Package 'ltertools'

March 26, 2025

Type Package

Title Tools Developed by the Long Term Ecological Research Community

Version 2.0.0

Date 2025-03-26

Maintainer Nicholas Lyon <lyon@nceas.ucsb.edu>

Description Set of the data science tools created by various members of the Long Term Ecological Research (LTER) community. These functions were initially written largely as standalone operations and have later been aggregated into this package.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

URL https://lter.github.io/ltertools/

BugReports https://github.com/lter/ltertools/issues

RoxygenNote 7.3.1

Depends R (>= 3.5)

Imports dplyr, generics, ggplot2, magrittr, purrr, readxl, stats, stringr, supportR, tidyr, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Nicholas Lyon [aut, cre] (https://njlyon0.github.io/),
Angel Chen [aut] (https://angelchen7.github.io),
Miguel C. Leon [ctb] (https://luquillo.lter.network/),
National Science Foundation [fnd] (NSF 1929393, 09/01/2019 - 08/31/2024),
University of California, Santa Barbara [cph]

Repository CRAN

Date/Publication 2025-03-26 18:30:02 UTC

Contents

begin_key . . . 2
check_key . . . 3
convert_temp . . . 4
cv . . . 5
expand_key . . . 5
harmonize . . . 7
lter_sites . . . 8
read . . . 9
site_subset . . . 10
site_timeline . . . 10
solar_day_info . . . 11
standardize . . . 12

Index **14**

`begin_key` *Generate the Skeleton of a Column Key*

Description

Creates the start of a 'column key' for harmonizing data. A column key includes a column for the file names to be harmonized into a single data object as well as a column for the column names in those files. Finally, it includes a column indicating the tidied name that corresponds with each raw column name. Harmonization can accept this key object and use it to rename all raw column names–in a reproducible way–to standardize across datasets. Currently supports raw files of the following formats: CSV, TXT, XLS, and XLSX

Usage

```
begin_key(
  raw_folder = NULL,
  data_format = c("csv", "txt", "xls", "xlsx"),
  guess_tidy = FALSE
)
```

Arguments

`raw_folder`	(character) folder / folder path containing data files to include in key
`data_format`	(character) file extensions to identify within the `raw_folder`. Default behavior is to search for all supported file types.
`guess_tidy`	(logical) whether to attempt to "guess" what the tidy name equivalent should be for each raw column name. This is accomplished via coercion to lowercase and removal of special character/repeated characters. If `FALSE` (the default) the "tidy_name" column is returned empty

Value

(dataframe) skeleton of column key

Examples

```
# Generate two simple tables
## Dataframe 1
df1 <- data.frame("xx" = c(1:3),
                  "unwanted" = c("not", "needed", "column"),
                  "yy" = letters[1:3])
## Dataframe 2
df2 <- data.frame("LETTERS" = letters[4:7],
                  "NUMBERS" = c(4:7),
                  "BONUS" = c("plantae", "animalia", "fungi", "protista"))

# Generate a local folder for exporting
temp_folder <- tempdir()

# Export both files to that folder
utils::write.csv(x = df1, file = file.path(temp_folder, "df1.csv"), row.names = FALSE)
utils::write.csv(x = df2, file = file.path(temp_folder, "df2.csv"), row.names = FALSE)

# Generate a column key with "guesses" at tidy column names
ltertools::begin_key(raw_folder = temp_folder, data_format = "csv", guess_tidy = TRUE)
```

---

check_key *Check and Prepare a Column Key Object*

Description

Accepts a column key dataframe and checks to make sure it has the needed structure for `ltertools::harmonize`. Also removes unnecessary columns and rows that lack a "tidy_name". Function invoked 'under the hood' by `ltertools::harmonize`.

Usage

```
check_key(key = NULL)
```

Arguments

`key`	(dataframe) key object including a "source", "raw_name" and "tidy_name" column. Additional columns are allowed but ignored

Value

(dataframe) key object with only "source", "raw_name" and "tidy_name" columns and only retains rows where a "tidy_name" is specified.

Examples

```
# Generate a column key object manually
key_obj <- data.frame("source" = c(rep("df1.csv", 3),
                                   rep("df2.csv", 3)),
                      "raw_name" = c("xx", "unwanted", "yy",
                                     "LETTERS", "NUMBERS", "BONUS"),
                     "tidy_name" = c("numbers", NA, "letters",
                                     "letters", "numbers", "kingdom"))

# Check it
ltertools::check_key(key = key_obj)
```

`convert_temp` *Convert Temperature Values*

Description

Converts a given set of temperature values from one unit to another

Usage

```
convert_temp(value = NULL, from = NULL, to = NULL)
```

Arguments

`value`	(numeric) temperature values to convert
`from`	(character) starting units of the value, not case sensitive.
`to`	(character) units to which to convert, not case sensitive.

