Package ‘erp.easy’

July 22, 2025

Type Package

Title Event-Related Potential (ERP) Data Exploration Made Easy

Version 1.1.0

URL `https://github.com/mooretm/erp.easy`

Description A set of user-friendly functions to aid in organizing, plotting
and analyzing event-related potential (ERP) data. Provides an easy-to-learn
method to explore ERP data. Should be useful to those without a background
in computer programming, and to those who are new to ERPs (or new to the
more advanced ERP software available). Emphasis has been placed on highly
automated processes using functions with as few arguments as possible.
Expects processed (cleaned) data.

Depends R (>= 3.2.0)

Imports plyr (>= 1.8.3), signal (>= 0.7-6), gtools (>= 3.5.0)

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Author Travis Moore [aut, cre]

Maintainer Travis Moore <`travis.m.moore@vanderbilt.edu`>

Repository CRAN

Date/Publication 2017-03-02 08:09:28

Contents

`butterfly` *Generate a butterfly plot for a specified condition*

Description

`butterfly` plots all individual waveforms for the condition specified by the `stim` argument (i.e., a butterfly plot). The grand average waveform is also plotted, using a red line.

Usage

```
butterfly(data, electrodes, stim = 1)
```

Arguments

`data`	A data frame in the format returned from `load.data`
`electrodes`	A single value or concatenation of several values (to be averaged) indicating which electrodes to include in generating the plot. At this time, if the raw data files imported using `load.data`) do not have a header, you must include a capital "V" in front of the number and enclose each electrode in quotes. (For example, electrodes = "V78", or electrodes = c("V78", "V76").)
`stim`	An integer specifying which condition to plot. Conditions are numbered in the order in which they are imported with `load.data`. The plot title will specify the name of the condition to avoid confusion.

Details

Single electrodes can be passed to the package functions, or several electrodes can be provided (i.e., when using dense arrays) and those electrodes will be averaged together as a single electrode.

Value

A single butterfly plot for the condition specified with `stim`.

Author(s)

Travis Moore

Examples

```
butterfly(ERPdata, electrodes = "V78", stim = 1)
```

`dif.wave` *Calculate a difference waveform from two data sets*

Description

`dif.wave` calculates a difference waveform from two data frames in the format returned from `load.data`

Usage

```
dif.wave(x, z, name = NULL, keep = NULL)
```

Arguments

`x`	A data frame in the format returned from `load.data`. This value serves as the minuend (i.e., value to be subtracted from).
`z`	A data frame in the format returned from `load.data`. This value serves as the subtrahend (i.e., value subtracted).
`name`	Specify the Stimulus column name of the difference data frame. Must provide the new name in quotations.
`keep`	Include one of the data frames used in subtraction in the returned difference data frame. By default `keep` = NULL and only the resulting difference data frame will be returned.

Details

The data frames must either:

1. have been imported using `load.data` - OR -
2. for subset data, the "Stimulus" column must be "refactored." See the example below for one way to "reset" the factors in the "Stimulus" column. Note this procedure is only necessary for subset data, and not data frames that were independently imported.

Value

One of two possible data frames, depending on the value of `keep`:

1. `keep` = "y": the difference (i.e., x - z) data frame and the minuend data frame (i.e., x)
2. `keep` = "n": just the difference data frame

Author(s)

Travis Moore

Examples

```
# Calculate a difference wave
Negative = ERPdata[1:6765, ]
Neutral = ERPdata[6766:13530, ]
refactor.neg <- factor(Negative$Stimulus)
refactor.neut <- factor(Neutral$Stimulus)
Negative$Stimulus <- refactor.neg
Neutral$Stimulus <- refactor.neut
difference <- dif.wave(Negative, Neutral, name = "Neg - Neut", keep = Neutral)
head(difference) # view the new Stimulus column name
grandaverage(difference, "V78") # plot the grand average difference wave
```

`easy.load` *An even easier way to import your data!*

Description

`easy.load` generates the code required to import your data by asking questions from the R console

Usage

```
easy.load()
```

Details

`easy.load` will begin the import wizard. A series of questions presented from the R console will ask for the necessary information to import your data in a user-friendly way. See `load.data` for parameter definitions and conventions for how to answer `easy.load` questions if you get stuck.

`easy.load` uses an interactive window to locate files to create the path name. The window usually appears BEHIND the R session and other open programs, so minimize open windows until the interactive pop-up window is visible. After you browse to your file location, bring the R session window back up to continue.

Follow any instructions accompanying the questions exactly, as some prompts require a very specific answer format.

