summary_definitions(by_definition_list = my_definition_list, by_definition_2_list = my_defi
```

---

get_temp_summaries *Get temperature summaries*

Description

Retrieves temperature summaries based on provided parameters.

Usage

```
get_temp_summaries(
  temp_summary_name,
  year,
  month,
  data_by_year,
  data_by_year_month = NULL
)
```

Arguments

temp_summary_name	Character vector specifying the name of the temperature summary.
year	Numeric vector specifying the year.
month	Numeric vector specifying the month.
data_by_year	A list of temperature summaries by definition (e.g., year).
data_by_year_month	An optional second list of temperature summaries by definition (e.g., year and month).

Value

A list containing temperature summary information.

Examples

```
# Example usage:
#get_temp_summaries("summary_name", year = 2023, month = 5, data_by_year = my_definition_list)
```

---

get_total_rain_counts *Get annual rain definitions*

Description

Retrieves annual rain definitions.

This function retrieves the definition for the total rainfall as well as the total number of rainy days for both annual and seasonal periods.

Usage

```
get_total_rain_counts(
  data_by_year = NULL,
 rain_name = data_book$get_climatic_column_name(data_name = "ghana", col_name = "rain")
)

get_total_rain_counts(
  data_by_year = NULL,
 rain_name = data_book$get_climatic_column_name(data_name = "ghana", col_name = "rain")
)
```

Arguments

data_by_year	List containing data by year.
rain_name	Character string specifying the name of the rainfall data.
data_name	The name of the main data frame.

Value

A list representation of annual rain definitions.

A list containing definitions of total rain counts for annual and seasonal periods.

Examples

```
# Example usage:
#get_total_rain_counts(annual_rain, ghana_defs)

#data <- your_data
#get_total_rain_counts(data_name = "ghana", data_by_year = data, rain_name = "rain")
```

`import_from_climsoft` *Import Data from Climsoft*

Description

Connects to a Climsoft database and imports data based on the specified filters for stations and elements, with options to include observation flags and station information.

Usage

```
import_from_climsoft(
  con = get_climsoft_conn(),
  stationfiltercolumn = "stationId",
  stations = c(),
  elementfiltercolumn = "elementId",
  elements = c(),
  include_observation_flags = FALSE,
  include_station_info = FALSE,
  unstack_data = TRUE,
  start_date = NULL,
  end_date = NULL
)
```

Arguments

`con`	Connection object to the Climsoft database, default is the result of `get_climsoft_conn()`.
`stationfiltercolumn`	Name of the column to filter by stations, default is 'stationId'.
`stations`	Vector of station IDs to filter the data, defaults to an empty vector.
`elementfiltercolumn`	Name of the column to filter by elements, default is 'elementId'.
`elements`	Vector of element IDs to filter the data, defaults to an empty vector.
`include_observation_flags`	Boolean, if TRUE includes observation flags in the output, defaults to FALSE.
`include_station_info`	Boolean, if TRUE includes station metadata in the output, defaults to FALSE.
`unstack_data`	Boolean. Option to unstack data once read in.
`start_date`	Start date for filtering the observations, format should be Date, defaults to NULL.
`end_date`	End date for filtering the observations, format should be Date, defaults to NULL.

Value

A list containing Climsoft station and observation data based on the filters applied. If `include_station_info` is TRUE, the list will have two elements: 'Metadata' with station details and 'Daily data' with observations.

Examples

```
#con <- get_climsoft_conn()
#data <- import_from_climsoft(con, stations = c("101", "102"), elements = c("1", "2"), start_date = as.Date("20
```

`join_null_data` *Join Null Data*

Description

This function joins two data frames, `summary_data` and `calculated_data`, using a full join if `summary_data` is not NULL. If `summary_data` is NULL, it assigns `calculated_data` to `summary_data`.

Usage

```
join_null_data(summary_data, calculated_data)
```

Arguments

`summary_data`	A data frame representing summary data.
`calculated_data`	A data frame containing calculated data.

Value

A data frame resulting from the full join of `summary_data` and `calculated_data`, or `calculated_data` if `summary_data` is NULL.

Examples

```
# summary_data is NULL
summary_data <- NULL
calculated_data <- data.frame(x = 1:5, y = letters[1:5])
join_null_data(summary_data, calculated_data)

# summary_data is not NULL
summary_data <- data.frame(x = 1:3, y = letters[1:3])
calculated_data <- data.frame(x = 4:5, y = letters[4:5])
join_null_data(summary_data, calculated_data)
```

monthly_temperature_summaries

Monthly Temperature Summaries

Description

Monthly Temperature Summaries

Usage

