Package: misc.debris (via r-universe)

May 12, 2026

Type Package

Title Various Utility Functions That Don't Really Belong Anywhere Else

Version 0.1.3

Description A ``catch-all'' package that holds utility functions commonly used in R programming projects that I work with.

License MIT + file LICENSE

URL https://codeberg.org/adamhsparks/misc.debris, https://adamhsparks.codeberg.page/misc.debris/

BugReports https://codeberg.org/adamhsparks/misc.debris/issues

Imports cli, data.table, ggplot2, lubridate, purrr, weatherOz

Suggests roxyglobals, scales, testthat (>= 3.0.0), vcr (>= 0.6.0), vdiffr

Remotes https://github.com/DPIRD-FSI/weatherOz

Config/roxyglobals/filename globals.R

Config/roxyglobals/unique FALSE

Config/testthat/edition 3

Encoding UTF-8

Language en-AU

LazyData true

Roxygen list(markdown = TRUE, roclets = c(``collate'', ``namespace'', ``rd'', ``roxyglobals::global_roclet''))

RoxygenNote 7.3.2.9000

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libmagick++-dev gsfonts libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2025-07-16 06:39:09 UTC

RemoteUrl https://codeberg.org/adamhsparks/misc.debris

RemoteRef HEAD

RemoteSha 917c326903fbc036cc671f4fe1a30d747d94b346

Contents

`check_date_seq` *Check Dates for Sequentialness and Completeness*

Description

Used to check vectors of dates mainly to ensure that weather data are complete. Invisibly returns a `NULL` value if no errors are found, else issues a `warning()` with an informative message providing the user with missing date values in the sequence and the optional location name if it's provided.

Usage

```
check_date_seq(x, y = NULL)
```

Arguments

`x`	A `vector` of dates.
`y`	A `vector` of values with location name, used for warning message.

Value

No return value, called for its side-effects, validates that dates are sequential and errors if they are not in order or some are missing.

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```
# passes
x <- seq(as.Date("2021-01-01"), as.Date("2021-01-31"), by = "1 day")
y <- "BA"
check_date_seq(x = x, y = y)

# fails
x <- c(
  seq(as.Date("2021-01-01"), as.Date("2021-01-31"), by = "1 day"),
  seq(as.Date("2021-02-02"), as.Date("2021-02-28"), by = "1 day")
```

```
)
y <- "BA"
check_date_seq(x = x, y = y)
```

---

check_oz_lonlat *Check User Provided Longitude and Latitude Values for Validity*

Description

Checks user-provided longitude and latitude values that they fall in Australia. Errors if any values do not.

Usage

```
check_oz_lonlat(longitude, latitude)
```

Arguments

longitude	user provided numeric value as decimal degrees
latitude	user provided numeric value as decimal degrees

Value

An invisible NULL, called for side-effects. Errors if the latitude or longitude values are outside of Australia.

Examples

```
# No error, Perth, WA
check_oz_lonlat(longitude = 115.86, latitude = -31.95)

## Not run:
# Error, in northern hemisphere
check_oz_lonlat(longitude = 115.86, latitude = 31.95)

## End(Not run)
```

`create_climograph` *Creates a Climograph of Weather data From Data Available Via weatherOz*

Description

Given weather data from **weatherOz**, create a climograph with maximum and minimum temperature on the left-hand y-axis and precipitation on the right- hand y-axis. The colours used are DPIRD's dark red (tmax), light blue (tmin) and dark blue (rain).

Usage

```
create_climograph(weather)
```

Arguments

`weather`	A `data.frame` containing weather station data from the SILO or DPIRD "Weather-2.0" APIs as provided by **weatherOz**.

Value

A **ggplot2** object.

Author(s)

Adam H. Sparks, <adamhsparks@gmail.com>

Examples

```
library(weatherOz)
wd <- get_dpird_summaries(
  station_code = "BI",
  start_date = "20220101",
  end_date = "20221231",
  api_key = Sys.getenv("DPIRD_API_KEY"),
  interval = "daily",
  values = c(
    "airTemperatureMin",
    "airTemperatureMax",
    "rainfall"
    )
)

# create the {ggplot2} object
c_graph <- create_climograph(wd)

# add a title, subtitle and caption
library(ggplot2)
```

```
main <- "Binnu Weather for 2022"
sub <-
 "Precipitation (left y-axis) and Min and Max Temperature (right y-axis)"
cap <-
  "Data from DPIRD Weather 2.0 API"
c_graph +
  labs(
    title = main,
    subtitle = sub,
    caption = cap
  )
```

`get_weather_list` *Get a List of Data Frame Objects of Weather Data From DPIRD and SILO APIs*

Description

A helper function that wraps two functions from **weatherOz** to fetch daily weather data from both the DPIRD Weather 2.0 and SILO Patched Point APIs and return a list of these objects. Values are predetermined and only those that are shared between the two APIS are returned.

Usage

