Package 'eegkit'

May 8, 2026

Type Package

Title Toolkit for Electroencephalography Data

Version 1.0-5

Date 2025-04-15

Maintainer Nathaniel E. Helwig <helwig@umn.edu>

Depends eegkitdata, bigsplines, ica, rgl, signal

Description Analysis and visualization tools for electroencephalography (EEG) data. Includes functions for (i) plotting EEG data, (ii) filtering EEG data, (iii) smoothing EEG data; (iv) frequency domain (Fourier) analysis of EEG data, (v) Independent Component Analysis of EEG data, and (vi) simulating event-related potential EEG data.

License GPL (>= 2)

NeedsCompilation no

Author Nathaniel E. Helwig [aut, cre]

Repository CRAN

Date/Publication 2025-04-15 20:10:14 UTC

Contents

`eegcap` *Draws EEG Cap with Selected Electrodes*

Description

Creates two- or three-dimensional plot of electroencephalography (EEG) cap with user-input electrodes. Three-dimensional plots are created using the `eegcoord` data and the `plot3d` function (from `rgl` package). Currently supports 84 scalp electrodes, and plots according to the international 10-10 system. Includes customization options (e.g., each electrode can have a unique plotting color, size, label color, etc.).

Usage

```
eegcap(electrodes = "10-10", type = c("2d", "3d"),
       plotlabels = TRUE, plotaxes = FALSE, main = "",
       xyzlab = NULL, cex.point = NULL, col.point = NULL,
       col.border = NULL, cex.label = NULL, col.label = NULL,
       nose = TRUE, ears = TRUE, head = TRUE,
       col.head = "AntiqueWhite", index = FALSE,
       plt = c(0.03,0.97,0.03,0.97), ...)
```

Arguments

`electrodes`	Character vector with electrodes to plot. Each element of `electrodes` must match one of the 89 reference electrodes (see Notes). Mismatches are ignored (not plotted). Input is NOT case sensitive. Default plots all available electrodes (full 10-10 system).
`type`	Type of plot to create: `type="3d"` produces three-dimensional plot, whereas `type="2d"` produces two-dimensional projection plot (bird's eye view).
`plotlabels`	If `TRUE`, the electrode labels are plotted.
`plotaxes`	If `TRUE`, the axes are plotted.
`main`	Title to use for plot. Default is no title
`xyzlab`	Axis labels to use for plot. If `type="2d"`, then `xyzlab` should be two-element character vector giving x and y axis labels. If `type="3d"`, then `xyzlab` should be three-element character vector giving x, y, and z axis labels.
`cex.point`	Size of electrode points. Can have a unique size for each electrode.
`col.point`	Color of electrode points. Can have a unique color for each electrode.
`col.border`	Color of electrode point borders. Can have a unique color for each electrode.
`cex.label`	Size of electrode labels. Can have a unique size for each electrode label. Input is ignored if `plotlabels=FALSE` is used.

col.label	Color of electrode labels. Can have a unique color for each electrode label. Input is ignored if plotlabels=FALSE is used.
nose	If TRUE, triangle is plotted to represent the subject's nose. Ignored if type="3d".
ears	If TRUE, ovals are plotted to represent the subject's ears. Ignored if type="3d".
head	If TRUE, head is plotted. Ignored if type="2d".
col.head	Color for dummy head in 3d plot. Ignored if type="2d".
index	Logical indicating if the cap row indices should be returned (see Note).
plt	A vector of the form c(x1, x2, y1, y2) giving the coordinates of the plot region as fractions of the current figure region. See par.
...	Optional inputs for plot or plot3d function.

Value

Produces plot of EEG cap and possibly returns cap row indices.

Note

Currently supports 84 scalp electrodes (plus ears and nose): A1 A2 AF1 AF2 AF3 AF4 AF5 AF6 AF7 AF8 AFZ C1 C2 C3 C4 C5 C6 CP1 CP2 CP3 CP4 CP5 CP6 CPZ CZ F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 FC1 FC2 FC3 FC4 FC5 FC6 FCZ FP1 FP2 FPZ FT7 FT8 FT9 FT10 FZ I1 I2 IZ NZ O1 O2 OZ P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 POZ PZ T7 T8 T9 T10 TP7 TP8 TP9 TP10

See eegcoord for the coordinates used to create plot. Setting index=TRUE returns the row indices of eegcoord that were used to plot the cap.

To save three-dimensional plots, use the rgl.postscript function (from rgl package).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Murdoch, D., and Adler, D. (2025). *rgl: 3D Visualization Using OpenGL*. doi:10.32614/CRAN.package.rgl, R package version 1.3.18, http://CRAN.R-project.org/package=rgl

Oostenveld, R., and Praamstra, P. (2001). The Five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology, 112*, 713-719. doi:10.1016/S13882457(00)00527-7

Examples

```
##########   EXAMPLE 1   ##########

# plot 10-10 system (default):

# plot full cap 2d (default options)
eegcap()
```

```
# plot full cap 2d (different color for ears and nose)
data(eegcoord)
mycols <- rep("white",87)
enames <- rownames(eegcoord)
mycols[enames=="A1"] <- "green"
mycols[enames=="A2"] <- "light blue"
mycols[enames=="NZ"] <- "pink"
eegcap(col.point = mycols)



##########   EXAMPLE 2   ##########

# plot 10-20 system:

# plot 2d cap with labels
eegcap("10-20")

# plot 2d cap without labels
eegcap("10-20", plotlabels = FALSE)



##########   EXAMPLE 3   ##########

# plot custom subset of electrodes
myelectrodes <- c("FP1","FP2","FPZ","F7","F3","FZ",
                  "F4","F8","T7","C3","CZ","C4","T8",
                  "P7","P3","PZ","P4","P8","O1","O2")
eegcap(myelectrodes)
```

---

eegcap2d *Draws 2D EEG Cap*

Description

Creates two-dimensional plot of electroencephalography (EEG) cap with user-input electrodes. Currently supports 84 scalp electrodes, and plots according to the international 10-10 system. Includes customization options (e.g., each electrode can have a unique plotting color, size, label color, etc.).

Usage

```
eegcap2d(electrodes = "10-10", axes = FALSE, asp = 1,
         cex.point = 2.75, col.point = "green", pch.point = 19,
         cex.border = 2.75, col.border = "black", pch.border = 21,
         cex.label = 0.5, col.label = "black",
         head = TRUE, nose = TRUE, ears = TRUE,
         main = "", xlab = "", ylab = "",
         xlim = c(-13.7, 13.7), ylim = c(-13.7, 13.7), ...)
```

Arguments

electrodes	Character vector with electrodes to plot. Each element of electrodes must match one of the 89 reference electrodes (see Details). Mismatches are ignored (not plotted). Input is NOT case sensitive. Default plots all available electrodes (full 10-10 system).
axes	If FALSE (default), no axes are plotted.
asp	Aspect ratio for plot (defaults to 1).
cex.point	Character EXpansion value for electrodes. Set to a negative value to suppress the electrode plotting.
col.point	Color for electrodes. Ignored if cex.point < 0.
pch.point	Plotting character for electrodes. Ignored if cex.point < 0.
cex.border	Character EXpansion value for electrode borders. Set to a negative value to suppress the electrode border plotting.
col.border	Color for electrode borders. Ignored if cex.border < 0.
pch.border	Plotting character for electrode borders. Ignored if cex.border < 0.
cex.label	Character EXpansion value for electrode labels. Set to a negative value to suppress the electrode label plotting.
col.label	Color for electrode labels. Ignored if cex.label < 0.
head	If TRUE, a circle is plotted to represent the subject's head.
nose	If TRUE, a triangle is plotted to represent the subject's nose.
ears	If TRUE, two ovals are plotted to represent the subject's ears.
main	Title to use for plot. Default is no title.
xlab, ylab	x-axis and y-axis labels for the plot. Default is no axis labels.
xlim, ylim	x-axis and y-axis limits for the plot.
...	Optional inputs for plot function.

