Package: cropgrowdays (via r-universe)

October 10, 2024

Title Crop Growing Degree Days and Agrometeorological Calculations

Version 0.2.1.9000

Description Calculate agrometeorological variables for crops including growing degree days (McMaster, GS & Wilhelm, WW (1997) <doi:10.1016/S0168-1923(97)00027-0>), cumulative rainfall, number of stress days and cumulative or mean radiation and evaporation. Convert dates to day of year and vice versa. Also, download curated and interpolated Australian weather data from the Queensland Government DES longpaddock website <https://www.longpaddock.qld.gov.au/>. This data is freely available under the Creative Commons 4.0 licence.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://gitlab.com/petebaker/cropgrowdays/

BugReports https://gitlab.com/petebaker/cropgrowdays/-/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends R (>= 4.2.0)

Imports lubridate, dplyr, tibble, purrr, purrrlyr, httr

Suggests rmarkdown, knitr, furrr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://petebaker.r-universe.dev

RemoteUrl https://gitlab.com/petebaker/cropgrowdays

RemoteRef HEAD

RemoteSha 03c1cf3946b71ca625cbafffed8de4e26b822565

Contents

---

boonah *SILO weather data extracted for Boonah SE Queensland*

Description

A dataset containing daily weather data in APSIM format for Boonah at -27.9927 S, 152.6906 E for the period 1 Jan 2019 to 31 May 2020.

Usage

```
boonah
```

Format

A data frame with 517 rows and 10 variables:

year Year

day Day

radn Radiation (MJ/m^2)

maxt Max Temperature (degrees C)

mint Min Temperature (degrees C)

rain Rainfall (mm)

evap Evaporation (mm)

vp Vapour Pressure (hPa)

code Source Code is a six digit string describing the source of the meteorological data where each digit is either 0, an actual observation; 1, an actual observation from a composite station; 2, a value interpolated from daily observations; 3, a value interpolated from daily observations using the anomaly interpolation method for CLIMARC data; 6, a synthetic pan value; or 7, an interpolated long term average

date_met The date on which daily weather was collected

Details

These data were obtained using

```
boonah <- get_silodata(latitude = "-27.9927", longitude = "152.6906", email = "MY_EMAIL_ADDRESS",
START = "20190101", FINISH = "20200531")
```

where "MY_EMAIL_ADDRESS" was replaced with an appropriate address.

Source

Meteorological SILO data obtained from the Longpaddock Qld Government web site https://www.longpaddock.qld.gov.au for Boonah -27.9927 S, 152.6906 E for the period 1 Jan 2019 to 31 May 2020.

---

crop *Crop data containing hypothetical sowing flowering and harvest dates*

Description

A dataset containing typical dates for agronomic data

Usage

```
crop
```

Format

A data frame with 10 rows and 3 variables:

sowing_date Sowing date of crop

flower_date Flowering date of crop

harvest_date Harvest date of crop

Details

Typical dates for crop data sets that would typically contain agronomic traits such as yield, dry matter yield and so on. However, here the dataset only contains dates which are used to demonstrate how to add computed weather variables either starting or ending at a specified date or between two dates.

Source

Hypothetical sowing, flowering and harvest dates based on typical horticultural crops in Queensland

`cumulative` *Sum of a weather variable between between two dates*

Description

Calculates the sum or total of daily values between two dates from a tibble or data frame of daily weather data. Alternatively, a number of days before or after a specific date may be specified. Typically this is used for solar radiation, evaporation or rainfall since the total rainfall, radiation or evaporation during a specified period may prove useful for modelling yield or plant growth.

Usage

```
cumulative(
  data,
  var = NULL,
  datevar = NULL,
  ndays = 5,
  na.rm = FALSE,
  startdate = NULL,
  enddate = NULL,
  monitor = FALSE,
  warn.consecutive = TRUE,
  ...
)
```

Arguments

`data`	Tibble or dataframe of daily weather data
`var`	Variable(s) to be extracted (Default: radn). Several columns may be specified using column names `c(variable1, variable2, ...)`
`datevar`	Date variable specifying day (Default: date_met)
`ndays`	Number of days after/before the start or end date, respectively. Ignored of both the start and end dates are specified (Default: 5)
`na.rm`	Used for calculations (Default: FALSE)
`startdate`	Start date of data to be extracted
`enddate`	Final date of data to be extracted
`monitor`	For debugging. Prints data and dates. (Default: FALSE)
`warn.consecutive`	A logical indicating whether to check that dates are consecutive, that none are missing and provide a warning if not (Default:TRUE)
`...`	options to be passed to `sum` calculation

Details

The sum is returned but if there are any missing values, then the sum is set to NA since the default na.rm is TRUE. Note that if there are any missing dates, then a warning is issued but the sum of non-missing values is returned.

