Package 'erpR'

July 2, 2014

Type Package

Title Event-related potentials (ERP) analysis, graphics and utility functions

Version 0.2.0

Date 2014-05-14

Author Giorgio Arcara, Anna Petrova

Maintainer Giorgio Arcara<giorgio.arcara@gmail.com>

Depends R (>= 3.0.2), rpanel

Description This package is dedicated to the analysis of event-related potentials (ERPs). Event-related potentials are the measured brain responses associated with a specific sensory, cognitive, or motor event and are obtained from electroencephalographic (EEG) signal. The erpR package contains a series of functions for importing ERP data, computing traditional ERP measures, exploratory ERP analyses and plotting.

Suggests akima

License GPL-3

Repository CRAN

Repository/R-Forge/Project erpr

Repository/R-Forge/Revision 67

Repository/R-Forge/DateTimeStamp 2014-05-14 07:56:17

Date/Publication 2014-05-14 11:59:08

NeedsCompilation no

R topics documented:

`erpR-package` *erpR package*

Description

The erpR package includes a series of functions for importing Event-related potential (ERP) data, carrying out exploratory analyses, computing traditional ERP measures and ERP plotting.

Details

Package:	erpR
Type:	Package
Version:	0.2.0
Date:	2014-05-14
License:	GPL-3

The erpR package is dedicated to the analysis of event-related potentials (ERPs). Event-related potentials are the measured brain responses associated with a specific sensory, cognitive, or motor event and are obtained from electroencephalographic (EEG) signal. The erpR package contains a series of functions for importing Event-related potentials (ERP) data, carrying out exploratory analyses, computing traditional ERP measures and ERP plotting. Many erpR functions require to specify two arguments: base and numbers. The names of the objects for the erpR functions are obtained from the combination of base and numbers. The argument base typically indicates the name of the experimental condition (or group, or both), while numbers indicate the numbers of the subjects on which function is applied. This system is crucial to keep order in objects and files names and to tell erpR the correspondence between the objects. In particular a given base will always refer to the same condition, and a given number will always refer to the same subject. The data frames containing ERP data (named by the combination of base and number) are stored in lists called erplists. The topoplot function of this package uses the function interp of package akima, that is released under a non-commercial use license.

Further information can be found on the maintainer's site https://sites.google.com/site/giorgioarcara/erpr.

Author(s)

Giorgio Acara <giorgio.arcara@gmail.com>

Anna Petrova <petrova@front.ru>

Mantainer: Giorgio Arcara <giorgio.arcara@gmail.com>

Examples

```
library(erpR)

data(ERPsets)

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)


erp(word$Fp1, smo=0, col="blue", startmsec=-200, endmsec=1500, ylim=c(-10,10))
```

`butterfly` *ERP butterfly plot*

Description

A butterfly plot of several ERP data frames.

Usage

```
butterfly(base, numbers,  electrode = NULL, startmsec = - 200,
endmsec = 1200, erplist = NULL, outline=NULL, out.col="black",
add = FALSE, ...)
```

Arguments

`base`	a string indicating the beginning of the name of the data frame containing ERP data.
`numbers`	the numbers of the subjects to be averaged by the function.
`electrode`	the electrode to be plotted.
`startmsec`	the start time (in ms) of the ERP data frames. It can be a negative value, indicating the baseline time frame.
`endmsec`	the end time (in ms) of the ERP data frames.
`erplist`	a list containing the ERP data frames specified in `base` and `numbers`.
`outline`	the number of the subject to be outlined (it will be plotted with lwd=2).
`out.col`	the color of the outlined ERP waveform.
`add`	add the current plot to a previous plot
`...`	further parameters to be passed to `link{erp}`.

Value

A butterfly plot.

Author(s)

Giorgio Arcara

See Also

`erp`, `erp.infl`, `scalp.infl`

Examples

```
data(ERPsets)
butterfly(base="Exp1_nonword_subj", numbers=1:20, electrode="Fp1", smo=0,
startmsec=-200, endmsec=1500, erplist=ERPsets, outline=1, ylim=c(-20,20), out.col="red")
```

---

char2fac *convert characters to factors*

Description

convert each `character` column in a data frame to a `factor`.

Usage

```
char2fac(x)
```

Arguments

`x`	a `data.frame` in which columns are to be converted.

Value

A `data.frame` with `character` columns converted in `factor` columns.

Note

This is an utility to quickly transform a `character` vector to `factor` for analysis purpose.

Author(s)

Giorgio Arcara.

Examples

```
dat=data.frame(RT=rnorm(20, mean=500, sd=100), Subject=rep(1:10, 2))

conditions=c(rep("a", 10), rep("b", 10))

dat$conditions=conditions

dat=char2fac(dat)
```

`check.erplist` *check consistency of an erplist*

Description

The function performs a series of check on an erplist (a list containing ERP data frames according to the rationale of the `erpR` package).

Usage

```
check.erplist(erplist=NULL)
```

Arguments

`erplist`	an erplist object containing ERP data frames.

Details

The function checks the following errors: 1) check for duplicate data frames; 2) check for data frames with names without `number` specified; 3) check for unbalanced combinations of `base` and `number`. For an explanation of the meaning of `base` and `numbers` see `erpR`.

Value

The function return the errors found on the erplist.

Author(s)

Giorgio Arcara

Examples

```
data(ERPsets)
check.erplist(ERPsets)
```

cn *display column names*

Description

Return the column names of a `data.frame` or a `matrix` in one-column format for an easy visual inspection.

Usage

```
cn(x)
```

Arguments

`x`	a data.frame or a matrix with column names.

Value

A one-column matrix with the column names of `x`.

Note

This function comes from an idea by Antoine Tremblay.

Author(s)

Giorgio Arcara

Examples

```
data(iris)

cn(iris)
```

`create.diff` *create an ERP data frame with differential values*

Description

This function creates a new set of data frames from two sets of paired ERP data frames. The values are obtained by subtracting the values of the second set from the first set. All data frames must be in the same erplist.

Usage

```
create.diff(base1, base2, numbers, outname = NULL, erplist,
fileinfo = 1)
```

Arguments

`base1`	a vector of strings indicating the beginning of the name of the first series of data frames containing ERP data.
`base2`	a vector of strings indicating the beginning of the name of the second series of data frames containing ERP data.
`numbers`	the numbers of the subjects to be subtracted by the function.
`outname`	the `base` of the names of the returned data frames.
`erplist`	a list containing the ERP data frames specified in `base` and `numbers`.
`fileinfo`	a number indicating from which set of objects specified in the argument `bases` the information of the data frame is retrieved.

Details

For more information on the `fileinfo` argument, see Details section of `import.erp`.

Value

A series of ERP data frames obtained by subtracting one specified set from the other.

Author(s)

Giorgio Arcara

See Also

`create.mean`

Examples

```
data(ERPsets)

ERPsetsdiff=create.diff("Exp1_word_subj", "Exp1_nonword_subj", 1:20,
outname="Exp1_diff_subj", erplist=ERPsets)

ERPsets=c(ERPsets, ERPsetsdiff)
```

`create.mean` *create ERP data frames with mean values from a series of data frames contained in an erplist.*

Description

This function creates a new set of data frames from two sets of paired ERP data frames by averaging the values of the original sets.

Usage

```
create.mean(bases, numbers, outname = NULL, erplist = NULL, fileinfo = 1)
```

Arguments

`bases`	a vector of strings indicating the beginning of the names of the data frames containing the ERP data to be averaged.
`numbers`	the numbers of the subjects to be averaged.
`outname`	the `base` of the names of the returned data frames.
`erplist`	a list containing the ERP data frames specified in `base` and `numbers`.
`fileinfo`	a number indicating from which set of objects specified in the argument `bases` the information of the data frame is retrieved.

Details

For more information on `fileinfo` argument see Details section of `import.erp`.

Value

A series of ERP data frames obtained by averaging the specified sets.

Author(s)

Giorgio Arcara

See Also

create.diff

Examples

```
data(ERPsets)

ERPsetsmean=create.mean(c("Exp1_word_subj", "Exp1_nonword_subj"), 1:20,
outname="Exp1_mean_subj", erplist=ERPsets)

ERPsets=c(ERPsets, ERPsetsmean)
```

---

create.roi *create electrode region of interests*

Description

An utility function to create a factor that collapses other factor levels (typically electrode names) in a new variable vector with the new ROI (region of interest) variable. It can be used only with data.frame in long format.

Usage

```
create.roi(datall, electrode="electrode", groups=NULL, roi.levels=NULL )
```

Arguments

datall	a data frame containing ERP data in long format.
electrode	name of the column in datall containing electrode names.
groups	a list containing (in separate vectors) electrode names to create the ROI. E.g. `list(c("Fp1", "Fp2"), c("P3", "P4"))`.
roi.levels	a vector with the names of the newly created factor variable. E.g. `c("Frontopolar", "Parietal")`.

