

# Package ‘RSEIS’

January 20, 2025

**Type** Package

**Title** Seismic Time Series Analysis Tools

**Version** 4.2-4

**Date** 2024-09-12

**Depends** R (>= 3.0)

**Imports** RPMG, Rwave, stats

**Suggests** GEOMap

**Author** Jonathan M. Lees [aut, cre],  
Jake Anderson [ctb],  
Leonard Lisapaly [ctb],  
Dave Harris [aut, cph]

**Maintainer** Jonathan M. Lees <jonathan.lees@unc.edu>

**Description** Multiple interactive codes to view and analyze seismic data, via spectrum analysis, wavelet transforms, particle motion, hodograms. Includes general time-series tools, plotting, filtering, interactive display.

**License** GPL (>= 2)

**Copyright** see inst/COPYRIGHTS

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2024-09-12 22:10:28 UTC

## Contents

RSEIS-package . . . . .	7
addpoints.hodo . . . . .	8
addtix . . . . .	9
addWPX . . . . .	10
applytaper . . . . .	11
ASCII.SEISN . . . . .	12
attime12 . . . . .	13
AUGMENTbutfilt . . . . .	14

autoreg . . . . .	16
brune.doom . . . . .	17
brune.func . . . . .	18
brune.search . . . . .	19
butfilt . . . . .	20
BUTREPLOTT . . . . .	22
catWPX . . . . .	24
CE1 . . . . .	25
checkWPX . . . . .	25
choosfilt . . . . .	26
CHOP.SEISN . . . . .	28
circ . . . . .	29
cleanpickfile . . . . .	29
cleanWPX . . . . .	30
colorwig . . . . .	31
combineSEIS . . . . .	32
comp.env . . . . .	33
Comp1Dvel . . . . .	34
Comp1Dvels . . . . .	35
complex.hodo . . . . .	36
COMPorder . . . . .	37
contwlet . . . . .	38
convert2Rseis . . . . .	39
convertATT . . . . .	40
correct.moveout . . . . .	41
DAYSpERYEAR . . . . .	42
DECIMATE.SEISN . . . . .	42
deconinst . . . . .	44
deleteWPX . . . . .	45
detail.pick . . . . .	46
detrend . . . . .	47
DISPLACE.SEISN . . . . .	48
distseisnXY . . . . .	49
DISTxsec . . . . .	50
DO.PMOT.ARR . . . . .	52
doGABOR.AR . . . . .	53
doGABOR.MTM . . . . .	54
doMYBUTTS . . . . .	56
DOsgram . . . . .	57
dowiggles . . . . .	58
downsample . . . . .	59
editDB . . . . .	60
EmptyPickfile . . . . .	62
EmptySEIS . . . . .	63
envelope . . . . .	63
EPOCHday . . . . .	64
EPOCHyear . . . . .	65
ETECTG . . . . .	66

evolAR . . . . .	67
evolfft . . . . .	69
evolMTM . . . . .	70
FAKEDATA . . . . .	72
filedatetime . . . . .	74
FILT.SEISN . . . . .	75
FILT.spread . . . . .	76
filterstamp . . . . .	77
finteg . . . . .	79
fixcompname . . . . .	80
fixcomps . . . . .	80
fixNA . . . . .	81
fixUWstasLL . . . . .	83
fromjul . . . . .	83
FRWDft . . . . .	84
gaddtix . . . . .	85
GAZI . . . . .	86
genrick . . . . .	87
get.corner . . . . .	88
GET.seis . . . . .	89
get.slepians . . . . .	92
Get1Dvel . . . . .	94
GETARAIC . . . . .	95
getb1b2 . . . . .	96
getEcard . . . . .	97
getFcard . . . . .	98
getGHtime . . . . .	99
getHcard . . . . .	100
getIRIS . . . . .	101
getjul . . . . .	102
getmoday . . . . .	103
getNcard . . . . .	104
getPDEcsv . . . . .	104
getpfile . . . . .	105
getphaselag2 . . . . .	106
getrdpix . . . . .	107
getseis24 . . . . .	108
getvertsorder . . . . .	110
GH . . . . .	112
ghstamp . . . . .	114
GLUE.GET.seis . . . . .	115
GLUEseisMAT . . . . .	115
gpoly . . . . .	116
GreatDist . . . . .	117
grotseis . . . . .	118
hilbert . . . . .	120
hilow . . . . .	121
hodogram . . . . .	122

hypot . . . . .	123
idpoints.hodo . . . . .	124
info.seis . . . . .	125
infoDB . . . . .	125
insertNAs . . . . .	127
INSTFREQS . . . . .	128
INSTresponse . . . . .	129
integ1 . . . . .	130
INVRft . . . . .	131
j2posix . . . . .	132
jadjust.length . . . . .	133
JBLACK . . . . .	133
JGRAY . . . . .	134
jitter.lab . . . . .	135
jlegend . . . . .	136
jpolyval . . . . .	137
JSAC.seis . . . . .	138
jstats . . . . .	140
Jtim . . . . .	141
KH . . . . .	142
lagplot . . . . .	143
leests . . . . .	144
legitpix . . . . .	145
letter.it . . . . .	145
LocalUnwrap . . . . .	146
logspace . . . . .	147
longfft . . . . .	148
makeDB . . . . .	150
makefreq . . . . .	152
markseis24 . . . . .	153
matsquiggle . . . . .	155
Mine.seis . . . . .	157
mirror.matrix . . . . .	159
Mmorlet . . . . .	160
mtapspec . . . . .	161
MTM.drive . . . . .	162
MTMdisp . . . . .	163
MTMgabor . . . . .	165
MTMplot . . . . .	166
NEW.getUWSTAS . . . . .	168
NEWPLOT.WPX . . . . .	168
next2 . . . . .	169
OH . . . . .	170
one . . . . .	171
P2GH . . . . .	171
parse.pde . . . . .	172
parseFN2STA . . . . .	173
partmotnet . . . . .	174