```
monthly_temperature_summaries(
  country,
  station_id,
  call = c("climsoft", "googlebuckets"),
 summaries = c("mean_tmin", "mean_tmax", "min_tmin", "min_tmax", "max_tmin", "max_tmax"),
  override = FALSE
)
```

Arguments

country	character(1) The country code of the data.
station_id	character The id's of the stations to analyse. Either a single value or a vector.
call	A character vector specifying where to call the raw data from if calling raw data.
summaries	character The names of the summaries to produce.
override	A logical argument default FALSE indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A data frame with monthly summaries.

Examples

```
# monthly_temperature_summaries(country = "zm", station_id = "1", summaries = c("min_tmin"))
# because it contains temperature data.
```

overall_extremes_summaries

Generate summary statistics for extreme weather events.

Description

This function generates summary statistics for extreme weather events based on given definitions.

Usage

```
overall_extremes_summaries(daily, data_names, definitions, summaries)
```

Arguments

daily	A dataframe containing daily weather data.
data_names	A list containing the data names in the daily data.
definitions	A list containing definitions of extreme events.
summaries	Name of the summary to be generated.

Value

A dataframe containing summary statistics for extreme events.

Examples

```
# Example usage:
# Generate summary statistics for extreme rain events
#rain_summary <- overall_extremes_summaries(daily_data, definitions_list, "extremes_rain")
```

---

reformat_annual_summaries

Reformat annual summaries data

Description

This function reformats annual summaries data by renaming columns and converting data types.

Usage

```
reformat_annual_summaries(
  data,
  station_col = NULL,
  year_col = NULL,
  start_rains_doy_col = NULL,
  start_rains_date_col = NULL,
  end_rains_doy_col = NULL,
  end_rains_date_col = NULL,
  end_season_doy_col = NULL,
  end_season_date_col = NULL,
  seasonal_rain_col = NULL,
  n_seasonal_rain_col = NULL,
  season_length_col = NULL,
  annual_rain_col = NULL,
  n_rain_col = NULL
)
```

Arguments

data	A data frame containing the annual summaries data.
station_col	Name of the column containing station information.
year_col	Name of the column containing year information.
start_rains_doy_col	Name of the column containing start of rains day of year.

start_rains_date_col
: Name of the column containing start of rains date.

end_rains_doy_col
: Name of the column containing end of rains day of year.

end_rains_date_col
: Name of the column containing end of rains date.

end_season_doy_col
: Name of the column containing end of season day of year.

end_season_date_col
: Name of the column containing end of season date.

seasonal_rain_col
: Name of the column containing seasonal rain data.

n_seasonal_rain_col
: Name of the column containing number of seasonal rain events.

season_length_col
: Name of the column containing season length.

annual_rain_col
: Name of the column containing annual rain data.

n_rain_col
: Name of the column containing number of rain events.

Value

The reformatted data frame.

---

reformat_crop_success *Reformat crop success data*

Description

This function reformats crop success data by renaming columns and converting data types.

Usage

```
reformat_crop_success(
  data,
  station_col = NULL,
  total_rain_col,
  plant_day_col,
  plant_length_col,
  prop_success_col
)
```

Arguments

data
: A data frame containing the crop success data.

station_col
: Name of the column containing station information.

total_rain_col
: Name of the column containing total rain data.

plant_day_col
: Name of the column containing plant day data.

plant_length_col
: Name of the column containing plant length data.

prop_success_col
: Name of the column containing proportion of success data.

Value

The reformatted data frame.

`reformat_season_start` *Reformat season start data*

Description

This function reformats season start data by renaming columns and calculating proportions.

Usage

```
reformat_season_start(
  data,
  station_col = NULL,
  year_col,
  plant_day_col,
  plant_day_cond_col
)
```

Arguments

`data`	A data frame containing the season start data.
`station_col`	Name of the column containing station information.
`year_col`	Name of the column containing year information.
`plant_day_col`	Name of the column containing plant day data.
`plant_day_cond_col`	Name of the column containing plant day condition data.

Value

The reformatted data frame.

`reformat_temperature_summaries`
Reformat temperature summaries data

Description

This function reformats temperature summaries data by renaming columns and converting data types.

Usage

