

Package ‘emu’

February 14, 2012

Version 4.2.1

Date 2009-11-23

Priority contrib

Title Interface to the Emu Speech Database System

Author Jonathan Harrington and others, IPS LMU Muenchen & IPDS CAU Kiel

Description Provides an interface to the Emu speech database system
and many special purpose functions for display and analysis of speech data.

License GPL (>= 2)

Depends R (>= 2.10), stats, methods, tcltk, MASS

LazyLoad yes

LazyData yes

LazyDataCompression xz

URL <http://emu.sourceforge.net/>

Maintainer Jonathan Harrington <jmh@phonetik.uni-muenchen.de>

Repository CRAN

Date/Publication 2011-02-11 10:05:49

R topics documented:

as.spectral	5
as.trackdata	7
bark	8
bind.trackdata	10
bridge	11
buildtrack	11
by.trackdata	12

cbind.trackdata	13
classify	15
classplot	16
coutts	17
coutts.epg	17
coutts.l	18
coutts.rms	18
coutts.sam	18
coutts2	18
coutts2.epg	19
coutts2.l	19
coutts2.sam	19
cr	20
crplot	22
dapply	23
dbnorm	25
dbtopower	26
dct	27
dcut	28
ddiff	30
demo.all	31
demo.all.rms	31
demo.vowels	32
demo.vowels.f0	32
demo.vowels.fm	33
dextract	33
dim.trackdata	34
dimnames.trackdata	35
dip	36
dip.fdat	36
dip.l	36
dip.spkr	36
dorfric	37
dorfric.dft	37
dorfric.l	37
dorfric.lv	37
dorfric.vl	38
dorfric.w	38
dorsal	38
dorsal.bound	39
dorsal.clab	39
dorsal.epg	39
dorsal.fm	39
dorsal.sam	40
dorsal.vlab	40
dplot	40
dsmooth	43
dur	43

e.dft	44
ellipse	44
emu.defaultpaths	45
emu.directory	45
emu.query	45
emu.requery	47
emu.track	49
emu.variables	50
emulink	51
engassim	52
engassim.epg	52
engassim.l	52
engassim.w	53
EPG contact indices	53
epgcog	54
epggs	56
epgplot	57
epgsum	59
eplot	61
euclidean	63
expand_labels	63
f2geraus	64
f2geraus.l	64
fapply	65
fr	66
fr.dft	66
fr.l	67
fr.sp	67
frames	67
frames.time	68
fretoint	68
fric	69
fric.dft	70
fric.l	70
fric.w	70
get.time.element	71
is.spectral	71
is.trackdata	72
isol	72
isol.fdat	73
isol.l	73
keng	73
keng.dft.5	73
keng.l	74
label	74
linear	75
locus	76
mahal	77

mahal.dist	78
make.seglist	79
makelab	80
matscan	81
mel	82
modify.seglist	84
moments	85
mu.colour	86
muclass	88
norm	89
palate	90
perform	91
plafit	91
plos	92
plos.asp	93
plos.dft	93
plos.l	93
plos.lv	93
plos.sam	94
plos.w	94
plot.spectral	94
plot.trackdata	96
polhom	98
polhom.epg	99
polhom.l	99
rad	99
radians	100
randomise.segs	100
rbind.trackdata	101
read.emusegs	102
read.trackdata	104
segmentlist	105
shift	105
sib	106
sib.dft	106
sib.l	107
sib.w	107
Slope.test	107
sortmatrix	108
splitstring	109
Start and end times for segment lists and trackdata objects	109
stops	110
stops.bark	111
stops.dct	111
stops.dft	111
stops.l	112
stops.sp	112
stops10	112

stops10.lab	112
stopsvow	113
stopsvow.fm	113
stopsvow.l	113
timevow	113
timevow.dft	114
timevow.fm	114
timevow.fm5	114
timevow.l	114
track.gradinfo	115
trackdata	116
trackfreq	117
trackinfo	118
tracktimes	119
train	120
trapply	121
v.sam	122
vowlax	122
vowlax.df	123
vowlax.dft.5	123
vowlax.fdat	123
vowlax.fdat.5	123
vowlax.fund	124
vowlax.fund.5	124
vowlax.l	124
vowlax.left	124
vowlax.right	125
vowlax.rms	125
vowlax.rms.5	125
vowlax.spkr	125
vowlax.word	126
wordlax.l	126
write.emusegs	126
write.trackdata	127
write.trackdata.get	128

Index**129**

as.spectral

*Function to convert an object into an object of class 'spectral'.***Description**

The function converts a vector, matrix, or EMU-trackdata object into an object of the same class and of class 'spectral'

Usage

```
as.spectral(trackdata, fs)
```

Arguments

trackdata	A vector, matrix, or EMU-trackdata object.
fs	Either a single element numeric vector, or a numeric vector of the same length as the length of trackdata if trackdata is a vector, or of the same number of rows as trackdata

Details

If fs is a single element numeric vector, then the frequencies of trackdata are defined to extend to fs/2. If fs is missing, then the frequencies are 0:(N-1) where N is the length of trackdata.

Value

The same object but of class 'spectral'.

Author(s)

Jonathan Harrington

See Also

[is.spectral](#) [plot.spectral](#)

Examples

```
vec = 1:10
as.spectral(vec, 2000)
mat = rbind(1:10, 1:10)
as.spectral(mat)
# turn a spectral trackdata object into a trackdata object
tr = as.trackdata(rbind(fric.dft$data), fric.dft$index, fric.dft$time)
# turn it into a spectral trackdata object with sampling freq 16 kHz
tr = as.spectral(tr, 16000)
# list the frequencies
trackfreq(tr)
# Notice that only the $data is made into a spectral matrix,
# not the entire trackdata object
# so this is trackdata
class(tr)
# this is a spectral matrix
class(tr$data)
```

as.trackdata	<i>Create an Emu trackdata object</i>
--------------	---------------------------------------

Description

Create an Emu trackdata object from a raw data matrix.

Usage

```
as.trackdata(data, index, ftime, trackname="")
```

Arguments

data	A two dimensional matrix of numerical data.
index	Segment index, one row per segment, two columns give the start and end rows in the data matrix for each segment.
ftime	A two column matrix with one row per segment, gives the start and end times in milliseconds for each segment.
trackname	The name of the track.

Details

Emu trackdata objects contain possibly multi-column numerical data corresponding to a set of segments from a database. Data for each segment takes up a number of rows in the main data matrix, the start and end rows are stored in the `index` component. The `ftime` component contains the start and end times of the segment data.

Trackdata objects are returned by the [emu.track](#) function.

Value

The components are bound into a trackdata object.

See Also

[emu.track dplot](#)

Examples

```
# make a trackdata object of two data segments
data1 <- matrix( 1:10, ncol=2 )
data2 <- matrix( 11:20, ncol=2 )

nd1 <- nrow(data1)
nd2 <- nrow(data2)
index <- rbind( c( 1, nd1 ), c(nd1+1,nd1+nd2) )
```

```

times <- rbind( c( 100.0, 110.0 ), c( 200.0, 210.0 ) )

tdata <- as.trackdata( rbind( data1, data2 ), index, times, trackname="fake")

# describe the data
summary(tdata)
# get the data for the first segment
tdata[1]
# and the second
tdata[2]

```

bark

Convert Hertz to Bark and Bark to Hertz

Description

The calculation is done using the formulae Traunmueller (1990)

Usage

```

## S3 method for class 'spectral'
bark(f,...)
## Default S3 method:
bark(f,inv=FALSE,...)

```

Arguments

f	A vector or matrix of data or a spectral object.
inv	A single element logical vector. If F, data are converted from Hertz to Bark, if T, data are converted from Bark to Hertz. (Does not apply if 'data' is an object of class 'spectral'.
...	for generic only

Details

If 'data' is a spectral object, then the frequencies are changed so that they are proportional to the Bark scale and such that the Bark intervals between frequencies are constant between the lowest and highest frequencies. More specifically, suppose that a spectral object has frequencies at 0, 1000, 2000, 3000, 4000 Hz. Then the corresponding frequencies extend in Bark between 0 and 17.46329 Bark

in four equal intervals, and linear interpolation is used with the 'approx' function to obtain the dB values at those frequencies. Negative frequencies which are obtained for values of about less than 40 Hz are removed in the case of spectral objects.

Value

A vector or matrix or spectral object of the same length and dimensions as data.

Author(s)

Jonathan Harrington

References

Traunmueller, H. (1990) "Analytical expressions for the tonotopic sensory scale" J. Acoust. Soc. Am. 88: 97-100.

See Also

[mel](#),
[plot.spectral](#)

Examples

```
# convert Hertz values to Bark
vec <- c(500, 1500, 2500)
vec
bark(vec)

# convert Hertz values to Bark and back to Hertz
bark(bark(vec, inv=TRUE))

# convert the $data values in a trackdata object to Bark
```

```
# create a new track data object

t1 <- dip.fdat

t1[1]

# convert Hertz to Bark

t1$data <- bark(t1$data)

t1[1]

# warp the frequency axis of a spectral object such
# that it is proportional to the Bark scale.

w = bark(e.dft)

par(mfrow=c(1,2))

plot(w, type="l")

# The values of w are at equal Bark intervals. Compare
# with

plot(e.dft, freq=bark(trackfreq(e.dft)))

# the latter has a greater concentration of values
# in a higher frequency range.
```

`bind.trackdata`

bind trackdata

Description

binds different trackdata objects together

Usage

```
## S3 method for class 'trackdata'
bind(...)
```

Arguments

```
...          trackdata objects
```

bridge	<i>Three-columned matrix</i>
--------	------------------------------

Description

An EMU dataset

Usage

```
bridge
```

buildtrack	<i>Build trackdata objects from the output of by()</i>
------------	--

Description

buildtrack() converts a list that is the output of by.trackdata() into a trackdata object if the list components are matrices whose rows are successive values in time.

Usage

```
buildtrack(mylist, ftime = NULL, trackname = "")
```

Arguments

```
mylist      a list that ist output from by()
ftime       ftime
trackname   name of track data object
```

Details

The default of by.trackdata() is to return a list. If each element of the list consists of a matrix whose rows are values occurring at the times given by the row dimension names of the matrix, then buildtrack() can be used to convert the list into a trackdata object. If the times are not given in the row dimension names, then these can be supplied as an additional argument to buildtrack()

Author(s)

Jonathan Harrington

See Also[by](#)**Examples**

```
#vowlax.fdat is a track data objects of formant of the vowlax segment list
#calculate the difference between adjacent formant values
p = by(vowlax.fdat[1,2],INDICES=NULL, diff)

p

#now build a track data object out of these values
m = buildtrack(p)

m
```

`by.trackdata`*A method of the generic function `by` for objects of class `'trackdata'`*

Description

A given function 'FUN' is applied to the data corresponding to each segment of data.

Usage

```
## S3 method for class 'trackdata'
by(data, INDICES=NULL,FUN, ..., simplify = FALSE
)
```

Arguments

<code>data</code>	a track data object
<code>INDICES</code>	a list of segment indices, like a label vector
<code>FUN</code>	a function that is applied to each segment
<code>...</code>	arguments of the function fun
<code>simplify</code>	<code>simplify = TRUE</code> , output is a matrix; <code>simplify = FALSE</code> a list is returned

Details

It is the same as `trapply` but with the extension to subsume calculation to groups of segments. Note, if you do not want to apply the function `fun` to a special group of segments, use `trapply` instead.

Value

list or vector

Author(s)

Jonathan Harrington

See Also

[trapply](#), [by](#), [trackdata](#) [daply](#) [smooth](#) [apply](#)

Examples

```
data(demo.vowels)
data(demo.vowels.fm)

#mean F1 subsumed for each vowel
#####
lab = label(demo.vowels)
by(demo.vowels.fm[,1], lab ,mean ,simplify=FALSE)

#mean F1 subsumed for segment onsets mids and offsets
#####
data = demo.vowels.fm
llabs = NULL
for (ind in 1:dim(data$time)[1]) {
  seglabs = rep("mid",data$index[ind,2]-data$index[ind,1]+1)
  seglabs[1] = "on"
  seglabs[length(seglabs)] = "off"
  llabs = as.vector(c(llabs , seglabs))
}

by(demo.vowels.fm[,1], llabs , mean , simplify=FALSE)

#mean F1 subsumed for segment onsets mids and offsets subsumed for each vowel
#####
by(demo.vowels.fm[,1], list(lab = lab, llabs = llabs) , mean , simplify=FALSE)
```

cbind.trackdata

A method of the generic function cbind for objects of class 'trackdata'

Description

Different track data objects from one segment list are bound by combining the `\$data` columns of the track data object by columns.

Usage

```
## S3 method for class 'trackdata'  
cbind(...)
```

Arguments

```
...          track data objects
```

Details

All track data objects have to be track data of the same segment list. Thus `\$index` and `\$ftime` values have to be identically for all track data objects. Track data objects are created by `emu.track()`. The number of rows of the track data objects must match.

Value

A track data object with the same `\$index` and `\$ftime` values of the source track data objects and with `\$data` that includes all columns of `\$data` of the source track data objects.

Author(s)

Jonathan Harrington

See Also

[cbind](#), [rbind.trackdata](#) [trackdata](#) [emu.track](#)

Examples

```
data(vowlax)  
  
#segment list vowlax - first segment only  
vowlax[1,]  
  
#F0 track data object for vowlax - first segment only  
vowlax.fund[1,]  
  
#rms track data object for vowlax - first segment only  
vowlax.rms[1,]  
  
#now combine both track data objects  
fund.rms.lax = cbind(vowlax.fund, vowlax.rms)  
  
#the combined track data object - first segment only  
#The first column keeps vowlax.fund data, the second keeps vowlax.rms data  
fund.rms.lax[1,]
```

classify	<i>classify</i>
----------	-----------------

Description

classifies data

Usage

```
classify(data, train, metric = "bayes")
```

Arguments

data	data to classify
train	training data
metric	bayes or mahal

Value

The classification matrix.

Author(s)

Jonathan Harrington

Examples

```
## The function is currently defined as
function (data, train, metric = "bayes")
{
  probs <- distance(data, train, metric = metric)
  if (metric == "bayes") {
    best <- apply(probs, 1, max)
  }
  else if (metric == "mahal") {
    best <- apply(probs, 1, min)
  }
  result <- rep("", length(best))
  for (lab in 1:length(train$label)) {
    tmp <- probs[, lab] == best
    result[tmp] <- train$label[lab]
  }
  result
}
```

classplot

*Produce a classification plot from discriminant or SVM modelling***Description**

The function classifies all point specified within the ranges of xlim and ylim based on the training model specified in model. It then produces a two-dimensional plot colour-coded for classifications.

Usage

```
classplot(model, xlim, ylim, N = 100, pch = 15, col = NULL, legend = TRUE, position = "topright", b
```

Arguments

model	A two-dimensional training model output from qda(), lda() of MASS package , or svm() of e1071 package
xlim	A vector of two numeric elements specifying the range on the x-axis (parameter 1) over which classifications should be made
ylim	A vector of two elements specifying the range on the y-axis (parameter 2) over which classifications should be made
N	A vector of one numeric element which specifies the density of classification (greater N gives higher density). The default is 100.
pch	A single element numeric vector specifying the plotting symbol to be used in the classification plot. Defaults to 15.
col	Either Null in which case the colours for the separate classes are col = c(1, 2, ...n) where n is the number of classes; or else a vector specifying the desired colours that is the same length as there are classes.
legend	A single element logical vector specifying whether a legend should be drawn. Defaults to T
position	A single element vector specifying the position in the figure where the legend should be drawn. Defaults to "topright"
bg	A single element vector specifying the background colour on which the legend should be drawn.
...	Further arguments to plot.

Author(s)

Jonathan Harrington

See Also

[qda](#), [lda](#), [svm](#) of e1071 package. There is a function [plot.svm](#) which produces a prettier plot for SVMs.

Examples

```

library(MASS)
# Data from female speaker 68
temp = vowlax.spkr=="68"
# Quadratic discriminant analysis
fm.qda = qda(vowlax.fdat.5[temp,1:2], vowlax.l[temp])
# Linear discriminant analysis
fm.lda = lda(vowlax.fdat.5[temp,1:2], vowlax.l[temp])

xlim=c(0,1000)
ylim=c(0,3000)

par(mfrow=c(1,2))
classplot(fm.qda, xlim=xlim, ylim=ylim, main="QDA")
classplot(fm.lda, xlim=xlim, ylim=ylim, main="LDA")

# install.packages("e1071")
# library(e1071)
# Support vector machine
## Not run: fm.svm = svm(vowlax.fdat.5[temp,1:2], factor(vowlax.l[temp]))
## Not run: xlim = range(vowlax.fdat.5[temp,1])
## Not run: ylim = range(vowlax.fdat.5[temp,2])
## Not run: classplot(fm.svm, xlim=xlim, ylim=ylim, xlab="F1", ylab="F2", main="SVM")

```

coutts	<i>Segment list of words, read speech, female speaker of Australian English from database epgcoutts</i>
--------	---

Description

An EMU dataset

Usage

coutts

coutts.epg	<i>EPG-compressed trackdata from the segment list coutts</i>
------------	--

Description

An EMU dataset

Usage

coutts.epg

coutts.l	<i>Vector of word label from the segment list coutts</i>
----------	--

Description

An EMU dataset

Usage

coutts.l

coutts.rms	<i>rms Data to coutts segment list</i>
------------	--

Description

An EMU dataset

Examples

data(coutts.rms)

coutts.sam	<i>Trackdata of acoustic waveforms from the segment list coutts</i>
------------	---

Description

An EMU dataset

Usage

coutts.sam

coutts2	<i>Segment list, same as coutts but at a slower speech rate</i>
---------	---

Description

An EMU dataset

Usage

coutts2

coutts2.epg	<i>EPG-compressed trackdata from the segment list coutts2</i>
-------------	---

Description

An EMU dataset

Usage

coutts2.epg

coutts2.1	<i>Vector of word label from the segment list coutts2</i>
-----------	---

Description

An EMU dataset

Usage

coutts2.1

coutts2.sam	<i>Trackdata of acoustic waveforms from the segment list coutts2</i>
-------------	--

Description

An EMU dataset

Usage

coutts2.sam

cr *Plot digital sinusoids.*

Description

The function plots and/or sums digital sinusoids for different parameter settings.