PDE2list . . . . .	175
peaks . . . . .	176
PICK.DOC . . . . .	177
pickgeninfo . . . . .	178
pickhandler . . . . .	178
pickit . . . . .	179
pickseis24 . . . . .	181
plocator . . . . .	183
PLOT.ALLPX . . . . .	184
PLOT.MATN . . . . .	185
PLOT.SEISN . . . . .	187
PLOT.TTCURVE . . . . .	189
Plot1Dvel . . . . .	191
plotarrivals . . . . .	192
plotDB . . . . .	193
plotevol . . . . .	194
plotGH . . . . .	196
plotJGET . . . . .	197
plotseis24 . . . . .	198
plotwlet . . . . .	200
plt.MTM0 . . . . .	202
PLTpicks . . . . .	203
PMOT.drive . . . . .	204
posix2RSEIS . . . . .	205
PPIX . . . . .	206
prep1wig . . . . .	207
prepSEIS . . . . .	209
PreSet.Instr . . . . .	210
PSTLTcurve . . . . .	211
Put1Dvel . . . . .	212
pwlet2freqs . . . . .	213
rangedatetime . . . . .	214
Ray.time1D . . . . .	215
rdistaz . . . . .	216
rDUMPLOC . . . . .	218
read1segy . . . . .	219
ReadInstr . . . . .	220
ReadSet.Instr . . . . .	221
readUW.OSTAS . . . . .	222
reccdate . . . . .	223
repairWPX . . . . .	224
replaceWPX . . . . .	225
rseis2segy . . . . .	226
rseis2ts . . . . .	227
rsspec.taper . . . . .	228
ruler . . . . .	229
save.wpix . . . . .	230
saveWPX . . . . .	231

scal2freqs . . . . .	232
screens . . . . .	233
SEARCHPIX . . . . .	234
secdif . . . . .	235
secdifL . . . . .	236
secdifv . . . . .	237
segy2rseis . . . . .	238
SEIS2list . . . . .	239
seiscols . . . . .	240
SEISNtime . . . . .	241
seisorder . . . . .	242
selAPX . . . . .	243
SELBUT . . . . .	244
selpgen . . . . .	245
SELSTA . . . . .	245
selstas . . . . .	246
SENSORsensitivity . . . . .	247
setPrePix . . . . .	248
setstas . . . . .	249
setupDB . . . . .	250
setwelch . . . . .	251
setwpix . . . . .	254
setWPX . . . . .	255
setypx . . . . .	257
showdatetime . . . . .	258
sigconv . . . . .	259
sigconvGR . . . . .	260
SNET.drive . . . . .	261
SPECT.drive . . . . .	262
Spectrum . . . . .	264
STALTA . . . . .	265
STLTcurve . . . . .	266
swig . . . . .	268
swig.ALLPX . . . . .	271
symshot1 . . . . .	272
sysinfo . . . . .	273
T12.pix . . . . .	274
TAPER.SEISN . . . . .	275
Thresh.J . . . . .	276
TOCART . . . . .	277
tojul . . . . .	278
tomo.colors . . . . .	278
trapz . . . . .	279
travel.time1D . . . . .	280
tung.pulse . . . . .	281
unpackAcard . . . . .	282
uwpfile2ypx . . . . .	283
varsquig . . . . .	284

varsquiggle . . . . .	286
VELMOD1D . . . . .	287
VELOCITY.SEISN . . . . .	288
view.seis . . . . .	289
vlen . . . . .	291
vline . . . . .	292
wiggle.env . . . . .	293
wiggleimage . . . . .	294
WINGH . . . . .	295
winmark . . . . .	296
winseis24 . . . . .	298
wlet.do . . . . .	299
wlet.drive . . . . .	301
write1segy . . . . .	302
writeUW.Acard . . . . .	303
writeUW.Commentcard . . . . .	304
writeUW.DOTcard . . . . .	304
writeUW.Ecard . . . . .	305
writeUW.Fcard . . . . .	305
writeUW.Hcard . . . . .	306
writeUW.Ncard . . . . .	306
writeUW.OSTAScard . . . . .	307
writeUWpickfile . . . . .	307
X2RSEIS . . . . .	308
X2SAC . . . . .	309
xcor2 . . . . .	310
xprod . . . . .	311
XTR . . . . .	312
xtract.trace . . . . .	313
yeardate . . . . .	314
YPIX . . . . .	315
YRsecdif . . . . .	317
Zdate . . . . .	319
zlocator . . . . .	320
ZOOM.SEISN . . . . .	321

**Index****323**

RSEIS-package

*Seismic Analysis and Display***Description**

Multiple interactive codes to view and analyze seismic data, via spectrum analysis, wavelet transforms, particle motion, hodograms. Includes general time-series tools, plotting, filtering, interactive display.

**Note**

**Seismic Sections** JGET.seis view.seis swig Mine.seis VELOCITY.SEISN DISPLACE.SEISN ZOOM.SEISN wlet.drive SENSORSensitivity PLOT.MATN PLOT.SEISN PLOT.TTCURVE PLOT.ALLPX plotevol MTMdisp MTMplot NEW.getUWSTAS NEWPLOT.WPX INSTFREQS INSTresponse GLUE.GET.seis GLUEseisMAT FILT.SEISN FILT.spread CHOP.SEISN get.corner grotseis

**Velocity-Travel Time:** Put1Dvel Ray.time1D setLQUAKE selAPX Get1Dvel Comp1Dvel Comp1Dvels travel.time1D

**Particle Motion:** hodogram PMOT.drive complex.hodo addpoints.hodo idpoints.hodo DO.PMOT.ARR partmotnet prep1wig prepSEIS EmptySEIS GAZI

**Time series:** xcor2 wlet.drive wlet.do wiggle.env plotwlet STLTcurve SPECT.drive rsspec.taper evolfft GETARAIC PSTLTcurve getphaselag2 envelope hilbert LocalUnwrap lagplot apply-taper autoreg butfilt choosfilt MTM.drive

**Date-Time Functions:** yeardate YRsecdif Zdate recdatel recdate tojul getjul getmoday secdifL secdif secdifv JtimL Jtim fromjul