Details

Currently supports 84 scalp electrodes (plus ears and nose): A1 A2 AF1 AF2 AF3 AF4 AF5 AF6 AF7 AF8 AFZ C1 C2 C3 C4 C5 C6 CP1 CP2 CP3 CP4 CP5 CP6 CPZ CZ F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 FC1 FC2 FC3 FC4 FC5 FC6 FCZ FP1 FP2 FPZ FT7 FT8 FT9 FT10 FZ I1 I2 IZ NZ O1 O2 OZ P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 POZ PZ T7 T8 T9 T10 TP7 TP8 TP9 TP10

See eegcoord for the coordinates used to create plot.

Value

Produces plot of EEG cap.

Note

Unlike the eegcap function, this function does not use par$plt for the figure positioning.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Oostenveld, R., and Praamstra, P. (2001). The Five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology, 112*, 713-719. doi:10.1016/S13882457(00)00527-7

See Also

See eegcap for a similar implementation, which also supports 3d EEG cap plotting.

Examples

```
##########   EXAMPLE 1   ##########

# plot 10-10 system (default):

# plot full cap (default options)
eegcap2d()

# plot full cap (different color for ears and nose)
data(eegcoord)
mycols <- rep(NA, 87)
enames <- rownames(eegcoord)
mycols[enames=="A1"] <- "green"
mycols[enames=="A2"] <- "light blue"
mycols[enames=="NZ"] <- "pink"
eegcap2d(col.point = mycols)



##########   EXAMPLE 2   ##########

# plot 10-20 system:

# plot cap with labels
eegcap2d("10-20")

# plot cap without labels
eegcap2d("10-20", cex.label = -1)



##########   EXAMPLE 3   ##########

# plot custom subset of electrodes
myelectrodes <- c("FP1","FP2","FPZ","F7","F3","FZ",
                  "F4","F8","T7","C3","CZ","C4","T8",
                  "P7","P3","PZ","P4","P8","O1","O2")
eegcap2d(myelectrodes)
```

eegcapdense *Draws Dense EEG Cap with Selected Electrodes*

Description

Creates two- or three-dimensional plot of dense electroencephalography (EEG) cap that spans user-input electrodes. Three-dimensional plots are created using the eegdense data and the plot3d function (from rgl package). Currently supports 933 scalp electrodes. Includes customization options (e.g., each electrode can have a unique plotting color, size, label color, etc.).

Usage

```
eegcapdense(electrodes = "10-10", type = c("2d", "3d"),
            plotlabels = TRUE, plotaxes = FALSE, main = "",
            xyzlab = NULL, cex.point = NULL, col.point = NULL,
            cex.label = NULL, col.label = NULL, nose = TRUE,
            ears = TRUE, head = TRUE, col.head = "AntiqueWhite",
            index = FALSE, zconst = 0.5, plt = c(0.03,0.97,0.03,0.97), ...)
```

Arguments

electrodes	Character vector with electrodes to plot. Each element of electrodes must match one of the 89 reference electrodes (see Notes). Mismatches are ignored (not plotted). Input is NOT case sensitive. Default plots all available electrodes (full 10-10 system).
type	Type of plot to create: type="3d" produces three-dimensional plot, whereas type="2d" produces two-dimensional projection plot (bird's eye view).
plotlabels	If TRUE, the electrode labels are plotted.
plotaxes	If TRUE, the axes are plotted.
main	Title to use for plot. Default is no title
xyzlab	Axis labels to use for plot. If type="2d", then xyzlab should be two-element character vector giving x and y axis labels. If type="3d", then xyzlab should be three-element character vector giving x, y, and z axis labels.
cex.point	Size of electrode points. Can have a unique size for each electrode.
col.point	Color of electrode points. Can have a unique color for each electrode.
cex.label	Size of electrode labels. Can have a unique size for each electrode label. Input is ignored if plotlabels=FALSE is used.
col.label	Color of electrode labels. Can have a unique color for each electrode label. Input is ignored if plotlabels=FALSE is used.
nose	If TRUE, triangle is plotted to represent the subject's nose. Ignored if type="3d".
ears	If TRUE, ovals are plotted to represent the subject's ears. Ignored if type="3d".

head	If TRUE, head is plotted. Ignored if type="2d".
col.head	Color for dummy head in 3d plot. Ignored if type="2d".
index	Logical indicating if the cap row indices should be returned (see Note).
zconst	Scalar controlling which row indices should be returned (see Note).
plt	A vector of the form c(x1, x2, y1, y2) giving the coordinates of the plot region as fractions of the current figure region. See par.
...	Optional inputs for plot or plot3d function.

Value

Produces plot of EEG cap and possibly returns cap row indices.

Note

Currently supports 84 scalp electrodes (plus ears and nose): A1 A2 AF1 AF2 AF3 AF4 AF5 AF6 AF7 AF8 AFZ C1 C2 C3 C4 C5 C6 CP1 CP2 CP3 CP4 CP5 CP6 CPZ CZ F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 FC1 FC2 FC3 FC4 FC5 FC6 FCZ FP1 FP2 FPZ FT7 FT8 FT9 FT10 FZ I1 I2 IZ NZ O1 O2 OZ P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 POZ PZ T7 T8 T9 T10 TP7 TP8 TP9 TP10

See eegdense for the coordinates used to create plot. Setting index=TRUE returns the row indices of eegdense that were used to plot the cap. Only returns row indices with z-coordinates >= (zmin-zconst), where zmin is minimum z-coordinate of input electrodes.

To save three-dimensional plots, use the rgl.postscript function (from rgl package).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Murdoch, D., and Adler, D. (2025). *rgl: 3D Visualization Using OpenGL*. doi:10.32614/CRAN.package.rgl, R package version 1.3.18, http://CRAN.R-project.org/package=rgl

Oostenveld, R., and Praamstra, P. (2001). The Five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology, 112*, 713-719. doi:10.1016/S13882457(00)00527-7

Examples

```
##########   EXAMPLE 1   ##########

# plot 10-10 system (default):
eegcapdense()



##########   EXAMPLE 2   ##########

# plot 10-20 system:
```

```
eegcapdense("10-20", plotlabels = FALSE)



##########   EXAMPLE 3   ##########

# plot custom subset of electrodes
myelectrodes <- c("FP1","FP2","FPZ","F7","F3","FZ",
                  "F4","F8","T7","C3","CZ","C4","T8",
                  "P7","P3","PZ","P4","P8","O1","O2")
eegcapdense(myelectrodes)
```

---

eegcoord *EEG Cap Coordinates*

Description

Three-dimensional electroencephalography (EEG) electrode coordinates (measured in cm), and corresponding projection onto two-dimensional xy plane. Contains 84 scalp electrodes, as well as nose and ears.

Usage

```
data(eegcoord)
```

Format

A data frame with 87 observations and the following 5 variables:

x x-coordinate of 3d cap (numeric).

y y-coordinate of 3d cap (numeric).

z z-coordinate of 3d cap (numeric).

xproj Projected x-coordinate of 2d cap (numeric).

yproj Projected y-coordinate of 2d cap (numeric).

Electrode channel name labels can be obtained using `rownames(eegcoord)`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Source

Created by Nathaniel E. Helwig (2014) using:

Murdoch, D., and Adler, D. (2025). *rgl: 3D Visualization Using OpenGL*. doi:10.32614/CRAN.package.rgl, R package version 1.3.18, http://CRAN.R-project.org/package=rgl

Oostenveld, R., and Praamstra, P. (2001). The Five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology, 112*, 713-719. doi:10.1016/S13882457(00)00527-7

Schlager S (2017). Morpho and Rvcg - Shape Analysis in R. In Zheng G, Li S, Szekely G (eds.), Statistical Shape and Deformation Analysis, 217-256. Academic Press. ISBN 9780128104934.