If any values are missing, while the calculated sum or total may prove useful, it will not include all the data and so may lead to biased underestimates. Hence, in these cases it may be unlikely that the sum is a good estimate but the appropriateness of the estimate will depend on the exact circumstances of the missing data and so this decision is left to the user.

Value

Numeric variable containing the sum of all values of the weather variable var during the specified period.

See Also

sum, daily_mean, growing_degree_days, stress_days_over, and weather_extract

Examples

```
## Selected calculations
## library(tidyverse)   # purrr used here for crop2 example
library(dplyr)
library(purrr)
cumulative(boonah, enddate = crop$flower_date[4], ndays = 3,
                    monitor = TRUE)
cumulative(boonah, enddate = crop$harvest_date[4], ndays = 3,
                    monitor = TRUE)
cumulative(boonah, startdate = crop$flower_date[4],
                    enddate = crop$harvest_date[4], monitor = TRUE)
cumulative(boonah, startdate = crop$flower_date[4],
                    enddate = crop$harvest_date[4])
cumulative(boonah, var = rain, startdate = crop$flower_date[4],
                    enddate = crop$harvest_date[4])

## Add selected totals of weather variables in 'boonah' to 'crop'' tibble
## using 'map2_dbl' from the 'purrr' package
## Note: using equivalent 'furrr' functions can speed up calculations
crop2 <- crop |>
  dplyr::mutate(totalrain_post_sow_7d =
          purrr::map_dbl(sowing_date, function(x)
            cumulative(boonah, var = rain, startdate = x, ndays = 7)),
          totalrad_flower_harvest =
            purrr::map2_dbl(flower_date, harvest_date, function(x, y)
              cumulative(boonah, var = radn, startdate = x, enddate = y)))
crop2
```

daily_mean *Mean of daily weather variable values between two dates*

Description

Calculates the average of daily values of the variable between two dates from a tibble or data frame of daily weather data. Alternatively, a number of days before or after a specific date may be specified. Typically this would be used for temperature, rainfall or solar radiation.

Usage

```
daily_mean(
  data,
  var = NULL,
  datevar = NULL,
  ndays = 5,
  na.rm = FALSE,
  startdate = NULL,
  enddate = NULL,
  monitor = FALSE,
  warn.consecutive = TRUE,
  ...
)
```

Arguments

data	Tibble or dataframe of daily weather data
var	Variable(s) to be extracted (Default: radn). Several columns may be specified using column names c(variable1, variable2, ...)
datevar	Date variable specifying day (Default: date_met)
ndays	Number of days after/before the start or end date, respectively. Ignored of both the start and end dates are specified (Default: 5)
na.rm	Used for calculations (Default: FALSE)
startdate	Start date of data to be extracted
enddate	Final date of data to be extracted
monitor	For debugging. Prints data and dates. (Default: FALSE)
warn.consecutive	A logical indicating whether to check that dates are consecutive, that none are missing and provide a warning if not (Default:TRUE)
...	options to be passed to mean calculation

Details

The mean is returned but if there are any missing values, then the mean is set to NA since the default na.rm is TRUE. Note that if there are any missing dates, then a warning is issued but the mean of non-missing values is returned.

If any values are missing, while the calculated mean may prove useful, it will not include all the data and so may lead to biased estimates. Hence, in these cases, the appropriateness of the estimate will depend on the exact circumstances of the missing data and so this decision is left to the user.

Value

Numeric variable containing the daily mean of the weather variable var during the specified period.