Details

All levels of the variable electrode that are not specified in groups will be coded as NA in the returned vector.

Value

The function returns a vector with the new coded ROI variable.

Author(s)

Giorgio Arcara

Examples

```
data(ERPsets)

datall=erp.mean(base = "Exp1_word_subj", numbers = 1:20,
 win.ini = 400, win.end = 600, startmsec= -200, endmsec=1500, erplist=ERPsets)

datall$caudality=create.roi(datall, "electrode",
groups=list(c("Fp1", "Fp2"), c("P3", "P4")),
roi.levels=c("Frontopolar", "Parietal"))

table(datall$caudality, datall$electrode)
```

---

erp *plot an ERP waveform*

Description

A function to plot an ERP waveform of a single electrode. If used with default values the function allows a fast plotting. Several parameters are available for plotting paper quality plots.

Usage

```
erp(el, startmsec=-200, endmsec = 1200, smo = NULL, interval = c(startmsec, endmsec),
xlim = c(startmsec, endmsec), ylim = c(-6,6), col = "black",
lwd = 1, xlab = "", ylab = "", main = "", frame.plot = TRUE,
draw.xaxis = TRUE, draw.yaxis = TRUE, type = "l", x0 = TRUE,
x.tick = seq(-200, 1200, 200), x.labels=x.tick, x.pos = NA,
x.outer = F, x.font = NA, x.lty = "solid", x.lwd = 1, x.lwd.ticks = 1,
x.col = NULL,  x.col.ticks = NULL, x.hadj = NA, x.padj = NA, x.tcl = -0.5,
x.tick.both= FALSE, x.cex = 1, y.tick = seq(-6,6,2), y.labels = y.tick,
y.pos = NA, y.outer = F, y.font = NA, y.lty = "solid", y.lwd = 1,
y.lwd.ticks = 1, y.col = NULL, y.col.ticks = NULL, y.hadj = NA, y.padj = NA,
y.tcl = -0.5, y.tick.both= FALSE, y.cex = 1, ...)
```

Arguments

el	a vector with the electrode to be plotted.
startmsec	the start time (in ms) of the ERP vector.
endmsec	the end time (in ms) of the ERP vector.
smo	the smoothing parameter to apply (see smooth.spline). Default is NULL and no smoothing is applied.
interval	a vector with a pair of values indicating the interval (in milliseconds) to be plotted.
xlim	a vector with a pair of values indicating the limits (in milliseconds) of the x axis.

ylim	a vector with a pair of values indicating the limits (in milliseconds) of the x axis.
col	the color of the ERP waveform.
lwd	the line width of the ERP waveform.
xlab	the label of the x axis.
ylab	the label of the y axis.
main	the title of the plot.
frame.plot	a logical indicating whether a box should be drawn around the plot.
draw.xaxis	a logical indicating whether the x axis should be drawn.
draw.yaxis	a logical indicating whether a box should be drawn around the plot.
type	1-character string giving the type of plot desired. Default is "l" and a line will be plotted. If "n" the ERP waveform is not plotted (see plot.default).
x0	a logical indicating whether a short segment should be drawn to indicate the 0 point (in millisecond)
x.tick	the position (in milliseconds) of x ticks.
x.labels	the labels (usually a vector of numbers) to be plotted on the x axis.
x.pos	the position of x axis, in terms of y axis coordinates. Default is NA and the axis is drawn at the bottom of the plot.
x.outer	a logical value indicating whether the x axis should be drawn in the outer plot margin, rather than the standard plot margin.
x.font	font for text of x labels. Defaults to par("font").
x.lty	line type for both the x axis line and the x tick marks.
x.lwd	line width for the x axis line.
x.lwd.ticks	line width for the x tick marks.
x.col	colors for the x axis line and the x tick marks.
x.col.ticks	color for the x tick marks (if specified, override x.col)
x.hadj	adjustment (see par("adj")) for all x labels parallel ("horizontal") to the reading direction. If this is not a finite value, the default is used (centring for strings parallel to the axis, justification of the end nearest the axis otherwise).
x.padj	adjustment for each x tick label perpendicular to the reading direction. For labels parallel to the axes, padj = 0 means right or top alignment, and padj = 1 means left or bottom alignment. This can be a vector given a value for each string, and will be recycled as necessary.
x.tcl	The length of tick marks as a fraction of the height of a line of text. The default value is -0.5.
x.tick.both	Logical. Should the tick marks be drawn both above and below the x axis?
x.cex	The magnification to be used for the x labels.
y.tick	the position (in milliseconds) of y ticks.
y.labels	the labels (usually a vector of numbers) to be plotted on the y axis.
y.pos	the position of y axis, in terms of x axis coordinates. Default is NA and the axis is drawn at the left of the plot.

y.outer	a logical value indicating whether the y axis should be drawn in the outer plot margin, rather than the standard plot margin.
y.font	font for teyt of y labels. Defaults to par("font").
y.lty	line type for both the y axis line and the y tick marks.
y.lwd	line width for the y axis line.
y.lwd.ticks	line width for the y tick marks.
y.col	colors for the y axis line and the y tick marks.
y.col.ticks	color for the y tick marks (if specified, override y.col)
y.hadj	adjustment (see par("adj")) for all y labels parallel ("horizontal") to the reading direction. If this is not a finite value, the default is used (centring for strings parallel to the axis, justification of the end nearest the axis otherwise).
y.padj	adjustment for each y tick label perpendicular to the reading direction. For labels parallel to the ayes, padj = 0 means right or top alignment, and padj = 1 means left or bottom alignment. This can be a vector given a value for each string, and will be recycled as necessary.
y.tcl	The length of tick marks as a fraction of the height of a line of teyt. The default value is -0.5.
y.tick.both	Logical. Should the tick marks be drawn both above and below the y axis?
y.cex	The magnification to be used for the y labels.
...	further parameters to be passed to plot.

Value

the plot of an ERP waveform.

Author(s)

Giorgio Arcara

See Also

erp.t, erp.infl, erp.cor, erp.xaxis, erp.yaxis

Examples

```
data(ERPsets)

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)

erp(word$F3, smo=0, col="blue", startmsec=-200, endmsec=1500, ylim=c(-6,6))
```

`erp.add` *add an ERP waveform to a plot*

Description

Add a waveform to a ERP plot created with `erp`, `erp.t`, or `butterfly` functions.

Usage

```
erp.add(el, startmsec=-200, endmsec=1200, interval=c(startmsec, endmsec),
smo = NULL, col = "black", lty = 1, lwd = 1, ...)
```

Arguments

`el`	the electrode to be plotted.
`startmsec`	the start time (in ms) of the ERP vector.
`endmsec`	the end time (in ms) of the ERP vector.
`smo`	the smoothing parameter to apply (see `smooth.spline`). Default is `NULL` and no smoothing is applied.
`interval`	a vector with a pair of values indicating the interval (in milliseconds) to be plotted.
`col`	the color of the waveform.
`lty`	line type of the waveform .
`lwd`	line width of the waveform.
`...`	further parameters passed to `lines`

Value

The specified waveform is added to an existing plot.

Author(s)

Giorgio Arcara

See Also

`erp`, `butterfly`, `erp.t`, `erp.infl`

Examples

```
data(ERPsets)

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)
nonword=grandaverage("Exp1_nonword_subj", 1:20, erplist=ERPsets)

erp(word$Fp1, col="blue", startmsec=-200, endmsec=1500)
erp.add(nonword$Fp1, col="red", startmsec=-200, endmsec=1500)
```

`erp.cor` *plot the timepoint correlations between ERPs (single electrode) and an external variable*

Description

This function performs a series of correlations on each timepoint between ERP voltages and an external (typically a behavioral) variable. The set of ERP data frames is specified by `base` and `numbers`. The behavioral variable is specified as a vector (of length equal to `numbers`) as the argument `external`. The function returns a plot of a single electrode showing the results of the correlation tests.

Usage

```
erp.cor(base, numbers, electrode, erplist=NULL,startmsec=-200,
endmsec=1200, external=NULL, smo=NULL, alpha=0.05,
method = c("pearson", "kendall", "spearman"),  sig=NULL,
main=electrode, ...)
```

Arguments

`base`	a string indicating the beginning of the name of the set of ERP data frames.
`numbers`	the numbers of the subjects of the set of ERP data frames.
`electrode`	the electrode to be plotted.
`erplist`	a list containing the ERP data frames specified in `base` and `numbers`.
`startmsec`	the start time (in ms) of the ERP vector.
`endmsec`	the end time (in ms) of the ERP vector.
`external`	a vector indicating the values of the external variable (typically a behavioural variable). This vector must be of the same length of `numbers`
`smo`	the smoothing parameter to apply (see `smooth.spline`). Default is `NULL` and no smoothing is applied.
`alpha`	the alpha of the correlation tests.
`method`	the method of the correlations. Available methods are `"pearson"`, `"kendall"`, `"spearman"`.

sig	if NULL, then a series of correlations is computed. If an appropriate object is specified (i.e. an object returned by scalp.cor or erp.cor) then the results are retrieved by this object and the function simply plots the results.
main	plot title.
...	further parameters to be passed to link{erp}.