Usage

```
cr(A = 1, k = 1, p = 0, N = 16, samfreq = NULL, duration = NULL,
  const = NULL, expon = NULL, plotf = TRUE, ylim = NULL,
  xlim = NULL, values = FALSE, xlabel = "Time (number of points)",
  ylabel = "Amplitude", type = "b", bw = NULL, dopoints = FALSE, ...)
```

Arguments

A	A vector of amplitude values. Defaults to A = 1
k	A vector of cycles (repetitions). Defaults to k = 1
p	A vector of phase values between $-\pi/2$ and $\pi/2$. Defaults to 0.
N	The number of points in the signal. Defaults to 16.
samfreq	If NULL, then a sinusoid is plotted with a frequency of k cycles per N points. Otherwise, if samfreq is a numeric, then the argument to k is interpreted as the frequency in Hz and the sinusoid at that frequency is plotted for however many points are specified by N. For example, if samfreq is 40 (Hz), and if N is 40 and k = 1, then 1 cycle of a 1 Hz sinusoid will be plotted.
duration	Specify the duration in ms. If NULL, the default, then the duration of the sinusoid is in points (N), otherwise if a numeric value is supplied, then in ms. For example, 1/2 second of a 1 cycle sinusoid at a sampling frequency of 40 Hz: duration = 500, k = 1, samfreq=40. A ms value can be supplied only if the sampling frequency is also specified.
const	A single numeric vector for shifting the entire sinusoid up or down the y-axis. For example, when const is 5, then 5 + s, where s is the sinusoid is plotted. Defaults to 0 (zero).
expon	A numeric vector. If supplied, then what is plotted is $\text{expon}[j]^{(c(0:(N-1)) * A \cos(2 * \pi * k/N * (0:(N-1))))}$. For example, a decaying sinusoid is produced with cr(expon=-0.9). Defaults to NULL (i.e. to expon = 1).
plotf	A single-valued logical vector. If T (default), the sinusoid is plotted.
ylim	A two-valued numeric vector for specifying the y-axis range.
xlim	A two-valued numeric vector for specifying the x-axis range.
values	If T, then the values of the sinusoid are listed. Defaults to F.
xlab	A character vector for plotting the x-axis title.
ylabel	A character vector for plotting the y-axis title.
type	A character vector for specifying the line type (see par)

bw	A numeric vector for specifying the bandwidth, if the sampling frequency is supplied. The bandwidth is converted to an exponential (see <code>expon</code> using <code>exp(-rad(bw/2, samfreq = samfreq))</code>).
dopoints	this is now redundant.
...	Option for supplying further graphical parameters - see <code>par</code> .

Author(s)

Jonathan Harrington

See Also[crplot](#)**Examples**

```
# cosine wave
cr()

# doubling the frequency, 1/3 amplitude, phase = pi/4, 50 points
cr(A=1/3, k=2, p=pi/4, N=50)

# sum 3 sinusoids of different frequencies)
cr(k=c(1, 3, 4))

# sum 2 sinusoids of different parameters
cr(c(1, 2), c(2, 10), c(0, -pi/3), N=200, type="l")

# store the above to a vector and overlay with noise
v = cr(c(1, 2), c(2, 10), c(0, -pi/3), N=200, type="l", values=TRUE)
r = runif(200, -3, 3)
v = v+r
plot(0:199, v, type="l")

# 100 points of a 50 Hz sinusoid with a 4 Hz bandwidth
# at a sampling frequency of 200 Hz
cr(k=50, bw=4, samfreq=2000, N=100)

# the same but shift the y-axis by +4 (d.c. offset=+4)
cr(const=4, k=50, bw=4, samfreq=2000, N=100)

# sinusoid multiplied by a decaying exponential (same effect as bandwidth)
cr(expon=-0.95, N=200, type="l")
```

crplot *Function to plot a digital sinusoid and the circle from which it is derived.*

Description

A digital sinusoid is derived the movement of a point around a circle. The function shows the relationship between the two for various parameter settings.

Usage

```
crplot(A = 1, k = 1, p = 0, N = 16, const = NULL, figsize = 8, npoints = 500, col = 1, cplot = TRUE,
splot = TRUE, numplot = TRUE, axes = TRUE, incircle = TRUE, arrow = TRUE, linetype = 1, textplot = NULL,
lineplot = NULL, ylab = "Amplitude", super = NULL, xlabel = NULL, type = "b",
xlabel = "Time (number of points)", fconst = 3.5/3.1, pointconst = 1.2)
```

Arguments

A	Amplitude of the circle/sinusoid.
k	Frequency of the sinusoid
p	Phase of the sinusoid
N	Number of points per cycle or revolution.
const	A constant corresponding to $k + A \cdot \cos(2 \cdot \pi \cdot k + p)$
figsize	Set the figure size as <code>pin <- c(figsize, figsize/2)</code> . Defaults to <code>figsize = 8</code> .
npoints	The number of points used in plotting the circle. Defaults to 500
col	An integer for the color in plotting the sinusoid and points around the circle
cplot	Now redundant
splot	Now redundant
numplot	Logical. If T (defaults), the digital points around the circle are numbered
axes	Logical. If T, plot axes.
incircle	Logical. If T, plot an the angle between digital points in the circle.
arrow	Logical. If T, plot an arrow on incircle showing the direction of movement.
linetype	Specify a linetype. Same function as <code>lty</code> in <code>plot</code>
textplot	A list containing <code>\\$radius</code> , <code>\\$textin</code> , <code>\\$pivals</code> for plotting text at specified angles and radii on the circle. <code>\\$radius</code> : a vector of amplitudes of the radii at which the text is to be plotted; <code>\\$textin</code> : a vector of character labels to be plotted; <code>\\$pivals</code> : the angle, in radians relative to zero radians (top of the circle) at which the text is to be plotted. Defaults to NULL
lineplot	Plot lines from the centre of the circle to the circumference. <code>lineplot</code> is a vector specifying the angle in radians (zero corresponds to the top of the circle)
ylab	Specify a y-axis label.

super	Superimpose a part solid circle and corresponding sinusoid. This needs to be a list containing $\$first$ and $\$last$, which are values between 0 and 2π defining the beginning and ending of the part circle which is to be superimposed
xaxlab	Now redundant
xlab	Specify an x-axis label.
type	Specify a type.
fconst	A single element numeric vector for the aspect ratio in a postscript plot. Defaults to 3.5/3.1 which is appropriate for a postscript setting of setps(h=4, w=4)
pointconst	The radius for plotting the numbers around the circle. Defaults to $1.2 * A$

Author(s)

Jonathan Harrington

References

Harrington, J, & Cassidy, S. 1999. Techniques in Speech Acoustics. Kluwer

See Also[cr](#)**Examples**

```
crplot()
# sine wave
crplot(p=-pi/2)

crplot(k=3)

# aliasing
crplot(k=15)
```

dapply*apply a function to each part of a trackdata object*

Description

Given an Emu trackdata object, dapply will apply a given function to the data corresponding to each segment of data. The result is a new trackdata object.

Usage

```
dapply(trackdata, fun, ...)
```

Arguments

trackdata	An Emu trackdata object
fun	A function taking a matrix of data and a vector of times and returning a list with components \$data and \$ftime.
...	Additional arguments to be passed to fun

Details

dapply can be used to apply an arbitrary function to trackdata extracted from an Emu database. It can be used for example to smooth the data (see [dsmooth](#)) or differentiate it (see [ddiff](#)).

Trackdata is made up of three components: a matrix of data \$data, a matrix of indexes (\$index) and a matrix of segment times (\$ftime). The indexes contain the start and end rows for each segment in the trackdata, the time matrix contains the start and end times of each data segment.

The function fun supplied to dapply should take one matrix of data (corresponding to one segment) and a vector of two times being the start and end of the data. It should return a modified data matrix, which can have any number of rows or columns, and a new pair of start and end times. The new start and end times are necessary because the operation applied might shorten or interpolate the data and hence change the times corresponding to the first and last rows of data.

Value

An Emu trackdata object with components:

data	A matrix of data with all segments concatenated by row.
index	A two column matrix of the start and end rows for each segment
ftime	A two column matrix of the start and end times for each segment

See Also

[dsmooth](#) [ddiff](#)

Examples

```
data(dip)
## formant data of the first segment in segment list dip
fm <- dip.fdat[1]

testfun <- function(data, ftime, n) {
  ## return only the first n rows of data
  ## doesn't check to see if there really are n rows...
  newdata <- data[1:n,]
  ## calculate a new end time
  interval <- (ftime[2]-ftime[1])/nrow(data)
  ftime[2] <- ftime[1] + interval*n
  ## now return the required list
  return( list( data=newdata, ftime=ftime ) )
}
```

```
fm.first3 <- dapply( fm, testfun, 3 )  
fm.first10 <- dapply( fm, testfun, 10 )
```

dbnorm

Function to dB-normalise spectral objects

Description

The function can be used to rescale a spectrum to a dB value at a particular frequency - for example, to rescale the spectrum so that 3000 Hz has 0 dB and all other values are shifted in relation to this.

Usage

```
dbnorm(specdata, f = 0, db = 0)
```

Arguments

specdata	An object of class 'spectral'
f	A single element vector specifying the frequency. Defaults to 0
db	A single element vector specifying the dB value to which the spectrum is to be rescaled. Defaults to zero

Value

An object of the same class with rescaled dB values. The default is to rescale the dB-values of the spectrum to 0 dB at 0 Hz.

Author(s)

Jonathan Harrington

See Also

[dbtopower](#) [plot.spectral](#)

Examples

```
# normalise to - 40 dB at 1500 Hz  
res = dbnorm(e.dft, 1500, 0)  
# compare the two  
ylim = range(c(res, e.dft))  
plot(e.dft, ylim=ylim, type="l")  
par(new=TRUE)  
plot(res, ylim=ylim, type="l", col=2)
```

`dbtopower`*Function for inter-converting between decibels and a linear scale*

Description

The function converts from decibels to a linear scale

Usage

```
dbtopower(specdata, const = 10, base = 10, inv = FALSE)
```

Arguments

<code>specdata</code>	A numeric object or an object of class <code>trackdata</code>
<code>const</code>	A single element numeric vector. Defaults to 10
<code>base</code>	A single element numeric vector. Defaults to 10
<code>inv</code>	Logical. If T, then the conversion is from a logarithmic to an anti-logarithmic form, otherwise the other way round

Details

The function returns $\text{base}^{(\text{specdata}/\text{const})}$ if `inv=F`, otherwise, $\text{const} * \log(\text{dat}, \text{base}=\text{base})$. If the object to which this function is applied is of class `'trackdata'` then this function is applied to `\$data`.

Value

An object of the same class.

Author(s)

Jonathan Harrington

See Also

[dbtopower](#) [plot.spectral](#)

Examples

```
# convert 10 dB to a power ratio
vec = dbtopower(10)
# convert dB-data to a power ratio and back to decibels
res = dbtopower(vowlax.dft.5)
res = dbtopower(res, inv=TRUE)
```

dct *Discrete Cosine Transformation*

Description

Obtain the coefficients of the discrete cosine transformation (DCTTRUE).

Usage

```
dct(data, m = NULL, fit = FALSE)
```

Arguments

data	a vector or single column matrix of numeric values to which the 2nd order polynomial is to be fitted.
fit	if F, return the DCT coefficients; if T, the values of the smoothed trajectory are returned based on summing the cosine waves of the k lowest ordered DCT coefficients, where k is the argument given below.
m	The number of DCT coefficients that are returned or on which the smoothed trajectory is based. Defaults to NULL which returns coefficients of frequencies $k = 0, 1, 2 \dots N-1$ where N is the length of the input signal, wav. If fit = TRUE and k = NULL, then the the sum of all the cosine waves whose amplitudes are the DCT coefficients are returned - which is equal to the original signal. k must be between 2 and the length of the signal.

Details

The function calculates the DCT coefficients for any vector or single-columned matrix. The function can also be used to obtain a smoothed trajectory of the input data by summing the cosine waves derived from the first few DCT coefficients.

The algorithm first reflects the input signal about the last data point, N. Thus if the input signal vec if of length N, the algorithm creates a vector $c(\text{vec}, \text{rev}(\text{vec}[-c(1,N)]))$. and the R fft function is applied to this reflected signal. The DCT coefficients are real part of what is returned by fft i.e. the amplitudes of the cosine waves of frequencies $k = 0, 1, 2, \dots 2 \cdot (N-1)$ radians per sample. The phase is zero in all cases. The amplitudes are calculated in such a way such that if these cosine waves are summed, the original (reflected) signal is reconstructed. What is returned by dct() are the amplitudes of the cosine waves (DCT coefficients) up to a frequency of N radians/sample, i.e. a vector of cosine wave amplitudes that has the same length as the original signal and of frequencies $k = 0, 1, 2, \dots (N-1)$. Alternatively, if fit=T, a smoothed signal of the same length as the original signal is obtained based on a summation of the lowest ordered DCT coefficients. This dct() algorithm returns very similar values to DCT() with inv=F written by Catherine Watson and used in Watson & Harrington (1999).

Author(s)

Jonathan Harrington

References

Watson, C. & Harrington, J. (1999). Acoustic evidence for dynamic formant trajectories in Australian English vowels. *Journal of the Acoustical Society of America*, 106, 458-468.

Zahorian, S., and Jagharghi, A. (1993). Spectral-shape features versus formants as acoustic correlates for vowels, *Journal of the Acoustical Society of America*, 94, 1966-1982.

See Also

[plafit by](#)

Examples

```
data(vowlax)
# obtain the first four DCT coefficients
# (frequencies k = 0, 1, 2, 3) for some
# first formant frequency data
vec <- vowlax.fdat[1,1]$data
dct(vec, m=4)

# obtain the corresponding smoothed
# trajectory
dct(vec, m=4 , fit=TRUE)
```

dcut

Function to extract a vector or matrix from EMU-Trackdata at a single time point or to create another EMU-trackdata object between two times.

Description

A general purpose tool for extracting data from track objects either at a particular time, or between two times. The times can be values in milliseconds or proportional times between zero (the onset) and one (the offset).

Usage

```
dcut(trackdata, left.time, right.time, single = TRUE, average = TRUE, prop = FALSE)
```

Arguments

trackdata	An Emu trackdata object.
left.time	Either: a numeric vector of the same length as there are observations in trackdata. Or: a single value between 0 and 1. In the first case, the left time boundary of trackdata[n,] is cut at left.time[n], in the second case, and if prop=T, it is cut at that proportional time.

right.time	Either: a numeric vector of the same length as there are observations in trackdata. Or: a single value between 0 and 1. In the first case, the right time boundary of trackdata[n,] is cut at right.time[n], in the second case, and if prop=T, it is cut at that proportional time.
single	If TRUE, one value is returned per segment. This applies when the requested time falls between two track frames. When single=TRUE, the preceding value is returned, unless average=TRUE (see below), in which case the average value of the two frames is returned. when the right.time argument is omitted
average	A single element logical vector - see single above. Applies only when the right.times argument is omitted and when single = TRUE
prop	If TRUE left.time and right.time are interpreted as proportions, if FALSE, they are interpreted as millisecond times

Details

This function extracts data from each segment of a trackdata object.

If 'prop=FALSE' the time arguments ('left.time' and 'right.time') are interpreted as millisecond times and each should be a vector with the same length as the number of segments in 'trackdata'. If 'prop=TRUE' the time arguments should be single values between zero (the onset of the segment) and one (the offset).

If 'right.time' is omitted then a single data point corresponding to 'left.time' for each segment is returned.

Value

A trackdata object if both 'left.time' and 'right.time' are specified, otherwise a matrix if 'right.time' is unspecified and the trackdata object has multiple columns of data or a vector if 'right.time' is unspecified and the trackdata object has a single column of data.

Author(s)

Jonathan Harrington

See Also

[emu.track](#), [dplot](#), [eplot](#)

Examples

```
# the data values of the trackdata object at the temporal midpoint
# (midvals is matrix of F1 and F2 data)
dip.fdat[1:10]
midvals <- dcut(dip.fdat, 0.5, prop=TRUE)
midvals[1:10,]

# the data values of the trackdata object between
# extending from 20
# (bet is a trackdata object of F1 and F2 values)
```

```
bet <- dcut(dip.fdat, 0.2, 0.8, prop=TRUE)
bet[1]

# the data values of the trackdata object at 30 ms after
# the start time of the trackdata object
# (time30 is a matrix of F1 and F2 data
times <- dip.fdat$ftime[,1]+30
times[1:10]
time30 <- dcut(dip.fdat, times)
time30[1:10]

# the data values of the trackdata object
# between the start time and 30 ms after the start time
# (int is a trackdata object of F1 and F2 values extending
# from the start of the diphthongs up to 30 ms after the diphthongs)
int <- dcut(dip.fdat, dip.fdat$ftime[,1], times)
int[1]
```

ddiff

Differentiation of tracks

Description

Differentiates a list, as returned by track, to the nth order, readjusting the index and ftime values each time.

Usage

```
ddiff(dataset, n = 1, smoothing = TRUE)
```

Arguments

dataset	track data object - a list as returned by track
n	the order of differentiation
smoothing	if TRUE track is smoothed

Author(s)

Jonathan Harrington

demo.all	<i>Emu segment list</i>
----------	-------------------------

Description

Segment list of the demo database that is part of the Emu system. It is the result of a database query, that searched all segments at level Phonetic.

Usage

```
data(demo.all)
```

Format

First Column labels Second start time of the segment Third end time of the segment Fourth utterance name of the utterance the segment was found

Details

A segment list is created via `emu.query()` or by using the EMU Query Tool.

See Also

[demo.vowels segmentlist](#)

demo.all.rms	<i>Emu track data for a rms track for segment list demo.all</i>
--------------	---

Description

A track list of the demo database that is part of the Emu system. It is the result of get rms data for the segment list demo.all (`data(demo.all)`).

Usage

```
data(demo.all.rms)
```

Format

A object with `\$index`, `\$ftime` and `\$data`

index: a two columned matrix with the range of the `\$data` rows that belong to the segment
ftime: a two columned matrix with the times marks of the segment
data: a vector with the rms data

Details

A track list is created via `emu.track()` or via get data within the EMU Query Tool.

See Also

[demo.vowels.fm segmentlist trackdata](#)

demo.vowels

Emu segment List

Description

Segment list of the demo database that is part of the Emu system. It is the result of a database query, that searched all vowel segments at level Phonetic.

Usage

data(demo.vowels)

Format

First Column labels Second start time of the segment Third end time of the segment Fourth utterance name of the utterance the segment was found

Details

A segment list is created via emu.query() or by using the EMU Query Tool.

See Also

[demo.all segmentlist](#)

demo.vowels.f0

F0 track data for segment list demo.vowels

Description

A track list of the demo database that is part of the Emu system. It is the result of get F0 data for the segment list demo.vowels (see data(demo.vowels)).

Usage

data(demo.vowels.f0)

Format

An object with `\$index`, `\$ftime` and `\$data`

index: a two columned matrix with the range of the `\$data` rows that belong to the segment ftime:

a two columned matrix with the times marks of the segment data: a one columned matrix with the

F0 values

Details

A track list is created via `emu.track()` or via get data within the EMU Query Tool.

See Also

[demo.all.rms segmentlist trackdata](#)

demo.vowels.fm

Formant track data for segment list demo.vowels

Description

A track list of the demo database that is part of the Emu system. It is the result of get fm data for the segment list demo.vowels (see `data(demo.vowels)`).

Usage

```
data(demo.vowels.fm)
```

Format

index: a two columned matrix with the range of the \Sdata rows that belong to the segment
 ftime: a two columned matrix with the times marks of the segment
 data: a three columned matrix with the formant values of the first three formants for each segment

Details

A track list is created via `emu.track()` or via get data within the EMU Query Tool.