See Also

mean, cumulative, growing_degree_days, stress_days_over, and weather_extract

Examples

```
##  Selected calculations
## library(tidyverse)   # only purrr used here for crop2 example
library(dplyr)
library(purrr)
daily_mean(boonah, enddate = crop$flower_date[4], ndays = 3,
                    monitor = TRUE)
daily_mean(boonah, enddate = crop$harvest_date[4], ndays = 3,
                    monitor = TRUE)
daily_mean(boonah, startdate = crop$flower_date[4],
                    enddate = crop$harvest_date[4], monitor = TRUE)

## Add selected daily means of weather variables in 'boonah' to 'crop'
## tibble using 'map2_dbl' from the 'purrr' package
## Note: using equivalent 'furrr' functions can speed up calculations
crop2 <- crop |>
  mutate(mean_maxtemp_post_sow_7d =
          purrr::map_dbl(sowing_date, function(x)
            daily_mean(boonah, var = maxt, startdate = x, ndays = 7)),
          mean_rad_flower_harvest =
            purrr::map2_dbl(flower_date, harvest_date, function(x, y)
              daily_mean(boonah, var = radn, startdate = x, enddate = y)))
crop2
```

---

date_from_day_year *Calculate a date from the day of the year and the year*

Description

date_from_day_of_year returns the date using the day of the year and a year for a calendar or financial year. The first day of the year is 1 January for the calendar year, 1 July for the Australian financial year or can be specified. Alternatively, the first day of the year can be any day of the year if desired.

Usage

```
date_from_day_year(
  day,
  year,
  type = c("calendar", "financial", "other"),
  base = NULL
)
```

Arguments

day	Day as an numeric (integer) used for calculation the date
year	Year as an integer (integer) used for calculation the date
type	A character string specifying the type of year. "calendar" is a calendar year starting on 1 January, "financial" an Australian financial year beginning on 1 July and "other" is for a year starting on another date which is specified in base. Default: "calendar"
base	A list specifying the day and month of the first day to be used in calculations. The name of the day component must be named either "day" or "d", and the month must be named either "month" or "m".

Value

A date of class Date calculated from the day of the year and the year

Examples

```
library(lubridate)
date_from_day_year(day = 366, year = 2020)
date_from_day_year(21,2021)
date_from_day_year(day = c(21, 24, 30), year = rep(2021, 3))
date_from_day_year(21,2021, type = "financial")
date_from_day_year(21,2021, type = "other", base = list(d=1, m=9))
```

---

day_of_harvest *The day of harvest in the year of sowing*

Description

day_of_harvest calculates the day of year of harvest (or another quantity such as petal fall) if harvest occurs in the next year after sowing. For instance, if the sowing date is near the end of the year, then the harvest date will fall in the next year. Hence, the harvest day (as a day of year) will be be smaller than the sowing date (as a day of the year). The day_of_harvest function rectifies this situation by calculating the harvest date as the day of year in the previous year. Hence, the resulting day of year will be greater than 366. Of course, this is not necessary if the sowing and harvest dates are in the same year, in which case day_of_year will provide exactly the same result.

Usage

```
day_of_harvest(
  x,
  sowing,
  type = c("calendar", "financial", "other"),
  base = NULL,
  verbose = FALSE
)
```

Arguments

x	A harvest Date used for calculation of the day of the year.
sowing	A sowing Date used for calculation.
type	A character string specifying the type of year. "calendar" is a calendar year starting on 1 January, "financial" an Australian financial year beginning on 1 July and "other" is for a year starting on another date which is specified in base. Default: "calendar"
base	A list specifying the day and month of the first day to be used in calculations. The name of the day component must be named either "day" or "d", and the month must be named either "month" or "m".
verbose	Logical to indicate whether to indicate data errors. (Default:FALSE)

Value

An numeric vector containing the day of harvest in the year of sowing which will differ from the day of harvest if the sowing date is in the previous year

Examples

```
library(lubridate)
day_of_harvest(x = ymd("2020-06-15"), sowing = ymd("2020-06-01")) # leap year
day_of_harvest(x = ymd("2021-06-15"), sowing = ymd("2021-06-01")) # not
day_of_harvest(x = ymd("2021-06-15"), sowing = ymd("2020-06-01")) # 366+166
day_of_harvest(x = ymd("2021-02-05"), sowing = ymd("2021-01-28"))
day_of_harvest(x = ymd("2021-02-05"), sowing = ymd("2021-01-28"),
               type = "financial")
day_of_harvest(x = ymd("2021-09-05"), sowing = ymd("2021-01-28"),
               type = "financial") # 67 + 365
```

```
## number_of_days(ymd("2021-02-05"), ymd("2020-09-01")) + 1
day_of_harvest(x = ymd("2021-02-05"), sowing = ymd("2021-01-28"),
               type = "other", base = list(m = 9, day = 1))
```

`day_of_year` *Calculate day of year from a date*

Description

`day_of_year` returns the day of the year as an integer. The first day of the year is 1 January for the calendar year, 1 July for the Australian financial year or can be specified as any day of the year if desired.