Examples

```
##########   EXAMPLE   ##########

data(eegcoord)
enames <- rownames(eegcoord)
# plot3d(eegcoord[,1],eegcoord[,2],eegcoord[,3],size=10,col="green")
# text3d(eegcoord[,1],eegcoord[,2],eegcoord[,3],texts=enames,col="blue")
plot(eegcoord[,4],eegcoord[,5],cex=2,col="green",pch=19)
text(eegcoord[,4],eegcoord[,5],labels=enames,col="blue")
```

---

eegdense *Dense EEG Cap Coordinates*

Description

Dense (hypothetical) three-dimensional electroencephalography (EEG) electrode coordinates, and corresponding projection onto two-dimensional plane. Dense cap spans the 84 scalp electrodes defined in eegcoord.

Usage

```
data(eegdense)
```

Format

A data frame with 977 observations and the following 5 variables:

x x-coordinate of 3d cap (numeric).

y y-coordinate of 3d cap (numeric).

z z-coordinate of 3d cap (numeric).

xproj Projected x-coordinate of 2d cap (numeric).

yproj Projected y-coordinate of 2d cap (numeric).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Source

Created by Nathaniel E. Helwig (2014) using:

Murdoch, D., and Adler, D. (2025). *rgl: 3D Visualization Using OpenGL*. doi:10.32614/CRAN.package.rgl, R package version 1.3.18, http://CRAN.R-project.org/package=rgl

Oostenveld, R., and Praamstra, P. (2001). The Five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology, 112*, 713-719. doi:10.1016/S13882457(00)00527-7

Schlager S (2017). Morpho and Rvcg - Shape Analysis in R. In Zheng G, Li S, Szekely G (eds.), Statistical Shape and Deformation Analysis, 217-256. Academic Press. ISBN 9780128104934.

Examples

```
##########   EXAMPLE   ##########

data(eegdense)
# plot3d(eegdense[,1],eegdense[,2],eegdense[,3],size=10,col="green")
plot(eegdense[,4],eegdense[,5],cex=1,col="green",pch=19)
```

---

eegfft *Fast Fourier Transform of EEG Data*

Description

Finds the strength (amplitude) and phase shift of the input signal(s) at a particular range of frequencies via a Discrete Fast Fourier Transform (FFT). Can input single or multi-channel data.

Usage

```
eegfft(x, Fs, lower, upper)
```

Arguments

`x`	Vector or matrix (time by channel) of EEG data with `n` time points.
`Fs`	Sampling rate of `x` in Hz such that `n = s * Fs` where `s` is the number of seconds of input data (some positive integer).
`lower`	Lower band in Hz. Smallest frequency to keep (defaults to `0`).
`upper`	Upper band in Hz. Largest frequency to keep (defaults to `Fs/2 - Fs/n`).

Details

The `fft` function (or `mvfft` function) is used to implement the FFT (or multivatiate FFT). Given the FFT, the *strength* of the signal is the modulus (`Mod`), and the *phase.shift* is the angle (`Arg`).

Value

If x is a vector, returns a data frame with variables:

frequency	vector of frequencies
strength	strength (amplitude) of signal at each frequency
phase.shift	phase shift of signal at each frequency

If x is a matrix with J channels, returns a list with elements:

frequency	vector of frequencies of length F
strength	F by J matrix: strength (amplitude) of signal at each frequency and channel
phase.shift	F by J matrix: phase shift of signal at each frequency and channel

Note

The strength of the signal has the same unit as the input (typically microvolts), and the phase shift is measured in radians (range -pi to pi).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Cooley, James W., and Tukey, John W. (1965) An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19(90), 297-301. doi:10.7551/mitpress/5222.003.0014

Singleton, R. C. (1979) Mixed Radix Fast Fourier Transforms, in Programs for Digital Signal Processing, IEEE Digital Signal Processing Committee eds. IEEE Press.

Examples

```
##########   EXAMPLE   ##########

### Data Generation ###

# parameters for signal
Fs <- 1000                              # 1000 Hz signal
s <- 3                                  # 3 seconds of data
t <- seq(0, s - 1/Fs, by = 1/Fs)        # time sequence
n <- length(t)                          # number of data points
freqs <- c(1, 5, 10, 20)                # frequencies
amp <- c(2, 1.5, 3, 1.75)               # strengths (amplitudes)
phs <- c(0, pi/6, pi/4, pi/2)           # phase shifts

# create data generating signals
mu <- rep(0, n)
for(j in 1:length(freqs)){
  mu <- mu + amp[j] * sin(2*pi*t*freqs[j] + phs[j])
}
set.seed(1)                           # set random seed
```

```
e <- rnorm(n)                          # Gaussian error
y <- mu + e                            # data = mean + error


### FFT of Noise-Free Data ###

# fft of noise-free data
ef <- eegfft(mu, Fs = Fs, upper = 40)
head(ef)
ef[ef$strength > 0.25,]

# plot frequency strength
par(mfrow = c(1,2))
plot(x = ef$frequency, y = ef$strength, t = "b",
     xlab = "Frequency (Hz)",
     ylab = expression("Strength (" * mu * "V)"),
     main = "FFT of Noise-Free Data")

# compare to data generating parameters
cbind(amp, ef$strength[ef$strength > 0.25])
cbind(phs - pi/2, ef$phase[ef$strength > 0.25])


### FFT of Noisy Data ###

# fft of noisy data
ef <- eegfft(y, Fs = Fs, upper = 40)
head(ef)
ef[ef$strength > 0.25,]

# plot frequency strength
plot(x = ef$frequency, y = ef$strength, t = "b",
     xlab = "Frequency (Hz)",
     ylab = expression("Strength (" * mu * "V)"),
     main = "FFT of Noisy Data")

# compare to data generating parameters
cbind(amp, ef$strength[ef$strength > 0.25])
cbind(phs - pi/2, ef$phase[ef$strength > 0.25])
```

`eegfilter` *Filters EEG Data*

Description

Low-pass, high-pass, or band-pass filter EEG data using either a Butterworth filter (default) or a finite impulse response (FIR) filter.

Usage

```
eegfilter(x, Fs, lower, upper, method = "butter",
          order = 3L, forwardreverse = TRUE,
          scale = FALSE, plot = FALSE)
```

Arguments

x	Vector or matrix (time by channel) of EEG data with n time points.
Fs	Sampling rate of x in Hz.
lower	Lower band in Hz. Smallest frequency to keep.
upper	Upper band in Hz. Largest frequency to keep.
method	Filtering method. Either "butter" for a Butterworth filter or "fir1" for a FIR filter.
order	Order of the filter. See corresponding argument of butter or fir1.
forwardreverse	If TRUE (default), the data are forward and reverse filtered via filtfilt. Otherwise the data are (forward) filtered via filter.
scale	If FALSE (default), the filter is not normalized. Otherwise the magnitude of the center of the first passband is normalized to 1.
plot	If TRUE, the filter is plotted via freqz_plot.

Details

For a low-pass filter, only enter the upper frequency to keep. For a high-pass filter, only enter the lower frequency to keep. For a band-pass filter, enter both the lower and upper frequency bounds.