Value

A complete line of code ready to be copy/pasted into the command line or R script

Author(s)

Travis Moore

Examples

```
## Not run:
# example output from easy.load (after answering questions)
myData = load.data('/Users/Username/Folder/', 'Positive', 20, -200, 899, 'n', FALSE)

## End(Not run)
```

`erp.easy` *erp.easy: A user-friendly package for exploring event-related potential (ERP) data*

Description

So you've recorded and cleaned (processed) some ERP data... Now what? If you're not a programmer, or are new to ERPs, the next step may be a bit daunting, or at the very least, time consuming if done by hand. The erp.easy package provides an intuitive approach to exploring your processed data, without requiring a background in computer programming. The erp.easy package provides three categories of functions, optimized to be easy to use: loading ERP files, plotting ERP data, and analyzing ERP data.

Loading functions

The function `load.data` exists to save you the hassle of opening each individual ERP file and adding a header or other identifying information to the files. This function expects data formatted with electrodes across the columns and time points as rows. Additional columns (i.e., "Subject", "Stimulus", and "Time") will be added upon import to help organize your data. The erp.easy package will use existing headers provided in raw data files to refer to electrodes for use in functions, or will assign headers if none are present (see `load.data`.) Single electrodes can be passed to the package functions, or several electrodes can be provided (i.e., when using dense arrays) and those electrodes will be averaged together as a single electrode. See also `easy.load` for a more user-friendly way to import your data using the erp.easy data import wizard.

Plotting functions

The plotting functions `grandaverage`, `individual`, and `mosaic` provide several ways to visualize both grand average and individual data. Color-coding and labeling happens automatically for ease of use.

Analysis functions

The analysis functions `m.measures` and `p.measures` calculate standard ERP measures such as mean amplitude, standard deviation, peak amplitude and peak latency for both grand average and individual waveforms.

Author(s)

Travis Moore

Examples

```
library(erp.easy)

data(ERPdata)

grandaverage(ERPdata, electrodes = "V78")

mosaic(ERPdata, electrodes = "V78")

m.measures(ERPdata, electrodes = "V78", window = c(1000, 1500))
```

ERPdata *ERP example data*

Description

A data frame containing example ERP data:

- 15 subjects (S1, S2, S3, ...)
- 2 conditions (Negative and Neutral)

Usage

```
ERPdata
```

Format

A data frame with 13,530 timepoints (rows) and 23 columns (Subject, Stimulus, Time, electrodes)

Note

These data have been heavily modified for use with this R package. Electrodes, subjects, and conditions have been truncated to reduce the size of the sample data set to comply with CRAN file size allowances.

Source

Original data collected by Hatun Zengin-Bolatkale. Data were altered for R package examples by Travis Moore.

Examples

```
## Not run:
data(ERPdata)

## End(Not run)
```

grandaverage *Plot the grand average waveform for all loaded conditions*

Description

`grandaverage` plots the grand average waveform for each condition present in the data frame you provide. A color-coded and labeled legend is generated with the plot for ease of identification of each condition. The legend can be suppressed if it interferes with the data presentation (i.e., hides part of the waveform).

Usage

```
grandaverage(data, electrodes, window = NULL, lgnd = NULL)
```

Arguments

`data`	A data frame in the format returned from `load.data`
`electrodes`	A single value or concatenation of several values (to be averaged) indicating which electrodes to include in generating the plot. At this time, if the raw data files imported using `load.data`) do not have a header, you must include a capital "V" in front of the number and enclose each electrode in quotes. (For example, electrodes = "V78", or electrodes = c("V78", "V76").)
`window`	The beginning and end points of a time window of interest; this is different from the beginning and ending times `epoch.st` and `epoch.end` defined in `load.data` (you only need to define the epoch once upon importing the data). The purpose of the `window` argument in this function is merely to highlight a window of interest; its default value is NULL.
`lgnd`	Whether or not a legend should appear on the plot. By default a legend will appear, but can be suppressed by setting `lgnd` equal to "n".

Details

`grandaverage` will return a plot of the grand average waveform for each condition present in the data frame you provide. For ease of use, colors are automatically assigned. The legend displays the value provided in the `condition` argument of `load.data`.

Single electrodes can be passed to the package functions, or several electrodes can be provided (i.e., when using dense arrays) and those electrodes will be averaged together as a single electrode.

Value

A single plot of grand average waveforms for each condition. Includes a color-coded and labeled legend (that can be suppressed if specified). Also returns the raw waveform data for each condition.

Author(s)

Travis Moore

Examples

```
# Create a plot of the grand average waveforms for each imported condition
grandaverage(ERPdata, electrodes = "V78", window = c(1000, 1500))
```

`individual` *Return individual average raw data and plotted waveforms (optional) for all loaded conditions*

Description

`individual` plots individual, averaged waveforms for each condition present in the data frame you provide. Separate plots for each individual can be generated, if specified.

