Package ‘pressuRe’

April 11, 2024

Type Package

Title Imports, Processes, and Visualizes Biomechanical Pressure Data

Version 0.2.4

Description Allows biomechanical pressure data from a range of systems to be imported and processed in a reproducible manner. Automatic and manual tools are included to let the user define regions (masks) to be analyzed. Also includes functions for visualizing and animating pressure data. Example methods are described in Shi et al., (2022) <doi:10.1038/s41598-022-19814-0>, Lee et al., (2014) <doi:10.1186/1757-1146-7-18>, van der Zward et al., (2014) <doi:10.1186/1757-1146-7-20>, Najafi et al., (2010) <doi:10.1016/j.gaitpost.2009.09.003>, Cavanagh and Rodgers (1987) <doi:10.1016/0021-9290(87)90255-7>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

URL https://github.com/Telfer/pressuRe

Imports abind, dplyr, gdistance, ggmap, ggplot2, magick, magrittr, pracma, raster, readxl, scales, sf, stringr, zoo

Depends R (>= 2.10)

NeedsCompilation no

Author Scott Telfer [aut, cre, cph] (<https://orcid.org/0000-0002-0104-4027>), Ellen Li [aut] (<https://orcid.org/0000-0001-5545-7364>)

Maintainer Scott Telfer <scott.telfer@gmail.com>

Repository CRAN

Date/Publication 2024-04-10 23:40:31 UTC

R topics documented:

`animate_pressure` *Animate pressure*

Description

Produces animation (gif) of pressure data

Usage

```
animate_pressure(
  pressure_data,
  plot_colors = "default",
  fps,
  dpi = 96,
  file_name
)
```

Arguments

`pressure_data`	Array. A 3D array covering each timepoint of the measurement. z dimension represents time
`plot_colors`	String
`fps`	Numeric. Number of frames per second in animation
`dpi`	Numeric. Resolution of gif
`file_name`	Name (inlcuding path) of export file

Value

Animation in gif format

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
animate_pressure(pressure_data, fps = 10, file_name = "pli_gif.gif")
```

`arch_index` *Calculate Arch Index.*

Description

Calculate Arch Index.

Usage

```
arch_index(pressure_data, plot = TRUE)
```

Arguments

`pressure_data`	List. Includes a 3D array covering each timepoint of the measurement. z dimension represents time
`plot`	Logical. Not implemented yet

Value

Numeric. Arch index value

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
arch_index(pressure_data)
```

auto_detect_side *Detect foot side*

Description

Detects which foot plantar pressure data is from (left or right), usually would only be needed for barefoot pressure plate data. Generally reliable but may be thrown off by severe deformities or abnormal walking patterns

Usage

```
auto_detect_side(pressure_data)
```

Arguments

pressure_data	List. First item should be a 3D array covering each timepoint of the measurement. z dimension represents time

Value

String. "LEFT" or "RIGHT"

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
auto_detect_side(pressure_data)
```

cop *Center of pressure*

Description

Generates xy coordinates for center of pressure during each frame of measurement

Usage

```
cop(pressure_data)
```

Arguments

pressure_data	List. First item is a 3D array covering each timepoint of the measurement. z dimension represents time

Value

Data frame with x and y coordinates of COP throughout trial

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
cop(pressure_data)
```

---

cpei *CPEI*

Description

Determine Center of Pressure Excursion Index (CPEI) for footprint pressure data

Usage

```
cpei(pressure_data, foot_side, plot_result = TRUE)
```

Arguments

pressure_data	List. First item is a 3D array covering each timepoint of the measurement. Not currently available for pedar.
foot_side	String. "right" or "left". Required for automatic detection of points
plot_result	Logical. Plots pressure image with COP and CPEI overlaid

Value

Numeric. CPEI value

Author(s)

Scott Telfer <scott.telfer@gmail.com>

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
cpei(pressure_data, foot_side = "auto", plot_result = FALSE)
```

`create_mask_auto` *Automatically mask pressure footprint*

Description

Automatically creates mask for footprint data

Usage