Value

Filtered version of input data.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

http://en.wikipedia.org/wiki/Butterworth_filter

http://en.wikipedia.org/wiki/Fir_filter

See Also

filter, filtfilt, butter, fir1

Examples

```
##########   EXAMPLE   ##########

# create data generating signals
n <- 1000                              # 1000 Hz signal
s <- 2                                 # 2 seconds of data
t <- seq(0, s, length.out = s * n)     # time vector
s1 <- sin(2*pi*t)                      # 1 Hz sinusoid
s5 <- sin(2*pi*t*5)                    # 5 Hz sinusoid
s10 <- sin(2*pi*t*10)                  # 10 Hz sinusoid
s20 <- sin(2*pi*t*20)                  # 20 Hz sinusoid

# create data
set.seed(1)                            # set random seed
e <- rnorm(s * n, sd = 0.25)           # Gaussian error
mu <- s1 + s5 + s10 + s20              # 1 + 5 + 10 + 20 Hz mean
y <- mu + e                            # data = mean + error

# 4-th order Butterworth filter (2 to 15 Hz band-pass)
yf.but <- eegfilter(y, Fs = n, lower = 2, upper = 15, method = "butter", order = 4)

# 350-th order FIR filter (2 to 15 Hz band-pass)
yf.fir <- eegfilter(y, Fs = n, lower = 2, upper = 15, method = "fir1", order = 350)

# check quality of results
yftrue <- s5 + s10                     # true (filtered) mean signal
mean((yf.but - yftrue)^2)              # mse between yf.but and yftrue
mean((yf.fir - yftrue)^2)              # mse between yf.fir and yftrue

# plot true and estimated filtered signals
plot(t, yftrue, type = "l", lty = 1, lwd = 2, ylim = c(-3, 3))
lines(t, yf.but, col = "blue", lty = 2, lwd = 2)
lines(t, yf.fir, col = "red", lty = 3, lwd = 2)
legend("topright", legend = c("Truth", "Butterworth", "FIR"),
       lty = 1:3, lwd = 2, col = c("black", "blue", "red"), bty = "n")

# power spectral density before and after filtering (dB)
par(mfrow=c(1,3), mar = c(5, 4.5, 4, 2) + 0.1)
eegpsd(y, Fs = n, upper = 50, t = "b",
       main = "Before Filtering", lwd = 2)
rect(2, -63, 15, 1, col = rgb(0.5,0.5,0.5,1/4))
legend("topright", legend = "2-15 Hz Filter",
       fill = rgb(0.5,0.5,0.5,1/4), bty = "n")
eegpsd(yf.but, Fs = n, upper = 50, t = "b",
       main = "After Butterworth Filter", lwd = 2)
eegpsd(yf.fir, Fs = n, upper = 50, t = "b",
       main = "After FIR Filter", lwd = 2)

# power spectral density before and after filtering (mv^2)
par(mfrow=c(1,3), mar = c(5, 4.5, 4, 2) + 0.1)
eegpsd(y, Fs = n, upper = 50, unit = "mV^2", t = "b",
       main = "Before Filtering", lwd = 2)
```

```
rect(2, 0, 15, 1.05, col = rgb(0.5,0.5,0.5,1/4))
legend("topright", legend = "2-15 Hz Filter",
       fill = rgb(0.5,0.5,0.5,1/4), bty = "n")
eegpsd(yf.but, Fs = n, upper = 50, unit = "mV^2", t = "b",
       main = "After Butterworth Filter", lwd = 2)
eegpsd(yf.fir, Fs = n, upper = 50, unit = "mV^2", t = "b",
       main = "After FIR Filter", lwd = 2)
```

---

eeghead *Dummy Head for 3d EEG Plots*

Description

Contains mesh3d object of dummy head, which is used in the plotting functions eegcap and eegspace. This is a transformed (translated, rotated, and rescaled) vesion of the dummyhead object from the Rvcg package.

Usage

```
data(eeghead)
```

Format

mesh3d object

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Source

Created by Nathaniel E. Helwig (2014) using:

Murdoch, D., and Adler, D. (2025). *rgl: 3D Visualization Using OpenGL*. doi:10.32614/CRAN.package.rgl, R package version 1.3.18, http://CRAN.R-project.org/package=rgl

Schlager S (2017). Morpho and Rvcg - Shape Analysis in R. In Zheng G, Li S, Szekely G (eds.), Statistical Shape and Deformation Analysis, 217-256. Academic Press. ISBN 9780128104934.

Examples

```
##########   EXAMPLE   ##########

# data(eeghead)
# shade3d(eeghead)
# eeghead$material$color <- rep("black",length(eeghead$material$color))
# wire3d(eeghead)
```

eegica *Independent Component Analysis of EEG Data*

Description

Computes temporal (default) or spatial ICA decomposition of EEG data. Can use Infomax (default), FastICA, or JADE algorithm. ICA computations are conducted via icaimax, icafast, or icajade from the ica package.

Usage

```
eegica(X, nc, center = TRUE, maxit = 100, tol = 1e-6,
       Rmat = diag(nc), type = c("time", "space"),
       method = c("imax", "fast", "jade"), ...)
```

Arguments

X	Data matrix with n rows (channels) and p columns (time points).
nc	Number of components to extract.
center	If TRUE, columns of X are mean-centered before ICA decomposition.
maxit	Maximum number of algorithm iterations to allow.
tol	Convergence tolerance.
Rmat	Initial estimate of the nc-by-nc orthogonal rotation matrix.
type	Type of ICA decomposition: type="time" extracts temporally independent components, and type="space" extracts spatially independent components.
method	Method for ICA decomposition: method="imax" uses Infomax, method="fast" uses FastICA, and method="jade" uses JADE.
...	Additional inputs to icaimax or icafast function.

Details

ICA Model The ICA model can be written as X = tcrossprod(S, M) + E, where columns of S contain the source signals, M is the mixing matrix, and columns of E contain the noise signals. Columns of X are assumed to have zero mean. The goal is to find the unmixing matrix W such that columns of S = tcrossprod(X, W) are independent as possible.

Whitening Without loss of generality, we can write M = P %*% R where P is a tall matrix and R is an orthogonal rotation matrix. Letting Q denote the pseudoinverse of P, we can whiten the data using Y = tcrossprod(X,Q). The goal is to find the orthongal rotation matrix R such that the source signal estimates S = Y %*% R are as independent as possible. Note that W = crossprod(R,Q).

Infomax The Infomax approach finds the orthogonal rotation matrix R that (approximately) maximizes the joint entropy of a nonlinear function of the estimated source signals. See Bell and Sejnowski (1995) and Helwig (in prep) for specifics of algorithms.

FastICA The FastICA algorithm finds the orthogonal rotation matrix `R` that (approximately) maximizes the negentropy of the estimated source signals. Negentropy is approximated using

$$J(s) = [E\{G(s)\} - E\{G(z)\}]^2$$

where *E* denotes the expectation, *G* is the contrast function, and z is a standard normal variable. See Hyvarinen (1999) for specifics of fixed-point algorithm.

JADE The JADE approach finds the orthogonal rotation matrix `R` that (approximately) diagonalizes the cumulant array of the source signals. See Cardoso and Souloumiac (1993,1996) and Helwig and Hong (2013) for specifics of the JADE algorithm.

Value

`S`	Matrix of source signal estimates (`S=Y%*%R`).
`M`	Estimated mixing matrix.
`W`	Estimated unmixing matrix (`W=crossprod(R,Q)`).
`Y`	Whitened data matrix.
`Q`	Whitening matrix.
`R`	Orthogonal rotation matrix.
`vafs`	Variance-accounted-for by each component.
`iter`	Number of algorithm iterations.
`type`	ICA type (same as input).
`method`	ICA method (same as input).