Usage

```
individual(data, electrodes, plots = "n")
```

Arguments

`data`	A data frame in the format returned from `load.data`
`electrodes`	A single value or concatenation of several values (to be averaged) indicating which electrodes to include in generating the plot. At this time, if the raw data files imported using `load.data`) do not have a header, you must include a capital "V" in front of the number and enclose each electrode in quotes. (For example, electrodes = "V78", or electrodes = c("V78", "V76").)
`plots`	Creates plots of individual averaged data in separate windows. By default, plots are suppressed, but can be activated by setting `plots` to "y".

Details

`individual` will generate individual average data and separate plots of averaged waveforms (optional) for each subject in the data frame you provide. Raw data are organized in columns by subject and condition. Plots will be generated by setting `plots = "y"`.

Single electrodes can be passed to the package functions, or several electrodes can be provided (i.e., when using dense arrays) and those electrodes will be averaged together as a single electrode.

Value

Data frame of individual average data for each subject in each condition. If `plot = "y"`, then multiple plots (1 per subject) will also be generated.

Author(s)

Travis Moore

Examples

```
# Return data frame of individual average data and create average waveform
# plots for each subject
individual(ERPdata, electrodes = "V78", plots="y")
```

`load.data` *Import your ERP data files.*

Description

`load.data` imports your individual ERP data files. File extensions must be .txt and file names must be in the format: YourFile_Condition.txt (e.g., SS34_Positive.txt). Raw data files to be imported should be organized as follows:

- each electrode must be a separate column
- voltages at each time point should be listed under the appropriate electrode column as rows
- no other data should be present in the raw data file (e.g., subject, condition, time, etc.)

Usage

```
load.data(path, condition, num.subs, epoch.st, epoch.end, bsln.cor = "n",
  header = FALSE)
```

Arguments

`path`	The folder path containing your ERP files
`condition`	In quotes, a string indicating which trial type to be imported (i.e., the condition indicated in the file name)
`num.subs`	The number of files (subjects) to import for a given condition
`epoch.st`	The earliest time point sampled in the ERP files, including the basline (e.g., -200)
`epoch.end`	The final time point sampled in the ERP files (typically $finaltimepoint - 1$)
`bsln.cor`	If "y", applies baseline correction to the imported data. Baseline correction is achieved by subtracting the mean voltage prior to 0 ms on a channel- by-channel basis.
`header`	Only accepts values of TRUE or FALSE. Used to specify whether or not there is an existing header row in the ERP files. If there is no header, `load.data` will supply one (see details below).

Details

See also `easy.load` for a more user-friendly way to generate the appropriate code.

- Name each individual file following the format mentioned above (e.g., SS34_Positive.txt). `load.data` will ignore all text preceding the "_", and treat all text following the "_" as the `condition`, (e.g., Positive). Use only one "_" in the file name (i.e., to separate your own naming convention from the `condition`); using multiple "_" characters will lead to faulty importing of data. The erp.easy convention for subjects is a capital "S" followed by the number corresponding to the order in which the file was loaded (e.g., S1, S2, etc.). Subjects will be loaded into the "Subject" column of the returned data frame.
- If no header is present in the ERP files, one will be supplied, using the standard R convention of a capital "V" followed by increasing integers (e.g., V1, V2, V3). Use these automatically assigned column name values to refer to the electrodes (unless a header is provided in the raw data file).
- Enter the starting time of the baseline, if present in your individual files, in `epoch.st` (e.g., -200).
- Once the desired data frames have been loaded, they can be exported as a number of different file types.
- The sample rate will be calculated for you, based on the starting (`epoch.st`) and ending (`epoch.end`) time points of the recording epoch and the number of time points in a given condition (the number of rows in your file for each condition).

Value

A single, concatenated data frame of all electrode data for all subjects organized into columns, with three added columns:

1. "Subject" containing repeating subject names
2. "Stimulus" containing repeating condition names (e.g., Neutral)
3. "Time" containing a repeating list of timepoints sampled

Note

While importing data must be done using a separate function call for each condition, it can be convenient to use R's native `rbind.data.frame()` command to bind several loaded conditions (variables) into a single data frame consisting of multiple conditions. All erp.easy functions will act on all conditions included in the data frame passed to the function. For example, if you'd like to see all conditions plotted, simply use `rbind.data.frame()` to make a single data frame to pass to an erp.easy plotting function, and you will see all added conditions plotted simultaneously in the same figure (as opposed to making separate data frames for each condition, then passing each data frame separately to a function).

Author(s)

Travis Moore

Examples

```
## Not run:
# Importing data for a condition named "Neutral" (file names: "Sub1_Neutral.txt",
"Sub2_Neutral.txt", etc.)
neutral <- load.data(path = "/Users/Username/Folder/", condition = "Neutral",
num.subs = 20, epoch.st = -200, epoch.end = 899, header = FALSE)

# Adding imported data named "positive" to the imported "neutral" data
combo <- rbind.data.frame(neutral, positive)

## End(Not run)
```

`m.measures` *Calculate grand average and individual mean amplitudes and standard deviations*

Description

`m.measures` calculates mean amplitude and standard deviation for each condition in the data frame, for the specified time window. Values are calculated based on grand average waveforms, as well as for each individual subject. Values are based on the electrode, or electrode cluster for dense arrays, provided in `electrodes`.