See Also

[demo.all.rms segmentlist trackdata](#)

dextract

Extract a subset of data from a trackdata object

Description

A function that cuts up trackdata either at a proportional time or proportionally between two times. It is a subsidiary function of `dplot()`

Usage

```
dextract(dataset, start, end)
```

Arguments

dataset	A trackdata object
start	A single valued numeric vector corresponding to a proportional time between zero (the onset of the trackdata) and one (the offset of the trackdata).
end	As start, but optional

Value

If both start and end are specified, a trackdata object is returned, otherwise a vector if the original trackdata is one-dimensional and the end argument is not used, or a matrix if the original trackdata has more than one dimension and the end argument is not used

Author(s)

Jonathan Harrington

See Also

dcut

Examples

```
data(demo.vowels.f0)
data(demo.vowels.fm)

form = demo.vowels.fm
# get the formants at the midpoint: f50 is a matrix
# same as dcut(form, .5, prop=TRUE)
f50 = dextract(form, 0.5)
# get the formants between the 25% and 75% time points
# fcut is a trackdata object
# same as dcut(form, .25, .75, prop=TRUE)
fcut = dextract(form, 0.25, 0.75)
# get F0 at the midpoint. fzero50 is a vector
# same as dcut(fzero, .5, prop=TRUE)
fzero = demo.vowels.f0
fzero50 = dextract(fzero, 0.5)
```

dim.trackdata

A method of the generic function dim for objects of class 'trackdata'

Description

The function returns the dimension attributes of a track data object.

Usage

```
## S3 method for class 'trackdata'  
dim(x)
```

Arguments

x a track data object

Details

The function returns the dimension attributes of a track data object as the number of segments x number of tracks. `c(nrow(x$index), ncol(x$data))`

Author(s)

Jonathan Harrington

Examples

```
#isol.fdat is the formant track of the segment list isol  
  
#write out the dimension of the track data object  
dim.trackdata(isol.fdat)  
  
#because there are 13 segments  
isol.fdat$time  
  
#and there are 4 rows for each segment (see here for the first segment)  
isol.fdat$data[1,]
```

dimnames.trackdata *Dimnames of trackdata object*

Description

returns dimension names of trackdata objects

Usage

```
## S3 method for class 'trackdata'  
dimnames(x)
```

Arguments

x trackdata object

dip	<i>Segment list of diphthongs, two speakers one male, one female , Standard North German, read speech from database kielread</i>
-----	--

Description

An EMU dataset

Usage

dip

dip.fdat	<i>Trackdata of formants from the segment list dip</i>
----------	--

Description

An EMU dataset

Usage

dip.fdat

dip.l	<i>Vector of phoneme labels from the segment list dip</i>
-------	---

Description

An EMU dataset

Usage

dip.l

dip.spkr	<i>Vector of speaker labels from the segment list dip</i>
----------	---

Description

An EMU dataset

Usage

dip.spkr

dorfric	<i>Segment list of palatal and velar fricatives, read speech, one female speaker of Standard North German from database kielread</i>
---------	--

Description

An EMU dataset

Usage

dorfric

dorfric.dft	<i>Spectral trackdata object from the segment list dorfric.</i>
-------------	---

Description

An EMU dataset

Usage

dorfric.dft

dorfric.l	<i>Vector of phoneme labels from the segment list dorfric</i>
-----------	---

Description

An EMU dataset

Usage

dorfric.l

dorfric.lv	<i>Vector of preceding phoneme labels from the segment list dorfric</i>
------------	---

Description

An EMU dataset

Usage

dorfric.lv

dorfric.v1	<i>A vector of the preceding vowel labels in the dorfric dataset</i>
------------	--

Description

A vector of labels preceding the labels: "C" or "x"

Usage

dorfric.v1

See Also

[segmentlist](#)

dorfric.w	<i>Vector of the word labels from the segment list dorfric</i>
-----------	--

Description

An EMU dataset

Usage

dorfric.w

dorsal	<i>Segment list of a a sequence of a vowel and K where K is either a voiceless dorsal fricative or k , isolated words, one speaker, Standard North German from database epgdorsal</i>
--------	---

Description

An EMU dataset

Usage

dorsal

dorsal.bound	<i>Vector of the acoustic boundary times between V and C from the segment list dorsal</i>
--------------	---

Description

An EMU dataset

Usage

dorsal.bound

dorsal.clab	<i>Vector of consonant labels from the segment list dorsal.</i>
-------------	---

Description

An EMU dataset

Usage

dorsal.clab

dorsal.epg	<i>EPG-compressed trackdata from the segment list dorsal</i>
------------	--

Description

An EMU dataset

Usage

dorsal.epg

dorsal.fm	<i>Trackdata of formants from the segment list dorsal</i>
-----------	---

Description

An EMU dataset

Usage

dorsal.fm

dorsal.sam	<i>Trackdata of acoustic waveform from the segment list dorsal</i>
------------	--

Description

An EMU dataset

Usage

dorsal.sam

dorsal.vlab	<i>Vector of vowel labels from the segment list dorsal.</i>
-------------	---

Description

An EMU dataset

Usage

dorsal.vlab

dplot	<i>A function to plot one or more columns of EMU-trackdata as a function of time</i>
-------	--

Description

A general purpose routine for plotting EMU-trackdata on a single plot. Tracks can be aligned at an arbitrary position, length normalised or averaged. The plots can be colour-coded for different category types.

Usage

```
dplot(x, labs = NULL, offset = 0, prop = TRUE, average = FALSE,
      xlim = NULL, ylim = NULL, lty = FALSE, normalise = FALSE,
      colour = TRUE, lwd = NULL, pch = NULL, legend = "topright",
      axes = TRUE, type = "l", n = 20, ...)
```

Arguments

x	An EMU-trackdata object
labs	A label vector with one element for each row in 'dataset'
offset	Either: A single numeric vector between 0 and 1. 0 and 1 denote synchronize the trackdata at their temporal onsets and offsets respectively; 0.5 denotes synchronization at the temporal midpoint, etc. Or a numeric vector of the same length as x specifying the synchronisation point per segment
prop	A single element character vector specifying whether the tracks should be aligned proportionally or relative to millisecond times. Defaults to proportional alignment
average	If TRUE, the data for each unique label in 'labs' is averaged
xlim	A vector of two numeric values specifying the x-axis range
ylim	A vector of two numeric values specifying the y-axis range
lty	A single element logical vector. Defaults to F. If TRUE, plot each label type in a different linetype
normalise	If TRUE, the data for each segment is linearly time normalised so that all observations have the same length. The number of points used in the linear time normalisation is control by the argument n.
colour	A single element logical vector. Defaults to T to plot each label type in a different colour
lwd	A code passed to the lwd argument in plotting functions. 'lwd' can be either a single element numeric vector, or its length must be equal to the number of unique types in labs. For example, if lwd=3 and if labs = c("a", "b", "a", "c"), then the output is c(3, 3, 3, 3). Alternatively, if lwd = c(2,3,1), then the output is c(2, 3, 2, 1) for the same example. The default is NULL in which case all lines are drawn with lwd=1
pch	A code passed to the pch argument in plotting functions. Functions in the same way as lwd above
legend	Either a character vector to plot the legend. Possible values are: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". This places the legend on the inside of the plot frame at the given location. Partial argument matching is used. Or a logical vector: legend = FALSE suppresses legend plotting. legend = TRUE plots it at the default, legend = "topright"
axes	A single element logical vector. Defaults to T to plot the axes
type	The default line type. Default to "l" for a line plot
n	A single element numeric vector. Only used if normalise=T. The number of data points used to linearly time normalise each track
...	graphical options par

Value

NULL

Author(s)

Jonathan Harrington

See Also

[dcut emu.track](#)

Examples

```
# Plot of column 1 (which happens to be the 1st formant) of an EMU-trackdata object
dplot(dip.fdat[,1])

# As above but only observations 1 to 5
dplot(dip.fdat[1:5,1])

# column 2 (which happens to be of the second formant) and colour-coded
# for each label-type
dplot(dip.fdat[,2], dip.l)

# put the legend bottom left
dplot(dip.fdat[,2], dip.l, legend="bottomleft")

# as above with no legend and averaged per category
dplot(dip.fdat[,2], dip.l, legend=FALSE, average=TRUE)

# both formants averaged
dplot(dip.fdat[,1:2], dip.l, average=TRUE)

# F2 only with linear-time normalisation
dplot(dip.fdat[,2], dip.l, norm=TRUE)

# linear time-normalisation, both formants and averaged
dplot(dip.fdat[,1:2], dip.l, norm=TRUE, average=TRUE)

# synchronise at the temporal midpoint before averaging, F2 only
dplot(dip.fdat[,2], dip.l, offset=0.5, average=TRUE)

# synchronise 60 ms before the diphthong offset
dplot(dip.fdat[,2], dip.l, offset=dip.fdat$time[,2]-60, prop=FALSE)

# as above averaged, no colour with linetype,
```

```
# different plot symbols double line thickness in the range between +- 20 ms
dplot(dip.fdat[,2], dip.l, offset=dip.fdat$time[,2]-60, prop=FALSE,
      average=TRUE, colour=FALSE, lty=TRUE, pch=1:3, lwd =2, type="b", xlim=c(-20, 20))
```

dsmooth

Smooth the data in a trackdata object.

Description

Smooths each dataset in a trackdata object using a running mean smoother.

Usage

```
dsmooth(dataset)
```

Arguments

dataset A trackdata object as returned from track.

Details

This function uses the dapply function to apply smooth to the data for each segment.

Value

The result of applying the smooth function to each column of the data for each segment in the trackdata object.

See Also

smooth, dapply

dur

duration

Description

calculates durations

e.dft	<i>Spectral vector of a single E vowel produced by a male speaker of Standard North German.</i>
-------	---

Description

An EMU dataset

Usage

e.dft

ellipse	<i>Calculate ellipse coordinates</i>
---------	--------------------------------------

Description

Calculates ellipse coordinates for eplot

Usage

ellipse(x, y, rx, ry, orient, incr=360/100)

Arguments

x	X coordinate of center
y	y coordinate of center
rx	Radius in the x direction
ry	Radius in the y direction
orient	Orientation, in radians. The angle of the major axis to the x axis.
incr	The increment between points, in degrees.

Value

A matrix of x and y coordinates for the ellipse.

See Also

eplot()

emu.defaultpaths	<i>Default paths of the emu and tcl/tk libraries</i>
------------------	--

Description

Returns the paths that are suggested for the system for the emu and tcl/tk libraries.

Usage

```
emu.defaultpaths()
```

Author(s)

Tina John

Examples

```
emu.defaultpaths()
```

emu.directory	<i>EMU directory</i>
---------------	----------------------

Description

Gives the EMU installation path

emu.query	<i>Query an Emu Database</i>
-----------	------------------------------

Description

Perform a query on an Emu speech database selecting segments for subsequent analysis. The return value is a segment list containing the labels, start and end times and utterance name of each token matching the query.

Usage

```
emu.query(template, pattern=NULL, query="")
```

Arguments

template	The name of the Emu database to query (in quotes)
pattern	A pattern matching utterances to be searched from the database
query	a valid Emu query

Details

The Emu query language is described in the Emu documentation. `template` must refer to a valid database template on your system, ie. the template file name without the `.tpl` extension.

Value

An object of type `emusegs` with one row per token matched by the query and columns for the token label, start time, end time and utterance name. This can be passed to `emu.track` to extract speech data corresponding to each token.

Note

This function calls an external which are scripts via `tcltk` part of the Emu speech database system and so requires this system to be installed on your computer. See the Emu web site for details.

Author(s)

Steve Cassidy <Steve.Cassidy@mq.edu.au>

References

See the Emu documentation at: <http://www.shlrc.mq.edu.au/emu>

See Also

[emu.track](#)

Examples

```
## assumes a database called demo is available on your system and that
## the Emu system is installed.
data(vowlax)
# find all Phonetic vowels in the database
## Not run: segs <- emu.query("demo", "*", "Phonetic=vowel")

# display summary information on the segments found
summary(segs)
# get formant data at the midpoint and plot it
## Not run: data <- emu.track( segs, "fm", cut=0.5 )

eplot( data[,1:2], label( segs ), dopoints=TRUE )
```

emu.requery	<i>Generate a segment list or label list derived from an existing Emu segment list</i>
-------------	--

Description

Given an Emu segment list, `emu.requery` can find other segments which are related to the segments in the original list. For example, the segment that follows or one dominated by the original segment.

Usage

```
emu.requery(segs, level, targetlevel=level, justlabels=FALSE, sequence=0, longerok=FALSE)
```

Arguments

<code>segs</code>	An Emu segment list
<code>level</code>	The level of the segments in <code>segs</code> (eg. Phonetic)
<code>targetlevel</code>	The level of the segments/events to find
<code>justlabels</code>	If TRUE, a label vector is returned instead of a segment list.
<code>sequence</code>	An integer value (positive or negative) which denotes the relative position of the target segments.
<code>longerok</code>	If TRUE, the returned segment list or label list may be longer than the input in cases where one segment in the input dominates more than one segment at the target level. If FALSE, the result will always have the same length as the input and multiple labels will be concatenated.

Details

`emu.requery` provides a way of locating segments related to those in an existing segment list either by sequence (follows or precedes) or hierarchically (dominates, is dominated by).

To find sequentially related segments or events, use the `sequence` argument to define the number of steps to take. For example `sequence=1` will find the immediately following segment, `sequence=-1` will find the preceding segment and `sequence=3` will find the third following segment.

To find hierarchically related segments, use the `targetlevel` argument to denote a different level to that of the original segment list. Segments at the target level related to the original segments will be retrieved. For example, if the original segment list consists of Phonetic segments, `targetlevel="Word"` would find the Word level segment dominating each Phonetic segment.

If both `targetlevel` and `sequence` are supplied together, the `targetlevel` argument is applied first and then the `sequence` argument is applied to find the following or preceding segment at the target level. For example, `targetlevel="Word" sequence=1` would find the Word following the word that dominates the original segment.

In some cases, Emu can't locate an appropriate segment (for example, because there is no following segment). In this case the corresponding segment label in the result will be "no-segment".

If `justlabels` is `TRUE`, only the vector of labels for the matching segments is returned, otherwise a complete segment list is returned.

If `longerok` is `TRUE`, the resulting segment list or label vector might be longer than the input in cases where more than one segment is dominated by (or dominates) the original segment. For example, if the original level is `Word` and the target level is `Phoneme` the segment `cat` might dominate `/k/`, `/a/` and `/t/` Phonemes. If `longerok` is `TRUE` these will be included as separate entries in the result, if it is `FALSE` the labels and times will be concatenated (eg. `k-a-t`).

Note that `emu.requery` might fail in some cases since it relies on finding segments according to their times. In some cases this might result in not finding a given segment due to rounding errors.

Value

If `longerok` is `FALSE`: an Emu segment list, otherwise a vector of segment labels.

Author(s)

Steve Cassidy

See Also

[emu.query](#)

Examples

```
data(vowlax)

# find all Phonetic vowels in the database
## Not run: segs <- emu.query("demo", "*", "Phonetic=vowel")

## find the word level segments
## Not run: wsegs <- emu.requery(segs, level="Phonetic", targetlevel="Text")

# now find the Phonetic segment that follows the original vowels:
## Not run: nsegs <- emu.requery(segs, level="Phonetic", sequence=1)

# and the one that precedes them, but only get the labels this time
## Not run: prelabs <- emu.requery(segs, level="Phonetic", sequence=-1, justlabels=TRUE)
```

 emu.track

Retrieve Numerical Data

Description

Takes the result of a database query and retrieves corresponding time-series data from the database.

Usage

```
emu.track(seglist, trackname, cut=NULL, npoints=NULL, template=attr(seglist, "database"), rmfile=TRUE)
```

Arguments

seglist	An Emu segment list.
trackname	The name of the data track to retrieve, a string. This must be a track name defined in the database template.
cut	An optional cut time for segment data, ranges between 0 and 1, a value of 0.5 will extract data only at the segment midpoint.
npoints	An optional number of points to retrieve for each segment or event. For segments this requires a cut= argument and data is extracted around the cut time. For events data is extracted around the event time.
template	The database to retrieve the data from, this should not normally be set.
rmfile	A trackdata matrix is written to users home directory. This file is removed (TRUE) or kept (FALSE).

Details

emu.track takes a segment list as input and retrieves associated numerical data from the corresponding database. The segment list will usually be the result of a call to [emu.query](#) but could be constructed with the [make.seglist](#) function. The result is either a two dimensional array of data or an object of class trackdata which may contain multi-column data from many tokens.

Value

If only two arguments are supplied the entire data track is retrieved for each segment in the segment list. The amount of data returned will depend on the sample rate and number of columns in the track requested. The returned data is packaged up as a trackdata object.

The optional cut argument specifies a cut point as a fraction of the duration of each segment in the segment list. If this is specified the data at this single cutpoint will be extracted rather than that for the entire track. The result of emu.track with the cut argument is a two dimensional array of data with one row per segment in the original segment list and one column per column in the requested track. This array can be treated like any other array in Splus.

If the input segment list is in fact an event list (ie. is derived from an annotation level defined as events in the database template) then the result of emu.track is the same as if the cut argument was specified.

If the `npoints` argument is specified, it defines the number of points that will be returned for each segment or event. The `cut` argument is required if the input is a segment list (as opposed to an event list).

Note

This function calls external scripts via `telnet` which are part of the Emu speech database system and so requires this system to be installed on your computer. See the Emu web site for details.

Author(s)

Steve Cassidy <Steve.Cassidy@mq.edu.au>

References

See the Emu documentation at: <http://www.shlrc.mq.edu.au/emu>

See Also

[dplot](#) [dapply](#)

Examples

```
## assumes a database called demo is available on your system and that
## the Emu system is installed.
data(vowlax)

# find all Phonetic vowels in the database
## Not run: segs <- emu.query("demo", "*", "Phonetic=vowel")

# get formant data at the midpoint, returns an array
## Not run: data.mid <- emu.track( segs, "fm", cut=0.5 )

# get formant data for entire tracks, returns trackdata
## Not run: data.all <- emu.track( segs, "fm" )

summary(data.all)
```

emu.variables

Emu variables

Description

`emu.version` Current version of the emu R package

`emu.date` Date of package creation

`emu.year` Year of package creation, copyright.

Usage

```
emu.version  
emu.date  
emu.year
```

emulink

Establishs the link to an Emu installation

Description

Without argument `emulink()` tries to read a configuration file and tries to find the necessary package in the stored directory paths. If this fails it tries to link against default directories. If this fails, the user is asked to enter the path. With `paths` argument the first two steps are ignored.

Usage

```
emulink(paths = "")
```

Arguments

`paths` The directories separated by ";" in which Emu libraries and `tellib >= 1.8` are installed on the system. Without `paths` argument link is established with default configurations.

Details

The paths are added to the `tcl auto_path` variable to make the `emuR` library available that is part of the Emu Speech Database System <http://www.emu.sf.net>.

Value

After success the file path is returned where this configuration was written to.

Note

Use `emulink()` without `paths` argument first. It is not necessary to link an Emu installation to this library. You can use all functions of the library but `emu.query`, `emu.requery` and `emu.track`. The functionality of `emu.query` and `emu.track` function is provided by the Emu Speech Database System <http://www.emu.sf.net> also. Thus export the respective files from the software and import it to R using `read.emusegs` or `read.trackdata` respectively.

