Package: rambur (via r-universe)

October 17, 2026

Title Design and Analysis of Intertidal Chamber Experiments

Version 0.5.6

Description Facilitates the design and analysis of intertidal chamber experiments as described in Pham & Sokolova (2026) <doi:10.64898/2026.09.28.754745>. It supports generating experimental profiles for 'ElectricBlue' intertidal chambers, reading log files, and checking actual chamber conditions against designed profiles. It also helps read log files from 'EnvLogger' robomussels (or temperature loggers) and 'Pulse V2' heart rate loggers, and provides an algorithm to extract heart rates from 'Pulse V2' records. The acronym stands for Research Assistant in Marine Biology at the University of Rostock.

License GPL (>= 3)

BugReports https://github.com/phamdn/rambur/issues

Depends R (>= 4.6.0)

Imports cowplot, dplyr, ggplot2, graphics, hms, lubridate, patchwork, purrr, readr, rlang, stats, tidyselect, utils

Config/roxygen2/version 8.1.1

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libx11-dev

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

Date/Publication 2026-10-16 17:17:02 UTC

RemoteUrl https://github.com/phamdn/rambur

RemoteRef HEAD

RemoteSha 40edbdabc52c56876038464251da7afc9dab8b71

Contents

add.datetime 2

chamber.check 3
chamber.design 4
chamber.diurnal 5
chamber.read 7
chamber.split 8
plot.chamber.design 8
plot.chamber.diurnal 9
pulse.extract 10
pulse.hr 11
pulse.read 12
robo.read 13
robos.read 13

Index **15**

`add.datetime` *Adding Local Datetime*

Description

A helper function to convert UTC datetime to local datetime.

Usage

```
add.datetime(df, type = 3, timezone, reverse = FALSE)
```

Arguments

`df`	a data frame.
`type`	an integer from 1 to 3. Use 1 to add datetime, 3 to add datetime, date, and time, and 2 to add date and time.
`timezone`	a character string, local time zone.
`reverse`	a logical, whether to convert local datetime to UTC datetime.

Value

a data frame with additional columns.

Examples

```
# example a
df <- data.frame(datetime.UTC = as.POSIXct(c("2025-01-01 01:00:00",
"2025-07-01 01:00:00"), tz = "UTC"))
df
df1 <- add.datetime(df, type = 1, timezone = "")
df1
df1$datetime.UTC
df1$datetime
```

```
df3 <- add.datetime(df, type = 3, timezone = "")
df3


# example b
data <- data.frame(datetime = as.POSIXct(c("2025-02-01 01:00:00", "2025-08-01 01:00:00")))
data$datetime
data2 <- add.datetime(data, type = 2)
data2
```

`chamber.check` *Chamber: Checking Performance*

Description

A function to compare the design and the actual performance of the chamber.

Usage

```
chamber.check(
  design,
  chamber.data,
  robo.data = NULL,
  dttm.limits = c(NA, NA),
  dttm.breaks = waiver(),
  dttm.labels = waiver(),
  ambient.label = c("Ambient", "Room"),
  temp.limits = NULL
)
```

Arguments

`design`	a data frame, multiday patterns of environmental variables. Use `design` output of `chamber.design`.
`chamber.data`	a data frame, actual records of environmental variables. Use `enhanced.data` output of `chamber.read`.
`robo.data`	a data frame, actual records of robomussels. Optional.
`dttm.limits`	a vector of two character strings, limits of dates.
`dttm.breaks`	a character string, duration between date breaks.
`dttm.labels`	a character string, format of dates.
`ambient.label`	a character string, how to ambient temperature.
`temp.limits`	a vector of two numeric, limits of temperature.

Value

a three-panel plot of light, immersion, and temperature.

Examples

```
acc.daily.settings <- data.frame(day = seq(0, 34))

acc <- chamber.design(acc.daily.settings,
                              start.date = "2025-05-15", export = FALSE)

folder <- system.file("extdata/chamber", package = "rambur")
chamber.data <- chamber.read(folder)

robo.folder <- system.file("extdata/robo", package = "rambur")
robo.data <- robos.read(robo.folder)

chamber.check(acc$design, chamber.data$enhanced.data, robo.data$synchronized.data,
dttm.limits = c("2025-05-21", "2025-05-24"), ambient.label = "Room")
```

`chamber.design` *Chamber: Designing Multiday Profile*

Description

A function to expand the diurnal patterns of environmental variables into multiday patterns.

