Package 'LightFitR'

July 25, 2025

Type Package

Title Design Complex Light Regimes

Version 1.0.0

Description A system for accurately designing complex light regimes using LEDs. Takes calibration data and user-defined target irradiances and it tells you what intensities to use.
For more details see Vong et al. (2025) <doi:10.1101/2025.06.06.658293>.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/ginavong/LightFitR/

BugReports https://github.com/ginavong/LightFitR/issues

LazyData true

Depends R (>= 3.5)

Imports utils, graphics, lubridate, nnls, stringr

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Gina Vong [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-3913-7667>)

Maintainer Gina Vong <gywv500@york.ac.uk>

Repository CRAN

Date/Publication 2025-07-25 16:30:14 UTC

Contents

`calibration` *calibration data*

Description

Example calibration data

Usage

```
calibration
```

Format

A data frame with 12 columns:

filename File that the raw data came from

time The time when a given measurement was taken

led LED channel being calibrated at that timepoint

intensity Intensity the light is set to

wavelength The wavelength this row describes

irradiance The irradiance measured at that wavelength by the spectrometer

Source

<https://github.com/ginavong/2024_LightFitR_MethodsPaper/blob/master/data/heliospectra_measurements/calibration/Apo

checkRange *Check that the intended irradiances are acheivable by the lights*

Description

Check that the intended irradiances are acheivable by the lights

Usage

```
checkRange(
  intended_irradiance,
  calibration_leds,
  calibration_wavelengths,
  calibration_intensities,
  calibration_irradiances
)
```

Arguments

`intended_irradiance`	Matrix of intended irradiances to be checked. Each row corresponds to an LED channel.
`calibration_leds`	A numeric vector of LED values from calibration, mapping to intensities and irradiances (i.e. the same length)
`calibration_wavelengths`	A numeric vector of wavelengths from calibration, corresponding to intensities and irradiances
`calibration_intensities`	A numeric vector of intensities (heliospectra units) from calibration
`calibration_irradiances`	A numeric vector of measured irradiances (any units, as long as it is consistently used) from calibration

Value

Boolean: TRUE = pass, FALSE = fail

Examples

```
calib <- LightFitR::calibration
irradiances <- LightFitR::target_irradiance
checkRange(irradiances, calib$led, calib$wavelength, calib$intensity, calib$irradiance)
```

`example_closest` *closest intensities*

Description

Matrix of closest intensities for example purposes. Generated from ‘target_irradiance‘

Usage

```
example_closest
```

Format

A matrix with 9 rows and 10 columns: each row represents an LED channel and each column represents an event

`example_intensities` *example intensities*

Description

Matrix of random intensities for example purposes

Usage

```
example_intensities
```

Format

A matrix with 9 rows and 10 columns: each row represents an LED channel and each column represents an event

example_regime *regime matrix*

Description

Example regime matrix

Usage

```
example_regime
```

Format

A matrix with 13 rows and 10 columns:

time time in HH:MM:SS format

hour The hour of the event

minute The minute of the event

second The second of the event

380nm Intensity at 380nm LED channel

400nm Intensity at 400nm LED channel

etc.

helio.checkFormat *Check formatting of the heliospectra matrices*

Description

Heliospectra DYNA matrices should have 9 rows (1 for each LED channel) and up to 150 columns (max number of events that Heliospectra DYNA can store)

Usage

```
helio.checkFormat(check_matrix)
```

Arguments

check_matrix	Matrix to be checked. Rows correspond to LEDs and columns are events / time-points.

Value

Vector of booleans: TRUE = pass, FALSE = fail

Examples

```
matrix_to_check <- LightFitR::target_irradiance
helio.checkFormat(target_irradiance)
```

`helio.csv_schedule` *Format regime_matrix for csv output that Heliospectra lights can parse*

Description

Format regime_matrix for csv output that Heliospectra lights can parse

Usage

```
helio.csv_schedule(regime_matrix, filename)
```

Arguments

`regime_matrix`	Matrix containing light regime, as generated by makeRegime
`filename`	Character. Filename to export to

Value

Matrix formatted for Heliospectra lights csv

Examples

```
tempfile_name = tempfile(fileext='.csv')
helio.csv_schedule(LightFitR::example_regime, tempfile_name)
```

`helio.disco` *Turn Heliospectra into disco lights*

Description

Congratulations! You've found the easter egg function!