Note

If `type="time"`, the data matrix is transposed before calling ICA algorithm (i.e., `X = t(X)`), and the columns of the tranposed data matrix are centered.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Bell, A.J. & Sejnowski, T.J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation, 7*, 1129-1159. doi:10.1162/neco.1995.7.6.1129

Cardoso, J.F., & Souloumiac, A. (1993). Blind beamforming for non-Gaussian signals. *IEE Proceedings-F, 140*, 362-370. doi:10.1049/ipf2.1993.0054

Cardoso, J.F., & Souloumiac, A. (1996). Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications, 17*, 161-164. doi:10.1137/S0895479893259546

Helwig, N.E. (2022). *ica: Independent Component Analysis*, doi:10.32614/CRAN.package.ica, R package version 1.0-3. http://CRAN.R-project.org/package=ica

Helwig, N.E. & Hong, S. (2013). A critique of Tensor Probabilistic Independent Component Analysis: Implications and recommendations for multi-subject fMRI data analysis. *Journal of Neuroscience Methods, 213*, 263-273. doi:10.1016/j.jneumeth.2012.12.009

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks, 10*, 626-634. doi:10.1109/72.761722

Examples

```
##########   EXAMPLE   ##########

# get "c" subjects of "eegdata" data
data(eegdata)
idx <- which(eegdata$group=="c")
eegdata <- eegdata[idx,]

# get average data (across subjects)
eegmean <- tapply(eegdata$voltage,list(eegdata$channel,eegdata$time),mean)

# remove ears and nose
acnames <- rownames(eegmean)
idx <- c(which(acnames=="X"),which(acnames=="Y"),which(acnames=="nd"))
eegmean <- eegmean[-idx,]

# get spatial coordinates (for plotting)
data(eegcoord)
cidx <- match(rownames(eegmean),rownames(eegcoord))

# temporal ICA with 4 components
icatime <- eegica(eegmean,4)
icatime$vafs
# quartz()
# par(mfrow=c(4,2))
# tseq <- (0:255)*1000/255
# for(j in 1:4){
#   par(mar=c(5.1,4.6,4.1,2.1))
#   sptitle <- bquote("VAF:  "*.(round(icatime$vafs[j],4)))
#   eegtime(tseq,icatime$S[,j],main=bquote("Component  "*.(j)),cex.main=1.5)
#   eegspace(eegcoord[cidx,4:5],icatime$M[,j],main=sptitle)
# }

# spatial ICA with 4 components
icaspace <- eegica(eegmean,4,type="space")
icaspace$vafs
# quartz()
# par(mfrow=c(4,2))
# tseq <- (0:255)*1000/255
# for(j in 1:4){
#   par(mar=c(5.1,4.6,4.1,2.1))
#   sptitle <- bquote("VAF:  "*.(round(icaspace$vafs[j],4)))
#   eegtime(tseq,icaspace$M[,j],main=bquote("Component  "*.(j)),cex.main=1.5)
#   eegspace(eegcoord[cidx,4:5],icaspace$S[,j],main=sptitle)
# }
```

---

eegmesh *EEG Cap for Dense Coordinates*

Description

Contains mesh3d object of eegdense, which is used in the plotting function eegspace.

Usage

```
data(eegmesh)
```

Format

mesh3d object

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Source

Created by Nathaniel E. Helwig (2014) using:

Murdoch, D., and Adler, D. (2025). *rgl: 3D Visualization Using OpenGL*. doi:10.32614/CRAN.package.rgl, R package version 1.3.18, http://CRAN.R-project.org/package=rgl

Oostenveld, R., and Praamstra, P. (2001). The Five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology, 112*, 713-719. doi:10.1016/S13882457(00)00527-7

Schlager S (2017). Morpho and Rvcg - Shape Analysis in R. In Zheng G, Li S, Szekely G (eds.), Statistical Shape and Deformation Analysis, 217-256. Academic Press. ISBN 9780128104934.

Examples

```
##########   EXAMPLE   ##########

# data(eegmesh)
# wire3d(eegmesh)
# eegmesh$material$color <- rep("red",length(eegmesh$material$color))
# shade3d(eegmesh)
```

---

eegpsd *Plots Power Spectral Density of EEG Data*

Description

Uses a fast discrete Fourier transform (eegfft) to estimate the power spectral density of EEG data, and plots the power esimate using the plot (single channel) or imagebar (multi-channel) function.

Usage

```
eegpsd(x, Fs, lower, upper, units = "dB",
       xlab = NULL, ylab = NULL, zlab = NULL, ...)
```

Arguments

x	Vector or matrix (time by channel) of EEG data with n time points.
Fs	Sampling rate of x in Hz.
lower	Lower band in Hz. Smallest frequency to keep.
upper	Upper band in Hz. Largest frequency to keep.
units	Units for plot. Options include "dB" for decibals (default), "mV" for microvolts, and "mV^2" for squared microvolts. Note dB = 10*log10(mV^2).
xlab	x-axis label for the plot/image.
ylab	y-axis label for the plot/image.
zlab	z-axis label for the plot/image.
...	Optional inputs for the plot or imagebar function.

Value

Produces a plot (single channel) or image (multi-channel).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Cooley, James W., and Tukey, John W. (1965) An algorithm for the machine calculation of complex Fourier series, Math. Comput. 19(90), 297-301.

Singleton, R. C. (1979) Mixed Radix Fast Fourier Transforms, in Programs for Digital Signal Processing, IEEE Digital Signal Processing Committee eds. IEEE Press.

Examples

```
##########   EXAMPLE   ##########

# create data generating signals
n <- 1000                              # 1000 Hz signal
s <- 2                                 # 2 seconds of data
t <- seq(0, s, length.out = s * n)     # time vector
s1 <- sin(2*pi*t)                      # 1 Hz sinusoid
s5 <- sin(2*pi*t*5)                    # 5 Hz sinusoid
s10 <- sin(2*pi*t*10)                  # 10 Hz sinusoid
s20 <- sin(2*pi*t*20)                  # 20 Hz sinusoid

# create data
set.seed(1)                            # set random seed
e <- rnorm(s * n, sd = 0.25)           # Gaussian error
mu <- s1 + s5 + s10 + s20              # 1 + 5 + 10 + 20 Hz mean
y <- mu + e                            # data = mean + error

# plot psd (single channel)
eegpsd(y, Fs = n, upper = 30, t = "b")
```

```
# plot psd (multi-channel)
ym <- cbind(s1, s5, s10, s20)
eegpsd(ym, Fs = n, upper = 30, units = "mV")
```

`eegresample` *Change Sampling Rate of EEG Data*

Description

Turn a signal of length `N` into a signal of length `n` via linear interpolation.

Usage

```
eegresample(x, n)
```

Arguments

`x`	Vector or matrix (time by channel) of EEG data with `N` time points.
`n`	Number of time points for the resampled data.

Details

Data are resampled using the "Linear Length Normalization" approach described in Helwig et al. (2011). Let $\mathbf{x} = (x_1, \ldots, x_N)'$ denote the input vector of length N, and define a vector $\mathbf{t} = (t_1, \ldots, t_n)$ with entries

$$t_i = 1 + (i-1)\delta$$

for $i = 1, \ldots, n$ where $\delta = (N-1)/(n-1)$. The resampled vector is calculated as

$$y_i = x_{\lfloor t_i \rfloor} + (x_{\lceil t_i \rceil} - x_{\lfloor t_i \rfloor})(t_i - \lfloor t_i \rfloor)$$

for $i = 1, \ldots, n$ where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling functions.

Value

Resampled version of input data with `n` time points.

Note

Typical usage is to down-sample (i.e., decrease the sampling rate of) a signal: `n < N`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E., Hong, S., Hsiao-Wecksler E. T., & Polk, J. D. (2011). Methods to temporally align gait cycle data. Journal of Biomechanics, 44(3), 561-566. doi:10.1016/j.jbiomech.2010.09.015