Value

(numeric) converted temperature values

Examples

```
# Convert from Fahrenheit to Celsius
convert_temp(value = 32, from = "Fahrenheit", to = "c")
```

cv *Calculate Coefficient of Variation*

Description

Computes the coefficient of variation (CV), by dividing the standard deviation (SD) by the arithmetic mean of a set of numbers. If `na_rm` is `TRUE` then missing values are removed before calculation is completed

Usage

```
cv(x, na_rm = TRUE)
```

Arguments

`x`	(numeric) vector of numbers for which to calculate CV
`na_rm`	(logical) whether to remove missing values from both average and SD calculation

Value

(numeric) coefficient of variation

Examples

```
# Convert from Fahrenheit to Celsius
cv(x = c(4, 5, 6, 4, 5, 5), na_rm = TRUE)
```

expand_key *Generate the Skeleton of a Column Key for Only New Data Files*

Description

Data discovery–and harmonization–is an iterative process. For those already depending upon a column key and the `harmonize` function, it can be cumbersome to add rows to an existing column key. This function formats rows for an existing column key for only datasets that are not already (A) in the column key or (B) in the harmonized data table.

Usage

```
expand_key(
  key = NULL,
  raw_folder = NULL,
  harmonized_df = NULL,
  data_format = c("csv", "txt", "xls", "xlsx"),
  guess_tidy = FALSE
)
```

Arguments

key	(dataframe) key object including a "source", "raw_name" and "tidy_name" column. Additional columns are allowed but ignored
raw_folder	(character) folder / folder path containing data files to include in key
harmonized_df	(dataframe) harmonized data table produced with the current version of the column key. Must include a "source" column but other columns are ignored.
data_format	(character) file extensions to identify within the raw_folder. Default behavior is to search for all supported file types.
guess_tidy	(logical) whether to attempt to "guess" what the tidy name equivalent should be for each raw column name. This is accomplished via coercion to lowercase and removal of special character/repeated characters. If FALSE (the default) the "tidy_name" column is returned empty

Value

(dataframe) skeleton of rows to add to column key for data sources not already in harmonized data table

Examples

```
# Generate two simple tables
## Dataframe 1
df1 <- data.frame("xx" = c(1:3),
                  "unwanted" = c("not", "needed", "column"),
                  "yy" = letters[1:3])
## Dataframe 2
df2 <- data.frame("LETTERS" = letters[4:7],
                  "NUMBERS" = c(4:7),
                  "BONUS" = c("plantae", "animalia", "fungi", "protista"))

# Generate a local folder for exporting
temp_folder <- tempdir()

# Export both files to that folder
utils::write.csv(x = df1, file = file.path(temp_folder, "df1.csv"), row.names = FALSE)
utils::write.csv(x = df2, file = file.path(temp_folder, "df2.csv"), row.names = FALSE)

# Generate a column key with "guesses" at tidy column names
key1 <- ltertools::begin_key(raw_folder = temp_folder, data_format = "csv", guess_tidy = TRUE)

# Harmonize the data
harmony <- ltertools::harmonize(key = key1, raw_folder = temp_folder)

# Make a new data file
df3 <- data.frame("xx" = c(10:15),
                  "letters" = letters[10:15])

# Export this locally to the temp folder too
utils::write.csv(x = df3, file = file.path(temp_folder, "df3.csv"), row.names = FALSE)
```

```
# Identify what needs to be added to the existing column key
ltertools::expand_key(key = key1, raw_folder = temp_folder, harmonized_df = harmony,
                      data_format = "csv", guess_tidy = TRUE)
```

`harmonize` *Harmonize Data via a Column Key*

Description

A "column key" is meant to streamline harmonization of disparate datasets. This key must include three columns containing: (1) the name of each raw data file to be harmonized, (2) the name of all of the columns in each of those files, and (3) the "tidy name" that corresponds to each raw column name. This function accepts that key and the path to a folder containing all raw data files included in the key. Each dataset is then read in and the original column names are replaced with their respective "tidy_name" indicated in the key. Once this has been done to all files, a single dataframe is returned with only columns indicated in the column name. Currently the following file formats are supported for the raw data: CSV, TXT, XLS, and XLSX

Note that raw column names without an associated tidy name in the key are removed. We recommend using the `begin_key` function in this package to generate the skeleton of the key to make achieving the required structure simpler.