```
reformat_temperature_summaries(
  data,
  station_col = NULL,
  year_col = NULL,
  month_col = NULL,
  mean_tmin_col = NULL,
  min_tmin_col = NULL,
  max_tmin_col = NULL,
  mean_tmax_col = NULL,
  min_tmax_col = NULL,
  max_tmax_col = NULL
)
```

Arguments

data	A data frame containing the temperature summaries data.
station_col	Name of the column containing station information.
year_col	Name of the column containing year information.
month_col	Name of the column containing month information.
mean_tmin_col	Name of the column containing mean minimum temperature data.
min_tmin_col	Name of the column containing minimum minimum temperature data.
max_tmin_col	Name of the column containing maximum minimum temperature data.
mean_tmax_col	Name of the column containing mean maximum temperature data.
min_tmax_col	Name of the column containing minimum maximum temperature data.
max_tmax_col	Name of the column containing maximum maximum temperature data.

Value

The reformatted data frame.

---

season_start_probabilities

Season start date probabilities

Description

A table containing the probabilities of the season starting on or before a set of particular dates.

Usage

```
season_start_probabilities(
  country,
  station_id,
  call = c("climsoft", "googlebuckets"),
  start_dates = NULL,
  override = FALSE
)
```

Arguments

country	character(1) The country code of the data.
station_id	character The id's of the stations to analyse. Either a single value or a vector.
call	A character vector specifying where to call the raw data from if calling raw data.
start_dates	numeric A vector of start dates (in doy format) to calculate the probabilities of the season starting on or before.
override	A logical argument default FALSE indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A list containing the definitions and a data frame with probability summaries.

Examples

```
#
#library(epicsawrap)
#library(tidyverse)
#epicsawrap::setup(dir = getwd())
#epicsawrap::gcs_auth_file(file = "C:/Users/lclem/Downloads/e-picsa-e630400792e7.json")
#season_start_probabilities(country = "zm", station_id = "16")
# or you can manually define
#season_start_probabilities(country = "zm", station_id = "16", start_dates = c(10, 20, 100))
```

---

setup *Setup directories*

Description

Setup directories

Usage

```
setup(dir, countries = c("mw", "zm", "zm_test"))
```

Arguments

dir	character(1) The path to set as the working directory
countries	character(1) The set of countries to create directories for.

Value

TODO

Examples

```
#TODO
```

set_climsoft_conn *Set Climsoft Connection*

Description

Establishes a connection to a Climsoft database and stores it in a package environment for later use.

Usage

```
set_climsoft_conn(dbname, user, password, host, port)
```

Arguments

dbname	Name of the database.
user	Username for database access.
password	Password for database access.
host	Host where the database server is located.
port	Port number on which the database server is running.

Value

Invisible. The function does not return anything but stores the connection in a designated package environment.

Examples

```
#set_climsoft_conn("climsoft_db", "user", "password", "localhost", "3306")
```

split_list *Split Parameter Into List*

Description

This function takes a vector where each element is a string formatted as "identifier=val" and splits each element into an identifier and its corresponding numeric value. Each numeric value is then stored in a list with a dynamically generated name based on its index.

Usage

```
split_list(parameter)
```

Arguments

parameter	A character vector where each element is a string with the format "identifier=value".

Value

A list where each element is the numeric value extracted from the input vector, named dynamically as "val1", "val2", etc., corresponding to their original order in the input vector.

Examples

```
#parameter <- c("A1=100", "A2=200", "A3=150")
#split_list(parameter)
```

`station_metadata` *Get Station Metadata*

Description

This function retrieves metadata for weather stations in specified countries.

Usage

```
station_metadata(
  country = NULL,
  station_id = NULL,
  include_definitions_id = TRUE,
  include_definitions = FALSE,
  format = c("wide", "long", "nested", "list")
)
```

Arguments

`country`	A character vector specifying the country or countries from which to get the metadata. Options are defined in `get_bucket_name()` (e.g., `"zm"`, `"mw"`).
`station_id`	A character vector specifying the station IDs to filter by. If provided, only metadata for the specified station IDs will be returned.
`include_definitions_id`	A logical value indicating whether to include the definitions id. If `TRUE`, definitions_id is given.
`include_definitions`	A logical value indicating whether to include definitions data. If `TRUE`, additional information about station definitions will be included in the output.
`format`	A character vector specifying the format of the output. Options are `"wide"` (default), `"long"`, `"nested"`, or `"list"`.

Value

If `include_definitions` is FALSE, the function returns a data frame with metadata for the specified stations. If `include_definitions` is `TRUE`, it returns a data frame with both metadata and station definitions.

See Also

`update_metadata` for updating metadata files.

Examples

```
# TODO
```

`station_metadata_definitions` *Get Processed Station Metadata*

Description

This function retrieves and processes station metadata for the specified country and format. The station metadata includes information about station IDs and their associated definitions.

Usage