Usage

```
chamber.design(
  daily.settings,
  start.date = "2025-04-30",
  timezone = "",
  export = TRUE,
  folder.path = NULL
)
```

Arguments

`daily.settings`	a data frame, diurnal patterns of environmental variables. Columns must match arguments of `chamber.diurnal()`.
`start.date`	a character string, start date of the experiment. Time automatically set to 00:00:00.
`timezone`	a character string, time zone.
`export`	a logical, whether to export the output as files (e.g., Profile.txt).
`folder.path`	a character string, path to the export folder.

Value

a list of two data frames, `daily.settings` as input but with an extra column `date`, and `design` of expanded multiday patterns.

See Also

chamber.diurnal()

Examples

```
# acclimation phase
acc.daily.settings <- data.frame(day = seq(0, 34),
temp.air.mean = 17, temp.water.mean = 19)

acc <- chamber.design(acc.daily.settings,
start.date = "2025-05-15", export = FALSE)
## to save the output files, set 'export' to 'TRUE'.

acc
as.POSIXct(acc$design$timestamp[1], tz = "UTC")
acc$design$datetime[1]
acc$daily.settings$date[1]
```

---

chamber.diurnal *Chamber: Designing Diurnal Profile*

Description

A worker function to design the diurnal patterns of environmental variables.

Usage

```
chamber.diurnal(
  day = 0,
  time.step = 1,
  light.model = "gaussian",
  light.duration = 16,
  light.peak.time = 13,
  light.max = 100,
  temp.model = "sinusoidal",
  temp.air.mean = 17,
  temp.air.range = 8,
  temp.water.mean = 19,
  temp.water.range = 1,
  temp.peak.time = 15,
  tidal.day = 24,
  ie.cycle = c(0, 1, 0, 1),
  iec.start.time = 0,
  wc.time = NA
)
```

Arguments

`day`	an integer, the day of experiment.
`time.step`	a numeric, the resolution of the profile in hours.
`light.model`	a character string, pattern of light. Default to "gaussian", indicating gaussian function.
`light.duration`	a numeric, light duration (photoperiod) in hours, with brightness of at least 1%.
`light.peak.time`	a numeric, time of day when brightness peaks.
`light.max`	an integer, maximum brightness.
`temp.model`	a character string, pattern of temperature. Default to "sinusoidal", indicating sinusoidal function.
`temp.air.mean`	a numeric, the mean of air temperature during the day in °C.
`temp.air.range`	a numeric, the range of air temperature during the day in °C.
`temp.water.mean`	a numeric, the mean of water temperature during the day in °C.
`temp.water.range`	a numeric, the range of water temperature during the day in °C.
`temp.peak.time`	a numeric, time of day when temperature peaks.
`tidal.day`	a numeric, tidal day duration in hours. Default to 24, same as the solar day.
`ie.cycle`	a vector of 0 and 1, the immersion-emersion cycle (e.g., semi-diurnal).
`iec.start.time`	a numeric, time of day when immersion-emersion cycle starts.
`wc.time`	a numeric, time of day when automatic water change starts.

Value

a data frame with eight columns, including time as `day` and `hour`, and environmental variables as `light`, `temp.air`, `temp.water`, `immersion`, `temp`, `wc`.

See Also

`chamber.design()`

Examples

```
# default
day0 <- chamber.diurnal()
day0
plot(day0)

# custom
chamber.diurnal(day = -3, time.step = 0.5,
  light.duration = 17, light.peak.time = 12,
  temp.air.mean = 30, temp.air.range = 10,
  temp.water.mean = 20, temp.water.range = 2, temp.peak.time = 14,
  ie.cycle =  c(0, 1), iec.start.time = 2.5,
  wc.time = 16.5)
```

chamber.read *Chamber: Reading Log Files*

Description

A function to read the CSV log files of an intertidal chamber.