Value

This function plots a scalp map of the ERP waveforms in an array resembling the topographic locations of electrodes on the scalp. The waveforms represent the average of the first set of data frames specified. Timepoints with a positive (and significant) correlation are highlighted by red bars. Timepoints with a negative (and significant) correlation are highlighted by blue bars. The function also returns an object with the correlation results. This object can be used in future plots with erp.cor or scalp.cor, by specifying it as the argument sig.

Author(s)

Giorgio Arcara

See Also

scalp.cor

Examples

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

#generate some simulated RT data

RT=rnorm(20, mean = 500, sd = 100)
erp.cor("Exp1_word_subj", numbers=1:20, electrode="Fp1",
erplist=ERPsets, external=RT, startmsec=-200, endmsec=1500,
ylim=c(-6,6))

## End(Not run)
```

---

erp.infl *inspect the influence of a subject on the average (single electrode).*

Description

This function returns a diagnostic plot to investigate the influence of a given ERP data frame on the grandaverage of a series of ERP data frames. A GUI is provided to explore the effect of removing a data frame (that is expected to be associated with a Subject) on the grandaverage. Two plots are superposed: the original grandaverage and the average removing the subject selected via the GUI.

Usage

```
erp.infl(base, numbers, electrode, erplist=NULL,  startmsec=-200,
endmsec=1200, smo=NULL, outnumber=1, lwd=1, lty=1,  ...)
```

Arguments

base	a string indicating the beginning of the name of the data.frame containing the ERP data.
numbers	the numbers of the subjects to be averaged by the function.
electrode	the electrode to be plotted.
erplist	a list containing the ERP data frames specified in base and numbers.
startmsec	the start time (in ms) of the ERP vector.
endmsec	the end time (in ms) of the ERP vector.
smo	the smoothing parameter to apply (see smooth.spline). Default is NULL and no smoothing is applied.
outnumber	The number of the subject that will be initially removed from the averaged (this value could be changed also via the GUI).
lwd	line width.
lty	line type.
...	further parameters to be passed to erp function called within erp.infl.

Details

The function requires the package rpanel

Value

The function returns a plot of a single electrode showing the influence of a subject on the grandaverage.

Author(s)

Giorgio Arcara

See Also

erp, link{scalp.infl}

Examples

```
data(ERPsets)

erp.infl(base="Exp1_word_subj", numbers=1:20, electrode="OZ",
startmsec=-200, endmsec=1500, erplist=ERPsets)
```

```
# Notice that Subject 1 is clearly particularly influential
# for the average on OZ.
```

`erp.latency` *calculate latency of several ERP data frames on a given time window*

Description

This function calculates the peak latency on several ERP data frames. It returns the results in wide format (each electrode in a separate column) or in long format which is more suitable for standard statistical analysis in R (e.g. ANOVA). `erp.latency` applies the function `mean` to several ERP data frames with timepoints in rows and electrodes in columns. ERP data frames are expected to be named as a combination of strings specified in `base` and `numbers`.

Usage

```
erp.latency(base, numbers, win.ini, win.end, erplist = NULL,
startmsec=-200, endmsec=1200, others=NULL, format="long",
name.dep="Dep", name.newvar="electrode", peak.fun = "max", ...)
```

Arguments

`base`	a string indicating the beginning of the name of the `data.frame` containing the ERP data.
`numbers`	the numbers of the subjects to be averaged by the function.
`win.ini`	the initial edge (in ms) of the window in which the peak latency has to be computed.
`win.end`	the final edge (in ms) of the window in which the peak latency has to be computed.
`erplist`	a list containing the ERP data frames specified in `base` and `numbers`.
`startmsec`	the start time (in ms) of the ERP data frames. It can be a negative value, indicating the baseline time frame.
`endmsec`	the end time (in ms) of the ERP data frames.
`others`	the names of other variables to be included in the returned data frame. Ignored if the format is "wide".
`format`	the output format. It may be "wide" or "long".
`name.dep`	the name of the dependent variable in the returned dataframe. Ignored if the format is "wide".
`peak.fun`	the function to calculate the peak. Available functions are `max`, `min`, `localmax`, `localmin`.
`name.newvar`	If the format is "long", the name of the new variable codifying the electrodes. Default is "electrode".
`...`	arguments passed to the function specified in `peak.fun`.

Details

The function takes an ERP data frame with electrodes in separate columns as input. If the format is long, this function returns a data frame in long format ready for standard statistical analysis in R. The returned data frame has one column for the dependent variable and a new factor "electrode". The erpR functions `localmax` and `localmin` are written following Luck (2005).

Value

A ERP data frame with peak latency in long or wide format. Two more columns are added: the column `Subject` indicates the subject `number`; the column `Subject_name` reports the string retrieved from the comment of the data frame (see Details on `import.erp`).

Note

This function calls `rearrange`.

Author(s)

Giorgio Arcara

References

Luck, S., J. (2005). *An Introduction to the Event-Related Potential Technique.* Cambridge, Mass.: The MIT Press.

See Also

`erp.mean`, `erp.peak`, `rearrange`, `max`, `min`, `localmax`, `localmin`

Examples

```
data(ERPsets)

dat=erp.latency(base="Exp1_word_subj", numbers=1:20,
 win.ini=130, win.end=190, erplist=ERPsets, startmsec=-200, endmsec=1500,
others=c(condition="word", interval="130-190"),
name.dep="Ampl", format="long", peak.fun=max)
```

---

erp.mean *Calculate the mean of several ERP data frames in a given time window*

Description

This function calculates the mean amplitude of several ERP data frames. It returns the results in wide format (each electrode in a separate column) or in long format which is more suitable for standard statistical analysis in R (e.g. ANOVA). `erp.mean` applies the function `mean` on several ERP data frames with timepoints in rows and electrodes in columns. ERP data frames are expected to be named as a combination of strings specified in `base` and `numbers`.

Usage

```
erp.mean(base, numbers, win.ini, win.end, erplist = NULL,
startmsec=-200, endmsec=1200, others=NULL, format="long",
name.dep="Dep", name.newvar="electrode")
```

Arguments

base	a string indicating the beginning of the name of data.frame containing ERP data.
numbers	the numbers of the subjects to be averaged by the function.
win.ini	the initial edge (in ms) of the window on which the mean amplitude has to be computed.
win.end	the final edge (in ms) of the window on which the mean amplitude has to be computed.
erplist	a list containing the ERP data frames specified in base and numbers.
startmsec	the start time (in ms) of the ERP data frames. It can be a negative value, indicating the baseline time frame.
endmsec	the end time (in ms) of the ERP data frames.
others	the names of other variables to be included in the returned data frame. Ignored if the format is "wide".
format	the output format. It may be "wide" or "long".
name.dep	the name of the dependent variable in the returned data.frame. Ignored if format is "wide".
name.newvar	If the format is "long", the name of the new variable codifying the electrodes. Default is "electrode".

Details

The function takes an ERP data frame with electrodes in separate columns as input. If the format is long, this function returns a data frame in long format ready for standard statistical analysis in R. The returned data frame has one column for the dependent variable and a new factor "electrode".

Value

A ERP data frame with mean amplitude in long or wide format. Two more columns are added: the column Subject indicates the subject number; the column Subject_name reports the string retrieved from the comment of the data frame (see Details on import.erp).

Note

This function calls rearrange.

Author(s)

Giorgio Arcara

See Also

erp.peak, erp.latency, rearrange, mean

Examples

```
data(ERPsets)

dat=erp.mean(base="Exp1_word_subj", numbers=1:20,
win.ini=130, win.end=190, erplist=ERPsets, startmsec=-200, endmsec=1500,
others=c(condition="word", interval="130-190"), name.dep="Ampl", format="long")
```

---

erp.peak *calculate the peak of several ERP data frames in a given time window*

Description

This function calculates the peak amplitude on several ERP data frame. It returns results in wide format (each electrode in a separate column) or in long format which is more suitable for standard statistical analysis in R (e.g. ANOVA). erp.peak applies the function mean to several ERP data frames with timepoints in rows and electrodes in columns. ERP data frames are expected to be named as a combination of strings specified in base and numbers.

Usage