Usage

```
helio.disco(filename, format = c("csv", "json"))
```

Arguments

filename	Character. Filename to export to
format	Character. Which format to export to? csv or json. Use extensions '.csv' or '.txt'

Details

This writes a schedule for Heliospectra DYNA lights which randomly changes colour every second for a fun disco effect. The disco will last until the Heliospectra runs out of memory (150 events), so you can get 2 min 30s of disco out of your expensive lights... Enjoy!

Value

Disco schedule file for the heliospectra

Examples

```
tempfile_name = tempfile(fileext='.txt')
helio.disco(tempfile_name, format='json')
```

---

helio.dyna.leds *heliospectra DYNA LEDs*

Description

Data about the heliospectra DYNA LED channels

Usage

```
helio.dyna.leds
```

Format

A data frame with 9 rows and 3 columns:

name Name of the LED channel

wavelength Wavelength of the LED channel

colour Colour of the LED channel

Source

<https://heliospectra.com/led-grow-lights/dyna/>

helio.eventLimit *Maximum number of events*

Description

Maximum number of events programmable onto heliospectra

Usage

```
helio.eventLimit
```

Format

Integer representing maximum allowable events

helio.json_schedule *Format regime_matrix for json output that Heliospectra lights can parse*

Description

Format regime_matrix for json output that Heliospectra lights can parse

Usage

```
helio.json_schedule(regime_matrix, filename)
```

Arguments

regime_matrix	Matrix containing light regime, as generated by makeRegime
filename	Character. Filename to export to

Value

Character in json format that Heliospectra can parse

Examples

```
tempfile_name = tempfile(fileext='.txt')
helio.csv_schedule(LightFitR::example_regime, tempfile_name)
```

internal.closestIntensities *Internal function. Find the intensities corresponding to the closest irradiance match between intended and calibration.*

Description

Internal function. Find the intensities corresponding to the closest irradiance match between intended and calibration.

Usage

```
internal.closestIntensities(
  irradiance_matrix,
  calibration_df,
  peaks = LightFitR::helio.dyna.leds$wavelength
)
```

Arguments

irradiance_matrix	Matrix of intended irradiances. rows = leds and columns = events
calibration_df	Dataframe of calibration data generated by internal.calibCombine
peaks	Vector of length 8 or 9. Containing wavelengths at which each LED peaks.

Value

Matrix of closest intensities, in the same format as 'irradiance_matrix'

Examples

```
# Format calibration data correctly
calib <- LightFitR::calibration[, c(3, 5, 4, 6)]

# Run function
internal.closestIntensities(LightFitR::target_irradiance, calib)
```

`makeRegime` *Create a regime (matrix) to program lights to achieve intended irradiances*

Description

This is a wrapper function that carries out multiple steps:

1. Calculate closest intensities

2. Predict the intensities to use to achieve the target irradiance (via a system of linear equations or non-negative least squares)

3. Tidy the intensities (rounding to integer, keep within the range of intensities that the lights can be set to)

4. Format the intensities and timestamps into a human-readable regime matrix

Usage

```
makeRegime(
  timeVector_POSIXct,
  irradiance_matrix,
  calibration_leds,
  calibration_wavelengths,
  calibration_intensities,
  calibration_irradiances,
  peaks = LightFitR::helio.dyna.leds$wavelength,
  method = "nnls"
)
```

Arguments

`timeVector_POSIXct`	Vector of schedule timepoints in POSICxt format
`irradiance_matrix`	Matrix of intended irradiances. rows = leds and columns = events
`calibration_leds`	A numeric vector of LED values from calibration, mapping to intensities and irradiances (i.e. the same length)
`calibration_wavelengths`	A numeric vector of wavelengths from calibration, corresponding to intensities and irradiances
`calibration_intensities`	A numeric vector of intensities (heliospectra units) from calibration
`calibration_irradiances`	A numeric vector of measured irradiances (any units, as long as it is consistently used) from calibration
`peaks`	Vector of length 8 or 9. Containing wavelengths at which each LED peaks.
`method`	Use 'nnls' (non-negative least squares) or 'sle' (system of linear equations)

Value

Matrix with light regime needed to program the lights

NNLS vs SLE

NNLS and SLE largely predict the same intensities, except in outlier cases. The default is NNLS, but if your predicted intensities end up being very far off, try SLE.