Author(s)

Tina John

References

<http://www.emu.sf.net>

See Also

[emu.query](#) [emu.requery](#) [emu.track](#) [read.emusegs](#) [read.trackdata](#)

Examples

```
## Not run: emulink(".')
```

engassim	<i>Segment list of a a sequence of syllable final n or N preceding k or g , isolated words single speaker, Australian English female from database epgassim.</i>
----------	--

Description

An EMU dataset

Usage

engassim

engassim.epg	<i>EPG-compressed trackdata from the segment list engassim</i>
--------------	--

Description

An EMU dataset

Usage

engassim.epg

engassim.l	<i>Vector of phonetic labels from the segment list engassim: $nK = nk, ng$, $sK = sk, sg$</i>
------------	--

Description

An EMU dataset

Usage

engassim.l

engassim.w

Vector of word labels from the segment list engassim.

Description

An EMU dataset

Usage

engassim.w

EPG contact indices

Electropalatographic contact indices

Description

epgai(), epgci(), epgdi() return the anteriority index, the centrality index, the dorsopalatal index respectively as a trackdata object or a vector

Usage

```
epgai (epgdata, weights = c(1, 9, 81, 729, 4921))
epgci (epgdata, weights = c(1, 17, 289, 4913))
epgdi (epgdata)
```

Arguments

epgdata	An eight-columned EPG-compressed trackdata object, or an eight columned matrix of EPG-compressed trackdata, or a 3D palatographic array that is the output of palate()
weights	A vector of five values that are applied to EPG rows 1-5 respectively in epgai(). A vector of four values that are applied to columns 1 and 8, to columns 2 and 7, columns 3 and 6, columns 4 and 5 respectively. Defaults to the values given in Recasens & Pallares (2001).

Details

These are exact implementations of the formulae for calculating the EPG anteriority, EPG centrality, and EPG dorsopalatal indices as described in Recasens & Pallares (2001).

Value

These functions return a trackdata object if they are applied to an eight-columned EPG-compressed trackdata object, otherwise a one-columned matrix.

Author(s)

Jonathan Harrington

References

GIBBON, F. AND NICOLAIDIS, K. (1999). Palatography. In W.J. Hardcastle & N. Hewlett (eds). Coarticulation. (pp. 229-245). Cambridge University Press: Cambridge.

RECASENS, D. & PALLARES, M. (2001) Coarticulation, assimilation and blending in Catalan consonant clusters. *Journal of Phonetics*, 29, 273-301.

See Also

[epgcog](#) [epggs](#) [palate](#)

Examples

```
# Anteriority index: trackdata
ai <- epgai(coutts.epg)
# Dorsopalatal index, one-columned matrix
di <- epgdi(dcut(coutts.epg, 0.5, prop=TRUE))
# Next to examples: Centrality index, one-columned matrix
ci <- epgci(palate(coutts.epg))
ci <- epgci(palate(dcut(coutts.epg, 0.5, prop=TRUE)))
```

 epgcog

Electropalatographic centre of gravity

Description

Calculate the centre of gravity in palatographic data.

Usage

```
epgcog (epgdata, weights = seq(7.5, 0.5, by = -1), rows = 1:8,
       columns = 1:8, row1 = NULL)
```

Arguments

epgdata	An eight-columned EPG-compressed trackdata object, or an eight columned matrix of EPG-compressed trackdata, or a 3D palatographic array that is the output of <code>palate()</code>
weights	A vector of 8 values that are applied to EPG rows 1-8 respectively. Defaults to 7.5, 7.0, 6.5...0.5.
rows	Calculate EPG-COG over selected row number(s). <code>rows = 5:8</code> , <code>columns = 3:6</code> is an implementation of posterior centre of gravity, as defined by Gibbon & Nicolaidis (1999,p. 239). See examples below.

columns	Calculate EPG-COG over selected column number(s).
row1	an optional single valued numeric vector to allow a separate weighting of the electrodes in row1. For example, if row1=4/3, then all the electrodes in row1 are multiplied by that value, before EPG-COG is calculated. Defaults to NULL (no weighting).

Details

The centre of gravity is a key function in palatographic research and gives an value per palate that is indicative of the overall location of contacts along the anterior-posterior dimension. The formula is an implementation of the ones discussed in Hardcastle et al. (1991), Gibbon et al (1993), and Gibbon & Nicolaidis (1999).

Value

These functions return a trackdata object if they are applied to an eight-columned EPG-compressed trackdata object, otherwise a one-columned matrix.

Author(s)

Jonathan Harrington

References

- GIBBON, F., HARDCASTLE, W. and NICOLAIDIS, K. (1993) Temporal and spatial aspects of lingual coarticulation in /kl/ sequences: a cross-linguistic investigation. *Language & Speech*, 36, 261-277.
- GIBBON, F. AND NICOLAIDIS, K. (1999). Palatography. In W.J. Hardcastle & N. Hewlett (eds). *Coarticulation*. (pp. 229-245). Cambridge University Press: Cambridge.
- HARDCASTLE, W, GIBBON, F. and NICOLAIDIS, K. (1991) EPG data reduction methods and thier implications for studies of lingual coarticulation. *Journal of Phonetics*, 19, 251-266.

See Also

[epgai](#) [epgsum](#) [palate](#)

Examples

```
# COG: trackdata
cog <- epgcog(coutts.epg)
# cog, one-columned matrix
cog <- epgcog(dcut(coutts.epg, 0.5, prop=TRUE))
# posterior cog for Fig. 10.5, p. 239 in Gibbon & Nicolaidis (1999)
r = array(0, c(8, 8, 2))
r[6,c(1, 8),1] <- 1
r[7,c(1, 2, 7, 8), 1] <- 1
r[8, ,1] <- 1
r[4, c(1, 2, 8), 2] <- 1
r[5, c(1, 2, 7, 8), 2] <- 1
r[6, c(1, 2, 3, 7, 8), 2] <- 1
```

```
r[7:8, , 2] = 1
class(r) <- "EPG"
epgcog(r, rows=5:8, columns=3:6)
```

epggs

Plot a grey-scale image of palatographic data.

Description

The function plots a grey-scale image of palatographic data such that the greyness in cell r, c is in proportion to the frequency of contacts in cells of row r and columns c of all palatograms in the object passed to this function.

Usage

```
epggs (epgdata, gscale = 100, gridlines = TRUE, gridcol = "gray",
      gridlty = 1, axes = TRUE, xlab = "", ylab = "", ...)
```

Arguments

epgdata	An eight-columned EPG-compressed trackdata object, or an eight columned matrix of EPG-compressed trackdata, or a 3D palatographic array that is the output of <code>palate()</code>
gscale	a single valued numeric vector that defines the granularity of the greyscale. Defaults to 100.
gridlines	if T (default) grid lines over the palatographic image are drawn are drawn.
gridlty	A single-valued numeric vector that defines the linetype for plotting the grid.
gridcol	color of grid
axes	T for show axes, F for no axes
xlab	A character vector for the x-axis label.
ylab	A character vector for the y-axis label.
...	graphical parameters can be given as arguments to <code>'epggs'</code> .

Details

The function plots a grey-scale image of up to 62 values arranged over an 8 x 8 grid with columns 1 and 8 unfilled for row 1. If cell row r column c is contacted for all palatograms in the object that is passed to this function, the corresponding cell is black; if none of of the cells in row r column c are contacted, then the cell is white (unfilled).

Author(s)

Jonathan Harrington

See Also

[epgai](#) [epgcog](#) [epgplot](#) [palate](#)

Examples

```
# greyscale image across the first two segments 'just relax'
# with title
epggs(coutts.epg[1:2,], main="just relax")

# as above but with dotted gridlines in blue
epggs(coutts.epg[1:2,], main="just relax", gridlty=2, gridcol="blue")

# as the first example, but with greyscale set to 2
epggs(coutts.epg[1:2,], 2, main="just relax")

# get palatograms for "S" from the polhom.epg database
temp = polhom.l == "S"
# greyscale image of all "S" segments at their temporal midpoint
epggs(dcut(polhom.epg[temp,], 0.5, prop=TRUE))

# greyscale image of all "S" segments from their onset to offset
epggs(polhom.epg[temp,])

# the same but derived from palates
p <- palate(polhom.epg[temp,])
epggs(p)
```

epgplot

Plot palatographic data

Description

Function to plot palatograms from EPG compressed objects or from a 3D-palatographic array that is output from `palate()`.

Usage

```
epgplot(epgdata, select = NULL, numbering = "times", gridlines = TRUE,
        mfrow = NULL, col = 1, mar = c(0.8, 0.1, 0.8, 0.1), xlim = NULL)
```

Arguments

epgdata	An eight-columned EPG-compressed trackdata object, or an eight columned matrix of EPG-compressed trackdata, or a 3D palatographic array that is the output of <code>palate()</code>
select	A vector of times. Palatograms are plotted at these times only. Note: this argument should only be used if <code>epgdata</code> is temporally contiguous, i.e. the entire trackdata object contains palatograms at successive multiple times of the EPG

	sampling frequency. (as in <code>coutts.epg\$time</code>). Defaults to NULL, in which case palatograms are plotted for all times available in <code>epgdata</code> .
numbering	Either "times" (default), or logical T, or a character vector of the same length as the number of segments in <code>epgdata</code> . In the default case, the times at which the palatograms occur are printed above the palatograms. If logical T, then the palatograms are numbered 1, 2, ... number of segments and this value is printed above the palatograms. If a character vector, then this must be the same length as the number of segments in <code>epgdata</code> .
gridlines	if T (default) grid lines over the palatogram are drawn.
mfrow	By default, the function tries to work out a sensible number of rows and columns for plotting the palatograms. Otherwise, this can be user-specified, in which case <code>mfrow</code> is a vector of two integer numeric values.
xlim	A numeric vector of two time values over which the <code>epgdata</code> should be plotted. Note: this argument should only be used if <code>epgdata</code> is temporally contiguous, i.e. the entire trackdata object contains palatograms at successive multiple times of the EPG sampling frequency. (as in <code>coutts.epg\$time</code>). Defaults to NULL (plot all time values).
col	specify a colour for plotting the filled EPG cells.
mar	A numerical vector of the form 'c(bottom, left, top, right)' which gives the number of lines of margin to be specified on the four sides of the plot. The default in this function is <code>c(0.8, 0.1, 0.8, 0.1)</code> . (The default in the R <code>plot()</code> function is <code>c(5, 4, 4, 2) + 0.1</code>).

Details

The function plots 62 values arranged over an 8 x 8 grid with columns 1 and 8 unfilled for row 1. When there is a contact (1), the corresponding rectangle of the grid is filled otherwise the rectangle is empty.

Author(s)

Jonathan Harrington

See Also

[epgai](#) [epgcog](#) [epggs](#) [palate](#)

Examples

```
epgplot(polhom.epg[10,])

# as above but between times 1295 ms and 1330 ms
epgplot(polhom.epg[10,], xlim=c(1295, 1330))

# the same as above, but the data is first
# converted to a 3D palatographic array
p <- palate(polhom.epg[10,])
epgplot(p, xlim=c(1295, 1330))
```

```

# plot palatograms 2 and 8
epgplot(p[,c(2, 8)])

# as above but
# no gridlines, different colour, numbering rather than times
epgplot(p[,c(2, 8)], gridlines=FALSE, col="pink", numbering=TRUE)

# as above but with a user-specified title

epgplot(p[,c(2, 8)], gridlines=FALSE, col="pink", numbering=c("s1", "s2"))

# plot the palatograms in the second
# segment of coutts.epg that are closest in time
# to 16377 ms and 16633 ms
epgplot(coutts.epg[2,], c(16377, 16633))

```

epgsum	<i>Sum contacts in palatograms.</i>
--------	-------------------------------------

Description

The function calculates EPG contact profiles, i.e. sums active or inactive electrodes optionally by row and/or column in palatographic data.

Usage

```
epgsum (epgdata, profile = c(1, 3), inactive = FALSE, rows = 1:8,
        columns = 1:8, trackname = "EPG-sum")
```

Arguments

epgdata	An eight-columned EPG-compressed trackdata object, or an eight columned matrix of EPG-compressed trackdata, or a 3D palatographic array that is the output of <code>palate()</code>
profile	A numeric vector of one or two values. The options are as follows. <code>c(1,3)</code> and <code>c(1)</code> sum the contacts by row, but the latter outputs the summation in the rows. <code>c(2,3)</code> and <code>c(2)</code> sum the contacts by column, but the latter outputs the summation in the columns. (see also rows and columns arguments and the examples below for further details).
inactive	a single element logical vector. If F (the default), then the active electrodes (i.e., 1s) are summed, otherwise the inactive electrodes (i.e., 0s) are summed.
rows	vector of rows to sum
columns	vector of columns to sum
trackname	single element character vector of the name of the track (defaults to "EPG-sum")

Details

Contact profiles are standard tools in electropalatographic analysis. See e.g., Byrd (1996) for details.

Value

These functions return a trackdata object if they are applied to an eight-columned EPG-compressed trackdata object, otherwise a one-columned matrix.

Author(s)

Jonathan Harrington

References

BYRD, D. (1996). Influences on articulatory timing in consonant sequences. *Journal of Phonetics*, 24, 209-244.

GIBBON, F. AND NICOLAIDIS, K. (1999). Palatography. In W.J. Hardcastle & N. Hewlett (eds). *Coarticulation*. (pp. 229-245). Cambridge University Press: Cambridge.

See Also

[epgai](#) [epgcog](#) [epggs](#) [palate](#)

Examples

```
# Trackdata object of the sum of contacts in the 1st segment of polhom.epg
epgsum(polhom.epg[1,])
# as above, but the summation is in rows 1-3 only.
epgsum(polhom.epg[1,], rows=c(1:3))
# as epgsum(polhom.epg[1,]), except sum the inactive electrodes in columns 3-6.
epgsum(polhom.epg[1,], columns=3:6, inactive=TRUE)
# Obtain compressed EPG-trackdata object for the 1st four segments of polhom.epg
# at the temporal midpoint
mid <- dcut(polhom.epg[1:4,], .5, prop=TRUE)
# sum of contacts in these four palatograms.
epgsum(mid)
# gives the same result as the previous command.
p <- palate(mid)
# sum the contacts in the palatograms.
epgsum(p)
# as above, but show the separate row summations.
epgsum(p, 1)
# as above, but show the separate column summations.
epgsum(p, 2)
# sum of the contacts in rows 1-4 showing the separate row summations.
epgsum(p, 1, rows=1:4)
# sum of the contacts in rows 1-4 showing the separate column summations.
epgsum(p, 2, rows=1:4)
# sum of the contacts in columns 3-6 showing the separate row summations.
epgsum(p, 1, columns=3:6)
# sum of the contacts in columns 3-6 showing the separate column summations.
```

```
epgsum(p, 2, columns=3:6)
```

eplot

Plot ellipses for two-dimensional data

Description

The function plots ellipses for different categories from two-dimensional data.

Usage

```
eplot (x, labs, chars, formant = FALSE, scaling = "linear",
      prob = 0.95, nsdev = NULL, dopoints = FALSE, doellipse = TRUE,
      centroid = FALSE, axes = TRUE, xlim, ylim, col = TRUE, lty = FALSE,
      lwd = NULL, ...)
```

Arguments

x	A two-columned matrix of data
labs	An optional vector of labels, parallel to 'data'
chars	An optional vector of labels, parallel to 'data'. If this argument is specified these labels will be plotted rather than the labels in 'labs'.
formant	If TRUE) then the data is negated and the axes are switched so that, for formant data, the plot is made with decreasing F2 on the x-axis and decreasing F1 on the y-axis.
scaling	Either "mel" or "bark" for mel or bark scaling of the data
prob	A single numeric vector greater than zero and less than 1 representing the confidence interval of the ellipse contours. Defaults to 0.95
nsdev	Defines the length of the major and minor axes of the ellipses in terms of the standard deviation of the data and overrides the prob argument.
dopoints	If TRUE) character labels (from 'labs' or 'chars') are plotted for each data point
doellipse	If TRUE, ellipses are drawn on the plot. If FALSE, no ellipses are drawn and, if 'dopoints' is also FALSE, 'centroids' is set to T
centroid	One label for each ellipse is drawn
axes	If TRUE axes are drawn on the plot.
xlim	A vector of two numeric values giving the range of the x-axis.
ylim	A vector of two numeric values giving the range of the y-axis.
col	If colour is TRUE) the ellipses and labels will be plotted in different colours
lty	If linetype is TRUE) the ellipses will be plotted with different linetypes. This is useful for plots that will be printed.

`lwd` A code passed to the `lwd` argument in plotting functions. 'lwd' can be either a single element numeric vector, or its length must be equal to the number of unique types in `labs`. For example, if `lwd=3` and if `labs = c("a", "b", "a", "c")`, then the output is `c(3, 3, 3, 3)`. Alternatively, if `lwd = c(2,3,1)`, then the output is `c(2, 3, 2, 1)` for the same example. The default is `NULL` in which case all lines are drawn with `lwd=1`

... graphical options [par](#)

Value

`NULL`

Author(s)

Jonathan Harrington jmh@ipds.uni-kiel.de, Steve Cassidy, Gordon Watson

See Also

[dcut](#)

Examples

```
data(vowlax)
data <- cbind(vowlax.df$F1,vowlax.df$F2)
phonetic = vowlax.l
word = vowlax.word
```

```
eplot(data, phonetic)
```

```
eplot(data, phonetic, form=TRUE, main="F1 x F2 plane", centroid=TRUE)
eplot(data, phonetic, form=TRUE, main="F1 x F2 plane", dopoints=TRUE)
eplot(data, phonetic, form=TRUE, main="F1 x F2 plane in Bark", dopoints=TRUE, scaling="bark")
eplot(data, phonetic, form=TRUE, main="F1 x F2 plane in Bark b/w with linetype", col=FALSE, lty=TRUE, dopoints=TRUE)
eplot(data, phonetic, form=TRUE, main="F1 x F2 plane", doellipse=FALSE, dopoints=TRUE)
eplot(data, phonetic, form=TRUE, dopoints=TRUE, prob=0.5, main="F1 x F2 plane, 50% confidence intervals")
eplot(data, phonetic, form=TRUE, dopoints=TRUE, nsdev=2, main="F1 x F2 plane, 2 standard deviations")
```

```
temp <- phonetic %in% c("a", "0")
eplot(data[temp,], phonetic[temp], form=TRUE, main="F1 x F2 [A] and [0] only", centroid=TRUE)
```

```
temp <- phonetic=="0"
eplot(data[temp,], phonetic[temp], word[temp], form=TRUE, dopoints=TRUE, main="[0] only showing word labels")
```

euclidean	<i>Find the inter-euclidean distance for a data matrix</i>
-----------	--

Description

Finds the inter-euclidean distance for a data matrix

Usage

```
euclidean(data, m=1, n=ncol(data))
```

Arguments

data	A vector or matrix of numerical data.
m	The first column of data to be used in the distance calculation.
n	The last column of data to be used in the distance calculation.

Value

Calculates the euclidean distance between successive rows of the matrix based on columns m:n.

See Also

steady

Examples

```
euclidean(cbind(c(1,2,3,4), c(2,3,2,2)))
```

expand_labels	<i>Label each data sample</i>
---------------	-------------------------------

Description

Labels each data sample

Usage

```
expand_labels(indvals, labs)
```

Arguments

indvals	Index component of a trackdata object as returned by frames, or track.
labs	A label vector parallel to indvals.