Usage

```
day_of_year(
  x,
  type = c("calendar", "financial", "other"),
  return_year = FALSE,
  base = NULL
)
```

Arguments

`x`	A `date` used for calculation the day of the year.
`type`	A character string specifying the type of year. "calendar" is a calendar year starting on 1 January, "financial" an Australian financial year beginning on 1 July and "other" is for a year starting on another date which is specified in `base`. Default: "calendar"
`return_year`	A logical indicating whether to return the year or not. Default: FALSE
`base`	A `list` specifying the day and month of the first day to be used in calculations. The name of the day component must be named either "day" or "d", and the month must be named either "month" or "m".

Value

A numeric vector containing day of the year. If `return_year` is TRUE then a `data.frame` is returned containing two columns `day` and `year`. The first column `day` is always numeric but the class of the `year` column depends on `type`. For `type` "calendar", "financial", "other" then `year` is numeric, character and the base date as a `Date`, respectively.

See Also

`date_from_day_year` for converting day of year and year to a `Date` and `day_of_harvest` for calculating the day of harvest given a sowing date where the start of the year is the first day of the year which contains the sowing date

Examples

```
library(lubridate)
## Day of Calendar Year
day_of_year(ymd("2020-01-05"))
day_of_year(ymd("2021-01-05"))
day_of_year(ymd(c("2020-06-05"), "2021-06-05")) # 29 Feb in 2020 only
day_of_year(ymd("2020-12-31"))
day_of_year(ymd("2021-12-31"))
day_of_year(ymd("2020-12-31"), return_year = TRUE)
day_of_year(ymd(c("2020-12-31", "2020-07-01", "2020-01-01")))
day_of_year(ymd(c("2020-12-31", "2020-07-01", "2020-01-01")),
            return_year = TRUE)

## Day of Financial Year
day_of_year(ymd(c("2020-12-31", "2020-07-01", "2020-01-01")),
            type = "financial")
day_of_year(ymd(c("2020-12-31", "2020-07-01", "2020-01-01")),
            type = "financial", return_year = TRUE)
day_of_year(x = ymd("2021-09-05"), type = "financial") # 67

## Specify the year starts on 1 September
day_of_year(ymd(c("2020-12-31", "2020-07-01", "2020-01-01")),
            type = "other", base = list(d = 1, m = 9))
day_of_year(ymd(c("2020-12-31", "2020-07-01", "2020-01-01")),
            type = "other", base = list(d = 1, m = 9), return_year = TRUE)
```

---

get_multi_silodata *Retrieve SILO data for multiple sites from Qld DES longpaddock website*

Description

Uses get_silodata to retrieve SILO (Scientific Information for Land Owners) data for multiple sites. SILO products are provided free of charge to the public for use under the Creative Commons Attribution 4.0 license and appear to be subject to fair use limits. Note that SILO may be unavailable between 11am and 1pm (Brisbane time) each Wednesday and Thursday to allow for essential system maintenance.

Usage

```
get_multi_silodata(
  latitude,
  longitude,
  Sitename,
  email,
  START = "20201101",
  FINISH = "20201231",
  FORMAT = "apsim",
```

```
  PASSWORD = "apitest",
  URL = "https://www.longpaddock.qld.gov.au/cgi-bin/silo/DataDrillDataset.php"
)
```

Arguments

latitude	A numerical vector containing site latitudes for data retrieval
longitude	A numerical vector containing site longitudes for data retrieval
Sitename	A vector of strings containing site names or labels which provide an extra column in the dataset named Sitename
email	A string containing your email which is required by DES in order to access the data
START	Start date as a character string "YYYYMMDD" with no spaces. Default: "20201101"
FINISH	Last date as a character string "YYYYMMDD" with no spaces. Default: "20201231"
FORMAT	of data file required. While this function was originally constructed to obtain apsim format, other formats are now available but not as tested. rainman is not included since it returns six separate files. See https://www.longpaddock.qld.gov.au/silo/about/file-formats-and-samples/ Default: "apsim"
PASSWORD	Default: "apitest"
URL	is the URL for querying the website and probably will not need to be changed. Default: "https://www.longpaddock.qld.gov.au/cgi-bin/silo/DataDrillDataset.php"

Value

A tibble (dataframe) containing specified or default climate variables. If "APSIM" format is specified then an extra column date_met, containing the date, is returned along with the usual year and day of year day. The Sitename variable contains the name of each site.