**Graphics:** plocator ilocator meshgrid ymarginfo zlocator winmark vline screens RESCALE pwlet2freqs addtix circle circ letter.it jpostscript JBLACK JGRAY HOZscale gaddtix Gcols jlegend tomo.colors

**Misc:** BKpfile2ypx brune.doom brune.func brune.search comp.env contwlet deconinst detail.pick rdistaz rDUMPLOC EmptyPickfile ETECTG finteg fixcompname fixcomps fixUWstasLL fmod FRWDft getb1b2 getNcard getpfile getseisinfo getvertsorder gpoly GreatDist gwpix2ypx hilow hypot integ1 INVRft itoxyz jadjust.length jpolyval jstats local.file logspace makefreq mirror.matrix Mmorlet mtapspec peaks PICK.DOC pickit plt.MTM0 PLTpicks PPIX Pre-Set.Instr ReadSet.Instr readUW.OSTAS scal2freqs SEARCHPIX setstas setwelch shade.col SNET.drive T12.pix Thresh.J TOCART trapz tung.pulse unpackAcard uwppfile2ypx

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu> Maintainer:Jonathan M. Lees<jonathan.lees.edu>

**See Also**

RPGM, RFOC

**Examples**

```
data("GH")
swig(GH)
```

---

addpoints.hodo

*Add points to a hodogram plot*

---

**Description**

Add points to a hodogram plot



**Usage**

```
addpoints.hodo(nbaz, dt, sx, flag = 1:10, pch = 3, col = 1)
```

**Arguments**

nbaz	matrix 3 by n
dt	sample interval, s
sx	x vector
flag	output of idpoints.hodo
pch	plot character
col	color for plotting

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PMOT.drive, idpoints.hodo

---

addtix	<i>add tix to plot</i>
--------	------------------------

---

**Description**

Add tick marks to edge of plot

**Usage**

```
addtix(side = 3, pos = 0, tck = 0.005, at =c(0, 1), labels = FALSE, col = 2, ...)
```

**Arguments**

side	side of plot 1-4
pos	position relative to side
tck	tick size
at	locations along axis
labels	labels for tics
col	color for ticks
...	graphical parameters, par

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

par

---

addWPX

*Add one pick to WPX file*

---

**Description**

Add one pick to WPX file

**Usage**

addWPX(WPX, ppx)

**Arguments**

WPX	WPX list
ppx	WPX list

**Details**

Adds one pick to end of list.

**Value**

WPX list

**Note**

Uses, the last pick as a reference.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

catWPX

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
s2 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(1))

s3 <- addWPX(s1, s2)
```

---

applytaper	<i>Apply taper to seismic trace</i>
------------	-------------------------------------

---

**Description**

Apply taper to ends of a time series for spectrum analysis.

**Usage**

```
applytaper(f, p = 0.05)
```

**Arguments**

f	signal
p	percent taper

**Details**

10 percent taper is 5 percent on each end.

**Value**

Tapered time series.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data(CE1)
Xamp <- CE1$y[CE1$x>5.443754 & CE1$x<5.615951]
Tamp <- applytaper(Xamp, p = 0.05)
```

---

`ASCII.SEISN`*ASCII RSEIS data dump*

---

**Description**

Write RSEIS list to a file in ASCII format.

**Usage**

```
ASCII.SEISN(GH, sel = 1, HEAD = TRUE, destdir='.' )
```

**Arguments**

GH	RSEIS list
sel	vector, select which ttraces to write
HEAD	logical, TRUE will put a header in the file
destdir	character, path to folder to deposit output file

**Details**

Used for data exchange for users who do not want to use RSEIS. The header consists of one line start date (yr, jd, hr, min, sec) and sample rate (dt).

**Value**

Side effects - files are created.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
## Not run:
##### this example creates an ascii version of the
##### seismic data for exchange purposes
data("GH")
tempd = tempdir()
sel <- which(GH$COMPS == "V" & GH$STNS=="CE1" )
ASCII.SEISN(GH, sel = 1, HEAD = TRUE, destdir=tempd)

## End(Not run)
```

---

attime12

*Epoch Time Window*


---

### Description

Set a time window in Epoch days for extraction from a DB file

### Usage

```
attime12(t1, t2 = t1, origyr = 1972, pre = 0, post = 0)
```

### Arguments

t1	list date-time 1
t2	list date-time 2
origyr	origin year
pre	seconds before time 1
post	seconds afer time 2

### Details

If t2 is missing, t1=t2.

### Value

vector            c(t1, t2)

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### Examples

```
j1 <- list(yr = 2005, jd= 214 , hr= 7 , mi= 1 ,sec= 0.5235)
j2 <- list(yr=2005, jd= 214 , hr= 7 , mi= 1 ,sec= 0.5235+6)

at <- attime12(j1, t2=j1, origyr=2005, pre=100, post=100)

### given an RSEIS format list
data(GH)
AT = SEISNtime(GH)
ats = attime12(AT[[1]], t2 = AT[[2]],
  origyr =AT[[1]]$yr , pre = 0, post= 0)
```

---

AUGMENTbutfilt                      *Butterworth filter with Augmentation*

---

### Description

Design and apply butterworth low/high/band pass filters with augmentation of the signal on either end to suppress edge effects.

### Usage