```
erp.peak(base, numbers, win.ini, win.end, erplist = NULL,
startmsec=-200, endmsec=1200, others=NULL, format="long",
name.dep="Dep", name.newvar="electrode", peak.fun = "max", ...)
```

Arguments

base	a string indicating the beginning of the name of data.frame containing the ERP data.
numbers	the numbers of the subjects to be averaged by the function.
win.ini	the initial edge (in ms) of the window on which the peak amplitude has to be computed.
win.end	the final edge (in ms) of the window on which the peak amplitude has to be computed.
erplist	a list containing the ERP data frames specified in base and numbers.
startmsec	the start time (in ms) of the ERP data frames. It can be a negative value, indicating the baseline time frame.
endmsec	the end time (in ms) of the ERP data frames.
others	the names of other variables to be included in the returned data frame. Ignored if the format is "wide".

format	the output format. It may be "wide" or "long".
name.dep	the name of the dependent variable in the returned data frame. Ignored if the format is "wide".
peak.fun	the function to calculate the peak. Available functions are max, min, localmax, localmin.
name.newvar	If the format is "long", the name of the new variable codifying the electrodes. Default is "electrode".
...	arguments passed to the function specified in peak.fun.

Details

The function takes an ERP data frame with electrodes in separate columns as input. If the format is long, this function returns a data frame in long format ready for standard statistical analysis in R. The returned data frame has one column for dependent variable and a new factor "electrode". The erpR functions localmax and localmin are are written following Luck (2005).

Value

A ERP data frame with peak amplitude in long or wide format. Two more columns are added: the column Subject indicates the subject number; the column Subject_name reports the string retrieved from the comment of the data frame (see Details on import.erp).

Note

This function calls rearrange.

Author(s)

Giorgio Arcara

References

Luck, S., J. (2005). *An Introduction to the Event-Related Potential Technique.* Cambridge, Mass.: The MIT Press.

See Also

erp.mean, erp.latency, rearrange, max, min, localmax, localmin

Examples

```
data(ERPsets)

dat=erp.peak(base="Exp1_word_subj", numbers=1:20,
win.ini=130, win.end=190, erplist=ERPsets, startmsec=-200, endmsec=1500,
others=c(condition="word", interval="130-190"), name.dep="Ampl", format="long", peak.fun=max)
```

`erp.t` *plot t-test comparisons at all timepoints between two sets of ERP data frames (single electrode)*

Description

This function performs a series of t-tests on each timepoint comparing two sets of ERP data frames. The first set is specified by `base1` and `numbers1`. The second set is specified by `base2` and `numbers2`.

Usage

```
erp.t(base1, base2, numbers1, numbers2 = numbers1, startmsec = -200,
endmsec=1200, electrode, smo = NULL, paired = TRUE, alpha = 0.05,
erplist1 = NULL, erplist2 = erplist1, sig = NULL, main = electrode,
col = c("blue", "red"), ...)
```

Arguments

`base1`	a string indicating the beginning of the name of the first set of ERP data frames.
`base2`	a string indicating the beginning of the name of the second set of ERP data frames.
`numbers1`	the numbers of the subjects of the first set of ERP data frames.
`numbers2`	The numbers of the subjects of the second set of ERP data frames.
`startmsec`	the start time (in ms) of the ERP vector.
`endmsec`	the end time (in ms) of the ERP vector.
`electrode`	a string indicating the electrode to be considered in the analysis.
`smo`	the smoothing parameter to apply (see `smooth.spline`). Default is `NULL` and no smoothing is applied.
`paired`	logical. This argument specifies if the t-tests are paired or not.
`alpha`	the alpha of the t-tests.
`erplist1`	a list containing the ERP data frames specified in `base1` and `numbers1`.
`erplist2`	a list containing the ERP data frames specified in `base2` and `numbers2`.
`sig`	if `NULL`, then a series of t-tests is computed. If an appropriate object is specified (i.e. the object returned by `erp.t` or `scalp.t`) then the results are retrieved by this object and the function simply plots the results.
`main`	plot title.
`col`	the colors of the waveforms.
`...`	further parameters to be passed to `erp`.

Value

The function plots two ERP waveforms. One is an average of the first set of specified data frames, and the other is an average of the second set of specified data frames. The colors of the waveforms are specified in `color.list`. Timepoints on which the t-test is significant are highlighted by grey bars. The function also returns an object with the results of the t-tests. This object can be used for future plots with `erp.t` or `scalp.t`, by specifying it as the argument `sig`.

Note

There is also a function `scalp.infl` able to perform the t-tests on all the electrodes, and not only on a single one.

Author(s)

Giorgio Arcara

See Also

`erp`, `scalp.t`, ~~~

Examples

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

erp.t("Exp1_word_subj", "Exp1_nonword_subj", 1:20, 1:20,
electrode="Fp1", ylim=c(-10,10),  startmsec=-200,
endmsec=1500, erplist1=ERPsets, erplist2=ERPsets,
col=c("blue", "red"))

## End(Not run)
```

---

erp.xaxis *Draw an erp x axis*

Description

This function plots an x axis for an erp waveform. Numbers and tick positions are determined according to three parameters: `length.erp`, `startmsec` and `endmsec` (the function calls `msectopoints`).

Usage

```
erp.xaxis(length.erp = NULL, startmsec=-200, endmsec = 1200,
x.tick=seq(-200, 1200, 200), x.labels = x.tick,  x.pos = NA,
x.outer = FALSE, x.font = NA, x.lty = "solid", x.lwd = 1,
x.lwd.ticks = 1, x.col = NULL, x.col.ticks = NULL, x.hadj = NA,
x.padj = NA, x.tcl = -0.5, x.tick.both= FALSE, x.cex = 1)
```

Arguments

length.erp	the length of the ERP vector the axis refers to.
startmsec	start time (in ms) of the ERP vector.
endmsec	end time (in ms) of the ERP vector.
x.tick	the position (in ms) of x ticks.
x.labels	the labels (usually a vector of numbers) to be plotted on the x axis.
x.pos	the position of x axis, in terms of y axis coordinates. Default is NA and the axis is drawn at the bottom of the plot.
x.outer	a logical value indicating whether the x axis should be drawn in the outer plot margin, rather than the standard plot margin.
x.font	font for text of x labels. Default is par("font").
x.lty	line type of both the x axis line and the x tick marks.
x.lwd	line width of the x axis line.
x.lwd.ticks	line width of the x tick marks.
x.col	colors of the x axis line and the x tick marks.
x.col.ticks	the color of the x tick marks (if specified, overrides x.col)
x.hadj	adjustment (see par("adj")) for all x labels parallel ("horizontal") to the reading direction. If this is not a finite value, the default is used (centering for strings parallel to the axis, alignment of the end nearest the axis otherwise).
x.padj	adjustment for each x tick label purpendicular to the reading direction. For labels parallel to the axes, padj = 0 means right or top alignment, and padj = 1 means left or bottom alignment. This can be a vector given a value for each string, and will be recycled as necessary.
x.tcl	The length of tick marks as a fraction of the height of a line of text. The default value is -0.5.
x.tick.both	Logical. Should the tick marks be drawn both above and below the x axis?
x.cex	The magnification to be used for the x labels.

Value

An erp x axis is drawn on an existing plot.

Note

This function is called internally by the erp function.

Author(s)

Giorgio Arcara

See Also

erp, erp.yaxis

Examples

```
plot(1, ylim=c(-5,5), xlim=c(0,200), frame.plot=FALSE,
type="n", axes=FALSE, xlab="", ylab="")

erp.xaxis(200, startmsec=-200, endmsec=1500,
x.pos=0, x.lwd=2)
```

`erp.yaxis` *Draw an erp y axis*

Description

This function plots an y axis of an erp waveform. The position of the axis is determined according to three parameters: `length.erp`, `startmsec` and `endmsec` (the function calls `msectopoints`).

Usage

```
erp.yaxis(length.erp = NULL, startmsec=-200, endmsec = 1200, y.tick=seq(-6,6,2),
y.labels = y.tick, y.pos = NA,  y.outer = FALSE, y.font = NA, y.lty = "solid",
y.lwd = 1, y.lwd.ticks = 1, y.col = NULL, y.col.ticks = NULL, y.hadj=NA,
y.padj = NA, y.tcl = -0.5, y.tick.both= FALSE, y.cex = 1)
```

Arguments

`length.erp`	the length of the ERP vector the axis is referred to.
`startmsec`	the start time (in ms) of the ERP vector.
`endmsec`	the end time (in ms) of the ERP vector.
`y.labels`	the labels (usually a vector of numbers) to be plotted on the y axis.
`y.pos`	the position of y axis, in terms of x axis coordinates (in ms). Default is `NA` and the axis is drawn at the left of the plot.
`y.tick`	the position (in ms) of y ticks.
`y.outer`	a logical value indicating whether the y axis should be drawn in the outer plot margin, rather than the standard plot margin.
`y.font`	font for the text of y labels. Default is par("font").
`y.lty`	line type for both the y axis line and the y tick marks.
`y.lwd`	line width for the y axis line.
`y.lwd.ticks`	line width for the y tick marks.
`y.col`	colors for the y axis line and the y tick marks.
`y.col.ticks`	color for the y tick marks (if specified, overrides y.col)

y.hadj	adjustment (see par("adj")) for all y labels parallel ("horizontal") to the reading direction. If this is not a finite value, the default is used (centring for strings parallel to the axis, alignment of the end nearest the axis otherwise).
y.padj	adjustment for each y tick label purpendicular to the reading direction. For labels parallel to the axes, padj = 0 means right or top alignment, and padj = 1 means left or bottom alignment. This can be a vector given a value for each string, and will be recycled as necessary.
y.tcl	The length of tick marks as a fraction of the height of a line of text. The default value is -0.5.
y.tick.both	Logical. Should the tick marks be drawn both above and below the y axis?
y.cex	The magnification to be used for the y labels.