See Also

get_silodata

Examples

```
## Not run:
## Example: Replace MY_EMAIL_ADDRESS with your email address below
two_sites  <-
  get_multi_silodata(latitude = c(-27.00, -28.00),
                     longitude = c(151.00, 152.00),
                     Sitename = c("Site_1", "Site_2"),
                     START = "20201101", FINISH = "20201105",
                     FORMAT = "allmort",
                     email = "MY_EMAIL_ADDRESS")
two_sites

## End(Not run)
```

get_silodata *Retrieve weather data from Queensland Government DES longpaddock website*

Description

SILO (Scientific Information for Land Owners) is a database of Australian climate data from 1889 (current to yesterday). It provides data sets for a range of climate variables. SILO is hosted by the Science and Technology Division of the Queensland Government's Department of Environment and Science (DES). For more information please see https://www.longpaddock.qld.gov.au/silo/about. A number of data formats are available via the FORMAT option, but we have mainly used the "apsim" format. Other formats may be retrieved but these are largely untested. Please lodge an issue if there are problems. For details, please see https://www.longpaddock.qld.gov.au/silo/about/file-formats-and-samples/. SILO products are provided free of charge to the public for use under the Creative Commons Attribution 4.0 license and appear to be subject to fair use limits. Note that SILO may be unavailable between 11am and 1pm (Brisbane time) each Wednesday and Thursday to allow for essential system maintenance.

Usage

```
get_silodata(
  latitude,
  longitude,
  email,
  START = "20201101",
  FINISH = "20201231",
 FORMAT = c("apsim", "fao56", "standard", "allmort", "ascepm", "evap_span", "span",
    "all2016", "alldata", "p51", "rainonly", "monthly", "cenw"),
  PASSWORD = "apitest",
  extras = NULL,
  URL = "https://www.longpaddock.qld.gov.au/cgi-bin/silo/DataDrillDataset.php"
)
```

Arguments

latitude	A number or character string of the site latitude for data retrieval
longitude	A number or character string of the site longitude for data retrieval
email	A string containing your email which is required by DES in order to access the data
START	Start date as a character string "YYYYMMDD" with no spaces. Default: "20201101"
FINISH	Last date as a character string "YYYYMMDD" with no spaces. Default: "20201231"
FORMAT	of data file required. While this function was originally constructed to obtain apsim format, other formats are now available but not as tested. rainman is not included since it returns six separate files. See https://www.longpaddock.qld.gov.au/silo/about/file-formats-and-samples/ Default: "apsim"

PASSWORD	Default: "apitest"
extras	A list containing variable names and values for extra columns. Note that the new variable(s) will have the same value. Default: NULL
URL	is the URL for querying the website and probably will not need to be changed. Default: "https://www.longpaddock.qld.gov.au/cgi-bin/silo/DataDrillDataset.php"

Details

When extrais specified, then extra columns which could for instance include the site name are added to each row for later calculations. This can also be employed in a loop using `for()` loops or a tidyverse approach, such as `purrrlyr::by_row()` in order to manipulate data by site.

Value

Data frame/tibble containing specified or default climate variables. If "APSIM" format is specified then an extra column date_met, containing the date, is returned along with the usual year and day of year day.