```
AUGMENTbutfilt(a, fl = 0, fh = 0.5, deltat = 1, type = "BP",
proto = "BU", npoles = 5, chebstop = 30, trbndw = 0.3,
RM = FALSE, zp = TRUE, pct = 0.1)
```

### Arguments

a	vector signal
fl	low frequency cut-off, default=0
fh	high frequency cut-off, DEFAULT= (1/2dt)
deltat	sample rate, s, deFAULT=1
type	type of filter, one of c("LP", "HP", "BP", "BR" ), DEFAULT="BP"
proto	prototype, c("BU", "BE", "C1", "C2"), DEFAULT="BU"
npoles	number of poles or order, DEFAULT=5
chebstop	Chebyshev stop band attenuation, DEFAULT=30.0
trbndw	Chebyshev transition bandwidth, DEFAULT=0.3
RM	Remove mean value from trace, default=FALSE
zp	zero phase filter, default=TRUE
pct	Percent augmentation applied to each side, default=0.1

### Details

Creation of butfilt is a described by the following arguments:

**LP** low pass  
**HP** high pass  
**BP** band pass  
**BR** band reject  
**BU** Butterworth  
**BE** Bessel  
**C1** Chebyshev type 1  
**C2** Chebyshev type 2

Arguments `chebstop` , `trbdnw` are ignored for non-chebyshev filters. LP and HP filters are set by specifying `fl` for HP filters and `fh` for LP filters, the other argument in each case is ignored.

Mean values should be removed prior to calling this function, and then set `RM=FALSE`. This is true especially if tapering is applied prior to filtering.

Zero phase filter is achieved by running filter back and forth. Otherwise a single pass is returned. This should be equivalent to package `signal` `filtfilt` (from MATLAB).

Augmentation involves copying the first and last percent of the signal, reversing the time and adding to the signal on each end. This is then filtered, and removed after filter is complete. It is assumed that the important part of the signal is in the center of the time series and the edges are less critical. Then the augmented part has the same statistical content as the edges of the signal (presumably noise) and will not affect the filtered signal considerably. This is then thrown away prior to return.

### Value

Filtered time series with the augmentation removed after filter.

### Author(s)

Jonathan M. Lees<[jonathan.lees.edu](mailto:jonathan.lees.edu)>

### See Also

`butfilt`

### Examples

```
data(CE1)

ts1 <- CE1$y
zz <- AUGMENTbutfilt(ts1, fl=1, fh=15, deltat=CE1$dtt, type="LP" , proto="BU",
npoles=5 )

##### second example with plotting

data(KH, package = 'RSEIS' )
w = KH$JSTR[[1]]
dt = KH$dtt[1]

x = seq(from=0, by=dt, length=length(w));
plot(x,w, type='l')

par(mfrow=c(2,1) )

i=1
  fl = 1/50
fh= 1/2
  ftype = 'BP'
##### normal band pass filter
```

```

zz = butfilt(w, fl, fh, dt, ftype , "BU")
    f.stamp = filterstamp(fl=f1, fh=fh, type=ftype)

plot(x, zz, type='l', xlab='s', ylab='amp', main= f.stamp)
title(sub='butfilt')

####
zz1 = AUGMENTbutfilt(w, fl, fh, dt, type=ftype , proto="BU", zp=TRUE, pct=0.2 )
    f.stamp = filterstamp(fl=f1, fh=fh, type=ftype)
plot(x, zz1, type='l', xlab='s', ylab='amp', main= f.stamp)
title(sub='AUGMENTbutfilt')

```

---

autoreg

*Auto-Regressive Spectrum Estimate*

---

## Description

Auto-Regressive Spectrum Estimate

## Usage

```
autoreg(a, numf = 1024, pord = 100, PLOT = FALSE, f1 = 0.01, f2 = 50)
```

## Arguments

a	signal
numf	number of frequency points to calculate
pord	order
PLOT	logical, TRUE=plot
f1	low frequency
f2	high frequency

## Value

LIST:

amp	amplitudes
freq	frequencies, Hz



**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**See Also**

fft, mtapspec, plt.MTM0

**Examples**

```
data(CE1)
Xamp <- CE1$y[CE1$x>5.443754 & CE1$x<5.615951]

ZIM <- autoreg(Xamp , numf=length(Xamp) , pord = 100, PLOT=FALSE, f1=.01, f2=50)
```

brune.doom

*Brune Modeling***Description**

Modeling the Brune spectrum with Graphical Diagnostics

**Usage**

```
brune.doom(amp, dt = 1, f1 = 0.01, f2 = 15, PLOTB = FALSE, tit = "")
```

**Arguments**

amp	signal
dt	deltaT
f1	low frequency for modeling
f2	high frequency for modeling
PLOTB	logical, TRUE=show diagnostic plots
tit	title for plot

**Value**

List:

SUCCESS	(0,1) for success or failure of modeling
WARN	flag = "OK"
tstar0	tstar0
gamma	gamma
omega0	omega0
fc	fc
alpha	alpha
chisqrd	chi-squared misfit over region of fitting

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Lees, J. M. and G. T. Lindley (1994): Three-dimensional Attenuation Tomography at Loma Prieta: Inverting  $t^*$  for  $Q$ , J. Geophys. Res., 99(B4), 6843-6863.

**Examples**

```
data(CE1)
plot(CE1$x, CE1$y, type='l')
Xamp = CE1$y[CE1$x>5.443754 & CE1$x<5.615951]

BF = brune.doom( Xamp, CE1$dt , f1=.5, f2=12 , PLOTB = TRUE)
```

---

brune.func

*Brune Earquake Model*

---

**Description**

Calculate Forward Brune model

**Usage**

```
brune.func(freq, omega0, tstar0, fc, alpha, gamma)
```

**Arguments**

freq	frequency vector
omega0	low freq asymptote
tstar0	T-star value
fc	corner frequency
alpha	alpha parameter
gamma	gamma parameter

**Details**

Brune model.

**Value**

returns displacement spectrum from given parameters

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**References**

Lees, J. M. and G. T. Lindley (1994): Three-dimensional Attenuation Tomography at Loma Prieta: Inverting  $t^*$  for  $Q$ , J. Geophys. Res., 99(B4), 6843-6863.

**See Also**

brune.doom

brune.search

*Search for Brune fit to displacement spectrum***Description**

Model of the spectrum of a seismic arrival. Uses Brune's Model.

**Usage**

```
brune.search(infreq, inspec, f1, f2, omega0, fcorn, tstar0, gamma)
```

**Arguments**

infreq	vector of frequencies
inspec	spectrum
f1	low frequency, Hz
f2	high frequency, Hz
omega0	initial starting low frequency asymptote
fcorn	initial starting corner frequency
tstar0	initial starting $t^*$
gamma	initial starting gamma

**Details**

see paper by Lees and Lindley

**Value**