```
get_weather_list(stations, first, last = Sys.Date(), api_keys)
```

Arguments

`stations`	A `vector` of values that provides station codes, `station_code` from **weatherOz**.
`first`	The date on which the weather data should start, provided in ISO8601 format, *e.g.* "2021-01-01".
`last`	The last date on which the weather data should end, provided in ISO8601 format, *e.g.* "2021-01-01". Defaults to the current system date.
`api_keys`	A vector of API keys for the DPIRD Weather 2.0 API and the SILO API in that order (it's alphabetical, DPIRD, SILO).

Value

A `list` of `data.frame` objects containing weather data from DPIRD and SILO APIs as requested.

Examples

```
# Note that you must provide your own API keys to use this function

get_weather_list(
  stations = c("4041", "BA"),
  first = "2021-01-01",
  last = "2021-01-02",
  api_keys = c(Sys.getenv("DPIRD_API_KEY"),
               Sys.getenv("SILO_API_KEY"))
)
```

---

mode *Arithmetic Mode*

Description

Generic function for finding the mode, supports multiple modes.

Usage

```
mode(x, max = FALSE, min = FALSE)
```

Arguments

x	An R numeric vector object.
max	Boolean Return only the maximum value for mode if there are multiple observed modes. Defaults to FALSE.
min	Boolean Return only the minimum value for mode if there are multiple observed modes. Defaults to FALSE.

Value

The mode, most commonly occurring value in the vector, as a numeric value. In cases where there are more than one mode, all values will be returned by default.

Examples

```
x <- c(1, 2, 3, 3, 3, 4, 4, 5, 5, 5)
mode(x)
mode(x, max = TRUE)
mode(x, min = TRUE)
```

rle2 *Count Consecutive Sequences of Events*

Description

Run length encoding to detect consecutive sequences of events.

Usage

```
rle2(x, index, l_run, value)
```

Arguments

x	A vector of values to check for consecutive sequences of events.
index	An index of event values, *e.g.* a vector of date values.
l_run	An integer value indicating the length of the run for the period of interest.
value	The base value to check against, for which values greater than will be recorded as TRUE.

Value

A data.table with three columns, the 'index', 'x', the original value that was tested and 'test' a column of Boolean values indicating whether the event was TRUE (greater than value) or FALSE (less than value), "test".

Note

R already has an base::rle(), so this function is suffixed with 2 to avoid NAMESPACE clashes.

Author(s)

Adam Sparks, <adamhsparks@gmail.com>

Source

Raphael Saldanha, https://rfsaldanha.github.io/posts/run_length_encoding.html with modifications by Adam Sparks (DPIRD).

Examples

```
# Get rainfall data since 2017 for Northam
library(weatherOz)
library(scales)
library(ggplot2)

wd <- get_dpird_summaries(
  station_code = "NO",
```

```
  start_date = "20170101",
  end_date = "20171231",
  api_key = Sys.getenv("DPIRD_API_KEY"),
  interval = "daily",
)

# Determine where runs of 2 or more days with rain over 0.5mm occur
rain <- rle2(
  x = wd$rainfall,
  index = wd$date,
  l_run = 2,
  value = 0.5
)

# Plot rainfall values
ggplot(data = rain, aes(x = index, y = value)) +
  geom_line(colour = "light blue", alpha = 0.7) +
  geom_point(aes(color = test), alpha = 0.85) +
  scale_color_manual(values = c("blue", "darkred")) +
  scale_x_date(labels = date_format("%m-%Y")) +
  xlab("Date") +
  ylab("Rainfall (mm)") +
  guides(colour = guide_legend("Two or more consecutive days of rain")) +
  theme(legend.position = "bottom", legend.direction = "horizontal")

# Determine where runs of 2 or more days avg temp above 28 C occur
tavg <- rle2(
  x = wd$air_tavg,
  index = wd$date,
  l_run = 2,
  value = 28
)

# Plot rainfall values
ggplot(data = tavg, aes(x = index, y = value)) +
  geom_line(colour = "lightblue", alpha = 0.7) +
  geom_point(aes(color = test), alpha = 0.85) +
  scale_color_manual(values = c("blue", "darkred")) +
  geom_hline(yintercept = 28, colour = "darkred") +
  scale_x_date(labels = date_format("%m-%Y")) +
  xlab("Date") +
  ylab("Temperature (C)") +
  guides(colour = guide_legend("Four or more consecutive days above 28 C")) +
  theme(legend.position = "bottom", legend.direction = "horizontal")
```

Index