Examples

```
# Prep variables
calib <- LightFitR::calibration
times <- LightFitR::time_vector
target_irradiance <- LightFitR::target_irradiance

# Run function
makeRegime(times, target_irradiance, calib$led, calib$wavelength, calib$intensity, calib$irradiance)
```

---

nnls_intensities *Use non-linear least squares to interpolate intensities*

Description

Use non-linear least squares to interpolate intensities

Usage

```
nnls_intensities(
  irradiance_matrix,
  closest_intensities,
  calibration_leds,
  calibration_wavelengths,
  calibration_intensities,
  calibration_irradiances,
  peaks = LightFitR::helio.dyna.leds$wavelength
)
```

Arguments

`irradiance_matrix`
: Matrix of intended irradiances. rows = leds and columns = events

`closest_intensities`
: Matrix of closest intensities, generated by ‘internal.closestIntensities‘. Same format as ‘irradiance_matrix‘

`calibration_leds`
: A numeric vector of LED values from calibration, mapping to intensities and irradiances (i.e. the same length)

`calibration_wavelengths`
: A numeric vector of wavelengths from calibration, corresponding to intensities and irradiances

`calibration_intensities`
: A numeric vector of intensities (heliospectra units) from calibration

`calibration_irradiances`
: A numeric vector of measured irradiances (any units, as long as it is consistently used) from calibration

`peaks`
: Vector of length 8 or 9. Containing wavelengths at which each LED peaks.

Value

Matrix of intensities to set the lights to, to achieve desired irradiances

Examples

```
# Prep variables
target_irradiance = LightFitR::target_irradiance
closest = LightFitR::example_closest
calib = LightFitR::calibration

# Run the function
nnls_intensities(target_irradiance, closest,
  calib$led, calib$wavelength, calib$intensity, calib$irradiance)
```

---

read.helio_json *Read a heliospectra script (json format) into a matrix.*

Description

Read a heliospectra script (json format) into a matrix.

Usage

```
read.helio_json(helio_script)
```

Arguments

`helio_script` File (.txt or .json) containing heliospectra regime script

Value

Matrix containing the regime encoded by the Heliospectra script

Examples

```
example_file <- system.file("extdata", "example_json_schedule.txt",
  package = "LightFitR", mustWork = TRUE)
read.helio_json(example_file)
```

`sle_intensities` *Use a system of linear equations to calculate intensities*

Description

Use a system of linear equations to calculate intensities

Usage

```
sle_intensities(
  irradiance_matrix,
  closest_intensities,
  calibration_leds,
  calibration_wavelengths,
  calibration_intensities,
  calibration_irradiances,
  peaks = LightFitR::helio.dyna.leds$wavelength
)
```

Arguments

`irradiance_matrix`	Matrix of intended irradiances. rows = leds and columns = events
`closest_intensities`	Matrix of closest intensities, generated by ‘internal.closestIntensities‘. Same format as ‘irradiance_matrix‘
`calibration_leds`	A numeric vector of LED values from calibration, mapping to intensities and irradiances (i.e. the same length)
`calibration_wavelengths`	A numeric vector of wavelengths from calibration, corresponding to intensities and irradiances
`calibration_intensities`	A numeric vector of intensities (heliospectra units) from calibration
`calibration_irradiances`	A numeric vector of measured irradiances (any units, as long as it is consistently used) from calibration
`peaks`	Vector of length 8 or 9. Containing wavelengths at which each LED peaks.

Value

Matrix of intensities to set the lights to, to achieve desired irradiances

Examples

```
#' # Prep variables
target_irradiance = LightFitR::target_irradiance
closest = LightFitR::example_closest
calib = LightFitR::calibration

# Run the function
sle_intensities(target_irradiance, closest,
  calib$led, calib$wavelength, calib$intensity, calib$irradiance)
```

---

target_irradiance *target irradiances*

Description

Matrix of random target irradiances for example purposes

Usage

```
target_irradiance
```

Format

A matrix with 9 rows and 10 columns: each row represents an LED channel and each column represents an event

---

time_vector *time vector*

Description

Example timepoints for events

Usage

```
time_vector
```

Format

A vector of length 10 with timepoints in POSIXct format

`write.helioSchedule` *Write the schedule to file that Heliospectra can parse*

Description

Writes to json or csv format

Usage

```
write.helioSchedule(regime_matrix, filename, format = c("csv", "json"))
```

Arguments

`regime_matrix`	Matrix containing light regime, as generated by makeRegime
`filename`	Character. Filename to export to
`format`	Character. Which format to export to? csv or json. Use extensions '.csv' or '.txt'

Value

Heliospectra schedule file in either the csv or json format

Examples

```
tempcsv_name = tempfile(fileext='.csv')
write.helioSchedule(LightFitR::example_regime, tempcsv_name, format='csv')

temptxt_name = tempfile(fileext='.txt')
write.helioSchedule(LightFitR::example_regime, temptxt_name, format='json')
```

Index