Value

An erp y axis is drawn on an existing plot.

Note

This function is called internally by the erp function.

Author(s)

Giorgio Arcara

See Also

erp, erp.xaxis

Examples

```
plot(1, ylim=c(-5,5), xlim=c(0,200), frame.plot=FALSE,
type="n", axes=FALSE, xlab="", ylab="")

erp.yaxis(200, startmsec=-200, endmsec=1500, y.pos=0,
y.lwd=2, y.tick=seq(-4,4,2))
```

ERPsets *list of ERP data frames*

Description

In the erpR package, ERPsets is an erplist, that is a list of data frames containing ERP data, named according to the rationale of erpR (see erpR). ERPsets contains 40 data frames, for the combinations of 20 subjects and two experimental conditions. Each data frame has 34 electrodes in columns and 426 timepoints in rows. Data are recorded from -200 ms to 1500 ms. Data were sampled at 250 Hz.

Usage

```
data(ERPsets)
```

Format

A list of data frames with 426 timepoints (rows) and 34 electrodes (columns).

Note

some artifact were added to the original data for didactical purposes

Source

Data collected by Giorgio Arcara.

Examples

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

## End(Not run)
```

---

factorall *Factorize one or more variables in a data frame*

Description

A convenience function to transform on or more variables of a data frame into factors.

Usage

```
factorall(x)
```

Arguments

x	one or more data frame columns to be transformed into factor.

Value

The vectors supplied are transformed into factors. Notice that if a variable is already a factor, it is updated and all unused levels are dropped.

Author(s)

Giorgio Arcara.

Examples

```
data(iris)

iris2 <- iris[iris$Species!="setosa", ]
levels(iris2$Species)

iris2[,c("Petal.Width", "Species")]<-factorall(iris2[,c("Petal.Width", "Species")])
levels(iris2$Species)
str(iris2)
```

`grandaverage` *calculate the grandaverage of ERP data frames for graphical purposes*

Description

A function to average ERP objects, i.e. data frames with timepoints in rows, electrodes in columns and amplitude values in each cell. It is used to compute the grandaverage across several subjects, for plotting purposes.

Usage

```
grandaverage(base, numbers, electrodes="all", erplist = NULL, NA.sub=TRUE)
```

Arguments

`base`	a string indicating the beginning of the name of the `data.frame` containing the ERP data.
`numbers`	the numbers of subjects to be averaged.
`electrodes`	a character vector indicating the electrodes on which perform the grandaverage. Default is `"all"`, and the `grandaverage` is performed on all electrodes of the data frames.
`erplist`	a list containing ERP data frames.
`NA.sub`	logical. If `TRUE`, electrodes columns with `NA` are ignored in grandaverage computation. In this case, a Warning is displayed.

Value

A data frame with the values of subjects specified by the combination of `base` and `numbers`.

Note

The function keeps track of the averaged objects in a `comment`.

Author(s)

Giorgio Arcara

Examples

```
data(ERPsets)
# compute the average of subjects 1 to 20 for the condition
# specified by the string "Exp1_word_subj".

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)
```

`import.erp` *import ERP data matrices from ASCII files (file extension should be specified). Data are imported as a list containing a data frame for each files imported.*

Description

Import a series of .txt files with ERP data.

Usage

```
import.erp(filenamebase, numbers, ext=".txt",
outname = "ERP_subj", fileinfo=FALSE, erplist = NULL, path=getwd())
```

Arguments

`filenamebase`	a string indicating the beginning of the name of the .txt files containing the ERP data.
`numbers`	the numbers (indicating the subjects) of the files to be imported.
`ext`	A string indicating file extension. Default is ".txt"
`outname`	a string indicating the beginning of the name of the objects that will be created.
`fileinfo`	If `TRUE` the function expects that in the first row of imported file there is a string indicating some information on the subject to be stored as a comment in the data frame created.
`erplist`	If an erplist is specified, then the imported files will be added to this erplist.
`path`	A string indicating the directory where the ASCII files are contained. Otherwise the files will be searched from the current working directory.

Details

The optional argument `fileinfo` can be used to store some short information on the file. This is usually an ID for the Subject or the name of the original file from which the ASCII file has been exported (for example, if the files have been exported from eeglab, the name of the .set file). All objects that will be created from a file imported with `import.erp` will keep this information as a `comment`. Type `comment(objectname)` to access this information. If `fileinfo=FALSE`, the name of the file imported will be stored as a comment.

Value

A list containing several ERP data frames (one for each file imported).

Author(s)

Giorgio Arcara

Examples

```
## Not run:
### The following is an example for importing a series of file
## named Exp1_word_subj1.txt, Exp1_word_subj2.txt, etc.

#Exp1 <- import.erp(filenamebase="Exp1_word_subj", numbers=1:20,
# outname="Exp1_word_subj", ext=".txt" fileinfo = T, erplist=NULL)

## End(Not run)
```

`localmax` *calculate a local maximum*

Description

The function calculates a local maximum of the vector `x` surrounded by at least `n.points` lower numbers.

Usage

```
localmax(x, n.points=2)
```

Arguments

`x`	a vector of numbers.
`n.points`	a minimum number of lower values to determine the local maximum. Default is two points.

Details

The function calculates a local maximum of the vector x, i.e. the highest value of the vector x surrounded by at least n lower numbers. The concept of the local maximum applied to ERP analysis is taken from Luck (2005).

Value

The local maximum of the vector x is returned.

Author(s)

Giorgio Arcara

References

Luck, S., J. (2005). *An Introduction to the Event-Related Potential Technique.* Cambridge, Mass.: The MIT Press.

Examples

```
x=seq(-5.5,1,0.001)

y=sin(x)+0.3*x

plot(x,y, type="l")

#print an horizontal line intersecting the local maximum
abline(h=localmax(y))
```

---

localmin *calculate a local minimum*

Description

The function calculates a local minimum of the vector x surrounded by at least n.points higher numbers.

Usage

```
localmin(x, n.points = 2)
```

Arguments

x	a vector of numbers.
n.points	a minimum number of higher values to determine the local minimum (see Details). Default is two points.

Details

The function calculates a local minimum of the vector x, i.e. the lowest value of the vector x surrounded by at least n higher numbers. The concept of the local minimum applied to ERP analysis is taken from Luck (2005).

Value

The local minimum of the vector x is returned.

Author(s)

Giorgio Arcara

References

Luck, S., J. (2005). *An Introduction to the Event-Related Potential Technique.* Cambridge, Mass.: The MIT Press.

Examples

```
x=seq(-5.5,1,0.001)
y=-(sin(x)+0.3*x)
plot(x,y, type="l")
abline(h=localmin(y)) #print an horizontal line intersecting the local minimum
```

---

msectopoints *convert points in milliseconds*

Description

An internal function of erpR to convert milliseconds in points.

Usage

```
msectopoints(a, lengthsegment, startmsec, endmsec)
```

Arguments

a	a number indicating the ms to be converted in points
lengthsegment	the length (in points) of the segment
startmsec	the time (in ms) of the start of the segment
endmsec	the time (in ms) of the end of the segment

Details

This is mostly an internal function of erpR.

Value

The function returns a single value that is the conversion of a from msec to points.

Author(s)

Giorgio Arcara

See Also

pointstomsec

Examples

```
# convert 400 ms to points in a segment of 500 points starting from -200 ms
# and ending at 1500 (500 Hz).

msectopoints(120, 500, startmsec=-500, endmsec=1500)
```

---

named.agg *aggregate and keep the original names*

Description

A wrapper of the aggregate function. Data are provided as a formula and the original names are kept in the returned data.frame.

Usage

```
named.agg(formula, data, FUN, ..., subset, na.action = na.omit)
```

Arguments

formula	a formula, such as y ~ x or cbind(y1, y2) ~ x1 + x2, where the y variables are numeric data to be split into groups according to the grouping x variables (typically factors).
data	the data.frame to be aggregated.
FUN	a function to compute the summary statistics which can be applied to all data subsets.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NA values. The default is to ignore missing values in the given variables.
...	further arguments passed to or used by methods.

Value

For the data frame method, teh function returns a data frame with columns corresponding to the grouping variables in the right part of the formula, followed by aggregated columns from the left part of the formula. The names of the variables remain the same as in the original data frame.

Note

named.agg is a wrapper of aggregate

Author(s)

Ananda Mahto

See Also

aggregate

Examples