```
list(omega0=omega0,tstar0=tstar0[3], fc=fcorn, alpha=0, gamma=gamma[3])
```

omega0	low frequency asymptote
fc	corner frequency
tstar0	$t^*$
gamma	gamma
alpha	alpha parameter
chisqrd	chi-squared misfit over region of fitting

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Lees and Lindley

**See Also**

MTM

**Examples**

```
data(CE1)

#### set frequency range for modeling
f1 <- 0.01
f2 <- 14.0

## set up data and parameters
amp <- CE1$y
len2 <- 2*next2(length(amp))
a <- list(y=amp, dt=CE1$dt)

Spec <- MTMdisp(a, f1=f1, f2=f2, len2=len2, PLOT=FALSE )

lspec <- Spec$displ

### get initial estimate of parameters
xc <- get.corner( Spec$f , lspec, dt, f1, f2, PLOT=FALSE)

jmod <- brune.search(Spec$f, lspec, f1, f2,
                    xc$omega0, xc$corn, xc$tstar0, 2.0)
```

---

butfilt

*Butterworth filter*

---

**Description**

Design and apply butterworth low/high/band pass filters.

**Usage**

```
butfilt(a, fl=0, fh=0.5, deltat=1, type="BP", proto="BU",
        npoles=5, chebstop=30.0, trbndw=0.3, RM=FALSE, zp=TRUE )
```

**Arguments**

a	vector signal
f1	low frequency cut-off, default=0
fh	high frequency cut-off, DEFAULT= (1/2dt)
deltat	sample rate, s, deFAULT=1
type	type of filter, one of c("LP", "HP", "BP", "BR" ), DEFAULT="BP"
proto	prototype, c("BU", "BE" , "C1" , "C2"), DEFAULT="BU"
npoles	number of poles or order, DEFAULT=5
chebstop	Chebyshev stop band attenuation, DEFAULT=30.0
trbndw	Chebyshev transition bandwidth, DEFAULT=0.3
RM	Remove mean value from trace, default=FALSE
zp	zero phase filter, default=TRUE

**Details**

Creation of butfilt is a described by the following arguments:

**LP** low pass

**HP** high pass

**BP** band pass

**BR** band reject

**BU** Butterworth

**BE** Bessel

**C1** Chebyshev type 1

**C2** Chebyshev type 2

Arguments chebstop , trbndw are ignored for non-chebyshev filters. LP and HP filters are set by specifying f1 for HP filters and fh for LP filters, the other argument in each case is ignored.

Mean values should be removed prior to calling this function, and then set RM=FALSE. This is true especially if tapering is applied prior to filtering.

Zero phase filter is achieved by running filter back and forth. Otherwise a single pass is returned. This should be equivalent to package signal filtfilt (from MATLAB).

**Value**

Filtered time series.

**Author(s)**

originally written in FORTRAN by David Harris, converted to C and modified by Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Harris, D., 1990: XAPiir: A recursive digital filtering package. United States: N. p., Web. doi:10.2172/6416972.

**See Also**

AUGMENTbutfilt

**Examples**

```
data(CE1)

ts1 <- CE1$y
zz <- butfilt(ts1, fl=1, fh=15, deltat=CE1$dt, type="LP" , proto="BU",
npoles=5 )

### try plotting:

### the above, by default, is zero phase.
##### next filter with non-zero-phase
z2 <- butfilt(ts1, fl=1, fh=15, deltat=CE1$dt, type="LP" , proto="BU",
npoles=5, zp=FALSE )
ex = seq(from=0, by=CE1$dt, length=length(ts1))

plot(ex, ts1, type='l')
lines(ex, zz, col='red')
lines(ex, z2, col='blue')

plot(ex[ex<0.5], ts1[ex<0.5], type='l')
lines(ex[ex<0.5], zz[ex<0.5], col='red')
lines(ex[ex<0.5], z2[ex<0.5], col='blue')
```

---

BUTREPLOT

*Replot Function for SELBUT*

---

**Description**

Replot Function for SELBUT

**Usage**

```
BUTREPLOT(opts, ncol = 5, HOZ = TRUE, TOP = TRUE,
cols = "white", main = "", xlim = c(0, 1),
ylim = c(0, 1), newplot = TRUE)
```

**Arguments**

opts	character list of options
ncol	number of columns
HOZ	logical, TRUE=plot horizontally
TOP	logical, TRUE=plot top-down
cols	colors
main	character title
xlim	x-limits in plotting region (user coordinates)
ylim	y-limits in plotting region (user coordinates)
newplot	logical, new plot?

**Details**

Used internally in SELBUT as a replotting function

**Value**

list	
M	x,y matrix of grid
dx	delta x
dy	delta y
rx	range of x
ry	range of y

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

SELBUT, swig

**Examples**

```
STDLAB <- c("DONE", "QUIT", "zoom.out", "zoom.in",  
  "SELBUT", "FILT", "UNFILT", "PSEL", "SGRAM",  
  "WLET", "SPEC", "XTR" )  
BUTREPLOT(STDLAB)
```

---

`catWPX`*Concatenate two WPX lists*

---

**Description**

Concatenate (combine) two WPX lists.

**Usage**

```
catWPX(WPX, ppx)
```

**Arguments**

WPX	WPX list
ppx	WPX list

**Details**

Adds second list to the end of the first list.

**Value**

WPX list

**Author(s)**

Jonathan M. Lees<[jonathan.lees@unc.edu](mailto:jonathan.lees@unc.edu)>

**See Also**

`addWPX`, `setWPX`, `checkWPX`, `cleanWPX`, `clusterWPX`, `repairWPX`, `saveWPX`

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
s2 <- setWPX(name="BYE", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
s3 <- catWPX(s1, s2)
```



---

CE1	<i>Single Seismogram</i>
-----	--------------------------

---

**Description**

Single Seismogram from Coso California

**Usage**

```
data(CE1)
```

**Format**

```
list(x=0, y=0, dt=0, name="", Tpick=0, mark="", deltat=0)
```

**References**

Lees, J.M., 2004. Scattering from a fault interface in the Coso geothermal field. *Journal of Volcanology and Geothermal Research*, 130(1-2): 61-75.