```
create_mask_auto(
  pressure_data,
  masking_scheme,
  foot_side = "auto",
  res_value = 10000,
  plot = TRUE
)
```

Arguments

`pressure_data` List. First item is a 3D array covering each timepoint of the measurement. z dimension represents time

`masking_scheme` String. "automask_simple", "automask_novel", "pedar_mask1", "pedar_mask2", "pedar_mask3". "simple_automask" applies a simple 3 part mask (hindfoot, midfoot, forefoot) "automask_novel" attempts to apply a 9-part mask (hindfoot, midfoot, mets, hallux, lesser toes), similar to the standard novel automask "pedar_mask1" splits the insole into 4 regions using sensel boundaries: hindfoot, midfoot, forefoot, and toes- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9470545/ "pedar_mask2" splits the insole into 4 regions using percentages: hindfoot, forefoot, hallux, and lesser toes- https://jfootankleres.biomedcentral.com/articles/10.1186/1757-1146-7-18 "pedar_mask3" splits the foot into 9 regions using sensel boundaries: medial hindfoot, lateral hindfoot, medial midfoot, lateral midfoot, MTPJ1, MTPJ2-3, MTPJ4-5, hallux, and lesser toes- https://jfootankleres.biomedcentral.com/articles/10.1186/175
1146-7-20

`foot_side` String. "RIGHT", "LEFT", or "auto". Auto uses auto_detect_side function

`res_value` Numeric. Adjusting this can help if the line between the forefoot and toes isn't correct. Default is 100000. This line is calculated using a least cost function and this parameter basically adjusts the resistance of the pressure value for that algorithm

`plot` Logical. Whether to play the animation

Value

List. Masks are added to pressure data variable

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time

- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
pressure_data <- create_mask_auto(pressure_data, "automask_novel",
res_value = 100000, foot_side = "auto", plot = FALSE)
```

`create_mask_manual` *Create masking*

Description

Allows user to manually define mask regions

Usage

```
create_mask_manual(
  pressure_data,
  mask_definition = "by_vertices",
  n_masks = 1,
  n_verts = 4,
  n_sens = 4,
  threshold = 0.005,
  plot_existing_masks = TRUE,
  mask_names = "default",
  plot = TRUE
)
```

Arguments

`pressure_data` List. First item is a matrix covering each timepoint of the measurement.

`mask_definition` String. "by_vertices" or "by_sensors". The first option let's you draw a shape around the area you want to select, the second allows you to define this area by clicking on specific sensors

`n_masks` Numeric. Number of masks to add

`n_verts` Numeric. Number of vertices in mask

`n_sens` Numeric. Number of sensors mask will contain `threshold` Numeric. Distance between adjacent mask vertices before sharing vertex coordinates

`plot_existing_masks` Logical. Show existing masks

`mask_names` List. Mask names. Default is "custom_mask#"

`plot` Logical. Show new maks on pressure image

Value

List. Mask(s) are added to pressure data variable

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix Matrix with maximum image

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
pressure_data <- create_mask_manual(pressure_data, mask_definition = "by_vertices",
n_masks = 1, n_verts = 4)
pressure_data <- create_mask_manual(pressure_data, mask_definition = "by_sensors",
n_masks = 1, n_sens = 8)
```

`edit_mask` *Edit mask*

Description

Allows user to manually adjust mask vertices

Usage

```
edit_mask(
  pressure_data,
  n_edit,
  threshold = 0.002,
  edit_list = seq(1, length(pressure_data[[5]])),
  image = "max"
)
```

Arguments

`pressure_data` List. First item is a 3D array covering each timepoint of the measurement.

`n_edit` Numeric. Number of vertices to edit

`threshold` Numeric. Distance between point clicked and vertex that is selected

`edit_list` List. Mask numbers that want to be edited. (Default is to load all masks so that adjacent masks with shared coordinates are modified together)

`image` String."max" = footprint of maximum sensors. "mean" average value of sensors over time

Value

List. Edited mask is added to the pressure data variable

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix Matrix with maximum image

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
pressure_data <- create_mask_auto(pressure_data, "automask_novel",
foot_side = "auto", plot = FALSE)
pressure_data <- edit_mask(pressure_data, n_edit = 1, threshold = 0.002,
image = "max")
```

`footprint` *Footprint*

Description

Determines footprint of pressure data

Usage

```
footprint(pressure_data, variable = "max", frame = NULL, plot = FALSE)
```

Arguments

`pressure_data`	List. Includes a 3D array covering each timepoint of the measurement. z dimension represents time
`variable`	String. "max" = maximum value of each sensor across full dataset. "mean" = average value of sensors over full dataset."frame" = an individual pressure frame. "meanmax" average max values across cycles (currently just for pedar)
`frame`	Integer. Only used if variable = "frame".
`plot`	Logical. Display pressure image

Value

Matrix. Maximum or mean values for all sensors

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
footprint(pressure_data, plot = FALSE)
```

`load_emed` *Load emed data*

Description

Imports and formats .lst files collected on emed system and exported from Novel software

Usage

```
load_emed(pressure_filepath)
```

Arguments

`pressure_filepath`	String. Filepath pointing to emed pressure file

Value

A list with information about the pressure data.

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the areas of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix Matrix with maximum image

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
```

`load_footscan` *Load footscan data*

Description

Imports and formats files collected on footscan systems (formerly RSScan)

Usage

```
load_footscan(pressure_filepath)
```

Arguments

`pressure_filepath`
String. Filepath pointing to emed pressure file

Value

A list with information about the pressure data.