```
data(ERPsets)

datall=erp.mean(base = "Exp1_word_subj", numbers = 1:20,
win.ini = 400, win.end = 600, erplist=ERPsets, startmsec= -200,
endmsec = 1500, format="long", name.dep="Ampl")

datagg=named.agg(Ampl~electrode+Subject, datall, FUN=mean)
```

---

pointstomsec *convert points in milliseconds*

Description

An internal function of erpR to convert points in milliseconds.

Usage

```
pointstomsec(a, lengthsegment, startmsec, endmsec)
```

Arguments

a	a number indicating the point to be converted in ms
lengthsegment	the length (in points) of the segment
startmsec	the time (in ms) of the start of the segment
endmsec	the time (in ms) of the end of the segment

Details

This is mostly an internal function of erpR.

Value

The function return a single value that is the conversion of a from points to msec.

Author(s)

Giorgio Arcara

See Also

msectopoints

Examples

```
# calculate the time in msec corresponding to the point 128
# in a segment of 500 points starting from -200 ms
# and ending at 1500 (500 Hz).

pointstomsec(128, 500, startmsec = - 500, endmsec = 1500)
```

---

rearrange *rearrange a data frame from wide to long format*

Description

This utility function of the erpR package converts an ERP data frame with a *single* variable in wide format to a long format.

Usage

```
rearrange(deps, oth = NULL, dataset, name.dep = "Dep", name.newvar = "New_Var")
```

Arguments

deps	the numbers of the columns in the specified data frame to be rearranged from wide format to long format.
oth	other variables that are to be kept in the returned data frame.
dataset	the name of the data frame to be converted from wide format to long format.
name.dep	the name of the dependent variable in the returned data frame.
name.newvar	the name of the variable indicating the levels associated with name.dep in the returned data frame.

Details

The function converts a dataframe with a *single* variable in wide format (levels are in separate columns) in a data frame in long format (the variable is in a single column and the levels are specified by other columns). The original variable in wide format is represented in two columns: one (labeled by `name.dep` parameter) indicating the values of the dependent variable, and the other that is a factor (labeled by `name.newvar` parameter) indicating the levels of the variable for each observation. The variables specified in `oth` are replicated appropriately. This function is called internally by `erp.mean`, `erp.peak`, and `erp.latency`.

Value

A data frame in which the variables specified in the columns `deps` are converted in a single column `name.dep` with levels specified in `name.newvar`

Note

This function is called internally by `erp.mean`, `erp.peak`, and `erp.latency`.

Author(s)

Giorgio Arcara

See Also

`erp.mean`, `erp.peak`, `erp.latency`

Examples

```
data(ERPsets)

datwide=erp.mean(base="Exp1_word_subj", numbers=1:20, win.ini=130,
win.end=190, erplist=ERPsets, startmsec=-200, endmsec=1500, format="wide")

dat.long=rearrange(deps=c(1:32), oth=33:34, dataset=datwide, name.newvar="electrode")
```

---

scalp *plot ERP waveforms in a scalp array*

Description

This function plots the waveforms of several electrodes in an rectangular array reflecting their positions on the scalp

Usage

```
scalp(categ, smo = NULL, layout = 1,
ylims = "auto", yrev = TRUE, startmsec = -200, endmsec = 1200,
lwd = 1, lty = 1, color.list = c("blue", "red", "darkgreen"),
legend = F, legend.lab = "default", t.axis = seq(-100, endmsec, 200),
scalp.array=NULL)
```

Arguments

categ	a list indicating the ERP data frame to be plotted. All electrodes specified in the parameter layout will be plotted. The other electrodes will be ignored.
smo	the smoothing parameter to apply (see smooth.spline). Default is NULL and no smoothing is applied.
layout	A number indicating the layout of electrodes to be plotted. Currently only few default layouts are supported (a number from 1 to 5). Alternatively, a character vector specifying the layout may be supplied (see Details).
ylims	a number indicating the upper and lower limits of the y-axis. If "auto", the function calculates the y limits automatically.
yrev	logical. If TRUE, the y-axis is plotted with the negative values upward.
startmsec	the start time (in ms) of the ERP vector
endmsec	the end time (in ms) of the ERP vector
lwd	a vector specifying the line width to be associated with each ERP data frame specified in categ
lty	a vector specifying the line types to be associated with each ERP data frame specified in categ
color.list	a vector specifying the colors to be associated with each ERP data frame specified in categ
legend	logical. If TRUE a legend with object names is drawn in the top-right corner of the array.
legend.lab	the legend labels. If "default", the names are retrieved from categ.
t.axis	the position of the ticks on an additional axis indicating the time. This axis is plotted in the bottom-left corner of the plot array.
scalp.array	This argument is vector of two values indicating the number of panels of the plot, as number of rows and number of columns respectively. If a default layout is specified, the number of panels is automatically computed. See also Details.

Details

The layout parameter can be a number (from 1 to 5), that specifies some default electrode arrangements (currently only a few default arrangements are supported). Otherwise, the argument may be a character vector indicating the names of the electrodes to be plotted. The plot is divided in a number of panels as specified in scalp.array. The names specified will be plotted from top to bottom and from left to right. If "blank" is specified an empty panel will be plotted. If "xaxis" is specified a x axis indicating time will be plotted. If "yaxis" is specified a y axis indicating voltage will be plotted.

Value

The function plots a map of ERP waveforms in an array resembling the topographic locations of the electrodes on the scalp.

Warning

In `categ` argument, a list must be specified even if only one object is plotted.

Warning

in the `categ` argument, a list must be specified even if the plot is on only one object

Note

Currently this function supports a very limited number of layouts. Further versions will allow to specify customized layouts.

Author(s)

Giorgio Arcara

See Also

`scalp.t`, `scalp.infl`, `erp`

Examples

```
data(ERPsets)

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)

nonword=grandaverage("Exp1_nonword_subj", 1:20, erplist=ERPsets)

scalp(list(word), layout=1, ylim=10)

scalp(list(word, nonword), layout=1, ylim=10, legend=TRUE)
```

---

scalp.cor *plot the timepoint correlations between ERPs and an external variable (scalp array)*

Description

This function performs a series of correlations on each timepoint between ERP voltages and an external (typically a behavioural) variable. The set of ERP data frames is specified by `base` and `numbers`. The external variable is specified as a vector (of length equal to `numbers`) in the argument `external`. A scalp plot of the results is returned.

Usage

```
scalp.cor(base, numbers, external = NULL, alpha = 0.05,
method = c("pearson", "kendall", "spearman"), sig = NULL,
erplist = NULL, smo = NULL, layout = 1, ylims = "auto",
 yrev = TRUE, startmsec = -200, endmsec = 1200, lwd = c(1, 1),
 lty = c(1, 1), col = "blue", legend = TRUE, legend.lab = "default",
 t.axis=seq(-100,endmsec,200), scalp.array=NULL)
```

Arguments

base	a string indicating the beginning of the name of the set of ERP data frames.
numbers	the numbers of the subjects of the set of ERP data frames .
external	a vector indicating the values of the external variable (typically a behavioural variable). This vector must be of the same length of numbers
alpha	the alpha of the correlation tests.
method	the method of the correlations. Available methods are "pearson", "kendall", "spearman".
sig	if NULL, then a series of correlations is computed. If an appropriate object is specified (i.e. an object returned by scalp.cor or erp.cor) then the results are retrieved by this object and the function simply plots the results.
erplist	a list containing ERP data frames.
smo	the smoothing parameter to apply (see smooth.spline). Default is NULL and no smoothing is applied.
layout	A number indicating the layout of electrodes to be plotted. Currently only few default layouts are supported (a number from 1 to 5). Alternatively, a character vector specifying the layout may be supplied (see Details).
ylims	a number indicating the upper and lower limits of the y-axis. If "auto", the function calculates the y limits automatically.
yrev	logical. If TRUE, the y-axis is plotted with the negative values upward.
startmsec	the start time (in ms) of the ERP vector
endmsec	the end time (in ms) of the ERP vector
lwd	A vector specifying the line width to be associated with each ERP data frame specfied in categ
lty	A vector specifying the line types to be associated with each ERP data frame specfied in categ
col	the color of the ERP waveform.
legend	logical. If TRUE, a legend with object names is drawn in the top-right corner of the array.
legend.lab	the legend labels. If "default", the names are retrieved from the categ argument.
t.axis	the position of the ticks on an additional axis indicating the time. This axis is plotted in the bottom-left corner of the plot array.
scalp.array	This argument is vector of two values indicating the number of panels of the plot, as number of rows and number of columns respectively. If a default layout is specified, the number of panels is automatically computed. See also Details.

Details

The layout parameter can be a number (from 1 to 5), that specifies some default electrode arrangements (currently only a few default arrangements are supported). Otherwise, the argument may be a character vector indicating the names of the electrodes to be plotted. The plot is divided in a number of panels as specified in `scalp.array`. The names specified will be plotted from top to bottom and from left to right. If `"blank"` is specified an empty panel will be plotted. If `"xaxis"` is specified a x axis indicating time will be plotted. If `"yaxis"` is specified a y axis indicating voltage will be plotted.

Value

The function plots a scalp map of ERP waveforms in an array resembling the topographic locations of the electrodes on the scalp. The waveforms represent an average of the first set of specified data frames. Timepoints with a positive (and significant) correlation are highlighted by red bars. Timepoints with a negative (and significant) correlation are highlighted by blue bars. The function also returns an object with the results of the correlations. This object can be used for future plots with `erp.cor` or `scalp.cor`, by specifying it as the argument `sig`.

Author(s)

Giorgio Arcara

See Also

`erp.cor`, `~~~`

Examples

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

#generate an hypothetic external variables
RT=rnorm(20, mean=500, sd=200)

scalp.cor("Exp1_word_subj", 1:20, external = RT, layout=1,
erplist=ERPsets, ylim=10, legend=TRUE)

## End(Not run)
```