See Also

`get_multi_silodata`

Examples

```
## Not run:
## Example 1: Replace MY_EMAIL_ADDRESS with your email address below
##            Latitude and Longitude character strings
boonah_data <-
  get_silodata(latitude = "-27.9927", longitude = "152.6906",
               email = "MY_EMAIL_ADDRESS",
               START = "20190101", FINISH = "20200531")
## Example 2: Replace MY_EMAIL_ADDRESS below with yours - adds extra column
##            Latitude and Longitude are numeric
boonah_data2 <-
  get_silodata(latitude = -27.9927, longitude = 152.6906,
               email = "MY_EMAIL_ADDRESS",
               START = "20190101", FINISH = "20200531",
               extras = list(Sitename = "Boonah"))
## Example 3: Replace MY_EMAIL_ADDRESS below with yours - adds extra column
##            Latitude and Longitude are numeric.
##            Retreives all Morton hydrological evapotranspirations
boonah_data3 <-
  get_silodata(latitude = -27.9927, longitude = 152.6906,
               email = "MY_EMAIL_ADDRESS", FORMAT  =  "allmort",
               START = "20190101", FINISH = "20190115",
               extras = list(Sitename = "Boonah"))
## Example 4: Replace MY_EMAIL_ADDRESS below with yours - adds extra column
##            Latitude and Longitude are numeric. Retreives two months of
##            total rainfall and evaporation and means for max/min
##            temperatures, solar radiation and vapour pressure
boonah_data4 <-
```

```
  get_silodata(latitude = -27.9927, longitude = 152.6906,
               email = "drpetebaker@gmail.com", FORMAT = "monthly",
               START = "20190101", FINISH = "20190228",
               extras = list(Sitename = "Boonah"))

## End(Not run)
```

`growing_degree_days` *Degree days as the sum of capped average daily temperature above a baseline value*

Description

Calculate the sum of of degree days (average temperature - base temperature `base_temp` for each day) during specified dates for a tibble or data frame of daily weather data. Alternatively, a number of days before or after a specific date may be specified. Note that the maximum temperature is capped at `maxt_cap` when calculating the average temperature.

Usage

```
growing_degree_days(
  data,
  varmax = NULL,
  varmin = NULL,
  datevar = NULL,
  maxt_cap = 30,
  base_temp = 5,
  na.rm = FALSE,
  ndays = 5,
  startdate = NULL,
  enddate = NULL,
  monitor = FALSE,
  warn.consecutive = TRUE
)
```

Arguments

`data`	Tibble or dataframe of daily weather data
`varmax`	Name of variable containing max temp (default 'maxt')
`varmin`	Name of variable containing min temp (default 'mint')
`datevar`	Date variable specifying day (Default: date_met)
`maxt_cap`	A numeric value set to the temperature considered to be the maximum necessary for plant growth. Maximum temperature is capped at this value for calculating average daily temp (Default: 30)
`base_temp`	A numeric value set to the temperature considered to be the minimum necessary for plant growth (Default: 5)

na.rm	Used for calculations (Default: FALSE)
ndays	Number of days after/before the start or end date, respectively. Ignored of both the start and end dates are specified (Default: 5)
startdate	Start date of data to be extracted
enddate	Final date of data to be extracted
monitor	For debugging. Prints data and dates. (Default: FALSE)
warn.consecutive	A logical indicating whether to check that dates are consecutive, that none are missing and provide a warning if not (Default:TRUE)

Details

The value returned is the sum of of degree days $GDD = \sum_i (Tmax_i + Tmin_i)/2 - Tbase$ during specified dates for a tibble or data frame of daily weather data. The maximum temperature Tmax is capped at maxt_cap degrees when calculating average temp (see https://farmwest.com/climate/calculator-information/gdd/ or McMaster, GS, & Wilhelm, WW (1997)). Baskerville, G & Emin, P (1969) provide variations on this method.

The sum of degree days is returned but if there are any missing values, then the value returned will be NA since the default na.rm is TRUE. Note that if there are any missing dates, then a warning is issued but the sum of non-missing values is returned.

If any values are missing, while the sum of degree days may prove useful, it will not include all the data and so will lead to biased underestimates. Hence, in these cases it is unlikely that the value returned is a good estimate but the appropriateness of the estimate will depend on the exact circumstances of the missing data and so this decision is left to the user.