Value

Returns a vector of labels, one for each row in the data matrix that corresponds to indivals.

See Also

frames, track

f2geraus

Trackdata of formants of i: vowels from one male speaker of Australian English and one speaker of Standard North German.

Description

An EMU dataset

Usage

f2geraus

f2geraus.l

Vector of labels parallel to f2geraus to identify the language.

Description

An EMU dataset

Usage

f2geraus.l

fapply

Function that applies a function to an EMU spectral object

Description

Applies a function to an EMU spectral object.

Usage

```
fapply(specdata, fun, ..., power = FALSE, powcoeffs = c(10, 10))
```

Arguments

specdata	A matrix or trackdata object of class spectral
fun	A function to be applied.
...	Optional arguments to fun
power	A single element logical vector. If T, convert specdata to power values i.e. apply the function to $a * \text{specdata}^b$ or $a * \text{specdata}^{\$data}^b$ where a and b powcoeffs defined below.
powcoeffs	A 2 element numeric vector for converting dB values to power values. Defaults to a = 10 and b = 10. See power.

Details

fapply performs a similar operation to apply except that it is specifically designed for handling EMU spectral objects.

Value

If the output has the same dimensions as the input, then an object of the same dimensionality and class is returned. Otherwise it may be a vector or matrix depending on the function that is applied.
...

Warning

The function can be very slow if applied to a large trackdata object. In this case, it may be faster to use a for-loop with the desired function around `\$data`

Author(s)

Jonathan Harrington

See Also

[apply by trackdata](#)

Examples

```

# mean value per spectrum, input is a spectral matrix
m <- fapply(vowlax.dft.5, mean)
# as above but after converting dB to powers before
# applying the function
m <- fapply(vowlax.dft.5, mean, power=TRUE)
# spectral range
r <- fapply(vowlax.dft.5, range)
# spectral moments applied to a trackdata object
# m is a four-dimensional trackdata object
m <- fapply(fric.dft, moments)
# 1st 3 DCT coefficients calculated in a spectral matrix
# d is a 3-columned matrix
d <- fapply(vowlax.dft.5, dct, 3)
# dct-smooth with 10 coefficients. d2 is spectral matrix
d2 <- fapply(vowlax.dft.5, dct, 10, TRUE)
# dct-smooth a trackdata object with 10 coefficients
d3 <- fapply(fric.dft[1:4,], dct, 10, TRUE)

```

fr	<i>Segment list 200 German fricatives, 2 speakers, one male, one female</i>
----	---

Description

An EMU dataset

Examples

```
data(fr)
```

fr.dft	<i>Spectral trackdata object</i>
--------	----------------------------------

Description

calculated with a 512 point DFT and frameshift of 5 ms

Examples

```
data(fr.dft)
```

fr.l	<i>a Vector of fricative labels</i>
------	-------------------------------------

Description

An EMU dataset

Examples

```
data(fr.l)
```

fr.sp	<i>A vector of speaker labels</i>
-------	-----------------------------------

Description

An EMU dataset

Examples

```
data(fr.sp)
```

frames	<i>frames</i>
--------	---------------

Description

Get frames from trackdata objects

Usage

```
frames(trackdata)
```

Arguments

trackdata an object of class trackdata

Value

Data frames from the input object.

Author(s)

Jonathan Harrington

See Also

[trackdata](#)

frames.time	<i>Find the time and position of a data element.</i>
-------------	--

Description

Finds the time and position of a data element.

Usage

```
frames.time(dataset, datanum)
```

Arguments

dataset	A dataset returned by track or frames.
datanum	An integer, an index into the data component of dataset.

Details

The dataset returned from track or frames consists of a matrix of data (the data component) and two index components (index and ftime). The data for all segments is concatenated together in \$data. This function can be used to find out which segment a particular row of \$data corresponds to.

Value

The segment number which contains the element datanum of dataset\$data.

See Also

track, frames

freqtoint	<i>Function to find the column number corresponding to frequencies of a spectral object</i>
-----------	---

Description

Find the column number corresponding to frequencies of a spectral object.

Usage

```
freqtoint(trackdata, j)
```

Arguments

trackdata	A spectral object
j	A vector of frequencies

Details

This function is used in conjunction with object oriented programming of EMU spectral objects. It should not in general be called from inside a function. Its principal use is to determine the column number(s) corresponding to frequencies for spectral trackdata objects or spectral matrices or the element number for spectral vectors.

Author(s)

Jonathan Harrington

Examples

```
freqtoint(fric.dft,1000:2000)
# all frequencies except 1000-2000
freqtoint(vowlax.dft.5, -(1000:2000))
# all frequencies except 1000 Hz
freqtoint(e.dft, -1000)
# the d.c. offset - i.e. column 1
freqtoint(vowlax.dft.5, 0)
# all freqs except the d.c. offset - i.e. not column 1
freqtoint(vowlax.dft.5, -1)
```

fric

Segment list of word-medial s or z one male speaker of Standard North German, read speech from database kielread.

Description

An EMU dataset

Usage

```
fric
```

fric.dft	<i>Spectral trackdata object from the segment list fric.</i>
----------	--

Description

An EMU dataset

Usage

fric.dft

fric.l	<i>Vector of labels from the segment list fric</i>
--------	--

Description

An EMU dataset

Usage

fric.l

fric.w	<i>Vector of word labels from the segment list fric.</i>
--------	--

Description

An EMU dataset

Usage

fric.w

get.time.element *Get data for a given time*

Description

Gets data for a given time

Usage

```
get.time.element(timeval, dataset)
```

Arguments

timeval A time in milliseconds
dataset A trackdata object as returned by track.

Value

The element number of trackdata\$data corresponding to time

See Also

track, frames

is.spectral *Function to test whether the object is of class "spectral"*

Description

Returns T or F depending on whether the object is of class "spectral"

Usage

```
is.spectral(dat)
```

Arguments

dat An R object

Value

A single element logical vector: T or F

Author(s)

Jonathan Harrington

See Also[as.spectral](#)**Examples**

```

is.spectral(vowlax.dft.5)
is.spectral(fric.dft)
is.spectral(fric.dft$data)
is.spectral(vowlax.dft.5[1,])
is.spectral(fric.dft[1,1])

```

<code>is.trackdata</code>	<i>Test whether an object is an Emu trackdata object</i>
---------------------------	--

Description

Test whether an object is an Emu trackdata object

Usage

```
is.trackdata(object)
```

Arguments

`object` A data object to be tested

Value

Returns TRUE if the argument is a trackdata object.

See Also[emu.track](#)

<code>isol</code>	<i>Segment list of vowels in a d d context isolated word speech, one male speaker of Australian English from database isolated.</i>
-------------------	---

Description

An EMU dataset

Usage

```
isol
```

isol.fdat	<i>Trackdata of formants from the segment list isol</i>
-----------	---

Description

An EMU dataset

Usage

isol.fdat

isol.l	<i>Vector of vowel phoneme labels from the segment list isol</i>
--------	--

Description

An EMU dataset

Usage

isol.l

keng	<i>Segment list of the aspiration of syllable-initial k preceding front and back vowels, two speakers of Australian English, read speech from database andosl.</i>
------	--

Description

An EMU dataset

Usage

keng

keng.dft.5	<i>Spectral matrix extracted at the segment's temporal midpoint from the segment list keng.</i>
------------	---

Description

An EMU dataset

Usage

keng.dft.5

keng.l	<i>Vector of labels designating whether the following vowel is front or back from the segment list keng</i>
--------	---

Description

An EMU dataset

Usage

keng.l

label	<i>Get labels / utterances from segment list</i>
-------	--

Description

label: extracts the labels from the segment list. utt: extracts the utterances from the segment list.

Usage

```
## S3 method for class 'emusegs'  
label(segs)  
## S3 method for class 'emusegs'  
utt(x)
```

Arguments

segs	segment list
x	segment list

Value

label / utterance vector

Author(s)

Jonathan Harrington

See Also

[segmentlist](#) [start](#) [end](#)

Examples

```
data(dip)
#dip is a segment list - first ten segments only
dip[1:10,]

#extract labels from the segment list
dips.labs = label(dip)
dips.labs
```

linear	<i>Perform linear time normalisation on trackdata.</i>
--------	--

Description

Performs linear time normalisation on trackdata.

Usage

```
linear(dataset, n=20)
```

Arguments

dataset	A trackdata object as returned by track.
n	The number of points (samples) required for each segment.

Details

The data for each segment is normalised using the approx function.

Value

A new trackdata object where the data for each segment has the same number (n) of samples.

See Also

approx

locus *Calculate locus equations for two-dimensional data*

Description

The function plots a locus equation and returns associated statistical information.

Usage

```
locus (target, onset, labels.vow = NULL, yxline = TRUE, plotgraph = TRUE, axes = TRUE, ...)
```

Arguments

target	a numerical vector typically of F2 values at the vowel target
onset	a numerical vector typically of the same length as target of F2 values at the vowel onset
labels.vow	an optionally character vector for plotting labels at the points (target, onset) of the same length as target
yxline	optionally plot the line target = onset. Defaults to True.
plotgraph	a logical vector for specifying whether the data should be plotted. Defaults to True.
axes	A logical vector indicating whether the axes should be plotted
...	graphical options par

Details

A locus equation is a straight line regression fitted with `lm()` in which the F2- values typically at the vowel onset are regressed on those of the target. The slope can be used to give an indication of target-on-onset coarticulatory influences.

The best estimate of the locus frequency is where the locus equation bisects the line target = onset.

Value

A list containing regression diagnostics of the function `lm()` that can be accessed with `summary()` and the estimated locus frequency in `\$locus`. A plot of values in the onset x target plane with superimposed locus equation and line onset=target.

Author(s)

Jonathan Harrington

Examples

```
# calculate an F2-locus equation for initial [d]
# preceding lax vowels produced by female speaker "68".
# the onset is taken at the vowel onset; the
# vowel target is taken at the vowel's temporal midpoint.

# identify initial "d" of speaker "68"
temp <- vowelax.left == "d" & vowelax.spkr == "68"
# get the F2 value at the vowel's temporal midpoint
targ <- dcut(vowelax.fdat[temp,2], .5, prop=TRUE)
# F2 value at the vowel's acoustic onset.
on <- dcut(vowelax.fdat[temp,2], 0, prop=TRUE)

# locus equation plot
result <- locus(targ, on, vowelax.l[temp])
# statistical diagnostics of the regression line (locus equation)
summary(result)
# intercept and slope
result$coeff
# best estimate of the locus frequency, i.e. the
# point of bisection of on = TRUEarg with the regression line
result$locus
```

 mahal

Classify using Mahalanobis distance

Description

Classifies using Mahalanobis distance

Usage

```
mahal(data, train)
```

Arguments

data	A vector or matrix of data
train	A Gaussian model generated by train.

Details

The model argument contains the mean and inverse covariance matrix (or standard deviation if the data is one-dimensional) for each class in the training set as well as the class labels. This function calculates the Mahalanobis distance of each row of data from each class mean and assigns the label of the closest mean to that row. The result is a vector of labels corresponding to the rows of data.

The Mahalanobis distance between a data point and a class is the Euclidean distance between the point and the class mean divided by the covariance matrix for the class. This means that classes with large covariances will attract data points from a larger area than those with small covariances.

Value

A label vector with one element per row of data

References

O'Shaughnessy, D. Speech Communication (Addison-Wesley: Reading, MA. 1987)

See Also

train

mahal.dist	<i>Calculate mahalanobis distances</i>
------------	--

Description

Calculates mahalanobis distances

Usage

```
mahal.dist(data, train, labels = NULL)
```

Arguments

data	A matrix of numerical data points.
labels	A vector of labels..
train	A gaussian model as returned by the train function.

Details

The train function finds the centroids and covariance matrices for a set of data and corresponding labels: one per unique label. This function can be used to find the mahalanobis distance of every data point in a dataset to each of the class centroids. The columns of the resulting matrix are marked with the label of the centroid to which they refer. The function mahal should be used if you want to find the closest centroid to each data point.

Value

A matrix of distances with one column for every class (label) in the gaussian model.

See Also

train, mahal, bayes.lab, bayes.dist

make.seglist	<i>Make an Emu segment list from the various components</i>
--------------	---

Description

This is the appropriate way to make an Emu segment list and ensure that it has all of the required components.

Usage

```
make.seglist(labels, start, end, utts, query, type, database)
```

Arguments

labels	A character vector of labels for each segment
start	A vector of start times
end	A vector of end times
utts	A character vector of utterance names
query	A query string
type	segment or event
database	The database name associated with the segment list

Details

An Emu segment list is the result of a query to a speech database (see [emu.query](#)) and has one row per matching segment or event from the query. Each row lists the label, start and end times (in milliseconds) and utterance name for the segment. This information is used by [emu.track](#) and other functions to extract data corresponding to these segments.

In order to ensure the proper format for segment lists and to ensure against future changes to the format, `make.seglist` should be used whenever you wish to create a segment list. Another function, [modify.seglist](#) can be used to change some part of an existing segment list. The functions [label.emusegs](#), [start.emusegs](#), [end.emusegs](#) and [utt.emusegs](#) can be used to access the different columns of the segment list.

Value

An Emu segment list.

Author(s)

Steve Cassidy

See Also

[modify.seglist](#), [label.emusegs](#)

Examples

```

l <- c("A", "B", "C")
s <- 1:3
e <- 2:4
u <- c("u1", "u1", "u1")
segs <- make.seglist(l, s, e, u, "Fake Query", "segment", "fake")
segs
## summary gives an overview of the data in the segment list
summary(segs)

# The following should be TRUE
label(segs) == l
dur(segs) == s
end(segs) == e
utt(segs) == u
emusegs.database(segs) == "fake"
emusegs.type(segs) == "segment"
emusegs.query(segs) == "Fake Query"

# segment durations should all be 1
dur(segs) == c(1,1,1)

```

makelab

Write out ESPS-style label files

Description

Writes out separate ESPS-label files for each utterance to a specified directory.

Usage

```
makelab (vectimes, uttname, dir, extn = "xlab", labels = NULL)
```

Arguments

vectimes	a vector of times
uttname	a character vector of the same length as vectimes giving the utterance name associated with each element of vectimes
dir	a character specifying the directory
extn	a character specifying the extension of the resulting files. Defaults to xlab
labels	either a single character vector or a character vector the same length as vectimes. Defaults to "T"

Value

ESPS-style label files are written out to the directory of the user's choice. One ESPS-label file is created for each utterance containing all time values for that utterance.

Author(s)

Jonathan Harrington

Examples

```
#first two segments (for the whole example) of segmentlist vowlax
vowlax[1:2,]

#format track of vowlax
vowlax.fdat[1:2,]

#Formant values of the midpoint of the segment
vowlax.fdat.5 = dcut(vowlax.fdat,0.5,prop=TRUE)

#the time marks of the midpoint of the segment
times = vowlax.fdat.5[1:2,1]
times

#utterance names to the segments
utts = utt(vowlax[1:2,])
utts

#the path to save the label files to "." is the RHOME Directory
path = "."

#write the label files to path
## Not run: makelab(times, utts, path, labels="T")

#the first two segments are from the same utterance,
#thus one label file was created in the R_HOME directory
```

matscan

Read matrix data from a file

Description

Reads matrix data from a file

Usage

```
matscan(file, num.cols=count.fields(file)[1], what=0, sk=0)
```

Arguments

file	A filename.
num.cols	The number of columns of data in the file.
what	A template for the data elements in the file, it should be a number for numeric data (the default) or a string for string data. Note that an Splus matrix can only hold one type of data (string or numeric), for mixed types use data tables and the read.table function.
sk	The number of leading lines of the file to skip.

Details

This function has been partially superceded by the introduction of data frames and the read.table function. It is still useful however for reading data into Splus matrix objects.

Value

A matrix corresponding to the data in file.

See Also

read.table

mel	<i>Convert Hz to the mel scale</i>
-----	------------------------------------

Description

The calculation is done using the formulae $mel = 1/\log(2) * (\log(1 + (Hz/1000))) * 1000$ where Hz is the frequency in Hz.

Usage

mel(a)

Arguments

a A vector or matrix of data or a spectral object.

Details

If 'data' is a spectral object, then the frequencies are changed so that they are proportional to the mel scale and such that the mel intervals between frequencies are constant between the lowest and highest frequencies. More specifically, suppose that a spectral object has frequencies at 0, 1000, 2000, 3000, 4000 Hz. Then the corresponding frequencies extend in mel between 0 and 2321.928 mel (=4000 Hz in mels) in four equal intervals, and linear interpolation is used with the 'approx' function to obtain the dB values at those frequencies.

Value

A vector or matrix or spectral object of the same length and dimensions as data.

Author(s)

Jonathan Harrington

References

Traunmueller, H. (1990) "Analytical expressions for the tonotopic sensory scale" J. Acoust. Soc. Am. 88: 97-100.

See Also

[bark](#), [plot.spectral](#)

Examples

```
#convert Hertz values to mel

vec <- c(500, 1500, 2500)
vec
mel(vec)

# convert Hertz values to mel

mel(vec)

# convert the $data values in a trackdata object to mel
# create a new track data object

t1 <- dip.fdat
t1[1]

# convert Hertz to mel

t1$data <- mel(t1$data)
t1[1]

# warp the frequency axis of a spectral object such
# that it is proportional to the mel scale.

w = mel(e.dft)
par(mfrow=c(1,2))
plot(w, type="l")

# The values of w are at equal mel intervals. Compare
# with
```

```
plot(e.dft, freq=mel(trackfreq(e.dft)))

# the latter has a greater concentration of values
# in a higher frequency range.
```

modify.seglist	<i>Modify one of the components of an Emu segment list</i>
----------------	--

Description

This function can be used to modify one of the parts of an Emu segment list while leaving the other parts unchanged.

Usage

```
modify.seglist(segs, labels, start, end, utts, query, type, database)
```

Arguments

segs	A segment list to modify, a modified copy is returned
labels	A new label vector
start	A new start time vector
end	A new end time vector
utts	A new vector of utterance labels
query	A new query string to associate with the segment list
type	A new type string
database	A new database name

Details

An Emu segment list has a number of components and is stored as an R object of class `emusegs`. This function can be used to modify a segment list while retaining all of the proper structures.

Any new vectors passed to the function must have the same length as the segment list itself for this call to succeed.

All arguments are optional and default to not modifying the segment list if not supplied.

The original segment list is not modified, instead, a modified copy is returned.

Value

An Emu segment list.

Author(s)

Steve Cassidy

See Also[emu.query](#)**Examples**

```

data(vowlax)
segs = vowlax
# extend the start times by 10ms
newsegs <- modify.seglist( segs, start=start(segs)+10 )

# change the associated database name
# this will affect where emu.track looks to find data
newsegs <- modify.seglist( segs, database="notdemo" )

```

moments

Function to calculate statistical moments

Description

The function calculates the first 4 moments, i.e. the mean, variance, skew, kurtosis.