**Examples**

```
data(CE1)
plot(CE1$x, CE1$y, type='l')
```

---

checkWPX	<i>Check WPX</i>
----------	------------------

---

**Description**

Check and verify WPX list for compliance.

**Usage**

```
checkWPX(wpx)
```

**Arguments**

wpx	WPX list
-----	----------

**Details**

Perform several checks on completeness, length of components, station names, component names and date-times of the WPX lists.

**Value**

- 0 no problems
- 1 list incomplete
- 2 names incomplete
- 3 components incomplete
- 4 dates incomplete

**Note**

No action taken in the event an error occurs - see repairWPX to fix problems.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX,catWPX, saveWPX,cleanWPX,clusterWPX,repairWPX,setWPX

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))  
s1$col=NULL
```

---

choosfilt

*Interactive Choice of Filter*

---

**Description**

Choose Butterworth filter from a selection

**Usage**

```
choosfilt(thefilt = thefilt, ncol = 5)
```

**Arguments**

thefilt	list of filter parameters
ncol	number of columns

**Details**

Used for interactive choices in swig. See example below.

**Value**

filter parameter list:

ON	logical, TRUE=filter is on
f1	low frequency cut-off
fh	high frequency cut-off
type	type of filter, one of c("LP", "HP", "BP", "BR")
proto	prototype, c("BU", "BE", "C1", "C2")

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

butfilt, RPMG

**Examples**

```

thefilts <-
  list(flo=
    c(0.02, 0.02, 0.02, 0.02, 0.02, 0.02,
      0.02, 0.02, 0.02, 0.02, 0.02, 0.02,
      0.02,
      1/2, 1/50,1/100, 1/100,1,1,
      0.2, 15, 5, 2,1,
      100),
    fhi=
    c(1/10, 1/6, 1/5, 1/4, 1/3, 1/2,
      0.2, 0.5, 1.0, 2.0, 3.0, 4.0,
      7.0,
      8, 1/2.0,1/5.0,1/10.0,10,5,
      7.0, 100, 100, 100,10,
      100),
    type =
    c("LP", "LP", "LP", "LP", "LP", "LP",
      "LP", "LP", "LP", "LP", "LP", "LP",
      "LP",
      "BP", "BP", "BP", "BP", "BP", "BP",
      "HP", "HP", "HP", "HP", "HP",
      "None"))

if(interactive() ) choosfilt(thefilts = thefilts, ncol = 5)

```

---

`CHOP.SEISN`*CHOP SEISmic structure*

---

**Description**

Take a seismic structure and return a time limited version

**Usage**

```
CHOP.SEISN(GH, sel = 1:4, WIN = NULL)
```

**Arguments**

<code>GH</code>	Seismic trace structure
<code>sel</code>	selection of traces
<code>WIN</code>	time window $c(0,1)$

**Value**

Seismic trace structure

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

`swig`

**Examples**

```
data("GH")
sel <- which(GH$COMPS=="V")

KF <- CHOP.SEISN(GH, sel=sel, WIN = c(0 , 5) )

swig(KF, SHOWONLY=0)
```

---

circ	<i>Draw a circle</i>
------	----------------------

---

**Description**

Draw a circle

**Usage**

```
circ()
```

**Details**

Draw a circle on new plot.

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

net

**Examples**

```
circ()
```

---

cleanpickfile	<i>Clean up Pickfile structure</i>
---------------	------------------------------------

---

**Description**

Given a pickfile, clean out stations that do not ocnform

**Usage**

```
cleanpickfile(P)
```

**Arguments**

P Pickfile list

**Details**

stations with name="" are removed

**Value**

P Pickfile list

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EmptyPickfile

**Examples**

```
P <- EmptyPickfile()
cleanpickfile(P)
```

---

cleanWPX

*Clean WPX*

---

**Description**

Return an empty (clean) WPX.

**Usage**

```
cleanWPX()
```

**Details**

Returns an empty list with NA's and 0's

**Value**

WPX list

**Note**

Used internally.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX, catWPX, checkWPX, repairWPX, clusterWPX, saveWPX, setWPX

**Examples**

```
s0 <- cleanWPX()
```

---

colorwig

*Plot a seismic trace colored in time*

---

**Description**

Plot a seismic trace colored in time. useful for coordinating other plots to specific times along a seismic trace.

**Usage**

```
colorwig(x1, y1, COL = rainbow(100))
```

**Arguments**

x1	x-coordinate (time)
y1	y-coordinate (amplitude)
COL	color palette

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
data(KH)

x <- KH$ex[KH$ex>95 & KH$ex<125]
y <- KH$JSTR[[1]][KH$ex>95 & KH$ex<125]

colorwig(x, y, rainbow(100))
```

---

`combineSEIS`*Combine SEIS lists*

---

**Description**

Combine 2 SEIS format lists into one list suitable for swig.

**Usage**

```
combineSEIS(IH, IV)
```

**Arguments**

IH	SEIS list (swig input)
IV	SEIS list (swig input)

**Details**

This will take two SEIS lists and merge them into one.

**Value**

SEIS list suitable for swig.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig, Mine.seis, prepSEIS

**Examples**

```
##### say you have 2 databases - extract from each:
####GH = Mine.seis(at1, at2, DB1, grepsta, grepcomp, kind = -1)
####JH = Mine.seis(at1, at2, DB2, grepsta, grepcomp, kind = -1)
#### merge the 2 structures

data(KH)

MH = KH

BH = combineSEIS(KH, MH)
##### plot and interact
swig(BH, SHOWONLY=TRUE )
```



---

`comp.env`*Compare Envelopes*

---

**Description**

calculate and plot signal envelopes.

**Usage**

```
comp.env(ex, Y, PLOT = TRUE, stamps = stamps)
```

**Arguments**

<code>ex</code>	x-axis
<code>Y</code>	matrix of Y values
<code>PLOT</code>	logical, TRUE=plot
<code>stamps</code>	character vectors of ids

**Details**

Takes in a common x predictor and compares the envelopes of each column in the Y matrix. All the Y's must have the same length as ex.