Value

Numeric variable containing the sum of degree days during the period

References

- Baskerville, G., & Emin, P. (1969). Rapid Estimation of Heat Accumulation from Maximum and Minimum Temperatures. Ecology, 50(3), 514-517. doi:10.2307/1933912
- McMaster, G. S., & Wilhelm, W. W. (1997). Growing degree-days: One equation, two interpretations. Agricultural and Forest Meteorology, 87(4), 291–300. doi:10.1016/S0168-1923(97)000270
- Anon. (2021). GDD. Farmwest. Retrieved June 15, 2021, from https://farmwest.com/climate/calculator-information/gdd/

See Also

cumulative, daily_mean, stress_days_over, and weather_extract

Examples

```
## Selected calculations
## library(tidyverse)  # only purrr used here for crop2 example
library(dplyr)
library(purrr)
growing_degree_days(boonah, enddate = crop$flower_date[4], ndays = 3,
                    varmax = maxt, varmin = mint,
                    monitor = TRUE)
growing_degree_days(boonah, enddate = crop$harvest_date[4], ndays = 3,
                    varmax = maxt, varmin = mint,
                    monitor = TRUE)
growing_degree_days(boonah, startdate = crop$flower_date[4],
                    varmax = maxt, varmin = mint,
                    enddate = crop$harvest_date[4], monitor = TRUE)

## Add selected growing degree days at 'boonah' to 'crop' tibble
## using 'map2_dbl' from the 'purrr' package
## Note: using equivalent 'furrr' functions can speed up calculations
crop2 <- crop |>
  mutate(gddays8_post_sow_7d =
          purrr::map_dbl(sowing_date, function(x)
            growing_degree_days(boonah, startdate = x, ndays = 7,
                                base_temp = 8)),
          gddays_flower_harvest =
            purrr::map2_dbl(flower_date, harvest_date, function(x, y)
              growing_degree_days(boonah, startdate = x, enddate = y)))
crop2
```

`number_of_days` *The number of days between two dates*

Description

A convenience function which is simply `as.numeric(end_date -start_date)`

Usage

```
number_of_days(x, start)
```

Arguments

`x`	The end date with class `Date`
`start`	The start date with class `Date`

Value

A `numeric` variable containing the number of days between the two dates

Examples

```
library(lubridate)
number_of_days(x = ymd("2021-01-05"), start = ymd("2020-12-28"))
```

`stress_days_over` *The number of days that maximum temp is over a baseline value*

Description

Calculate the number of days when the maximum temperature exceeds a base line `stress_temp` during specified dates for a tibble or data frame of daily weather data. Alternatively, a number of days before or after a specific date may be specified. The default value of `stress_temp` is 30 degrees C.

Usage

```
stress_days_over(
  data,
  var = NULL,
  datevar = NULL,
  ndays = 5,
  na.rm = FALSE,
  stress_temp = 30,
  startdate = NULL,
  enddate = NULL,
  monitor = FALSE,
  warn.consecutive = TRUE,
  ...
)
```

Arguments

`data`	Tibble or dataframe of daily weather data
`var`	Variable to be extracted (Default: maxt)
`datevar`	Date variable specifying day (Default: date_met)
`ndays`	Number of days after/before the start or end date, respectively. Ignored of both the start and end dates are specified (Default: 5)
`na.rm`	Used for calculations (Default: FALSE)
`stress_temp`	A numeric value set to the temperature considered to be stressful if the maximum temperature exceeds (Default: 30)
`startdate`	Start date of data to be extracted
`enddate`	Final date of data to be extracted
`monitor`	For debugging. Prints data and dates. (Default: FALSE)
`warn.consecutive`	A logical indicating whether to check that dates are consecutive, that none are missing and provide a warning if not (Default:TRUE)
`...`	options to be passed to `sum` calculation

Details

The number of days is returned but if there are any missing values, then the sum is set to NA since the default na.rm is TRUE. Note that if there are any missing dates, then a warning is issued but the sum of non-missing values is returned.

If any values are missing, while the number of days over stress_temp total may prove useful, it will not include all the data and so may lead to biased underestimates. Hence, depending on the time of year, it may be unlikely that this is a good estimate but the appropriateness of the estimate will depend on the exact circumstances of the missing data and so this decision is left to the user.