Usage

```
moments(count, x, minval = FALSE)
```

Arguments

count	A vector of the observed instances per class
x	A vector of the same length as count defining the class. If missing, and if count is of class spectral, then x is equal to trackfreq(count). If x is missing and is not of class spectral, then x default to 0:(length(count)-1)
minval	If T, subtract min(count) from count so that the minimum value of count is zero. This is principally used in calculating spectral moments where count is in decibels, and more generally if count contains negative values.

Details

The units of the first moment are the same as x, the units of the second moment are x^2 , and the third and fourth moments are dimensionless.

Author(s)

Jonathan Harrington

References

Snedecor, G & Cochran, W. 'Statistical Methods' Iowa State Press. Wuensch,K., 2005

Examples

```
# first four moments of a vector
mom <- moments(bridge[,2])
# the above is the same as moments(bridge[,2], 0:12)
# first four moments of a spectral vector with the dB values
# reset so that the minimum dB value is 0. The d.c. offset is also
# excluded in the calculation
mom <- moments(e.dft[-1], minval=TRUE)
# the temporal skew of F1 for the 10th segment. Use
m <- moments(vowlax.fdat[10,1]$data)[3]
```

mu.colour	<i>Function for specifying color, linetype, and line-widths in EMU plotting functions.</i>
-----------	--

Description

The function specifies color, linetype and linewidths in EMU plotting functions as is used mostly in calls from within `plot.trackdata`, `plot.spectral`, `eplot`, and `dplot`

Usage

```
mu.colour(labs, col = TRUE, linetype = FALSE, lwd = NULL, pch = NULL)
```

Arguments

labs	A vector of character labels
col	A code passed to the 'col' argument in plotting functions. There are four possibilities. Either logical, a character vector, or a numeric vector. In the first case, if TRUE, then a different numeric code is given for each unique label type. For example, if labs is <code>c("a", "b", "a", "c")</code> , then the output is <code>c(1, 2, 1, 3)</code> . If FALSE, then for this example, the output is <code>c(1, 1, 1, 1)</code> . In the second case, the character vector can be either a single element specifying a character, or there can be as many elements as there are unique colors. Thus if <code>col = "red"</code> , then for the example <code>c("a", "b", "a", "c")</code> , the output is <code>c("red", "red", "red", "red")</code> . Alternatively, since there are three unique labels for this example, then the user could specify <code>col = c("green", "red", "blue")</code> and the output is <code>c("green", "red", "green", "blue")</code> if labs is <code>c("a", "b", "a", "c")</code> . In the third case, 'col' can be either a single element numeric vector, or its length must be equal to the number of unique types in labs. For example, if <code>col=3</code> and if labs = <code>c("a", "b", "a", "c")</code> , then the output is <code>c(3, 3, 3, 3)</code> . Alternatively, if <code>col = c(2,3,1)</code> , then the output is <code>c(2, 3, 2, 1)</code> for the same example. Finally, col can be specified as a character or numeric vector that is the same length as labs, allowing the user to choose the color in which each line should be drawn. The default is <code>col = TRUE</code> .

linetype	A code specifying linetypes, i.e. the values passed to lty in plotting functions. There are 2 possibilities. Either logical, a character vector, or a numeric vector. In the first case, if TRUE, then a different numeric code is given for each unique label type. For example, if labs is c("a", "b", "a", "c"), then the output is c(1, 2, 1, 3). If F, then for this example, the output is c(1, 1, 1, 1). In the second case, 'linetype' can be either a single element numeric vector, or its length must be equal to the number of unique types in labs. For example, if linetype=3 and if labs = c("a", "b", "a", "c"), then the output is c(3, 3, 3, 3). Alternatively, if linetype = c(2,3,1), then the output is c(2, 3, 2, 1) for the same example. Finally, linetype can be specified as a numeric vector that is the same length as labs, allowing the user to choose the linetype in which each line should be drawn. The default is linetype=F
lwd	A code passed to the lwd argument in plotting functions. 'lwd' can be either a single element numeric vector, or its length must be equal to the number of unique types in labs. For example, if lwd=3 and if labs = c("a", "b", "a", "c"), then the output is c(3, 3, 3, 3). Alternatively, if lwd = c(2,3,1), then the output is c(2, 3, 2, 1) for the same example. The default is NULL in which case all lines are drawn with lwd=1
pch	A code passed to the pch argument in plotting functions. Functions in the same way as lwd above

Details

Parameters are also supplied for use with the function 'legend'

Value

If it is a LISTTRUE, use

colour	A code for the color'
linetype	A code for the linetype
lwd	A code for the line width
legend	A list consisting of $\backslash\text{legend}\backslash\text{lab}$, $\backslash\text{legend}\backslash\text{lty}$ and $\backslash\text{legend}\backslash\text{lwd}$ that specify the parameters for the 'legend' function.
	...

Author(s)

Steve Cassidy, modified by Jonathan Harrington

See Also

[plot.trackdata](#) [dplot](#) [eplot](#) [plot.spectral](#)

Examples

```

# examples will be given using the above functions
# b/w but with different linetypes
eplot(vowlax.fdat.5[,1:2], vowlax.l, col=FALSE, lty=TRUE)

# user-defined colors
eplot(vowlax.fdat.5[,1:2], vowlax.l, col=c("green", "blue", "red", "orange"))

# spectral plot, user-defined colors, the last one is dotted
# and with a line-thickness of 2
plot(vowlax.dft.5[1:20,], vowlax.l[1:20],
col=c("green", "blue", "red", "orange"),
fun=mean, lty=c(1, 1, 1, 2), lwd=c(1, 1, 1, 2))

# similar but using dplot()
dplot(vowlax.fdat[1:20,2], vowlax.l,
col=c("green", "blue", "red", "orange"),
lwd=c(1, 1, 1, 2), lty=c(1, 1, 1, 2))

# the default except plot everything with a dotted line and plotting symbol 4
dplot(vowlax.fdat[,2], vowlax.l, average=TRUE, lty=2, pch=4, type="b", xlim=c(40, 60))

# the default except plot everything with a dotted line and
# with double line thickness
eplot(vowlax.fdat.5[,1:2], vowlax.l, lty=2, lwd=2)

```

muclass

Find common elements in vectors

Description

Finds common elements in vectors

Usage

```
muclass(labels, class)
```

Arguments

labels	A vector of labels.
class	A label or vector of labels.

Value

A logical vector which is T for each element in labels which matches class or an element of class.

See Also

match

Examples

```
muclass(c("a", "b", "c"), c("a", "c"))
#[1] T F T
```

norm	<i>Normalise speech data</i>
------	------------------------------

Description

Normalises speech data

Usage

```
norm(data, speakerlabs, type="gerst", rescale=FALSE)
```

Arguments

data	A matrix of data. Can be either an n-columned matrix or a trackdata object as returned by track.
speakerlabs	A parallel vector of speaker labels.
type	The type of extrinsic normalisation to be performed on data. type can be "nearey", "cen", "lob", "gerst" (default), for normalisation according to Nearey, centroid method, Lobanov, or Gerstman.
rescale	Currently only works for Lobanov normalisation. The normalised values are multiplied by the standard deviation and then the mean is added, where the standard deviation and mean are across all original speakers' unnormalised data.

Details

Types of normalisation: "nearey", Nearey : Find the log of each data element and subtract from each the mean of the logarithmic data. "cen", centroid: Find the mean of the data column and subtract it from each data element in that column. "lob", Lobanov: Find the mean and standard deviation of the data. Subtract the mean from each data element and divide each result by the standard deviation. "gerst", Gerstman: Subtract from the data the minimum formant value then divide by the formant range.

Value

Normalised values of data are returned, having the same structure as data.

See Also

track

palate	<i>Obtain a three-dimensional palatographic array</i>
--------	---

Description

Function to calculate a three-dimensional palatographic array from.

Usage

```
palate(egpdata)
```

Arguments

egpdata	An eight-columned EPG-compressed trackdata object or an eight columned matrix of EPG-compressed trackdata.
---------	--

Details

An EPG compressed trackdata object that is output from the Reading system contains eight columns of data and each row value when converted to binary numbers (after adding 1) gives the corresponding EPG contact patterns. This function does the conversion to binary values.

Value

An array of three dimensions of 8 rows x 8 columns x n segments where n is the number of segments in the trackdata object or matrix. The rows and columns are given dimension names, the dimension names of the third dimension contains the times at which the palatograms occur.

Author(s)

Jonathan Harrington

See Also

[epgcog](#) [epggs](#) [epgai](#) [eggplot](#)

Examples

```
# convert an EPG-compressed trackdata object to palatograms
p <- palate(coutts.epg)
```

```
# convert an EPG-compressed matrix to palatograms
p <- palate(dcut(coutts.epg, 0, prop=TRUE))
```

perform	<i>Performance (hit rate) of a confusion matrix</i>
---------	---

Description

Performs (hit rate) of a confusion matrix

Usage

```
perform(data)
```

Arguments

data	A confusion matrix.
------	---------------------

Value

Calculates the accuracy (total score) of the confusion matrix, returning percentage of correct, and incorrect matches.

See Also

confusion

plafit	<i>Calculate the coefficients of a parabola</i>
--------	---

Description

Fit a second ordered polynomial to a vector of values

Usage

```
plafit(wav, fit = FALSE, n = 101)
```

Arguments

wav	a vector or single column matrix of numeric values to which the 2nd order polynomial is to be fitted.
fit	if F, return the coefficients of the polynomial; if T, the values of the polynomial are returned to the same length as the vector wav.
n	in fitting the polynomial, linear time normalisation is first applied to the input vector wav to 101 points. The polynomial is fitted under the assumption that these points extend linearly in time between t = -1 and t = 1 with t = 0 occurring at the temporal midpoint.

Details

The function fits a parabola (2nd order polynomial) following the method of van Bergem, Speech Communication, 14, 1994, 143-162. The algorithm fixes the parabola at the onset, midpoint, and offset of the vector i.e. such that the fitted parabola and original vector have the same values at these points.

Value

The function returns the coefficients of c_0 , c_1 , c_2 in the parabola $y = c_0 + c_1t + c_2t^2$ where t extends between -1 and 1. The function can also be used to derive the values of the parabola as a function of time from the coefficients.

Author(s)

Jonathan Harrington

See Also

[dct](#)

Examples

```
# fit a polynomial to a segment of fundamental frequency data
plafit(vowlax.fund[1,]$data)

# return the fitted values of the polynomial
plafit(vowlax.fund[1,]$data, fit=TRUE)
```

plos

Segment list of syllable-initial German b and d , isolated words, one male speaker from database gerplosives.

Description

An EMU dataset

Usage

plos

plos.asp	<i>Vector of times at which the burst onset i.e. onset of stop release occurs for segment list plos</i>
----------	---

Description

An EMU dataset

Usage

plos.asp

plos.dft	<i>Spectral trackdata object from the segment list plos.</i>
----------	--

Description

An EMU dataset

Usage

plos.dft

plos.l	<i>Vector of labels from the segment list plos</i>
--------	--

Description

An EMU dataset

Usage

plos.l

plos.lv	<i>Vector of labels of the following vowels from the segment list plos</i>
---------	--

Description

An EMU dataset

Usage

plos.lv

plos.sam	<i>Trackdata object of sampled speech data from the segment list plos</i>
----------	---

Description

An EMU dataset

Usage

plos.sam

plos.w	<i>Vector of word labels from the segment list plos</i>
--------	---

Description

An EMU dataset

Usage

plos.w

plot.spectral	<i>Plot spectra from EMU spectral objects</i>
---------------	---

Description

The function plots spectrum of any EMU spectral object.

Usage

```
## S3 method for class 'spectral'
plot(x, labs, ylim, xlim, col, lty, lwd, fun, freq, type = "l",
     power = FALSE, powcoeffs = c(10, 10), dbnorm = FALSE, dbcoeffs = c(0,0), legend = TRUE, axes = TRUE, ..
```

Arguments

x	An EMU object of class 'spectral'
labs	An optional vector character labels. Must be the same length as specdata
ylim	A two-element numeric vector for the y-axis range (see 'par')
xlim	A two-element numeric vector for the x-axis range (see 'par')
col	Specify a color - see 'mu.colour')
lty	Specify a linetype - see 'mu.colour'
lwd	Specify line thickness - see 'mu.colour'
fun	An R function name e.g., mean, var, sum, etc. The function is applied separately to each category type specified in labs
freq	A numeric vector the same length as the number of columns in specdata specifying the frequencies at which the spectral data is to be plotted. If not supplied, defaults to trackfreq(specdata)
type	A single element character vector for the linetype
power	Logical. If T, then specdata (or specdata\$data if specdata is a trackdata object, is converted to a * specdata^b, where a and b have the values given in powcoeffs. This operation is applied before b
powcoeffs	A two-element numeric vector. Defaults to c(10, 10)
dbnorm	Logical. If T, apply dB-level normalization per spectrum as defined by dbcoeffs below. Defaults to F.
dbcoeffs	A two element numeric vector (x, y). The spectra are normalised in such a way that the values of each spectrum at a frequency of y are set to a dB level of x. For example, to normalise the spectrum to 10 dB at 2000 Hz, set dbnorm to T and dbcoeffs to c(2000, 10)
legend	Parameters for defining the legend. See 'mu.legend' for further details
axes	A logical vector indicating whether the axes should be plotted
...	Further graphical parameters may be supplied.

Details

This function is implemented when a spectral trackdata object is called with the 'plot' function.

Note

To plot spectral data from a spectral trackdata object, then call the function explicitly with 'plot/spectral' rather than with just 'plot'

Author(s)

Jonathan Harrington

See Also

[plot](#) [plot.trackdata](#) [as.spectral](#)

Examples

```

plot(vowlax.dft.5[1,])

# with label types
plot(vowlax.dft.5[1:20,], vowlax.l[1:20])

# As above but averaged after converting to power ratios.
plot(vowlax.dft.5[1:20,], vowlax.l[1:20], fun=mean, power=TRUE)

# All the spectra of one segment in a trackdata object
plot.spectral(fric.dft[1,])

```

plot.trackdata	<i>Produces time-series plots from trackdata</i>
----------------	--

Description

The function produces a plot as a function of time for a single segment or multiple plots as a function of time for several segments.

Usage

```

## S3 method for class 'trackdata'
plot(x, timestart = NULL, xlim = NULL, ylim = NULL, labels = NULL,
      col = TRUE, lty = FALSE, type = "p", pch = NULL, contig = TRUE, ...)

```

Arguments

x	A trackdata object.
timestart	A single valued numeric vector for setting the time at which the trackdata should start. Defaults to NULL which means that the start time is taken from start(trackdata), i.e. the time at which the trackdata object starts.
xlim	A numeric vector of two values for specifying the time interval over which the trackdata is to be plotted. Defaults to NULL which means that the trackdata object is plotted between between the start time of the first segment and the end time of the last segment.
ylim	Specify a yaxis range.
labels	A character vector the same length as the number of segments in the trackdata object. Each label is plotted at side = 3 on the plotted at the temporal midpoint of each segment in the trackdata object. Defaults to NULL (plot no labels). Labels will only be plotted if xlim=NULL.
col	A single element logical vector. Defaults to T to plot each label type in a different colour

lty	A single element logical vector. Defaults to F. If TRUE, plot each label type in a different linetype
type	Specify the type of plot. See plot for the various possibilities
pch	The symbol types to be used for plotting. Should be specified as a numeric vector of the same length as there are unique label classes
contig	A single valued logical vector T or F. If T, then all the segments of the trackdata object are assumed to be temporally contiguous, i.e. the boundaries of the segments are abutting in time and the start time of segment[j,] is the end time of segment[j-1,]. In this case, all the segments of the trackdata object are plotted on the same plot as a function of time. An example of a contiguous trackdata object is <code>coutts.sam</code> . <code>contig = FALSE</code> is when a trackdata object is non-contiguous e.g. all "i:" vowels in a database. An example of a non-contiguous trackdata object is <code>vowlax.fdat</code> . If <code>contig=F</code> then each segment of the trackdata object is plotted separately.
...	the same graphical parameters can be supplied to this function as for <code>plot</code> e.g. <code>type="l"</code> , <code>lty=2</code> etc.

Details

The function plots a single segment of trackdata as a function of time. If the segment contains multiple tracks, then these will be overlaid. If there are several temporally non-contiguous segments in the trackdata object, each segment is plotted in a different panel by specifying `contig=F`. This function is not suitable for overlaying trackdata from more than one segments on the same plot as a function of time: for this use `dplot()`.

Author(s)

Jonathan Harrington

See Also

[plot](#), [dplot](#)

Examples

```
# a single segment of trackdata (F1) plotted as a function of time.
plot(vowlax.fdat[1,1])

# as above, but limits are set for the time axis.
plot(vowlax.fdat[1,1], xlim=c(880, 920))

# the the start-time of the x-axis is set to 0 ms, plot F1 and F3, lineplot
plot(vowlax.fdat[1,c(1,3)], timestart=0, type="l")

# plot F1-F4, same colour, same plotting symbol, between 900
# and 920 ms, type is line and points plot, different linetype per track, no box
plot(vowlax.fdat[1,], col="blue", pch=20, xlim=c(900, 920), type="b", lty=TRUE, bty="n")
```

```

# F1 and F2 of six vowels with labels, separate windows
par(mfrow=c(2,3))
plot(vowlax.fdat[1:6,1:2], contig=FALSE, labels=vowlax.l[1:6], ylab="F1 and F2",
xlab="Time (ms)", type="b", ylim=c(300, 2400))

# As above, timestart set to zero, colour set to blue, different plotting
# symbols for the two tracks
plot(vowlax.fdat[1:6,1:2], contig=FALSE, labels=vowlax.l[1:6], ylab="F1 and F2",
xlab="Time (ms)", type="b", col="blue", pch=c(1,2), ylim=c(300, 2400), timestart=0)

# RMS energy for the utterance 'just relax said Coutts'
plot(coutts.rms, type="l")
# as above a different colour
plot(coutts.rms, type="l", col="pink")
# as above, linetype 2, double line thickness, no box, times reset to 0 ms
plot(coutts.rms, type="l", col="pink", lty=2, lwd=2, bty="n", timestart=0)
# as above but plotted as non-contiguous segments, i.e one segment per panel
par(mfrow=c(2,3))
plot(coutts.rms, type="l", col="pink", lty=2, lwd=2, bty="n", timestart=0, contig=FALSE)
# plot with labels
labels = label(coutts)
par(mfrow=c(1,1))
plot(coutts.rms, labels=labels, type="l", bty="n")
# as above, double line-thickness, green, line type 3, no box,
# time start 0 ms with x and y axis labels
plot(coutts.rms, labels=labels, type="l", lwd=2, col="green", lty=3, bty="n", timestart=0, xlab="Time (ms)", ylab="RMS Energy", ylim=c(0, 200))
# as above with a different plotting symbol for the points
par(mfrow=c(2,3))
plot(coutts.rms, labels=labels, type="b", lwd=2, col="green", timestart=0, bty="n", contig=FALSE, pch=20)

```

polhom

Segment list of four Polish homorganic fricatives from database epg-polish.