`scalp.infl` *inspect the influence of a subject on an average (scalp array)*

Description

The function returns a diagnostic plot to investigate the influence of a given ERP data frame on the grandaverage of a series of ERP data frames. A GUI is provided to explore the effect of removing a data frame (that is expected to be associated with a subject) from the grandaverage. Two plots are superposed: the original grandaverage, and the average removing the subject selected via the GUI. The plot shows an array of electrodes with a topography reflecting their positions on the scalp.

Usage

```
scalp.infl(base, numbers, smo = NULL, layout = 1, outnumber=1,
ylims = 12,  yrev = FALSE, startmsec = -200, endmsec = 1200,
lwd = c(1, 2), lty = 1, col = "black", out.col = "red",
erplist = NULL, t.axis=seq(-100,endmsec,200), scalp.array=NULL)
```

Arguments

base	a string indicating the beginning of the name of the data.frame containing the ERP data.
numbers	the numbers of the subjects to be averaged.
smo	the smoothing parameter to apply (see smooth.spline). Default is NULL and no smoothing is applied.
layout	A number indicating the layout of electrodes to be plotted. Currently only few default layouts are supported (a number from 1 to 5). Alternatively, a character vector specifying the layout may be supplied (see Details).
outnumber	The number of the subject that will be initially removed from the averaged (this value could be changed also via the GUI).
ylims	a number indicating the upper and lower limits of the y axis. If "auto", the function calculates the y limits automatically.
yrev	logical. If TRUE, the y-axis is plotted with the negative values upward.
startmsec	the start time (in ms) of the ERP vector
endmsec	the end time (in ms) of the ERP vector
lwd	line width of the waveforms.
lty	line type of the waveforms.
col	the color of the grand average including all subjects.
out.col	the color of the grand average in which the selected subject is removed
erplist	a list containing ERP data frames.
t.axis	the position of the ticks on an additional axis indicating the time. This axis is plotted in the bottom-left corner of the plot array.
scalp.array	This argument is vector of two values indicating the number of panels of the plot, as number of rows and number of columns respectively. If a default layout is specified, the number of panels is automatically computed. See also Details.

Details

The layout parameter can be a number (from 1 to 5), that specifies some default electrode arrangements (currently only a few default arrangements are supported). Otherwise, the argument may be a character vector indicating the names of the electrodes to be plotted. The plot is divided in a number of panels as specified in `scalp.array`. The names specified will be plotted from top to bottom and from left to right. If `"blank"` is specified an empty panel will be plotted. If `"xaxis"` is specified a x axis indicating time will be plotted. If `"yaxis"` is specified a y axis indicating voltage will be plotted.

Value

The function returns a plot of a scalp array showing the influence of a subject on the grandaverage.

Author(s)

Giorgio Arcara

See Also

`erp.infl`, `scalp`

Examples

```
data(ERPsets)

# Notice that Subject 1 is clearly particularly influential
# for the average on OZ.
scalp.infl(base="Exp1_word_subj", numbers=1:20, layout=1,
startmsec=-200, endmsec=1500, erplist=ERPsets)
```

---

scalp.t *plot t-test comparisons at all timepoints between two ERP set of data frames (scalp array)*

Description

This function performs a series of t-tests on each timepoint comparing two sets of ERP data frames. The first set is specified by `base1` and `numbers1`. The second is specified by `base2` and `numbers2`. A scalp plot of the results is returned.

Usage

```
scalp.t(base1, base2, numbers1, numbers2 = NULL, paired = TRUE,
alpha = 0.05, sig = NULL, erplist1 = NULL, erplist2 = erplist1,
smo = NULL, layout = 1, ylims = "auto", yrev = TRUE,
startmsec = -200, endmsec = 1200, lwd = c(1, 1), lty = c(1, 1),
color.list = c("blue", "red"), legend = F, legend.lab = "default",
t.axis = seq(-100, endmsec, 200), scalp.array=NULL)
```

Arguments

base1	a string indicating the beginning of the name of the first set of ERP data frames.
base2	a string indicating the beginning of the name of the second set of ERP data frames.
numbers1	the numbers of the subjects of the first set of ERP data frames .
numbers2	the numbers of the subjects of the second set of ERP data frames .
paired	logical. This argument specifies if the t-tests are paired or not.
alpha	the alpha of the t-tests.
sig	if NULL, then a series of t-test is computed. If an appropriate object is specified (i.e. an object returned by erp.t or scalp.t) then the results are retrieved by this object and the function simply plots the results.
erplist1	a list containing the ERP data frames specified in base1 and numbers1.
erplist2	a list containing the ERP data frames specified in base2 and numbers2.
smo	the smoothing parameter to apply (see smooth.spline). Default is NULL and no smoothing is applied.
layout	A number indicating the layout of electrodes to be plotted. Currently only few default layouts are supported (a number from 1 to 5). Alternatively, a character vector specifying the layout may be supplied (see Details).
ylims	a number indicating the upper and lower limits of the y-axis. If "auto", the function calculates the y-limits automatically.
yrev	logical. If TRUE, the y-axis is plotted with negative values upward.
startmsec	the start time (in ms) of the ERP vector
endmsec	the end time (in ms) of the ERP vector
lwd	a vector specifying the line width to be associated with each ERP data frame specified in categ
lty	a vector specifying the line types to be associated with each ERP data frame specified in categ
color.list	a vector specifying the colors to be associated with each ERP data frame specified in categ
legend	logical. If TRUE, a legend with object names is drawn in the top-right corner of the array.
legend.lab	the legend labels. If "default", the names are retrieved from categ.

t.axis	the position of the ticks of an additional axis indicating the time. This axis is plotted in the bottom-left corner of the plot array.
scalp.array	This argument is vector of two values indicating the number of panels of the plot, as number of rows and number of columns respectively. If a default layout is specified, the number of panels is automatically computed. See also Details.

Details

The layout parameter can be a number (from 1 to 5), that specifies some default electrode arrangements (currently only a few default arrangements are supported). Otherwise, the argument may be a character vector indicating the names of the electrodes to be plotted. The plot is divided in a number of panels as specified in scalp.array. The names specified will be plotted from top to bottom and from left to right. If "blank" is specified an empty panel will be plotted. If "xaxis" is specified a x axis indicating time will be plotted. If "yaxis" is specified a y axis indicating voltage will be plotted.

Value

The function plots a map of two series of ERP waveforms in an array resembling the topographic locations of the electrodes on the scalp. One series is an average of the first set of specified data frames, and the other series is the average of the second set of specified data frames. The colors of the waveforms are specified in color.list. Timepoints in which the t-test is significant are highlighted by grey bars. The function also returns an object with the results of the t-tests. This object can be used for future plots with erp.t or scalp.t, by specifying it as the argument sig.