Examples

```
##########   EXAMPLE 1   ##########

# create vector with N = 200 time points
N <- 200
x <- sin(4 * pi * seq(0, 1, length.out = N))

# down-sample (i.e., decrease sampling rate) to n = 100
y <- eegresample(x, n = 100)
mean((y - sin(4 * pi * seq(0, 1, length.out = 100)))^2)

# up-sample (i.e., increase sampling rate) to n = 500
z <- eegresample(x, n = 500)
mean((z - sin(4 * pi * seq(0, 1, length.out = 500)))^2)

# plot results
par(mfrow = c(1,3))
plot(x, main = "Original (N = 200)")
plot(y, main = "Down-sampled (n = 100)")
plot(z, main = "Up-sampled (n = 500)")


##########   EXAMPLE 2   ##########

# create matrix with N = 500 time points and 2 columns
N <- 500
x <- cbind(sin(2 * pi * seq(0, 1, length.out = N)),
           sin(4 * pi * seq(0, 1, length.out = N)))

# down-sample (i.e., decrease sampling rate) to n = 250
y <- eegresample(x, n = 250)
ytrue <- cbind(sin(2 * pi * seq(0, 1, length.out = 250)),
               sin(4 * pi * seq(0, 1, length.out = 250)))
mean((y - ytrue)^2)

# up-sample (i.e., increase sampling rate) to n = 1000
z <- eegresample(x, n = 1000)
ztrue <- cbind(sin(2 * pi * seq(0, 1, length.out = 1000)),
               sin(4 * pi * seq(0, 1, length.out = 1000)))
mean((z - ztrue)^2)

# plot results
par(mfrow = c(1,3))
plot(x[,1], main = "Original (N = 500)", cex = 0.5)
points(x[,2], pch = 2, col = "blue", cex = 0.5)
plot(y[,1], main = "Down-sampled (n = 250)", cex = 0.5)
points(y[,2], pch = 2, col = "blue", cex = 0.5)
```

```
plot(z[,1], main = "Up-sampled (n = 1000)", cex = 0.5)
points(z[,2], pch = 2, col = "blue", cex = 0.5)
```

`eegsim` *Simulate Event-Related Potential EEG Data*

Description

Simulates event-related potential EEG data from hypothetical visual-stimulus ERP study. Data are simulated using a linear combination of five spatiotemporal component functions: P100, N100, P200, N200, and P300 components. User can control the coefficient (weight) given to each component, as well as the time shift (delay) of each component.

Usage

```
eegsim(channel, time, coefs = rep(1,5), tshift = rep(0,5))
```

Arguments

`channel`	Character vector of length `n` giving EEG channel of simulated data.
`time`	Numeric vector of length `n` giving time point of simulated data (should be in interval [0,1]).
`coefs`	Numeric vector of length 5 giving the coefficients (weights) to use for P100, N100, P200, N200, and P300 components (respectively).
`tshift`	Numeric vector of length 5 giving the time shifts (delays) to use for P100, N100, P200, N200, and P300 components (respectively).

Value

Returns a vector of simulated EEG data corresponding to the input channel(s), time point(s), coefficients, and time shifts.

Note

Simulates data for 39 parietal and occipital electrodes: CP1 CP2 CP3 CP4 CP5 CP6 CPZ I1 I2 IZ O1 O2 OZ P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 POZ PZ TP7 TP8 TP9 TP10

Returns simulated value of 0 for other electrodes.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Created by Nathaniel E. Helwig (2014) using data from:

Begleiter, H. (1995). EEG Database [Dataset]. UCI Machine Learning Repository. doi:10.24432/C5TS3D

Examples

```
##########   EXAMPLE   ##########

### plot spatiotemporal component functions

# data(eegcoord)
# chnames <- rownames(eegcoord)
# tseq <- seq(0,1,length.out=200)

# quartz(width=18,height=6)
# layout(matrix(c(1,2,3,4,5,6,7,8,9,10,11,11), 2, 6, byrow = TRUE))

# eegspace(eegcoord[,4:5],p1s(chnames),cex.point=1,main=expression(psi[p1]),cex.main=2,vlim=c(-3,9))
# eegtime(tseq,p1t(tseq),ylim=c(-1,1),asp=1/2,main=expression(tau[p1]),cex.main=2,
#         xlab="Time After Stimulus (sec)")
# eegspace(eegcoord[,4:5],p2s(chnames),cex.point=1,main=expression(psi[p2]),cex.main=2,vlim=c(-3,9))
# eegtime(tseq,p2t(tseq),ylim=c(-1,1),asp=1/2,main=expression(tau[p2]),cex.main=2,
#         xlab="Time After Stimulus (sec)")
# eegspace(eegcoord[,4:5],p3s(chnames),cex.point=1,main=expression(psi[p3]),cex.main=2,vlim=c(-3,9))
# eegtime(tseq,p3t(tseq),ylim=c(-1,1),asp=1/2,main=expression(tau[p3]),cex.main=2,
#         xlab="Time After Stimulus (sec)")
# eegspace(eegcoord[,4:5],n1s(chnames),cex.point=1,main=expression(psi[n1]),cex.main=2,vlim=c(-3,9))
# eegtime(tseq,n1t(tseq),ylim=c(-1,1),asp=1/2,main=expression(tau[n1]),cex.main=2,
#         xlab="Time After Stimulus (sec)")
# eegspace(eegcoord[,4:5],n2s(chnames),cex.point=1,main=expression(psi[n2]),cex.main=2,vlim=c(-3,9))
# eegtime(tseq,n2t(tseq),ylim=c(-1,1),asp=1/2,main=expression(tau[n2]),cex.main=2,
#         xlab="Time After Stimulus (sec)")
# plot(seq(-10,10),seq(-10,10),type="n",axes=FALSE,xlab="",ylab="")
# text(0,8,labels=expression(omega[p1]*" = "*psi[p1]*tau[p1]),cex=2)
# text(0,4,labels=expression(omega[n1]*" = "*psi[n1]*tau[n1]),cex=2)
# text(0,0,labels=expression(omega[p2]*" = "*psi[p2]*tau[p2]),cex=2)
# text(0,-4,labels=expression(omega[n2]*" = "*psi[n2]*tau[n2]),cex=2)
# text(0,-8,labels=expression(omega[p3]*" = "*psi[p3]*tau[p3]),cex=2)



### plot simulated data at various time points

# quartz(width=15,height=3)
# tseq <- c(50,150,250,350,450)/1000
# par(mfrow=c(1,5))
# for(j in 1:5){
#   eegspace(eegcoord[,4:5],eegsim(chnames,rep(tseq[j],87)),vlim=c(-6.8,5.5),
#            main=paste(tseq[j]*1000," ms"),cex.main=2)
# }
```

eegsmooth *Spatial and/or Temporal Smoothing of EEG Data*

Description

Smooths single- or multi-channel electroencephalography (EEG) with respect to space and/or time. Uses the bigspline, bigtps, and bigssa functions (from bigsplines package) for smoothing.

Usage

```
eegsmooth(voltage, space = NULL, time = NULL, nknots = NULL,
          rparm = NULL, lambdas = NULL, skip.iter = TRUE,
          se.fit = FALSE, rseed = 1234)
```

Arguments

voltage	Vector of recorded EEG voltage at each row in space.
space	Matrix of electrode coordinates (in three-dimensions) at which EEG was recorded. If space=NULL, data are temporally smoothed only.
time	Vector of time points at which EEG was recorded. If time=NULL, data are spatially smoothed only.
nknots	Number of knots to sample for smoothing. Positive integer.
rparm	Rounding parameter(s) to use for smoothing. See Notes and Examples.
lambdas	Smoothing parameter(s) to use for smoothing.
skip.iter	If FALSE, iterative spatial-temporal smoothing is skipped. Ignored if space=NULL or time=NULL.
se.fit	If TRUE, standard errors of smoothed values are calculated.
rseed	Random seed to use for knot selection. Set rseed=NULL to obtain different knots each time, or set rseed to any positive integer to use a different random seed.

Value

For temporal smoothing only: an object of class "bigspline" (see bigspline).

For spatial smoothing only: an object of class "bigtps" (see bigtps).

For spatial-temporal smoothing: an object of class "bigssa" (see bigssa).

Note

For temporal smoothing only (i.e., space=NULL), the input rparm should be a positive scalar less than 1. Larger values produce faster (but less accurate) approximations. Default is 0.01, which I recommend for temporal smoothing; rparm=0.005 may be needed for particularly rough signals, and rparm=0.02 could work for smoother signals.

For spatial smoothing only (i.e., time=NULL), the input rparm should be a positive scalar giving the rounding unit for the spatial coordinates. For example, rparm=0.1 rounds each coordinate to the nearest 0.1 (same as round(space,1)).