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix. Matrix

@examples footscan_data <- system.file("extdata", "footscan_test.xls", package = "pressuRe") pressure_data <- load_footscan(footscan_data) @importFrom readxl read_excel @export `load_pedar` *Load pedar data*

Description

Imports and formats .asc files collected on pedar system and exported from Novel software

Usage

```
load_pedar(pressure_filepath)
```

Arguments

`pressure_filepath`
String. Filepath pointing to pedar pressure file

Value

A list with information about the pressure data.

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. String with sensor type
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix Matrix with maximum image

Examples

```
pedar_data <- system.file("extdata", "pedar_example.asc", package = "pressuRe")
pressure_data <- load_pedar(pedar_data)
```

`load_pliance` *Load pliance data*

Description

Imports and formats .asc files collected on pliance system and exported from Novel software

Usage

```
load_pliance(pressure_filepath)
```

Arguments

`pressure_filepath`
String. Filepath pointing to pliance pressure file

Value

A list with information about the pressure data.

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. String with sensor type
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix. Matrix

Examples

```
pliance_data <- system.file("extdata", "pliance_test.asc", package = "pressuRe")
pressure_data <- load_pliance(pliance_data)
```

`load_tekscan` *Load Tekscan data*

Description

Imports and formats files collected on tekscan systems and exported from Tekscan software

Usage

```
load_tekscan(pressure_filepath)
```

Arguments

`pressure_filepath`
String. Filepath pointing to emed pressure file

Value

A list with information about the pressure data.

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix. Matrix

@examples tekscan_data <- system.file("extdata", "fscan_testL.asf", package = "pressuRe") pressure_data <- load_tekscan(tekscan_data) @importFrom @export

`load_xsensor` *Load xsensor data*

Description

Imports and formats files collected on xsensor insole systems

Usage

```
load_xsensor(pressure_filepath)
```

Arguments

`pressure_filepath`
String. Filepath pointing to emed pressure file

Value

A list with information about the pressure data.

- pressure_array. 2D array covering each timepoint of the measurement. row dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix. Matrix

@examples xsensor_data <- system.file("extdata", "xsensor_data.csv", package = "pressuRe") pressure_data <- load_xsensor(xsensor_data) @importFrom abind abind @export

---

`mask_analysis` *Analyze masked regions of pressure data*

Description

Analyze masked regions of pressure data

Usage

```
mask_analysis(
  pressure_data,
  partial_sensors = FALSE,
  variable = "press_peak_sensor",
  pressure_units = "kPa",
  area_units = "cm2"
)
```

Arguments

`pressure_data` List. Includes a 3D array covering each timepoint of the measurement. z dimension represents time

`partial_sensors`
Logical Defines how sensors that do not lie wholly within mask are dealt with. If FALSE, they will be excluded; if TRUE, for relevant variables their contribution will be weighted by the proportion of the sensor that falls within the mask border

`variable` String. Variable to be determined. "press_peak_sensor", "press_peak_mask", "contact_area_peak", "pti_1", "pti_2", "force_time_integral", "force_peak", "dpli", "press_peak_sensor_ts", "force_ts". Variables ending in "_ts" produce time series data

`pressure_units` String. Default "kPa". Other options: "MPa", "Ncm2" (Newtons per square centimeter)

`area_units` String. Default is "cm2" (square centimeters). Other options "m2" (square meters); "mm2" (square millimeters)

Value

Data frame. Contains values for each mask plus additional information relevant to the data including cycle/step and foot side

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
pressure_data <- create_mask_auto(pressure_data, "automask_simple", plot = FALSE)
mask_analysis(pressure_data, FALSE, variable = "press_peak_sensor")
```

`pedar_insole_areas` *pedar sensor size data*

Description

Sensor sizes for different pedar insoles

Usage

```
pedar_insole_areas
```

Format

‘pedar_insole_areas‘ A data frame with 198 rows and 8 columns:

u areas for size u

v areas for size v

w areas for size w

x areas for size x

y areas for size y

uw areas for size uw

xw areas for size xw

vw areas for size vw

Source

Scott Telfer

`pedar_insole_grid` *pedar sensor grid data*

Description

Sensor outline coordinates

Usage

```
pedar_insole_grid
```

Format

‘pedar_insole_grid‘ A data frame with 199 rows and 8 columns:

V1 x1 coordinate

V2 y1 coordinate

V3 x2 coordinate

V4 y2 coordinate

V5 x3 coordinate

V6 y3 coordinate

V7 x4 coordinate

V8 y4 coordinate

Source

Scott Telfer

`plot_pressure` *Plot pressure*

Description

Produces visualization of pressure data

Usage