Value

Numeric variable containing the number of days where the maximum temperature was above the specified stress temperature cutoff

See Also

cumulative, daily_mean, growing_degree_days, and weather_extract

Examples

```
##  Selected calculations
## library(tidyverse) # only purrr used here for crop2 example
library(dplyr)
library(purrr)
stress_days_over(boonah, enddate = crop$flower_date[4], ndays = 3,
                 monitor = TRUE)
stress_days_over(boonah, enddate = crop$harvest_date[4], ndays = 3,
                 monitor = TRUE)
stress_days_over(boonah, startdate = crop$flower_date[4],
                 enddate = crop$harvest_date[4], monitor = TRUE)

## Add selected stress days at 'boonah' to 'crop' tibble
## using 'map2_dbl' from the 'purrr' package
## Note: using equivalent 'furrr' functions can speed up calculations
crop2 <- crop |>
  mutate(stressdays25_post_sow_7d =
           purrr::map_dbl(sowing_date, function(x)
             stress_days_over(boonah, startdate = x, ndays = 7,
                              stress_temp = 25)),
           stressdays_flower_harvest =
             purrr::map2_dbl(flower_date, harvest_date, function(x, y)
               stress_days_over(boonah, startdate = x, enddate = y)))
crop2
```

`two_sites` *SILO weather data extracted for two arbitrary sites*

Description

A dataset containing daily weather data in APSIM format for Site_1: at -27 S, 151 E and Site_2: -28 S, 152 E for the period 1 to 5 Nov 2020

Usage

```
two_sites
```

Format

A data frame with 10 rows and 11 variables:

year Year

day Day

radn Radiation (MJ/m^2)

maxt Max Temperature (degrees C)

mint Min Temperature (degrees C)

rain Rainfall (mm)

evap Evaporation (mm)

vp Vapour Pressure (hPa)

code Source Code is a six digit string describing the source of the meteorological data where each digit is either 0, an actual observation; 1, an actual observation from a composite station; 2, a value interpolated from daily observations; 3, a value interpolated from daily observations using the anomaly interpolation method for CLIMARC data; 6, a synthetic pan value; or 7, an interpolated long term average

date_met The date on which daily weather was collected

Sitename Sites labelled "Site_1" and "Site_2"

Details

The data were obtained with

```
two_sites <- get_multi_silodata(latitude = c(-27.00, -28.00), longitude = c(151.00, 152.00),
Sitename = c("Site_1", "Site_2"), START = "20201101", FINISH = "20201105", email = "MY_EMAIL_ADDRESS")
```

where "MY_EMAIL_ADDRESS" was relaced with an appropriate address.

Source

Meteorological SILO data obtained from the Longpaddock Qld Government web site https://www.longpaddock.qld.gov.au for two sites for the period 1 to 5 Nov 2020

`weather_extract` *Extract one or more columns of weather data between two dates*

Description

Extract column(s) from a tibble/data frame of daily weather data between two specified dates. Either specify the start and end dates or specify one of these dates and also the number of days after or before, respectively.

Usage

```
weather_extract(
  data,
  var,
  datevar = NULL,
  ndays = 5,
  startdate = NULL,
  enddate = NULL,
  monitor = FALSE,
  return.dates = TRUE,
  warn.consecutive = TRUE
)
```

Arguments

`data`	Tibble or dataframe of daily weather data
`var`	Variable(s) to be extracted (Default: radn). Several columns may be specified using column names `c(variable1, variable2, ...)`
`datevar`	Date variable specifying day (Default: date_met)
`ndays`	Number of days after/before the start or end date, respectively. Ignored of both the start and end dates are specified (Default: 5)
`startdate`	Start date of data to be extracted
`enddate`	Final date of data to be extracted
`monitor`	For debugging. Prints data and dates. (Default: FALSE)
`return.dates`	A logical indicating whether to return the date column (Default: TRUE)
`warn.consecutive`	A logical indicating whether to check that dates are consecutive, that none are missing and provide a warning if not (Default:TRUE)

Value

A tibble (data frame) of extracted weather data

See Also

`between` `filter`, `cumulative`, `daily_mean`, `growing_degree_days`, and `stress_days_over`

Examples

```
library(lubridate)
boonah |>
weather_extract(rain, date = date_met, startdate = ymd("2019-08-16"),
                  enddate = ymd("2019-08-21"))
boonah |>
  weather_extract(rain, startdate = ymd("2019-08-16"),
                  enddate = ymd("2019-08-21"))
boonah |>
  weather_extract(maxt, date = date_met, startdate = ymd("2019-08-16"),
                  ndays = 3, return.dates = FALSE)
boonah |>
  weather_extract(mint, enddate = ymd("2019-08-16"), ndays = 1)
## extract multiple columns
boonah |>
  weather_extract(c(year, day, mint, maxt), enddate = ymd("2019-08-16"),
                  ndays = 3)
```

Index