For spatial-temporal smoothing (i.e., both `space` and `time` are non-null), the input `rparm` should be a list of the form `rparm=list(space=0.1,time=0.01)`, where the 0.1 and 0.01 can be replaced by your desired rounding parameters.

Setting `rparm=NA` will use the full data solution; this is more computationally expensive, and typically produces a solution very similar to using `rparm=0.01` (see references).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign. https://www.ideals.illinois.edu/items/44427

Helwig, N.E. (2015). *bigsplines: Smoothing Splines for Large Samples*. doi:10.32614/CRAN.package.bigsplines, R package version 1.1-1, https://CRAN.R-project.org/package=bigsplines

Helwig, N. E. & Ma, P. (2015). Fast and stable multiple smoothing parameter selection in smoothing spline analysis of variance models with large samples. *Journal of Computational and Graphical Statistics, 24*(3), 715-732. doi:10.1080/10618600.2014.926819

Helwig, N. E. & Ma, P. (2016). Smoothing spline ANOVA for super large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*(4), 433-444. doi:10.4310/SII.2016.v9.n4.a3

Examples

```
##########   EXAMPLE 1: Temporal    ##########

# get "PZ" electrode of "c" subjects in "eegdata" data
data(eegdata)
idx <- which(eegdata$channel=="PZ" & eegdata$group=="c")
eegdata <- eegdata[idx,]

# temporal smoothing
eegmod <- eegsmooth(eegdata$voltage,time=eegdata$time)

# define data for prediction
time <- seq(min(eegdata$time),max(eegdata$time),length.out=100)
yhat <- predict(eegmod,newdata=time,se.fit=TRUE)

# plot results using eegtime
eegtime(time*1000/255,yhat$fit,voltageSE=yhat$se.fit,ylim=c(-4,4),main="Pz")



##########   EXAMPLE 2: Spatial   ##########

# get time point 65 (approx 250 ms) of "c" subjects in "eegdata" data
data(eegdata)
idx <- which(eegdata$time==65L & eegdata$group=="c")
```

```
eegdata <- eegdata[idx,]

# remove ears, nose, and reference (Cz)
idx <- c(which(eegdata$channel=="X"),which(eegdata$channel=="Y"),
         which(eegdata$channel=="nd"),which(eegdata$channel=="Cz"))
eegdata <- eegdata[-idx,]

# match to eeg coordinates
data(eegcoord)
cidx <- match(eegdata$channel,rownames(eegcoord))

# spatial smoothing
eegmod <- eegsmooth(eegdata$voltage,space=eegcoord[cidx,1:3])

# use dense cap for prediction
mycap <- levels(factor(eegdata$channel))
ix <- eegcapdense(mycap,type="2d",index=TRUE)
data(eegdense)
space <- eegdense[ix,1:3]
yhat <- predict(eegmod,newdata=space)

# plot results using eegspace
#eegspace(space,yhat)
eegspace(eegdense[ix,4:5],yhat)



##########   EXAMPLE 3: Spatial-Temporal (not run)   ##########

# # get "c" subjects of "eegdata" data
# data(eegdata)
# idx <- which(eegdata$group=="c")
# eegdata <- eegdata[idx,]

# # remove ears, nose, and reference (Cz)
# idx <- c(which(eegdata$channel=="X"),which(eegdata$channel=="Y"),
#          which(eegdata$channel=="nd"),which(eegdata$channel=="Cz"))
# eegdata <- eegdata[-idx,]

# # match to eeg coordinates
# data(eegcoord)
# cidx <- match(eegdata$channel,rownames(eegcoord))

# # spatial-temporal smoothing
# eegmod <- eegsmooth(eegdata$voltage,space=eegcoord[cidx,1:3],time=eegdata$time)

# # time main effect
# newdata <- list(time=seq(min(eegdata$time),max(eegdata$time),length.out=100))
# yhat <- predict(eegmod,newdata=newdata,se.fit=TRUE,include="time")
# eegtime(newdata$time,yhat$fit,voltageSE=yhat$se.fit,ylim=c(-2,4),main="Time Main Effect")

# # space main effect
# mycap <- levels(factor(eegdata$channel))
```

```
# ix <- eegcapdense(mycap,type="2d",index=TRUE)
# data(eegdense)
# newdata <- list(space=eegdense[ix,1:3])
# yhat <- predict(eegmod,newdata=newdata,include="space")
# eegspace(newdata$space,yhat)

# # interaction effect (spatial map at time point 65)
# newdata <- list(space=eegdense[ix,1:3],time=rep(65,nrow(eegdense[ix,])))
# yhat <- predict(eegmod,newdata=newdata,include="space:time")
# eegspace(newdata$space,yhat)

# # full prediction (spatial map at time point 65)
# newdata <- list(space=eegdense[ix,1:3],time=rep(65,nrow(eegdense[ix,])))
# yhat <- predict(eegmod,newdata=newdata)
# eegspace(newdata$space,yhat)
```

---

eegspace *Plots Multi-Channel EEG Spatial Map*

Description

Creates plot of multi-channel electroencephalography (EEG) spatial map. User can control the plot type (2d or 3d), the colormap, color, etc.

Usage

```
eegspace(space, voltage, vlim = NULL, mycolors = NULL, ncolor = 25,
         colorbar = TRUE, nctick = 5, rtick = 1, cex.axis = 1,
         barloc = NULL, colorlab = NULL, colorlabline = 3, cex.lab = 1,
         plotaxes = FALSE, main = "", xyzlab = NULL, cex.point = 1,
         cex.main = 1, nose = TRUE, ears = TRUE, head = TRUE,
         col.head = "AntiqueWhite", mar = NULL, ...)
```

Arguments

space	Matrix of input electrode coordinates (3d or 2d).
voltage	Vector of recorded EEG voltage at each row in space.
vlim	Two-element vector giving the limits to use when mapping voltage to colors in mycolors. Default is vlim=range(voltage).
mycolors	Character vector of colors to use for color mapping (such that length(mycolors)<=ncolor). Default: mycolors=c("blueviolet","blue","cyan","green","yellow","orange","red").
ncolor	Number of colors to use in mapping (positive integer).
colorbar	If TRUE, colorbar is plotted.
nctick	Approximate number of ticks for colorbar. Ignored if colorbar=FALSE.
rtick	Round tick labels to given decimal. Ignored if colorbar=FALSE.

cex.axis	Cex of axis ticks for colorbar. Ignored if colorbar=FALSE.
barloc	Character vector giving location of color bar. See Notes.
colorlab	Character vector giving label for color bar. Ignored if colorbar=FALSE.
colorlabline	Line number for color bar label (for input to mtext).
cex.lab	Cex of axis labels for colorbar. Ignored if colorbar=FALSE.
plotaxes	If TRUE, axes labels are plotted. Ignored for 3d plots.
main	Plot title. Default is no title.
xyzlab	Axis labels to use for plot. If type="2d", then xyzlab should be two-element character vector giving x and y axis labels. If type="3d", then xyzlab should be three-element character vector giving x, y, and z axis labels.
cex.point	Cex for plotted electrodes.
cex.main	Cex for plot title. Ignored if main=NULL.
nose	If TRUE, triangle is plotted to represent the subject's nose. Ignored if ncol(space)==3.
ears	If TRUE, ovals are plotted to represent the subject's ears. Ignored if ncol(space)==3.
head	If TRUE, head is plotted. Ignored if type="2d".
col.head	Color for dummy head in 3d plot. Ignored if type="2d".
mar	Margins to use for plot (see par).
...	Optional inputs for plot or lines function.

Value

Produces plot of EEG spatial map with NULL return value.

Note

For 3d plots, barloc can be one of four options: "backright", "backleft", "frontright", or "frontleft". For 2d plots, barloc can be either "right" or "left".