**Value**

Graphical Side effects. returns envelope series.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig

**Examples**

```
data("GH")

temp <- cbind(GH$JSTR[[1]], GH$JSTR[[2]], GH$JSTR[[3]])

atemp <- temp[1168:1500, ]
ex <- seq(from=0,length=length(temp[1168:1500, 1]), by=GH$dt[1])

comp.env(ex, atemp, PLOT = TRUE, stamps = c("1", "2", "3") )
```

---

 Comp1Dvel

 Compare a pair of 1D models
 

---

### Description

plot a pair of 1D velocity Models for comparison

### Usage

```
Comp1Dvel(v, v2, col=c('blue', 'brown'), ...)
```

### Arguments

v	List structure for model 1
v2	List structure for model 2
col	2-colors for P and swave
...	other graphical parameters (e.g. lty, lwd)

### Value

Graphical Side Effects

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

Plot1Dvel, Get1Dvel, travel.time1D

### Examples

```
VEL <- list()
VEL$'zp' <- c(0,0.25,0.5,0.75,1,2,4,5,10,12)
VEL$'vp' <- c(1.1,2.15,3.2,4.25,5.3,6.25,6.7,6.9,7,7.2)
VEL$'ep' <- c(0,0,0,0,0,0,0,0,0,0)
VEL$'zs' <- c(0,0.25,0.5,0.75,1,2,4,5,10,12)
VEL$'vs' <- c(0.62,1.21,1.8,2.39,2.98,3.51,3.76,3.88,3.93,4.04)
VEL$'es' <- c(0,0,0,0,0,0,0,0,0,0)
VEL$'name' <- '/data/wadati/lees/Site/Hengil/krafla.vel'

VELNish <- list()
VELNish$'zp' <- c(0,0.1,0.6,1.1,21.1)
VELNish$'vp' <- c(2.8,3.4,4.1,4.7,4.7)
VELNish$'ep' <- c(0,0,0,0,0)
VELNish$'zs' <- c(0,0.1,0.6,1.1,21.1)
VELNish$'vs' <- c(1.6,2,2.4,2.7,2.7)
VELNish$'es' <- c(0,0,0,0,0)
```

```
VELNish$name' <- 'Nish'
Comp1Dvel(VEL, VELNish)
```

---

 Comp1Dvels

*Compare 1D models*


---

## Description

Plot 1D velocity Models for comparison.

## Usage

```
Comp1Dvels(INV, depth = 1:50)
```

## Arguments

INV	vector of velocity models in memory
depth	depth range for plotting

## Details

takes several velocity models, finds the range of all, makes a plot so that all models fit on figure.

## Value

Graphical Side Effects

## Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

## See Also

Plot1Dvel, Comp1Dvel, Get1Dvel

## Examples

```
VEL <- list()
VEL$'zp' <- c(0,0.25,0.5,0.75,1,2,4,5,10,12)
VEL$'vp' <- c(1.1,2.15,3.2,4.25,5.3,6.25,6.7,6.9,7,7.2)
VEL$'ep' <- c(0,0,0,0,0,0,0,0,0,0)
VEL$'zs' <- c(0,0.25,0.5,0.75,1,2,4,5,10,12)
VEL$'vs' <- c(0.62,1.21,1.8,2.39,2.98,3.51,3.76,3.88,3.93,4.04)
VEL$'es' <- c(0,0,0,0,0,0,0,0,0,0)
VEL$name' <- '/data/wadati/lees/Site/Hengil/krafla.vel'

VELNish <- list()
```

```

VELNish$'zp' <- c(0,0.1,0.6,1.1,21.1)
VELNish$'vp' <- c(2.8,3.4,4.1,4.7,4.7)
VELNish$'ep' <- c(0,0,0,0,0)
VELNish$'zs' <- c(0,0.1,0.6,1.1,21.1)
VELNish$'vs' <- c(1.6,2,2.4,2.7,2.7)
VELNish$'es' <- c(0,0,0,0,0)
VELNish$'name' <- 'Nish'

```

```
Comp1Dvels(c("VEL", "VELNish"))
```

---

complex.hodo

*HodoGram Plot*

---

## Description

HodoGram Plot

## Usage

```

complex.hodo(nbaz, dt = dt, labs = c("Vertical", "North", "East"),
  COL = rainbow(100), STAMP = "")

```

## Arguments

nbaz	n by 3 matrix
dt	time sample rate
labs	labels for the components
COL	color palette
STAMP	character stamp for identification

## Value

sx = list graphical side effect

## Author(s)

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data("GH")

temp <- cbind(GH$JSTR[[1]][1168:1500], GH$JSTR[[2]][1168:1500],
GH$JSTR[[3]][1168:1500])

pmolabs <- c("Vertical", "North", "East")

sx <- complex.hodo(temp, dt=GH$dt[1] ,labs=pmolabs,
STAMP="Example", COL=rainbow(100) )
```

---

COMPorder

*Seismic Component Order*

---

**Description**

Set seismic component order

**Usage**

```
COMPorder(STNS, COMPS)
```

**Arguments**

STNS	stations
COMPS	components

**Details**

Sets up components so they are ordered according to V, N, E. used internally in swig.

**Value**

order vector

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

---

`contwlet`*Contour Wavelet Transform*

---

**Description**

Contour Wavelet Transform

**Usage**

```
contwlet(baha, Ysig, dt, clev = 0.75, NLEV = 12,  
         zscale = 1, zbound = NULL, col = col, ygrid = FALSE,  
         WUNITS = "Volts", PEAX = NULL)
```

**Arguments**

<code>baha</code>	Output of wavelet transform (image)
<code>Ysig</code>	input signal to wavelet transform
<code>dt</code>	DeltaT, sample rate
<code>clev</code>	levels for contours
<code>NLEV</code>	number of levels
<code>zscale</code>	scale of amplitudes
<code>zbound</code>	bounds for scale of interest
<code>col</code>	color for contour lines
<code>ygrid</code>	logical, TRUE=add grid lines
<code>WUNITS</code>	Units of wavelet transform
<code>PEAX</code>	peaks structure

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**See Also**

plotwlet, wlet.do, wlet.drive

---

convert2Rseis	<i>Convert Seismic data to RSEIS</i>
---------------	--------------------------------------

---

**Description**

Convert Seismic in SAC or SEG Y format to RSEIS native format.

**Usage**