```
station_metadata_definitions(
  country,
  station_id,
  format = c("wide", "long", "nested", "list")
)
```

Arguments

`country`	A character vector specifying the country code for which station metadata should be retrieved and processed. Options are defined in `get_bucket_name()` (e.g., `"zm"`, `"mw"`).
`station_id`	A character vector specifying the station ID(s) for the given country.
`format`	A character vector indicating the desired format of the processed data. It can be "wide", "long", "nested", or "list".

Value

Depending on the specified format, the function returns the processed station metadata in either wide, long, nested, or list format.

Examples

```
# Retrieve and process station metadata for country "zm" in wide format
#station_metadata_definitions(country = "zm", format = "wide")

# Retrieve and process station metadata for countries "zm" and "mw" in long format
#station_metadata_definitions(country = c("zm", "mw"), format = "long")

# Retrieve and process station metadata for country "zm" in nested format
#station_metadata_definitions(country = "zm", format = "nested")
```

sum_rain_definitions *Summarize Rain Definitions*

Description

This function summarises rain definitions for a specific time period.

Usage

```
sum_rain_definitions(
  time = "annual_rain",
  total_rain,
  n_rain,
  sum_rain,
  n_raindays,
  data = NULL
)
```

Arguments

time	A character string specifying the time period for which the rain definitions are summarised ("annual_rain" or "seasonal_rain"").
total_rain	Logical indicating whether total rain is considered.
n_rain	Logical indicating whether the number of rainy days is considered.
sum_rain	Numeric vector containing the sum of rainfall.
n_raindays	Numeric vector containing the number of rainy days.
data	Optional additional data (default NULL).

Value

A list containing summarised rain definitions for the specified time period.

Examples

```
# TODO
```

total_temperature_summaries *Generate Annual and Monthly Temperature Summaries*

Description

This function calculates annual or monthly temperature summaries for specified stations in a given country.

Usage

```
total_temperature_summaries(
  country,
  station_id,
  call = c("climsoft", "googlebuckets"),
 summaries = c("mean_tmin", "mean_tmax", "min_tmin", "min_tmax", "max_tmin", "max_tmax"),
  to = c("annual", "monthly"),
  override = FALSE
)
```

Arguments

country	A character string specifying the country code of the data.
station_id	A character vector specifying the ID(s) of the station(s) to analyse.
call	A character vector specifying where to call the raw data from if calling raw data.
summaries	A character vector specifying the names of the summaries to produce.
to	A character string indicating whether the summaries should be generated for "annual" or "monthly" data.
override	A logical argument default FALSE indicating whether to calculate the summaries still, even if they are stored already in the bucket.

Value

A data frame containing the requested temperature summaries.

Examples

```
# Generate annual temperature summaries for station 16 in Zambia
#total_temperature_summaries(country = "zm", station_id = "1", summaries = c("mean_tmin", "mean_tmax", "min_tm
```

---

update_daily_data *Get the Daily Data*

Description

This function updates the daily data for a specific station in the specified country. It retrieves the data from Google Cloud Storage using the get_data function.

Usage

```
update_daily_data(country, station_id)
```

Arguments

country	A character vector specifying the country or countries from which to update the data. Options are defined in get_bucket_name() (e.g., "zm", "mw").
station_id	A character string specifying the ID of the station for which to update the daily data.

Details

The update_daily_data function is used to update the daily data for a specific station in the specified country. It internally calls the get_data function to retrieve the data from Google Cloud Storage. The country argument is a character vector that allows specifying one or more countries from which to update the data. The data will be updated for Mozambique ("mz") and Zambia ("zm"). You can modify this argument to update data for different countries. The station_id argument is a character string that specifies the ID of the station for which to update the daily data. The function will construct the filename by concatenating the "data/" directory, the station_id, and the file extension ".rds". The filename will be passed to the get_data function to retrieve the data. The function uses the invisible function to suppress the output of the get_data function, ensuring that the data retrieval process is not visible in the console.

Value

This function does not return any value explicitly. It gets the daily data for the specified station in the specified country.

Examples

```
# todo
```

---

update_definitions_data

Get the Definitions Data

Description

This function updates the definitions data for a specific station in the specified country. It retrieves the data from Google Cloud Storage using the get_data function.

Usage

```
update_definitions_data(country, station_id)
```

Arguments

country	A character vector specifying the country or countries from which to update the data. Options are defined in get_bucket_name() (e.g., "zm", "mw").
station_id	A character string specifying the ID of the station for which to update the daily data.

Details

The update_definitions function is used to update the daily data for a specific station in the specified country. It internally calls the get_data function to retrieve the data from Google Cloud Storage. The country argument is a character vector that allows specifying one or more countries from which to update the data. The data will be updated for Mozambique ("mz") and Zambia ("zm"). You can modify this argument to update data for different countries. The station_id argument is a character string that specifies the ID of the station for which to update the daily data. The function will construct the filename by concatenating the "definitions/" directory, the station_id, and the file extension ".json". The filename will be passed to the get_data function to retrieve the data. The function uses the invisible function to suppress the output of the get_data function, ensuring that the data retrieval process is not visible in the console.

Value

This function does not return any value explicitly. It gets the daily data for the specified station in the specified country.

Examples