Usage

```
chamber.read(
  folder.path = NULL,
  file.name = ".CSV",
  size.limits = c(0, Inf),
  metadata.lines = 16,
  timezone = "",
  agg.res = "1 minute"
)
```

Arguments

folder.path	a character string, path to the folder of CSV records.
file.name	a character string, filter the file name.
size.limits	a vector of two numerics, minimum and maximum size in bytes.
metadata.lines	an integer, number of lines to skip in the header.
timezone	a character string, time zone.
agg.res	a character string, duration to summarize the mean of the records.

Value

a list of three data frames, `original.data`, `enhanced.data`, and `aggregated.data` for original, enhanced, and summarized records, respectively.

Examples

```
folder <- system.file("extdata/chamber", package = "rambur")
chamber.data <- chamber.read(folder)
chamber.data
```

chamber.split *Chamber: Splitting Profile*

Description

A function to split the "Profile.txt" with more than 1000 lines into smaller files.

Usage

```
chamber.split(
  file.path,
  max.lines = 960,
  overlap.lines = 120,
  folder.path = NULL
)
```

Arguments

`file.path`	a character string, path to the original "Profile.txt" file.
`max.lines`	an integer, number of lines in new files.
`overlap.lines`	an integer, number of lines shared between two consecutive files.
`folder.path`	a character string, path to the new files after splitting.

Value

No return value, called for side effects. The function splits "Profile.txt" and saves each part into its own subdirectory.

plot.chamber.design *Chamber: Plotting Multiday Profile*

Description

A helper function to plot the multiday patterns of environmental variables.

Usage

```
## S3 method for class 'chamber.design'
plot(x, ...)
```

Arguments

`x`	a data frame, multiday patterns of environmental variables. Use `design` output of `chamber.design`.
`...`	additional arguments, currently ignored.

Value

a list of six plots of environmental variables.

Examples

```
# acclimation phase
acc.setup <- data.frame(day = seq(0, 34),
temp.air.mean = 17, temp.water.mean = 19)

acc.profile <- chamber.design(acc.setup,
start.date = "2025-05-15", export = FALSE)

plot(acc.profile$design)
```

---

plot.chamber.diurnal *Chamber: Plotting Diurnal Profile*

Description

A function to plot the diurnal patterns of environmental variables.

Usage

```
## S3 method for class 'chamber.diurnal'
plot(x, ..., display = c("all", "light", "immersion", "temperature"))
```

Arguments

x	a data frame, output of chamber.diurnal.
...	additional arguments, currently ignored.
display	a character string, which plots to display.

Value

a six-panel plot of environmental variables.

Examples

```
# default
day0 <- chamber.diurnal()
plot(day0)
plot(day0, display = "light")
plot(day0, display = "immersion")
plot(day0, display = "temp")
```

pulse.extract *Pulse: Extracting Heart Rate*

Description

A function to extract heart rate from records of pulse devices.

Usage

```
pulse.extract(
  pulse.data,
  sampling.rate = NULL,
  search.scope = 1,
  score.parameter = 0.5,
  cor.min = 0.5,
  display = "none",
  ppg.zoom = TRUE,
  time.window = "1 minute",
  agg.res = NULL
)
```

Arguments

pulse.data	a data frame, records of signal. Use output of pulse.read.
sampling.rate	an integer, sampling rate in Hz. Can be autocalculated from data.
search.scope	a numeric, the ratio of lag domain length to the total signal length.
score.parameter	a numeric, the exponent used in the score.
cor.min	a numeric, the correlation threshold for qualified signal.
display	a logical, whether to display plots.
ppg.zoom	a logical, whether to zoom the PPG plot to fit signal intensity.
time.window	a character string, window duration extract heart rate.
agg.res	a character string, duration to summarize the mean or median of the records.

Value

a list of two data frames, window.hr and aggregated.hr for window and summarized heart rate.