Usage

```
harmonize(
  key = NULL,
  raw_folder = NULL,
  data_format = c("csv", "txt", "xls", "xlsx"),
  quiet = TRUE
)
```

Arguments

`key`	(dataframe) key object including a "source", "raw_name" and "tidy_name" column. Additional columns are allowed but ignored
`raw_folder`	(character) folder / folder path containing data files to include in key
`data_format`	(character) file extensions to identify within the `raw_folder`. Default behavior is to search for all supported file types.
`quiet`	(logical) whether to suppress certain non-warning messages. Defaults to `TRUE`

Value

(dataframe) harmonized dataframe including all columns defined in the "tidy_name" column of the key object

Examples

```
# Generate two simple tables
## Dataframe 1
df1 <- data.frame("xx" = c(1:3),
                  "unwanted" = c("not", "needed", "column"),
                  "yy" = letters[1:3])
## Dataframe 2
df2 <- data.frame("LETTERS" = letters[4:7],
                  "NUMBERS" = c(4:7),
                  "BONUS" = c("plantae", "animalia", "fungi", "protista"))

# Generate a local folder for exporting
temp_folder <- tempdir()

# Export both files to that folder
utils::write.csv(x = df1, file = file.path(temp_folder, "df1.csv"), row.names = FALSE)
utils::write.csv(x = df2, file = file.path(temp_folder, "df2.csv"), row.names = FALSE)

# Generate a column key object manually
key_obj <- data.frame("source" = c(rep("df1.csv", 3),
                                   rep("df2.csv", 3)),
                      "raw_name" = c("xx", "unwanted", "yy",
                                     "LETTERS", "NUMBERS", "BONUS"),
                     "tidy_name" = c("numbers", NA, "letters",
                                     "letters", "numbers", "kingdom"))

# Use that to harmonize the 'raw' files we just created
ltertools::harmonize(key = key_obj, raw_folder = temp_folder, data_format = "csv")
```

---

lter_sites *Long Term Ecological Research Site Information*

Description

There are currently 28 field sites involved with the Long Term Ecological Research (LTER) network. These sites occupy a range of habitats and were started / are renewed on site-specific timelines. To make this information more readily available to interested parties, this data object summarizes the key components of each site in an easy-to-use data format.

Usage

```
lter_sites
```

Format

Dataframe with 8 columns and 32 rows

name Full name of the LTER site

code Abbreviation (typically three letters) of the site name

habitat Simplified habitat designation of the site (or "mixed" for more complex habitat contexts)

start_year Year of initial funding by NSF as an official LTER site

end_year End of current funding cycle grant

latitude Degrees latitude of site

longitude Degrees longitude of site

site_url Website URL for the site

Source

Long Term Ecological Research Network Office. https://lternet.edu/site/

`read` *Read Data from Folder*

Description

Reads in all data files of specified types found in the designated folder. Returns a list with one element for each data file. Currently supports CSV, TXT, XLS, and XLSX

Usage

```
read(raw_folder = NULL, data_format = c("csv", "txt", "xls", "xlsx"))
```

Arguments

`raw_folder`	(character) folder / folder path containing data files to read
`data_format`	(character) file extensions to identify within the `raw_folder`. Default behavior is to search for all supported file types.

Value

(list) data found in specified folder of specified file format(s)

Examples

```
# Generate two simple tables
## Dataframe 1
df1 <- data.frame("xx" = c(1:3),
                  "unwanted" = c("not", "needed", "column"),
                  "yy" = letters[1:3])
## Dataframe 2
df2 <- data.frame("LETTERS" = letters[4:7],
                  "NUMBERS" = c(4:7),
                  "BONUS" = c("plantae", "animalia", "fungi", "protista"))

# Generate a local folder for exporting
```

```
temp_folder <- tempdir()

# Export both files to that folder
utils::write.csv(x = df1, file = file.path(temp_folder, "df1.csv"), row.names = FALSE)
utils::write.csv(x = df2, file = file.path(temp_folder, "df2.csv"), row.names = FALSE)

# Read in all CSV files in that folder
read(raw_folder = temp_folder, data_format = "csv")
```

`site_subset` *Subsets the LTER Site Information Table by Site Codes and Habitats*

Description

Subsets the information on long term ecological research (LTER) sites based on user-specified site codes (i.e., three letter abbreviations), and/or desired habitats. See `lter_sites` for the full set of site information

Usage

```
site_subset(sites = NULL, habitats = NULL)
```

Arguments

`sites`	(character) three letter site code(s) identifying site(s) of interest
`habitats`	(character) habitat(s) of interest. See `unique(lter_sites$habitat)`

Value

(dataframe) complete site information (8 columns) for all sites that meet the provided site code and/or habitat criteria

---

`site_timeline` *Create a Timeline of Site(s) that Meet Criteria*

Description

Creates a ggplot2 plot of all sites that meet the user-specified site code (i.e., three letter abbreviation) and/or habitat criteria. See `lter_sites` for the full set of site information including accepted site codes and habitat designations (unrecognized entries will trigger a warning and be ignored). Lines are grouped and colored by habitat to better emphasize possible similarities among sites