```
plot_pressure(
  pressure_data,
  variable = "max",
  smooth = FALSE,
  frame,
  step_n = "max",
  plot_COP = FALSE,
  plot_outline = FALSE,
  plot_colors = "default",
  break_values,
  break_colors,
  sensor_outline = TRUE,
  plot = TRUE,
  legend = TRUE
)
```

Arguments

`pressure_data`	List. Includes a 3D array covering each timepoint of the measurement. z dimension represents time
`variable`	String. "max" = footprint of maximum sensors. "mean" = average value of sensors over time (usually for static analyses). "frame" = an individual frame
`smooth`	Logical. Not implemented. If TRUE, plot will interpolate between sensors to increase data density
`frame`	Integer.
`step_n`	If numeric, the step number to plot (only for insole data). If "max", the max across complete trial, if "meanmax", the max on a per step basis
`plot_COP`	Logical. If TRUE, overlay COP data on plot. Default = FALSE
`plot_outline`	Logical. If TRUE, overlay convex hull outline on plot
`plot_colors`	String. "default": novel color scheme; "custom": user supplied
`break_values`	Vector. If plot_colors is "custom", values to split colors at
`break_colors`	Vector. If plot_colors is "custom", colors to use. Should be one shorter than break_values
`sensor_outline`	Logical. Sensor outline to be shown
`plot`	Logical. If TRUE, plot will be displayed
`legend`	Logical. If TRUE, legend will be added to plot

Value

ggplot plot object

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
plot_pressure(pressure_data, variable = "max", plot_COP = FALSE)
plot_pressure(pressure_data, variable = "frame", frame = 20,
              plot_colors = "custom", break_values = c(100, 200, 300),
              break_colors = c("light blue", "light green", "yellow", "pink"))
```

`pressure_interp` *Interpolate pressure data*

Description

Resamples pressure data over time. Useful for normalizing to stance phase, for example

Usage

```
pressure_interp(pressure_data, interp_to)
```

Arguments

`pressure_data`	List. First item should be a 3D array covering each timepoint of the measurement. z dimension represents time.
`interp_to`	Integer. Number of frames to interpolate to

Value

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix. Matrix

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
pressure_data <- pressure_interp(pressure_data, interp_to = 101)
```

select_steps *Select steps*

Description

Select steps, usually from insole data, and format for analysis

Usage

```
select_steps(
  pressure_data,
  threshold = "auto",
  min_frames = 10,
  n_steps = 5,
  skip = 2
)
```

Arguments

pressure_data	List. First item should be a 3D array covering each timepoint of the measurement. z dimension represents time.
threshold	Numeric. Threshold force to define start and end of step. If "auto", function will set threshold at minimum force in trial + 10N
min_frames	Numeric. Minimum number of frames that need to be in step
n_steps	Numeric. Target number of steps/cycles. User will be asked to keep selected steps until this target is reached or they run out of candidate steps
skip	Numeric. Usually the first few steps of a trial are accelerating and not representative of steady state walking so this removes them

Value

- pressure_array. 3D array covering each timepoint of the measurement. z dimension represents time
- pressure_system. String defining pressure system
- sens_size. Numeric vector with the dimensions of the sensors
- time. Numeric value for time between measurements
- masks. List
- events. List
- sensor_polygons. Data frame with corners of sensors
- max_matrix. Matrix

Examples

```
pedar_data <- system.file("extdata", "pedar_example.asc", package = "pressuRe")
pressure_data <- load_pedar(pedar_data)
pressure_data <- select_steps(pressure_data)
```

whole_pressure_curve *Whole pressure curve*

Description

Generates vectors with option to plot for force, peak/mean pressure and area for complete measurement. Useful for checking data

Usage

```
whole_pressure_curve(
  pressure_data,
  variable,
  side,
  threshold = 10,
  plot = FALSE
)
```

Arguments

pressure_data	List. A 3D array covering each timepoint of the measurement. z dimension represents time
variable	String. "peak_pressure", "force", or "area"
side	For insole data only
threshold	Numeric. Threshold value for sensor to be considered active. Currently only applies to insole data
plot	Logical. If TRUE also plots data as line curve

Value

Numeric vector containing variable values

Examples

```
emed_data <- system.file("extdata", "emed_test.lst", package = "pressuRe")
pressure_data <- load_emed(emed_data)
whole_pressure_curve(pressure_data, variable = "peak_pressure", plot = FALSE)
whole_pressure_curve(pressure_data, variable = "area", plot = FALSE)
whole_pressure_curve(pressure_data, variable = "force", plot = FALSE)
```

Index