Author(s)

Giorgio Arcara

See Also

erp, erp.t

Examples

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

scalp.t("Exp1_word_subj", "Exp1_nonword_subj", 1:20, 1:20, smo=0,
layout=1, ylims=10, startmsec=-200, endmsec=1500,
color.list=c("blue", "red"), erplist1=ERPsets, erplist2=ERPsets)

## End(Not run)
```

topoplot *plot a topographic map of ERP*

Description

Plot a topographic map of an ERP on a specific timepoint or averaging the ERP in a time window. The head is represented as a circle.

Usage

```
topoplot(erpobj, startmsec=-200, endmsec=1200, win.ini,
win.end,  exclude = NULL, elec.coord=NULL, projection="orthographic",
palette.col="jet", palette.steps=10, return.coord = FALSE, zlim=NULL,
interpolation = "cubicspline", extrap = TRUE, interp.points = 500,
return.notfound=FALSE, mask = TRUE,  contour=TRUE, x.rev=FALSE,
draw.elec.pos=TRUE,  draw.nose=FALSE, draw.elec.lab=TRUE,
elec.lab.adj=c(0.5, NA), head.col="black", head.lwd=1, ...)
```

Arguments

erpobj	an ERP dataframe with electrodes in columns and timepoints in rows.
startmsec	start time (in ms) of the ERP data frame. It can be a negative value, indicating the baseline time frame.
endmsec	end time (in ms) of the ERP data frame.
win.ini	the initial edge (in ms) of the window on which the topographic map has to be plotted.
win.end	the final edge (in ms) of the window on which the topographic map has to be plotted. If equal to win.ini, then the map of a single time point is plotted.
exclude	a vector of characters indicating the channel names that are excluded from the plotting
elec.coord	an optional data frame with electrode coordinates (see Note).
projection	the type of electrode projection from 3d to 2d. Two methods available are "orthographic" and "equalarea".
palette.col	the color palette. Two default palettes are available: "jet" or "heat". Otherwise a customized palette can be created. A palette will be built as an interpolation trough the vector of colors provided.
palette.steps	the number of steps in the color palette.
return.coord	if TRUE, the function doesn't create a topoplot, but returns a data frame containing all the built-in electrode coordinates.
zlim	the limits of z axis (i.e. colors in the topographic map). If NULL, then the limits are automatically computed.
interpolation	The interpolation between electrodes. Two methods from package akima are available: "linear" and "cubicspline".

extrap	logical. Extrapolate the data outside the convex hull also. This parameter is ignored if interpolation method is "linear".
interp.points	the number of points to be used for interpolation. This parameter defines the smoothness of the plot.
return.notfound	logical. If TRUE, the function doesn't create a topoplot, but returns a vector with the electrode names in erpobj not contained in the supplied list of electrodes.
mask	logical. Should a circular mask around topographic map be drawn?
contour	logical. Should contour lines be drawn?
x.rev	logical. Should the x axis be reversed?
draw.elec.pos	logical. Should the black points be drawn in correspondence with electrode positions?
draw.nose	logical. Should a nose be drawn to indicate upper side of the mask?
draw.elec.lab	logical. Should electrode labels be drawn?
elec.lab.adj	a vector with a pair of values indicating horizontal and vertical adjustment of electrode labels (see text).
head.col	line color of head and nose.
head.lwd	line width of head and nose.
...	further parameters to be passed to image.

Value

The topographic map is plotted. The function also returns a list of values that can be used by other functions. The first item of the list is a pair of values indicating the zlim of the plot. The second item of the list is a vector indicating the palette created by the function (palette). This last vector with palette colors can be used by the topoplot.palette function.

Note

The topographic plots are created according the following steps. 1) The electrodes are projected from 3d space to 2d space. 2) Voltage values are interpolated (and extrapolated, if extrap=TRUE) on 2d space (the interpolation methods come from the nterp function of akima package). The spline interpolation is NOT a spherical spline interpolation.
The function already contains a built-in list of electrode coordinates, but some names may not be the same in the supplied data frame and in the built-in list (even if the case is ignored in the match). An optional object with electrode coordinates can be supplied. This object should be a data frame containing these four columns: a column labeled "el.name" with electrode names, three columns labeled "x", "y", "z" with spherical coordinates of the electrode.
Type topoplot(return.coord=TRUE) to see the data frame structure.
In the "orthographic" projection only "x" and "y" of electrode coordinates are mantained. The "equalarea" projection keeps in the 2d projection the area proprortions as in 3d representation. The topoplot function uses the function interp of package akima, that is released under a non-commercial use license.

Author(s)

Giorgio Arcara

Examples

```
if(require(akima)){

data(ERPsets)

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)

# check if some electrodes are not present in the list
# and create an object with these electrode names.

notfound=topoplot(word, return.notfound=TRUE)

#make a topoplot excluding not found electrode
topoplot(word, startmsec=-200, endmsec=1500, win.ini=400,
win.end=600, exclude=notfound)
}
```

`topoplot.palette` *Add a palette to a plot*

Description

This function draws a topoplot palette as a rectangle with colored areas indicating the palette shades.

Usage

```
topoplot.palette(cols, pos = c(0.5,0.5), p.width=0.2, p.height=0.8,
horizontal = FALSE, rev= FALSE, palette.lwd=1, palette.bord="black",
palette.lim = c(-5,5), labels=TRUE, lab.dist = 1, lab.cex = 1,
 lab.font = 1, lab.family = "")
```

Arguments

`cols`	a vector of colors to draw the palette. This vector may be the output of `topoplot`.
`pos`	the position of the palette as a proportion of the plot region. Default (`c(0.5, 0.5)`) is centered on the screen.
`p.width`	palette width as a proportion of the plot region.
`p.height`	palette height as a proportion of the plot region.
`horizontal`	logical. Should the color shades be plotted horizontally? (default is vertical)
`rev`	logical. Should the palette shades order be reversed?
`palette.lwd`	line width of the palette rectangle.
`palette.bord`	color of the palette border.
`palette.lim`	limits of the palette. They will be plotted as text if `labels=TRUE`.

labels	logical. Should labels indicating zlim be drawn?
lab.dist	a number indicating the distance of the labels from the palette rectangle. Positive numbers move the labels from the palette rectangle, whereas negative numbers move the labels towards the rectangle.
lab.cex	magnification factor for the labels.
lab.font	font type for the labels (see par).
lab.family	font family for the labels (see par).

Value

the function draws a palette on an existing plot.

Note

The two parameters cols and zlim may be supplied with values that are returned from topoplot.

Author(s)

Giorgio Arcara

Examples

```
if(require(akima)) {

data(ERPsets)

word=grandaverage("Exp1_word_subj", 1:20, erplist=ERPsets)

# check if some electrodes are not present in the list
# and create an object with these electrode names.
notfound=topoplot(word, return.notfound=TRUE)

#define a layout for
mat=matrix(c(1,2), 1, 2, byrow=TRUE)

layout(mat, widths=c(0.8, 0.2))

#make a topoplot excluding not found electrode
par(pty="s")
topo.data=topoplot(word, startmsec=-200, endmsec=1500, win.ini=400,
win.end=600, exclude=notfound)

#draw the palette on a new empty plot.
par(pty="m", mar=c(0,0,0,0))
plot.new()
topoplot.palette(cols=topo.data$palette,
palette.lim=topo.data$zlim, p.height=0.6)

}
```

tpairs *calculate pairwise t-tests*

Description

An utility function to perform a series of t-tests on several combinations of levels of one or more variables. P-value correction methods can be specified.

Usage

```
tpairs(dat, vars, contr, dep, wid, p.adjust.methods = "none",
paired = FALSE, ...)
```

Arguments

dat	a data.frame including all variables for the t-test computation.
vars	a vector of characters indicating the names of the independent variables to be considered for the t-tests. If length vars > 1, a new variable will be created by combining the levels of the variables specified in vars.
contr	a list specifying the t-test to be performed. If "all" is specified, t-tests on all possible combinations are performed. See Details for further specification information.
dep	a string specifying the name of the dependent variable (a column in dat).
wid	optional: a string specifying the variable that encodes the repeated measure identifier.
p.adjust.methods	p-value correction methods as in p.adjust.
paired	logical. If TRUE, paired t-tests are performed.
...	Arguments passed to t.test.

Details

The contr argument allows to specify a subset of combinations of levels of the variables listed in vars to be performed. The contrasts have to be specified as a list of lists. Each list within the main list contains two vectors with strings. The strings specify the levels of the variables in vars that have to be compared. The convention is to specify the levels separated by an underscore. For example suppose that vars=c("height", "color"), and contr=list(list(c("high_red"), c("low_red")), list(c("high_green", "low_green"))) will perform two t-tests. The first t-test it will compare the mean for height=="high" and color=="red" with the mean of height=="low", and color=="red". The second t-test will compare the mean of height=="high", and color=="green" with the mean of height=="low" and color=="green". If a p-value correction method is specified, only p-values of the t-test carried out will be taken into account.

Value

A `data.frame` with the results of the t-tests specified.

Note

The means for the combinations of the variables taken into account are computed internally by `tpairs`, according to the specification in `vars`, `dep`, `wid`. The function is not able to deal with mixed within and between t-tests. Please, perform the two kinds of t-tests separately and take into consideration if the variables specified in `vars` are mixed (both within and between).

Author(s)

Giorgio Arcara

See Also

`t.test`, `p.adjust`

Examples

```
# simulate some subjects
subjRT=rnorm(20, 500, 100)

#simulate the effects of three experimental conditions for each subject
condA=rnorm(20, 50, 10)
condB=rnorm(20, -40, 10)
condC=rnorm(20, 20, 10)

#create a data frame
dat=data.frame(Subject=rep(1:20,3),
condition=c(rep("A", 20), rep("B", 20), rep("C", 20)),
RT=c(subjRT+condA, subjRT+condB, subjRT+condC ))

#perform pairwise t.test
tpairs(dat, "condition", "all", "RT", "Subject", var.equal=TRUE)
```

Index