Usage

```
site_timeline(sites = NULL, habitats = NULL, colors = NULL)
```

Arguments

sites	(character) three letter site code(s) identifying site(s) of interest
habitats	(character) habitat(s) of interest. See `unique(lter_sites$habitat)`
colors	(character) colors to assign to the timelines expressed as a hexadecimal (e.g, #00FF00). Note there must be as many colors as habitats included in the graph

Value

(ggplot2) plot object of timeline of site(s) that meet user-specified criteria

Examples

```
# Make the full timeline of all sites with default colors by supplying no arguments
site_timeline()

# Or make a timeline of only sites that meet certain criteria
site_timeline(habitats = c("grassland", "forest"))
```

---

solar_day_info *Identify Solar Day Information*

Description

For all days between the specified start and end date, identify the time of sunrise, sunset, and solar noon (in UTC) as well as the day length. The idea for this function was contributed by Miguel C. Leon and a Python equivalent lives in the Luquillo site's LUQ-general-utils GitHub repository.

Usage

```
solar_day_info(
  lat = NULL,
  lon = NULL,
  start_date = NULL,
  end_date = NULL,
  quiet = FALSE
)
```

Arguments

lat	(numeric) latitude coordinate for which to find day length
lon	(numeric) longitude coordinate for which to find day length
start_date	(character) starting date in 'YYYY-MM-DD' format
end_date	(character) ending date in 'YYYY-MM-DD' format
quiet	(logical) whether to suppress certain non-warning messages. Defaults to TRUE

Value

(dataframe) table of 6 columns and a number of rows equal to the number of days between the specified start and end dates (inclusive). Columns contain: (1) date, (2) sunrise time, (3) sunset time, (4) solar noon, (5) day length, and (6) time zone of columns 2 to 4.

Examples

```
## Not run:
# Identify day information in Santa Barbara (California) for one week
solar_day_info(lat = 34.416857, lon = -119.712777,
               start_date = "2022-02-07", end_date = "2022-02-12",
               quiet = TRUE)

## End(Not run)
```

---

standardize *Standardize a Single Dataset via a Column Key*

Description

A "column key" is meant to streamline harmonization of disparate datasets. This key must include three columns containing: (1) the name of each raw data file to be harmonized, (2) the name of all of the columns in each of those files, and (3) the "tidy name" that corresponds to each raw column name. This function accepts that key and a list of datasets that can be standardized with that key. The function standardizes the specified dataset out of any number of datasets in the key or list. While usable on its own, this function is intended to streamline internal operations of `ltertools::harmonize` – which is the recommended tool for key-based harmonization.

Usage

```
standardize(focal_file = NULL, key = NULL, df_list = NULL)
```

Arguments

`focal_file`	(character) filename corresponding to one value of "source" column of "key" data and to one name in "df_list".
`key`	(dataframe) key object including a "source", "raw_name" and "tidy_name" column. Additional columns are allowed but ignored
`df_list`	(list) named list of dataframe-like objects where each name is the filename initially containing that data

Value

(dataframe) single standardized dataframe including all columns defined in the "tidy_name" column of the key object

Examples

```
#' # Generate two simple tables
## Dataframe 1
df1 <- data.frame("xx" = c(1:3),
                  "unwanted" = c("not", "needed", "column"),
                  "yy" = letters[1:3])
## Dataframe 2
df2 <- data.frame("LETTERS" = letters[4:7],
                  "NUMBERS" = c(4:7),
                  "BONUS" = c("plantae", "animalia", "fungi", "protista"))

# Generate a local folder for exporting
temp_folder <- tempdir()

# Export both files to that folder
utils::write.csv(x = df1, file = file.path(temp_folder, "df1.csv"), row.names = FALSE)
utils::write.csv(x = df2, file = file.path(temp_folder, "df2.csv"), row.names = FALSE)

# Read in list of these data files
data_list <- ltertools::read(raw_folder = temp_folder, data_format = "csv")

# Generate a column key object manually
key_obj <- data.frame("source" = c(rep("df1.csv", 3),
                                   rep("df2.csv", 3)),
                      "raw_name" = c("xx", "unwanted", "yy",
                                     "LETTERS", "NUMBERS", "BONUS"),
                     "tidy_name" = c("numbers", NA, "letters",
                                     "letters", "numbers", "kingdom"))
# Standardize one dataset
ltertools::standardize(focal_file = "df1.csv", key = key_obj, df_list = data_list)
```

Index

∗ **datasets**
    lter_sites, 8

begin_key, 2

check_key, 3
convert_temp, 4
cv, 5

expand_key, 5

harmonize, 7

lter_sites, 8

read, 9

site_subset, 10
site_timeline, 10
solar_day_info, 11
standardize, 12