```
# todo
```

`update_metadata` *Update Station Metadata*

Description

Update Station Metadata

Usage

```
update_metadata(country)
```

Arguments

`country`	A character vector specifying the country or countries from which to update the metadata. Options are defined in `get_bucket_name()` (e.g., `"zm"`, `"mw"`).

Value

This function updates the metadata for the specified station in the specified country.

Examples

```
#todo
```

`update_metadata_definition_id` *Update Metadata Definition ID*

Description

This function updates the definitions ID in the station metadata for a given country and station ID. It can either overwrite the existing definitions ID or append a new one.

Usage

```
update_metadata_definition_id(
  country,
  station_id,
  definition_id,
  overwrite = FALSE,
  add_climsoft = FALSE,
  elementfiltercolumn = "elementName",
  elements = c("Temp  Daily Max", "Temp  Daily Min", "Precip  Daily")
)
```

Arguments

country	A character string representing the country code.
station_id	A character string representing the station ID.
definition_id	A character string representing the new definition ID to be added.
overwrite	A logical value indicating whether to overwrite the existing definitions ID.
add_climsoft	A logical value indicating whether to add in climsoft details. This will add in the values from elementfiltercolumn and elements.
elementfiltercolumn	Name of the column to filter by elements, default is 'elementName'.
elements	Vector of element IDs to filter the data. If TRUE, the existing definitions ID will be overwritten. If FALSE, the new definition ID will be appended. Default is FALSE.

Value

None. The function updates the metadata in the specified cloud storage bucket.

---

update_metadata_info *Update Station Metadata*

Description

This function updates the metadata for specified stations in a given country by merging new data from metadata_data with existing metadata.

Usage

```
update_metadata_info(
  country,
  metadata_data,
  station_var,
  latitude_var = NULL,
  longitude_var = NULL,
  elevation_var = NULL,
  district_var = NULL
)
```

Arguments

country	A character vector specifying the country or countries from which to update the metadata. Options are defined in get_bucket_name() (e.g., "zm", "mw").
metadata_data	A data frame containing new metadata for the stations. This data should include columns for station identifiers and other metadata fields like latitude, longitude, etc.
station_var	Optional. The name of the column in metadata_data corresponding to station IDs. Defaults to NULL.
latitude_var	Optional. The name of the column in metadata_data containing latitude values. Defaults to NULL.

longitude_var	Optional. The name of the column in metadata_data containing longitude values. Defaults to NULL.
elevation_var	Optional. The name of the column in metadata_data containing elevation values. Defaults to NULL.
district_var	Optional. The name of the column in metadata_data containing district information. Defaults to NULL.

Value

A data frame with updated metadata for the specified stations.

Examples

```
# Assuming get_bucket_name and station_metadata functions are defined:
# update_metadata_info("zm", c("BINGA", "KARIBA"), new_metadata_data,
#                station_var = "stationName", latitude_var = "lat", longitude_var = "lon")
```

---

update_summaries_data *Get the Summaries Data*

Description

This function updates the summary data for a specific station in the specified country. It retrieves the data from Google Cloud Storage using the get_data function.

Usage

```
update_summaries_data(country, station_id, summary)
```

Arguments

country	A character vector specifying the country or countries from which to get the data. Options are defined in get_bucket_name() (e.g., "zm", "mw").
station_id	A character string specifying the ID of the station for which to get the summary data.
summary	A character string specifying the summary to retrieve.

Details

The update_daily_data function is used to update the daily data for a specific station in the specified country. It internally calls the get_data function to retrieve the data from Google Cloud Storage. The country argument is a character vector that allows specifying one or more countries from which to update the data. The data will be updated for Mozambique ("mz") and Zambia ("zm"). You can modify this argument to update data for different countries. The station_id argument is a character string that specifies the ID of the station for which to update the daily data. The function will construct the filename by concatenating the "data/" directory, the station_id, and the file extension ".rds". The filename will be passed to the get_data function to retrieve the data. The function uses the invisible function to suppress the output of the get_data function, ensuring that the data retrieval process is not visible in the console.

Value

This function does not return any value explicitly. It gets the summary data for the specified station in the specified country.

Examples

```
# todo
```

Index