Usage

```
m.measures(data, electrodes, window)
```

Arguments

`data`	A data frame in the format returned from `load.data`
`electrodes`	A single value or concatenation of several values (to be averaged) indicating which electrodes to include in generating the plot. At this time, if the raw data files imported using `load.data`) do not have a header, you must include a capital "V" in front of the number and enclose each electrode in quotes. (For example, electrodes = "V78", or electrodes = c("V78", "V76").)
`window`	The beginning and end points of a time window of interest; this is different from the beginning and ending times `epoch.st` and `epoch.end` defined in `load.data` (you only need to define the epoch once upon importing the data).

Details

Single electrodes can be passed to the package functions, or several electrodes can be provided (i.e., when using dense arrays) and those electrodes will be averaged together as a single electrode.

Value

A data frame with columns labeled:

- Subject
- Trial Type
- Standard Deviation
- Mean Amplitude

A plot indicating the specified time window

Author(s)

Travis Moore

Examples

```
# Calculate mean amplitude and standard deviation
m.measures(ERPdata, electrodes = "V78", window = c(1000, 1500))
```

`mosaic` *Create average waveform plots for each subject in a single, multiplot window*

Description

`mosaic` generates multiple plots in a single window. Plots are average waveforms for each subject. Each plot shows all conditions present in the data frame.

Usage

```
mosaic(data, electrodes, cols = 5, rows = 5)
```

Arguments

`data`	A data frame in the format returned from `load.data`
`electrodes`	A single value or concatenation of several values (to be averaged) indicating which electrodes to include in generating the plot. At this time, if the raw data files imported using `load.data`) do not have a header, you must include a capital "V" in front of the number and enclose each electrode in quotes. (For example, electrodes = "V78", or electrodes = c("V78", "V76").)
`cols`	An integer defining the number of desired columns of plots. The default is 5.
`rows`	An integer defining the number of desired rows of plots. The default is 5.

Details

The default values for columns and rows (i.e., 5 and 5) and higher are best suited to the graphical parameters specified in the code. At this time, graphical parameters (e.g., tick marks and labels) do not scale with the number of rows and colums. As this feature is not intended to produce manuscript-ready plots, the option to explore your data with any number of rows and columns is available, but fewer rows and columns will yield crowded plots.

Single electrodes can be passed to the package functions, or several electrodes can be provided (i.e., when using dense arrays) and those electrodes will be averaged together as a single electrode.

Value

A single window containing multiple plots (1 per subject)

Author(s)

Travis Moore

Examples

```
# Inspect average ERP waveforms for each subject
mosaic(ERPdata, electrodes = "V78", cols = 5, rows = 5)
```

`p.measures` *Calculate grand average and individual peak amplitude and latency*

Description

`p.measures` calculates local or simple peak amplitude and latency for each condition in the data frame. Values are calculated for grand average waveforms, as well as for each individual subject. Values are based on the electrode, or electrode cluster for dense arrays, provided in `electrodes`.

Usage

```
p.measures(data, electrodes, window, num.pts = 10, pol = "abs")
```

Arguments

`data`	A data frame in the format returned from `load.data`
`electrodes`	A single value or concatenation of several values (to be averaged) indicating which electrodes to include in generating the plot. At this time, if the raw data files imported using `load.data`) do not have a header, you must include a capital "V" in front of the number and enclose each electrode in quotes. (For example, electrodes = "V78", or electrodes = c("V78", "V76").)
`window`	The beginning and end points of a time window of interest; this is different from the beginning and ending times `epoch.st` and `epoch.end` defined in `load.data` (you only need to define the epoch once upon importing the data).

`num.pts`	The number of bins to check for local peak measures. If no local peaks are found, the simple peak will be returned. To force the simple peak, set `num.pts` to 0.
`pol`	The polarity of peaks to favor when multiple peaks are present. Entering "pos" will locate the most positive peak. Entering "neg" will locate the most negative peak. Entering "abs" will find the greatest deviation from 0, regardless of the polarity (i.e., the absolute value). If a single peak is located, it will be selected regardless of polarity specified. To avoid this, set `num.pts` to 0 (for simple peak measures).

Value

A data frame with columns labeled:

- Subject
- Trial Type
- Peak Latency
- Peak Amplitude

A plot indicating the time window and identified peak(s)

Author(s)

Travis Moore

Examples

```
# Calculate peak latency and amplitude
p.measures(ERPdata, electrodes = "V78", window = c(1000, 1500), num.pts=10, pol="abs")
```

Index