Currently supports spatial maps registered to the 84-channel cap produced by eegcap and eegcoord.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Begleiter, H. (1995). EEG Database [Dataset]. UCI Machine Learning Repository. doi:10.24432/C5TS3D

Examples

```
##########   EXAMPLE   ##########

# get time point 65 (approx 250 ms) from "eegdata" data
data(eegdata)
idx <- which(eegdata$time==65L)
eegdata <- eegdata[idx,]

# get average spatial map
eegmean <- tapply(eegdata$voltage,list(eegdata$channel,eegdata$group),mean)

# remove ears and nose
acnames <- rownames(eegmean)
idx <- c(which(acnames=="X"),which(acnames=="Y"),which(acnames=="nd"),which(acnames=="Cz"))
eegmean <- eegmean[-idx,]

# match to eeg coordinates
data(eegcoord)
cidx <- match(rownames(eegmean),rownames(eegcoord))

# # plot average control voltage in 3d
# open3d()
# eegspace(eegcoord[cidx,1:3],eegmean[,2])

# plot average control voltage in 2d
eegspace(eegcoord[cidx,4:5],eegmean[,2])

# # change 3d bar location and use play3d to rotate (not run)
# open3d()
# par3d(windowRect=c(0,0,600,600))
# eegspace(eegcoord[cidx,1:3],eegmean[,2],barloc="frontleft")
# play3d(spin3d(axis=c(0,0,1),rpm=5),duration=20)

# change 2d bar location
eegspace(eegcoord[cidx,4:5],eegmean[,2],barloc="left")
```

`eegtime` *Plots Single-Channel EEG Time Course*

Description

Creates plot of single-channel electroencephalography (EEG) time course with optional confidence interval. User can control the plot orientation, line types, line colors, etc.

Usage

```
eegtime(time, voltage, flipvoltage = TRUE, vlty = 1, vlwd = 2,
        vcol = "blue", voltageSE = NULL, slty = NA, slwd = 1,
```

```
scol = "cyan", salpha = 0.65, conflevel = 0.95,
plotzero = TRUE, zlty = 1, zlwd = 0.5, zcol = "black",
xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,
nxtick = 6, nytick = 6, xticks = NULL, yticks = NULL,
add = FALSE, ...)
```

Arguments

time	Vector of time points at which EEG was recorded.
voltage	Vector of recorded EEG voltage at each point in time.
flipvoltage	If TRUE, negative voltages are plotted upwards.
vlty	Line type for voltage.
vlwd	Line width for voltage.
vcol	Line color for voltage.
voltageSE	Vector of standard errors of EEG voltage at each point in time.
slty	Line type for voltageSE. If slty=NA (default) shaded polygons are plotted.
slwd	Line width for voltageSE. Ignored if slty=NA.
scol	Polygon or line color for voltageSE.
salpha	Transparency value for voltageSE polygon (only used if slty=NA).
conflevel	Confidence level to use for confidence intervals. Default forms 95% CI.
plotzero	If TRUE, horizontal reference line is plotted at 0 volts.
zlty	Line type for reference line. Ignored if plotzero=FALSE.
zlwd	Line width for reference line. Ignored if plotzero=FALSE.
zcol	Line color for reference line. Ignored if plotzero=FALSE.
xlim	Plot limits for time.
ylim	Plot limits for voltage.
xlab	Plot label for time.
ylab	Plot label for voltage.
nxtick	Approximate number of axis ticks for time.
nytick	Approximate number of axis ticks voltage.
xticks	x-axis ticks for time (overrides nxtick).
yticks	y-axis ticks voltage (overrides nytick).
add	If TRUE, lines are added to current plot; otherwise a new plot is created.
...	Optional inputs for plot or lines function.

Value

Produces plot of EEG time course with NULL return value.

Note

Confidence intervals are formed using the normal (Gaussian) distribution.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Begleiter, H. (1995). EEG Database [Dataset]. UCI Machine Learning Repository. doi:10.24432/C5TS3D

Examples

```
##########   EXAMPLE   ##########

# get "PZ" electrode from "eegdata" data
data(eegdata)
idx <- which(eegdata$channel=="PZ")
eegdata <- eegdata[idx,]

# get average and standard error (note se=sd/sqrt(n))
eegmean <- tapply(eegdata$voltage,list(eegdata$time,eegdata$group),mean)
eegse <- tapply(eegdata$voltage,list(eegdata$time,eegdata$group),sd)/sqrt(50)

# plot results with legend
tseq <- seq(0,1000,length.out=256)
eegtime(tseq,eegmean[,2],voltageSE=eegse[,2],ylim=c(-10,6),main="Pz")
eegtime(tseq,eegmean[,1],vlty=2,vcol="red",voltageSE=eegse[,1],scol="pink",add=TRUE)
legend("bottomright",c("controls","alcoholics"),lty=c(1,2),
       lwd=c(2,2),col=c("blue","red"),bty="n")
```

---

eegtimemc *Plots Multi-Channel EEG Time Course*

Description

Creates plot of multi-channel electroencephalography (EEG) time courses with subplots positioned according to electrode locations. User can control the plot orientation, line types, line colors, etc.

Usage

```
eegtimemc(time, voltmat, channel, size = c(0.75,0.75),
          vadj = 0.5, hadj = 0.5, xlab = "", ylab = "",
          voltSE = NULL, vlty = 1, slty = NA, vlwd = 1,
          slwd = 1, vcol = "blue", scol = "cyan", ...)
```

Arguments

time	Vector of time points at which EEG was recorded.
voltmat	Matrix of multi-channel EEG voltages (time by channel).
channel	Character vector giving name of channel for each column of voltmat.
size	Relative size of each subplot.
vadj	Vertical adjustment for each subplot.
hadj	Horizontal adjustment for each subplot.
xlab	X-axis label for each subplot.
ylab	Y-axis label for each subplot.
voltSE	Matrix of voltage standard errors (same size as voltmat).
vlty	Line type for voltmat.
slty	Line type for voltSE. If slty=NA (default) shaded polygons are plotted.
vlwd	Line width for voltmat.
slwd	Line width for voltSE. Ignored if slty=NA.
vcol	Line color for voltmat.
scol	Polygon or line color for voltSE.
...	Optional inputs for eegtime function.

Value

Produces plot of EEG time course with NULL return value.

Note

Currently supports 84 scalp electrodes (plus ears and nose): A1 A2 AF1 AF2 AF3 AF4 AF5 AF6 AF7 AF8 AFZ C1 C2 C3 C4 C5 C6 CP1 CP2 CP3 CP4 CP5 CP6 CPZ CZ F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 FC1 FC2 FC3 FC4 FC5 FC6 FCZ FP1 FP2 FPZ FT7 FT8 FT9 FT10 FZ I1 I2 IZ NZ O1 O2 OZ P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 PO1 PO2 PO3 PO4 PO5 PO6 PO7 PO8 PO9 PO10 POZ PZ T7 T8 T9 T10 TP7 TP8 TP9 TP10

Subplots are created using eegtime, so input ... can be any optional input for eegtime.

Inspired by Frank Harrell's subplot function (in Hmisc package).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Begleiter, H. (1995). EEG Database [Dataset]. UCI Machine Learning Repository. doi:10.24432/C5TS3D

Harrell Jr F (2025). Hmisc: Harrell Miscellaneous. doi:10.32614/CRAN.package.Hmisc, R package version 5.2-3, https://CRAN.R-project.org/package=Hmisc

Examples

```
##########   EXAMPLE   ##########

# # get control ("c") data from "eegdata" data
# data(eegdata)
# idx <- which(eegdata$group=="c")
# eegdata <- eegdata[idx,]

# # get average
# eegmean <- tapply(eegdata$voltage,list(eegdata$time,eegdata$channel),mean)
# eegse <- tapply(eegdata$voltage,list(eegdata$time,eegdata$channel),sd)/sqrt(50)

# # plot time course for all electrodes
# dev.new(height=15,width=15, noRStudioGD = TRUE)
# tseq <- seq(0,1000,length.out=256)
# eegtimemc(tseq,eegmean,colnames(eegmean),ylim=c(-11,14),voltSE=eegse)
```

Index