See Also

pulse.hr()

Examples

```
folder <- system.file("extdata/pulse", package = "rambur")
pulse.data <- pulse.read(folder)
pulse.extract(pulse.data)
```

pulse.hr *Pulse: Calculating Heart Rate*

Description

A worker function to calculate heart rate

Usage

```
pulse.hr(
  signal,
  sampling.rate = 5,
  search.scope = 1,
  score.parameter = 0.5,
  cor.min = 0.5,
  display = c("all", "ppg", "none"),
  ppg.zoom = TRUE
)
```

Arguments

signal	an integer vector, infrared signal from pulse device.
sampling.rate	an integer, sampling rate in Hz.
search.scope	a numeric, the ratio of lag domain length to the total signal length.
score.parameter	a numeric, the exponent used in the score.
cor.min	a numeric, the correlation threshold for qualified signal.
display	a character string, whether to display plots.
ppg.zoom	a logical, whether to zoom the PPG plot to fit signal intensity.

Value

a mixed list, `locmax` for local maxima, `nominee` for the best peak, `cor.pass` and `anticor.pass` for quality control, and `hr` for final heart rate.

See Also

`pulse.extract()`

Examples

```
folder <- system.file("extdata/pulse", package = "rambur")
pulse.data <- pulse.read(folder, timezone = "Europe/Berlin")
ex1 <- pulse.hr(subset(pulse.data,
datetime >= as.POSIXct("2025-05-21 00:01:00", tz = "Europe/Berlin") &
datetime < as.POSIXct("2025-05-21 00:02:00", tz = "Europe/Berlin"),
channel.1, drop = TRUE))
```

pulse.read *Pulse: Reading Log Files*

Description

A function to read the CSV log files of a PULSE V2 logger.

Usage

```
pulse.read(
  folder.path = NULL,
  file.name = ".CSV",
  size.limits = c(0, Inf),
  metadata.lines = 22,
  channel.names = NULL,
  timezone = ""
)
```

Arguments

folder.path	a character string, path to the folder of CSV records.
file.name	a character string, filter the file name.
size.limits	a vector of two numerics, minimum and maximum file sizes in bytes, e.g., 1e6 bytes (~1 MB).
metadata.lines	an integer, number of lines to skip in the header.
channel.names	a character vector of length 10. Optional names for channels.
timezone	a character string, time zone.

Value

a data frame of enhanced data.

Examples

```
folder <- system.file("extdata/pulse", package = "rambur")
pulse.data <- pulse.read(folder)
pulse.data
```

robo.read — *Robomussel: Reading a Log File*

Description

A function to read the CSV log file of a single robomussel or temperature EnvLogger.

Usage

```
robo.read(file.path, metadata.lines = 21, timezone = "", agg.res = NULL)
```

Arguments

`file.path`	a character string, path to the file.
`metadata.lines`	an integer, number of lines to skip in the header.
`timezone`	a character string, time zone.
`agg.res`	a character string, duration to summarize the mean of the records. Optional.

Value

a list of two data frames, `enhanced.data` and `aggregated.data` for enhanced and summarized records, respectively.

Examples

```
robo.file <- system.file("extdata/robo/RM1.csv", package = "rambur")
robo.read(robo.file, agg.res = "1 hour")
```

robos.read — *RobomusselS: Reading Log Files*

Description

A function to read the CSV log files of multiple robomussels or temperature EnvLoggers.

Usage

```
robos.read(
  folder.path = NULL,
  file.name = ".csv",
  metadata.lines = 21,
  timezone = "",
  agg.res = NULL
)
```

Arguments

`folder.path`	a character string, path to the folder.
`file.name`	a character string, filter the file name.
`metadata.lines`	an integer, number of lines to skip in the header.
`timezone`	a character string, time zone.
`agg.res`	a character string, duration to summarize the mean of the records. Optional.

Value

a list of three data frames, `enhanced.data`, `synchronized.data`, and `aggregated.data` for enhanced, synchronized, and aggregated data, respectively.

Examples

```
robo.folder <- system.file("extdata/robo", package = "rambur")
robos.read(robo.folder, agg.res = "1 hour")
```

Index

add.datetime, 2

chamber.check, 3
chamber.design, *3*, 4, *8*
chamber.design(), *6*
chamber.diurnal, 5, *9*
chamber.diurnal(), *4, 5*
chamber.read, *3*, 7
chamber.split, 8

plot.chamber.design, 8
plot.chamber.diurnal, 9
pulse.extract, 10
pulse.extract(), *11*
pulse.hr, 11
pulse.hr(), *10*
pulse.read, *10*, 12

robo.read, 13
robos.read, 13