Description

An EMU dataset

Usage

polhom

polhom.epg	<i>EPG-compressed trackdata from the segment list polhom</i>
------------	--

Description

An EMU dataset

Usage

polhom.epg

polhom.l	<i>Vector of phonetic labels from the segment list polhom</i>
----------	---

Description

An EMU dataset

Usage

polhom.l

rad	<i>Function to convert between Hertz and Radians</i>
-----	--

Description

convert between Hertz and Radians

Usage

rad(vec, samfreq = 20000, hz = TRUE)

Arguments

vec	A numerical vector of frequencies in Hz or radians
samfreq	A single element numerical vector of the sampling frequency. Defaults to 20000 Hz
hz	Logical. If T, convert from Hz to radians otherwise from radians to hz

Author(s)

Jonahtan Harrington

See Also[help](#)**Examples**

```
# 4000 Hz in radians at a sampling frequency of 8000 Hz
rad(4000, 8000)
# pi/2 and pi/4 radians in Hz at a sampling frequency of 10000 Hz
rad(c(pi/2, pi/4), 10000, FALSE)
```

radians	<i>Converts degrees to radians</i>
---------	------------------------------------

Description

Converts degrees to radians

Usage

```
radians(degrees)
```

Arguments

degrees Angular measurement for conversion.

Details

There are 360 degrees or $2 * \text{PI}$ radians in one full rotation.

Value

Angular measurement in radians.

randomise.segs	<i>Randomise or Reverse items in a segment list</i>
----------------	---

Description

Randomises or Reverses items in a segment list

Usage

```
randomise.segs(segs, rand=TRUE, bwd=FALSE)
```

Arguments

segs	An Emu segment list.
bwd	If T, reverse the order of the segment list.
rand	If T, randomise the order of the segment lists (default).

Value

A segment list containing the original elements in random or reversed order. This is useful if the segment list is to be used as the source for a set of stimuli in a perception experiment.

See Also

[emu.query](#)

Examples

```
data(vowlax)
## assumes a database called demo is available on your system and that
## the Emu system is installed.

# all Phonetic vowels in the database
segs <- vowlax

# randomise the segment list
rsegs <- randomise.segs( segs )
```

rbind.trackdata *A method of the generic function rbind for objects of class trackdata*

Description

Different track data objects from one segment list are bound by combining the `\$data` columns of the track data object by rows. Track data objects are created by `emu.track()`.

Usage

```
## S3 method for class 'trackdata'
rbind(...)
```

Arguments

... track data objects

Details

All track data objects have to be track data of the same segment list. Thus `\$index` and `\$ftime` values have to be identically for all track data objects. The number of columns of the track data objects must match. Thus a track data object of more than one formant and single columned F0 track data object can not be `rbind()`ed.

Value

A track data object with the same `\$index` and `\$ftime` values of the source track data objects and with `\$data` that includes all columns of `\$data` of the source track data objects.

Author(s)

Jonathan Harrington

See Also

[rbind](#) [cbind.trackdata](#) [trackdata](#) [emu.track](#)

Examples

```
data(vowlax)

#segment list vowlax - first segment only
vowlax[1,]

#F0 track data object for vowlax - first segment only
vowlax.fund[1]

#rms track data object for vowlax - first segment only
vowlax.rms[1]

#now combine both track data objects
fund.rms.lax = rbind(vowlax.fund[1:10,], vowlax.rms[1:10,])

#the combined track data object
#The first ten rows in $data keep vowlax.fund data, the 11th to last row keeps vowlax.rms data
fund.rms.lax
```

read.emusegs

Create an Emu segment list from a file

Description

Create an Emu segment list from a file saved by the Emu query tools.

Usage

```
read.emusegs(file)
```

Arguments

file The name of the file to read

Details

Reads segment lists created by programs external to R/Splus and stored in text files on disk.

Value

An Emu segment list.

Author(s)

Steve Cassidy

See Also

[emu.query](#)

Examples

```
## create a segment list file and write it out
seglist.txt <- "database:demo\
query:Phonetic=vowel\
type:segment\
#\
@: 3059.65 3343.65 msdjc001\
e: 5958.55 6244.55 msdjc002\
@u 8984.75 9288.75 msdjc003\
A 11880.8 12184.8 msdjc004\
E 17188.3 17366.4 msdjc005\
ei 20315.2 20655.2 msdjc006"

## Not run: cat(seglist.txt, file="seglist.txt")

# now read it back as a segment list
## Not run: segs <- read.emusegs("seglist.txt")
## Not run: segs
## and clean up
## Not run: unlink("seglist.txt")
```

read.trackdata	<i>Load track data from file</i>
----------------	----------------------------------

Description

read data from a file into a trackdata object, the files contain the data and time components of the object, they're produced by gettrack (within the EMU System) and [write.trackdata](#).

Usage

```
read.trackdata(filename, trackname = "data")
```

Arguments

filename	file name
trackname	track name of the a track that is written in FILE. without trackname, track is defined as data in the returned track data object

Value

track data object

Author(s)

Jonathan Harrington

See Also

[write.trackdata](#)

Examples

```
data(dip)
#Formant track data of the segment list dip (see data(dip)) - first segment only
dip.fdat[1]
## Not run: write.trackdata(dip.fdat, "emu.write.track.example.txt")

#There is a file emu_write.track.example.txt written to R_HOME/
#that includes the track data

#Now load the track data into R
## Not run: dip.fdat.r1 = read.trackdata("emu.write.track.example.txt", "fm")
## Not run: dip.fdat.r1[1]

#Now load the track data into R without argument trackname
## Not run: dip.fdat.r1 = read.trackdata("emu.write.track.example.txt")
## Not run: dip.fdat.r1[1]
## Not run: unlink("emu.write.track.example.txt")
```

segmentlist	<i>Segment list</i>
-------------	---------------------

Description

A segment list is the result of `emu.query()` or `read.emusegs()`.

Format

multi-columned matrix one row per segment

- first columnlabel
- second columnsegment onset time
- third columnsegment offset time
- fourth columnutterance name

See Also

[emu.query](#), [demo.vowels](#)

Examples

```
data(demo.vowels)

#demo.vowels is a segment list
demo.vowels
```

shift	<i>Function to shift the elements of a vector.</i>
-------	--

Description

The function makes use of the function 'fitler' to delay or advance a signal by k points.

Usage

```
shift(x, delta = 1, circular = TRUE)
```

Arguments

x	A numeric vector
delta	A single element numeric vector. Defines the number of points by which the signal should be shifted.
circular	Logical. If T, the signal is wrapped around itself so that if delta = 1, x[n] becomes x[1]. Otherwise, if delta is positive, the same number of zeros are prepended to the signal

Details

The function makes use of the function 'filter' for linear filtering to carry out the shifting.

Value

The signal shifted by a certain number of points. ...

Author(s)

Jonathan Harrington

See Also

filter

Examples

```
vec = 1:10
shift(vec, 2)
shift(vec, -2)
shift(vec, 2, circular=FALSE)
```

sib	<i>Segment list of syllable-initial z preceding i:, I one male speaker of Standard North German from database kielread.</i>
-----	---

Description

An EMU dataset

Usage

sib

sib.dft	<i>Spectral trackdata object from the segment list sib.</i>
---------	---

Description

An EMU dataset

Usage

sib.dft

sib.l	<i>Vector of phonetic labels from the segment list sib</i>
-------	--

Description

An EMU dataset

Usage

sib.l

sib.w	<i>Vector of word labels from the segment list sib</i>
-------	--

Description

An EMU dataset

Usage

sib.w

Slope.test	<i>Slope Test</i>
------------	-------------------

Description

Tests whether the difference between two or more regression lines is significant

Usage

Slope.test(...)

Arguments

... this function takes any number of two column matrices. The first column is the y-data (in the case of locus equations, this is the vowel onset) and the second column is the x-data (in the case of locus equations, vowel target).

Value

The return value consists of the following componenets:

separate	slope, intercept, r-squared, F-ratio, "d(egrees of) f(reedom)" and "prob(ability that) line fits data" for the separate data matrices entered.
combined	F-ratio, "d(egrees of) f(reedom)", and "Probability of them being DIFFERENT" for the slope and for the intercept of the combined data.
x	the combined x-data for all the matrices.
y	the combined y-data for all the matrices.
mat	the category vectors for the combined data (consists of 1, 0 and -1).
numrows	the number of rows in each matrix.
numcats	the sum number of matrices entered.

References

see E. Pedhazur, Multiple Regression in Behavioral Research p.436-450, 496-507.

See Also

lm(), summary.lm(), pf()

sortmatrix

Sort matrix by label

Description

Sorts matrix by label

Usage

```
sortmatrix(mat, labs=dimnames(mat)[[2]])
```

Arguments

mat	A mu+ sement matrix.
labs	A label vector which has the same number of columns as mat.

Value

Returns a sorted matrix by label, created from mat.

See Also

label, phon

splitstring	<i>Split a string into words.</i>
-------------	-----------------------------------

Description

Splits a string into words.

Usage

```
splitstring(str, char)
```

Arguments

str	A string.
char	A character to split on

Value

A vector of strings. The original str is split at every occurrence of char to generate a vector of strings.

Examples

```
splitstring("/home/recog/steve/foo", "/")  
#[1] "home" "recog" "steve" "foo"
```

Start and end times for segment lists and trackdata objects

Start and end times for EMU segment lists and trackdata objects

Description

Obtain start and end times for EMU segment lists and trackdata objects

Usage

```
## S3 method for class 'emusegs'  
start(x, ...)  
## S3 method for class 'emusegs'  
end(x, ...)  
## S3 method for class 'trackdata'  
start(x, ...)  
## S3 method for class 'trackdata'  
end(x, ...)
```

Arguments

x a segment list or a trackdata object
 ... due to the generic only

Details

The function returns the start and/or end times of either a segment list or a trackdata object. The former refers to the boundary times of segments, the latter the start and end times at which the tracks from segments occur. `start.emusegs` and `end.emusegs` give exactly the same output as `start` and `end` respectively.

Value

A vector of times.

Author(s)

Jonathan Harrington

See Also

[tracktimes](#)

Examples

```
# start time of a segment list
start(polhom)
# duration of a segment list
end(polhom) - start(polhom)
# duration from start time of segment list
# and start time of parallel EPG trackdata
start(polhom) - start(polhom.epg)
```

stops

Segment list of word-initial German /b, d, g/

Description

470 /b, d, g/ word-initial German stops in trochaic words between the closure onset and onset of periodicity of the following vowel

Examples

```
data(stops)
```

stops.bark	<i>Spectral data from the stop release/burst onset to the onset of the following vowel</i>
------------	--

Description

Spectral data calculated with a 512 pt FFT, frame shift 5 ms from the stop release/burst onset to the onset of periodicity of the following vowel with the frequency axis warped to the Bark scale and covering the frequency interval 200 - 7800 Hz

Examples

```
data(stops.bark)
## maybe str(stops.bark) ; plot(stops.bark) ...
```

stops.dct	<i>DCT-0, -1, and -2 applied to stops.bark</i>
-----------	--

Description

An EMU dataset

Examples

```
data(stops.dct)
```

stops.dft	<i>Spectral data from stop release/burst onset to the onset the following vowel</i>
-----------	---

Description

Spectral data calculated with a 512 pt FFT, frame shift 5 ms from the stop release/burst onset to the onset of periodicity of the following vowel

Examples

```
data(stops.dft)
```

stops.l *Vector of stop labels*

Description

An EMU dataset

Examples

data(stops.l)

stops.sp *A speaker identifier for stops*

Description

Adult speakers of Standard German, 3 male (gam, lbo, sbo) 4 female (agr, gbr, rlo, tjo).

Examples

data(stops.sp)

stops10 *Spectral matrix extracted 10 ms after stop release burst onset in syllable-initial b, d, g isolated words, one male speaker of Standard German.*

Description

An EMU dataset

Usage

stops10

stops10.lab *Vector of phoneme labels parallel to the spectral matrix stops10*

Description

An EMU dataset

Usage

stops10.lab

stopsvow	<i>Segment list of the following vowel of stops</i>
----------	---

Description

An EMU dataset

Examples

```
data(stopsvow)
```

stopsvow.fm	<i>Formant frequency data, F1-F4 for the segment list stopsvow</i>
-------------	--

Description

An EMU dataset

Examples

```
data(stopsvow.fm)
```

stopsvow.l	<i>Vector of the vowel labels</i>
------------	-----------------------------------

Description

An EMU dataset

Examples

```
data(stopsvow.l)
```

timevow	<i>Segment list of three lax vowels, male speaker of Standard German, railway timetable enquiries from database timetable.</i>
---------	--

Description

An EMU dataset

Usage

```
timevow
```

timevow.dft	<i>Spectral trackdata object from the segment list timevow</i>
-------------	--

Description

An EMU dataset

Usage

timevow.dft

timevow.fm	<i>Trackdata of the first four formants from the segment list timevow</i>
------------	---

Description

An EMU dataset

Usage

timevow.fm

timevow.fm5	<i>Matrix of formants extracted at the temporal midpoint from the segment list timevow</i>
-------------	--

Description

An EMU dataset

Usage

timevow.fm5

timevow.l	<i>Vector of phonetic labels from the segment list timevow</i>
-----------	--

Description

An EMU dataset

Usage

timevow.l

track.gradinfo	<i>Calculate gradient summary information for trackdata</i>
----------------	---

Description

Calculates a number of summary measures for a trackdata object: duration, start and end data points, delta values and slope.

Usage

```
track.gradinfo(trackdata)
```

Arguments

trackdata An Emu trackdata object as returned by [emu.track](#)

Details

track.gradinfo calculates a number of summary measure for the segments within a trackdata object. These are useful for data such as kinematic measures where segments might correspond to articulatory movements etc.

Measures returned are: duration, start and end data values (ie. the first and last rows of data for each segment), delta (the difference between the first and last rows of data) and slope (delta divided by the duration).

Value

A data frame with one row per segment and columns:

duration	Segment
startN	The starting value for each segment (start1 is the starting value for the first column)
endN	The ending value for each segment
deltaN	The delta value for each segment
slopeN	The slope value for each segment

Since the result is a data frame, the columns can be referred to by name (result\$duration) or as matrix columns (result[,1]).

Author(s)

Steve Cassidy

See Also

[emu.track](#), [dapply](#)

Examples

```

data(vowlax)
segs = vowlax
## fm has 4 columns
data.fm <-vowlax.fdat
## F0 has one
data.F0 <- vowlax.fund
## info.fm will have duration, 4xstart, 4xend, 4xdelta, 4xslope
info.fm <- track.gradinfo(data.fm)
## this should be true
ncol(info.fm) == 1+4+4+4+4

## info.F0 will have one of each
info.F0 <- track.gradinfo(data.F0)
## this should be true
ncol(info.F0) == 1+1+1+1+1

## plot the durations vs delta of the first formant
plot(info.F0$duration, info.fm$delta1, type="n", xlab="Duration", ylab="Delta")
text(info.fm$duration, info.fm$delta1, labels=label(segs))

## extract just the delta values from the formant info
## You need to eyeball the data to work out which columns to select
delta.fm <- info.fm[,10:13]

```

trackdata

Track data object

Description

A track data object is the result of `emu.track()`.

Format

\\$index a two columned matrix, each row keeps the first and last index of the `\$data` rows that belong to one segment

\\$ftime a two columned matrix, each row keeps the times marks of one segment

\\$data a multi-columned matrix with the real track values for each segment

Methods

The following generic methods are implemented for trackdata objects.

Arith "+", "-", "*", "^", "%%", "%/%", "/"

Compare "==", ">", "<", "!=", "<=", ">="

Logic "&", "|".

Ops "Arith", "Compare", "Logic"

```

Math "abs", "sign", "sqrt", "ceiling", "floor", "trunc", X "cummax", "cummin", "cumprod",
  "cumsum", "log", "log10", "log2", "log1p", "acos", "acosh", "asin", "asinh", "atan",
  "atanh", "exp", "expm1", "cos", "cosh", "sin", "sinh", "tan", "tanh", "gamma", "lgamma",
  "digamma", "trigamma"
Math2 "round", "signif"
Summary "max", "min", "range", "prod", "sum", "any", "all"

```

Note

The entire data track is retrieved for each segment in the segment list. The amount of data returned will depend on the sample rate and number of columns in the track requested.

See Also

[emu.track](#), [demo.vowels.fm](#) [demo.all.rms](#)

Examples

```

data(demo.vowels.fm)
data(demo.vowels)

#Formant track data for the first segment of the segment list demo.vowels
demo.vowels.fm[1]

```

trackfreq

function to find the frequencies of a spectral object

Description

Find the frequencies of a spectral object.

Usage

```
trackfreq(specdata)
```

Arguments

specdata A spectral object

Value

A vector of the frequencies at which the columns of a spectral matrix occur.

Author(s)

Jonathan Harrington

Examples

```

trackfreq(vowlax.dft.5)
# Frequency components between 1000 and 2000 Hz
trackfreq(vowlax.dft.5[,1000:2000])
# All frequency components of a trackdata object except the d.c. offset
trackfreq(fric.dft[,-1])
# All frequency components except the d.c. offset
# and except frequencies above 5000 Hz
trackfreq(fric.dft[,-c(1, 5000:20000)])
# Note the following syntax if the spectral object is a vector
# Frequencies 1000-3000 Hz
trackfreq(e.dft[1000:3000])

```

trackinfo

Tracks names and track file extensions

Description

Gets tracks names and track file extensions that are defined for an EMU data base.

Usage

```
trackinfo(template)
```

Arguments

template The name of an EMU data base (name of the template file).

Details

Data base must exist for the EMU system.

Value

2-columned matrix

first column name of track

second column file extension of the track files

Author(s)

Jonathan Harrington

Examples

```

#tracks that are defined for data base kielread06
## Not run: trackinfo("kielread06")

```

tracktimes	<i>Get the track times from EMU trackdata objects</i>
------------	---

Description

The function obtains the times at which track values occur.

Usage

```
tracktimes(trackdata)
```

Arguments

trackdata	An EMU trackdata object, or a matrix of track values obtained at a single time point using dcut()
-----------	---

Details

Every `\$data` value in a trackdata object is associated with a time at which it occurs in the utterance. This function returns those times.

Author(s)

Jonathan Harrington

See Also

[start.trackdata](#) [end.trackdata](#) [start.emusegs](#) [end.emusegs](#)

Examples

```
# track time values for a trackdata object
times <- tracktimes(vowlax.fdat)
# track time values for a matrix of trackdata values
# at the temporal midpoint
tracktimes(dcut(vowlax.fdat[1:3,], 0.5, prop=TRUE))
```

`train`*Train a Gaussian Model*

Description

Trains a Gaussian Model

Usage

```
train(x, lab)
```

Arguments

<code>x</code>	A data vector or matrix.
<code>lab</code>	A vector of labels parallel to <code>x</code> . If missing, all data is assumed to be from the same class.

Details

This function is used to train a gaussian model on a data set. The result can be passed to either the `mahal` or `bayes.lab` functions to classify either the training set (`x`) or a test set with the same number of dimensions. Train simply finds the mean and inverse covariance matrix/standard deviation for the data corresponding to each unique label in `labs`.

Value

A structure with the following components:

<code>label</code>	The unique labels in <code>lab</code> .
<code>means</code>	The means for each dimension per unique label.
<code>cov</code>	The combined covariance matrixes for each unique label. The matrixes are joined with <code>rbind</code> . If the input data is one-dimensional, this is just the standard deviation of the data.
<code>invcov</code>	The combined inverse covariance matrixes for each unique label. The matrixes are joined with <code>rbind</code> . If the input data is one-dimensional, this is just the reciprocal of the standard deviation of the data.

See Also

`mahal`, `bayes.lab`, `mahalplot`, `bayes.plot`

trapply	<i>A method of the generic function by for objects of class 'trackdata'</i>
---------	---

Description

A given function 'FUN' is applied to the data corresponding to each segment of data.

Usage

```
trapply (trackdata, fun, ..., simplify = FALSE, returntrack = FALSE)
```

Arguments

trackdata	a track data object
fun	a function that is applied to each segment
...	arguments of the function fun
simplify	simplify = TRUE , output is a matrix; simplify = FALSE a list is returned
returntrack	returntrack = FALSE , return a trackdata object

Details

trapply() applies a function iteratively to each segment of a trackdata object without the need for using a for-loop. It can be used to calculate, for example, the mean value of the data values of each segment separately. Any function that can be applied sensibly to trackdata[j]\$data where j is a segment number can be used as the fun argument to trapply(). It is also possible to write your own function and use trapply() to apply it separately to each segment. Care needs to be taken in using trapply() in the following two ways. Firstly, the argument simplify=T should only be set if it can be guaranteed that a vector of the same length or matrix of the same number of rows as the number of segments in the trackdata object is returned. For example, simplify=T can be used in calculating the mean per segment of a trackdata object, because there will only be one value (the mean) per segment. However, simplify should be set to F in calculating the range because here two values are returned per segment. Similarly use simplify=F n smoothing the data in which the number of values returned per segment is different. Secondly, trapply() only applies a function to a single parameter; the function can be used to apply to a function to multi-parameter trackdata such as F1-F4, but then the function needs to be put inside apply() - see examples below.

Value

list or vector or matrix

Author(s)

Jonathan Harrington

See Also

[apply](#)

Examples

```

# mean f0 one value per segment
m = trapply(vowlax.fund, mean, simplify=TRUE)
# mean F1 - F4
m = trapply(vowlax.fdat, apply, 2, mean, simplify=TRUE)
# make a logical vector of any segments that have an F1 value
# between their start time and end time greater than n Hz
pfun <- function(x, n=1000) any(x > n)
# greater than 1100 Hz
temp = trapply(vowlax.fdat[,1], pfun, 1100, simplify=TRUE)
# get the F2-range per segment
r = trapply(vowlax.fdat[,2], range)
# F2-range of 20th segment
r[[20]]
# DCT-smooth F2 with 10 coeffs
# get the first 4 DCT coefficients
f2.dct = trapply(vowlax.fdat[,2], dct, 3, simplify=TRUE)
# dct-smooth F2 with the first 5 DCT coeffs
f2sm = trapply(vowlax.fdat[,2], dct, 4, TRUE, returntrack=TRUE)
# Make new F2 trackdata such that each segment has
# F2 divided by its F2 range
pfun <- function(x) x/(diff(abs(range(x))))
newf2 = trapply(vowlax.fdat[,2], pfun, returntrack=TRUE)

```

v.sam

Trackdata of sampled speech data centred at the temporal midpoint of an E vowel produced by a male speaker of Standard North German.

Description

An EMU dataset

Usage

v.sam

vowlax

Segment list of four lax vowels, read speech, one male and one female speaker of Standard North German from database kielread.

Description

An EMU dataset

Usage

vowlax

vowlax.df	<i>Data frame of various parameters and labels from the segment list vowlax</i>
-----------	---

Description

An EMU dataset

Usage

vowlax.df

vowlax.dft.5	<i>Spectral matrix centred at the temporal midpoint of the vowels from the segment list vowlax.</i>
--------------	---

Description

An EMU dataset

Usage

vowlax.dft.5

vowlax.fdat	<i>Trackdata of formants from the segment list vowlax</i>
-------------	---

Description

An EMU dataset

Usage

vowlax.fdat

vowlax.fdat.5	<i>Matrix of formant data extracted at the temporal midpoint from the segment list vowlax.</i>
---------------	--

Description

An EMU dataset

Usage

vowlax.fdat.5

vowlax.fund	<i>Trackdata of fundamental frequency from the segment list vowlax</i>
-------------	--

Description

An EMU dataset

Usage

vowlax.fund

vowlax.fund.5	<i>Vector of fundamental frequency extracted at the temporal midpoint from the segment list vowlax.</i>
---------------	---

Description

An EMU dataset

Usage

vowlax.fund.5

vowlax.l	<i>Vector of phoneme labels from the segment list vowlax</i>
----------	--

Description

An EMU dataset

Usage

vowlax.l

vowlax.left	<i>Vector of labels preceding the vowels from the segment list vowlax</i>
-------------	---

Description

An EMU dataset

Usage

vowlax.left

vowlax.right	<i>Vector of labels following the vowels from the segment list vowlax</i>
--------------	---

Description

An EMU dataset

Usage

vowlax.right

vowlax.rms	<i>Trackdata of RMS energy from the segment list vowlax</i>
------------	---

Description

An EMU dataset

Usage

vowlax.rms

vowlax.rms.5	<i>Vector of RMS energy values at the temporal midpoint extracted at the temporal midpoint from the segment list vowlax</i>
--------------	---

Description

An EMU dataset

Usage

vowlax.rms.5

vowlax.spkr	<i>Vector of speaker labels from the segment list vowlax.</i>
-------------	---

Description

An EMU dataset

Usage

vowlax.spkr

vowlax.word	<i>Vector of word labels from the segment list vowlax.</i>
-------------	--

Description

An EMU dataset

Usage

vowlax.word

wordlax.l	<i>Vector of word labels from segment list wordlax</i>
-----------	--

Description

For wordlax (see data(vowlax))

Usage

data(vowlax)

write.emusegs	<i>Write an Emu segment list to a file</i>
---------------	--

Description

Writes an Emu segment list to a file

Usage

write.emusegs(seglist, file)

Arguments

seglist	An Emu segment list
file	The name of a file to write the segment list into.

Value

None.

Side Effects

The segment list is written to a file in the standard format, suitable for input to gettrack or other Emu utility programs.

See Also

[emu.query](#)

Examples

```
data(dip)
#dip a segment list - first 10 segments only
dip[1:10,]
## Not run: write.emusegs(dip, "write.emusegs.example.txt")

#The file write.emusegs.example.txt would have been written to R_HOME
## Not run: unlink("write.emusegs.example.txt")
```

write.trackdata	<i>Write track data objects to file</i>
-----------------	---

Description

The track data object can be saved to a text file in a format suitable for loading into other applications. Single point data is saved in a simple table. Multiple point per segment data is stored in columns with more than one entry per segment. Use [read.trackdata](#) to load the file into R.

Usage

```
write.trackdata(trackdata, file)
```

Arguments

trackdata	track data object or track data object as character
file	file name

Value

a file with the track data is written to the given path

Author(s)

Jonathan Harrington

See Also

[read.trackdata](#)

Examples

```
data(dip)
#Formant track data of the segment list dip (see data(dip)) - first segment only
dip.fdat[1]
## Not run: write.trackdata(dip.fdat, "emu.write.track.example.txt")

#There is a file emu.write.track.example.txt would have been written to R_HOME/
#that includes the track data

## Not run: unlink("emu.write.track.example.txt")
```

write.trackdata.get *This is the write.trackdata version for the a trackdata argument of type character*

Description

It does the same like write.trackdata but with the additional commands that are necessary to handle the string to get the data.

Usage

```
write.trackdata.get(trackdata, file)
```

Arguments

trackdata	as character value of the track data object variable
file	file name

Note

For larger objects this function is faster than write.trackdata

Author(s)

Tina John

See Also

[write.trackdata](#)

Index

*Topic **IO**

- emu.defaultpaths, 45
- emulink, 51
- makelab, 80
- read.emusegs, 102
- read.trackdata, 104
- write.trackdata, 127
- write.trackdata.get, 128

*Topic **attribute**

- as.spectral, 5
- dur, 43
- is.spectral, 71
- trackfreq, 117

*Topic **classes**

- segmentlist, 105
- trackdata, 116

*Topic **datagen**

- dcut, 28
- dextract, 33
- palate, 90
- tracktimes, 119

*Topic **datasets**

- bridge, 11
- coutts, 17
- coutts.epg, 17
- coutts.l, 18
- coutts.rms, 18
- coutts.sam, 18
- coutts2, 18
- coutts2.epg, 19
- coutts2.l, 19
- coutts2.sam, 19
- demo.all, 31
- demo.all.rms, 31
- demo.vowels, 32
- demo.vowels.f0, 32
- demo.vowels.fm, 33
- dip, 36
- dip.fdat, 36

- dip.l, 36
- dip.spkr, 36
- dorfric, 37
- dorfric.dft, 37
- dorfric.l, 37
- dorfric.lv, 37
- dorfric.vl, 38
- dorfric.w, 38
- dorsal, 38
- dorsal.bound, 39
- dorsal.clab, 39
- dorsal.epg, 39
- dorsal.fm, 39
- dorsal.sam, 40
- dorsal.vlab, 40
- e.dft, 44
- emu.directory, 45
- emu.variables, 50
- engassim, 52
- engassim.epg, 52
- engassim.l, 52
- engassim.w, 53
- f2geraus, 64
- f2geraus.l, 64
- fr, 66
- fr.dft, 66
- fr.l, 67
- fr.sp, 67
- fric, 69
- fric.dft, 70
- fric.l, 70
- fric.w, 70
- isol, 72
- isol.fdat, 73
- isol.l, 73
- keng, 73
- keng.dft.5, 73
- keng.l, 74
- plos, 92

- plos.asp, 93
- plos.dft, 93
- plos.l, 93
- plos.lv, 93
- plos.sam, 94
- plos.w, 94
- polhom, 98
- polhom.epg, 99
- polhom.l, 99
- sib, 106
- sib.dft, 106
- sib.l, 107
- sib.w, 107
- stops, 110
- stops.bark, 111
- stops.dct, 111
- stops.dft, 111
- stops.l, 112
- stops.sp, 112
- stops10, 112
- stops10.lab, 112
- stopsvow, 113
- stopsvow.fm, 113
- stopsvow.l, 113
- timevow, 113
- timevow.dft, 114
- timevow.fm, 114
- timevow.fm5, 114
- timevow.l, 114
- v.sam, 122
- vowlax, 122
- vowlax.df, 123
- vowlax.dft.5, 123
- vowlax.fdat, 123
- vowlax.fdat.5, 123
- vowlax.fund, 124
- vowlax.fund.5, 124
- vowlax.l, 124
- vowlax.left, 124
- vowlax.right, 125
- vowlax.rms, 125
- vowlax.rms.5, 125
- vowlax.spkr, 125
- vowlax.word, 126
- wordlax.l, 126
- *Topic **dplot**
 - cr, 20
 - crplot, 22
 - dplot, 40
 - epggs, 56
 - eggplot, 57
 - eplot, 61
 - plot.spectral, 94
 - plot.trackdata, 96
- *Topic **manip**
 - buildtrack, 11
 - dbnorm, 25
 - shift, 105
- *Topic **math**
 - bark, 8
 - dbtopower, 26
 - dct, 27
 - ddiff, 30
 - EPG contact indices, 53
 - epgcog, 54
 - eggsum, 59
 - freqtoint, 68
 - locus, 76
 - mel, 82
 - moments, 85
 - plafit, 91
 - rad, 99
- *Topic **methods**
 - bind.trackdata, 10
 - by.trackdata, 12
 - cbind.trackdata, 13
 - dim.trackdata, 34
 - dimnames.trackdata, 35
 - label, 74
 - rbind.trackdata, 101
 - trapply, 121
- *Topic **misc**
 - as.trackdata, 7
 - dapply, 23
 - dsmooth, 43
 - ellipse, 44
 - emu.query, 45
 - emu.requery, 47
 - emu.track, 49
 - euclidean, 63
 - expand_labels, 63
 - frames.time, 68
 - get.time.element, 71
 - is.trackdata, 72
 - linear, 75
 - mahal, 77

- mahal.dist, 78
- make.seglist, 79
- matscan, 81
- modify.seglist, 84
- muclass, 88
- norm, 89
- perform, 91
- radians, 100
- randomise.segs, 100
- Slope.test, 107
- sortmatrix, 108
- splitstring, 109
- track.gradinfo, 115
- train, 120
- write.emusegs, 126
- *Topic **models**
 - classify, 15
- *Topic **utilities**
 - fapply, 65
 - frames, 67
 - mu.colour, 86
 - Start and end times for segment
 - lists and trackdata objects, 109
 - trackinfo, 118
- apply, 13, 65, 121
- as.spectral, 5, 72, 95
- as.trackdata, 7
- bark, 8, 83
- bind.trackdata, 10
- bridge, 11
- buildtrack, 11
- by, 12, 13, 28
- by (by.trackdata), 12
- by.trackdata, 12, 65
- cbind, 14
- cbind (cbind.trackdata), 13
- cbind.trackdata, 13, 102
- classify, 15
- classplot, 16
- coutts, 17
- coutts.epg, 17
- coutts.l, 18
- coutts.rms, 18
- coutts.sam, 18
- coutts2, 18
- coutts2.epg, 19
- coutts2.l, 19
- coutts2.sam, 19
- cr, 20, 23
- crplot, 21, 22
- dapply, 13, 23, 50, 115
- dbnorm, 25
- dbtopower, 25, 26, 26
- dct, 27, 92
- dcut, 28, 42, 62
- ddiff, 24, 30
- demo.all, 31, 32
- demo.all.rms, 31, 33, 117
- demo.vowels, 31, 32, 105
- demo.vowels.f0, 32
- demo.vowels.fm, 32, 33, 117
- dextract, 33
- dim (dim.trackdata), 34
- dim.trackdata, 34
- dimnames.trackdata, 35
- dip, 36
- dip.fdat, 36
- dip.l, 36
- dip.spkr, 36
- dorfric, 37
- dorfric.dft, 37
- dorfric.l, 37
- dorfric.lv, 37
- dorfric.vl, 38
- dorfric.w, 38
- dorsal, 38
- dorsal.bound, 39
- dorsal.clab, 39
- dorsal.epg, 39
- dorsal.fm, 39
- dorsal.sam, 40
- dorsal.vlab, 40
- dplot, 7, 29, 40, 50, 87, 97
- dsmooth, 24, 43
- dur, 43
- e.dft, 44
- ellipse, 44
- emu.date (emu.variables), 50
- emu.defaultpaths, 45
- emu.directory, 45
- emu.query, 45, 48, 49, 51, 52, 79, 85, 101, 103, 105, 127
- emu.requery, 47, 51, 52

- emu.track, 7, 14, 29, 42, 46, 49, 51, 52, 72, 79, 102, 115, 117
- emu.variables, 50
- emu.version (emu.variables), 50
- emu.year (emu.variables), 50
- emulink, 51
- emusegs (segmentlist), 105
- end, 74
- end.emusegs, 79, 119
- end.emusegs (Start and end times for segment lists and trackdata objects), 109
- end.trackdata, 119
- end.trackdata (Start and end times for segment lists and trackdata objects), 109
- engassim, 52
- engassim.epg, 52
- engassim.l, 52
- engassim.w, 53
- EPG contact indices, 53
- epgai, 55, 57, 58, 60, 90
- epgai (EPG contact indices), 53
- epgci (EPG contact indices), 53
- epgcog, 54, 54, 57, 58, 60, 90
- epgdi (EPG contact indices), 53
- epggs, 54, 56, 58, 60, 90
- epgplot, 57, 57, 90
- epgsum, 55, 59
- eplot, 29, 61, 87
- euclidean, 63
- expand_labels, 63

- f2geraus, 64
- f2geraus.l, 64
- fapply, 65
- fr, 66
- fr.dft, 66
- fr.l, 67
- fr.sp, 67
- frames, 67
- frames.time, 68
- frequint, 68
- fric, 69
- fric.dft, 70
- fric.l, 70
- fric.w, 70

- get.time.element, 71

- help, 100

- is.spectral, 6, 71
- is.trackdata, 72
- isol, 72
- isol.fdat, 73
- isol.l, 73

- keng, 73
- keng.dft.5, 73
- keng.l, 74

- label, 74
- label.emusegs, 79
- lda, 16
- linear, 75
- locus, 76

- mahal, 77
- mahal.dist, 78
- make.seglist, 49, 79
- makelab, 80
- Math.trackdata (trackdata), 116
- Math2.trackdata (trackdata), 116
- matscan, 81
- mel, 9, 82
- modify.seglist, 79, 84
- moments, 85
- mu.colour, 86
- muclass, 88

- norm, 89

- Ops.trackdata (trackdata), 116

- palate, 54, 55, 57, 58, 60, 90
- par, 41, 62, 76
- perform, 91
- plafit, 28, 91
- plos, 92
- plos.asp, 93
- plos.dft, 93
- plos.l, 93
- plos.lv, 93
- plos.sam, 94
- plos.w, 94
- plot, 95, 97
- plot.spectral, 6, 9, 25, 26, 83, 87, 94
- plot.trackdata, 87, 95, 96
- polhom, 98

- polhom.epg, 99
- polhom.l, 99

- qda, 16

- rad, 99
- radians, 100
- randomise.segs, 100
- rbind, 102
- rbind(rbind.trackdata), 101
- rbind.trackdata, 14, 101
- read.emusegs, 51, 52, 102
- read.trackdata, 51, 52, 104, 127

- segmentlist, 31–33, 38, 74, 105
- shift, 105
- sib, 106
- sib.dft, 106
- sib.l, 107
- sib.w, 107
- Slope.test, 107
- smooth, 13
- sortmatrix, 108
- splitstring, 109
- start, 74
- Start and end times for segment lists and trackdata objects, 109
- start.emusegs, 79, 119
- start.emusegs (Start and end times for segment lists and trackdata objects), 109
- start.trackdata, 119
- start.trackdata (Start and end times for segment lists and trackdata objects), 109

- stops, 110
- stops.bark, 111
- stops.dct, 111
- stops.dft, 111
- stops.l, 112
- stops.sp, 112
- stops10, 112
- stops10.lab, 112
- stopsvow, 113
- stopsvow.fm, 113
- stopsvow.l, 113
- Summary.trackdata (trackdata), 116

- timevow, 113
- timevow.dft, 114
- timevow.fm, 114
- timevow.fm5, 114
- timevow.l, 114
- track.gradinfo, 115
- trackdata, 13, 14, 32, 33, 67, 102, 116
- trackfreq, 117
- trackinfo, 118
- tracktimes, 110, 119
- train, 120
- trapply, 12, 13, 121

- utt (label), 74
- utt.emusegs, 79

- v.sam, 122
- vowlax, 122
- vowlax.df, 123
- vowlax.dft.5, 123
- vowlax.fdat, 123
- vowlax.fdat.5, 123
- vowlax.fund, 124
- vowlax.fund.5, 124
- vowlax.l, 124
- vowlax.left, 124
- vowlax.right, 125
- vowlax.rms, 125
- vowlax.rms.5, 125
- vowlax.spkr, 125
- vowlax.word, 126

- wordlax.l, 126
- write.emusegs, 126
- write.trackdata, 104, 127, 128
- write.trackdata.get, 128