```
convert2Rseis(FLS, NEWDIR = ".", kind = 1, Iendian = "little", BIGLONG =
FALSE, NEWsta = "", NEWcomp = "")
```

**Arguments**

FLS	array of File names
NEWDIR	Destination directory path
kind	an integer 1, 2, 3; 0=R(DAT) , 1 = segy, 2 = sac, 3 = AH.
Iendian	Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE=long=8 bytes
NEWsta	character vector, stations associated with the vector of files
NEWcomp	character vector, component name associated with the vector of files

**Details**

Converts the data to R format so it can be loaded with the load command. After this conversion, files should be loaded in subsequent calls by using kind=0.

**Value**

Side effects - creates new files on local system

**Note**

JGET.seis extracts digital seismic data from binary files stored in the file system. The program uses readBin for I/O and passes data back to R. Currently SAC, SEG Y formats are installed but it is easy to extend. AH format is available for LINUX systems, but there were problems compiling in WINDOWS and MACOS so this feature was removed. A filter for mseed format is currently being developed.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

JGET.seis, JSAC.seis , Mine.seis

**Examples**

```

Iendian = .Platform$endian
data(GH)

##### create some SAC files:
apath = tempdir()
J = rseis2sac(GH, sel = 1:5, path = apath, BIGLONG =FALSE )
#### get SAC file file names:
Lname <- list.files(path=J , pattern='SAC', full.names=TRUE)

##### convert each file to a saved RSEIS file, saved in apath
#### reading in SAC files, kind=2
convert2Rseis(Lname, NEWDIR = apath, kind = 2, Iendian = Iendian, BIGLONG =
FALSE )
#### check if files are there
list.files(path=apath)

```

---

convertATT

*DateHour to List*


---

**Description**

Convert a julian day+time to an RSEIS date list.

**Usage**

```
convertATT(at1, yr)
```

**Arguments**

at1	julian day in Year, plus (hr+minutes+seconds)
yr	Year

**Details**

Calculates the data-list that RSEIS uses in calculations. The Month and Day-of-month are also returned.

**Value**

List with date and time

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>



**See Also**

recdate, recdate1, dateList, dateStamp, filedatetime, rangedatetime, yeardate, Zdate, as.POSIXct

**Examples**

```
yr = 2014
j = 233.1234
convertATT(j, yr)
```

---

correct.moveout	<i>Moveout Correction</i>
-----------------	---------------------------

---

**Description**

Shift traces according to given moveout times

**Usage**

```
correct.moveout(GH, sel = 1, tims = 0)
```

**Arguments**

GH	RSEIS structure list
sel	index of which traces to be shifted
tims	time shifts for each trace

**Details**

Each trace listed in sel gets shifted forward or backward according to time in tims. This is useful for shifting traces according to a given moveout curve.

**Value**

RSEIS list structure returned with adjusted traces

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

DAYSperYEAR

*Days per Year*

---

**Description**

Calculate the number of days per calendar year

**Usage**

```
DAYSperYEAR(yr)
```

**Arguments**

```
yr          year
```

**Value**

```
days      integer number of days for a given year
```

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
years <- seq(from=1850, to=2010, by=1)
```

```
DAYSperYEAR(years)
```

---

DECIMATE.SEISN*Decimate a set of traces*

---

**Description**

Decimate, or reduce the sample rate of a set of traces stored in event RSEIS format

**Usage**

```
DECIMATE.SEISN(TH, sel=1:length(TH$JSTR), dec=5 ,  
  type="LP", proto="BU" , fl=2, fh=10, RM=FALSE, zp=TRUE )
```

**Arguments**

TH	RSEIS list
sel	numeric, which traces to select
dec	numeric, number of samples to skip
type	type of filter (see butfilt), or FALSE for no filter
proto	filter proto type
f1	low pass frequency cut off
fh	high pass frequency cut off
RM	Remove mean value from trace, default=FALSE
zp	zero phase filter, default=TRUE

**Details**

Reduces the number of samples by skipping every "dec" sample.

To achieve smoothing prior to sampling, low pass filter may be applied to avoid spikes or other sampling issues.

If type is FALSE, no filter is applied and samples are taken from the input.

**Value**

an RSEIS list.

**Note**

The dt, n and t2 are modified in info.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

butfilt, downsample

**Examples**

```
data(GH)

dec = 250/50

##### resample all traces by reducing from 250 to 50 samples/s
DH = DECIMATE.SEISN(GH, sel=1:length(GH$JSTR), dec=dec ,
  type="LP", proto="BU" , f1=2, fh=50, RM=FALSE, zp=TRUE )

##### compare

##### times in
```

```

### starting second should be the same
GH$info$sec[1:5] - DH$info$sec[1:5]
#### number of samples should be reduced
cbind(GH$info$n[1:5] , DH$info$n[1:5] )
### ending seconds should be close but not identical
cbind(GH$info$t2[1:5] , DH$info$t2[1:5] )

cbind(GH$info$dt[1:5] , DH$info$dt[1:5] )

cbind( sapply(GH$JSTR, 'length'), sapply(DH$JSTR, 'length') )

#### for visual comparison:
### par(mfrow=c(2,1) )
## g = swig(GH, sel=which(GH$COMPS=="V" ), SHOWONLY=0 )
## d = swig(DH, sel=which(DH$COMPS=="V" ), SHOWONLY=0 )

```

---

deconinst

*Deconvolve instrument response from seismic data*


---

## Description

Deconvolve instrument response from seismic data

## Usage

```
deconinst(data, sintr, KAL, key, Calibnew, waterlevel = 1e-08)
```

## Arguments

data	Real vector of data
sintr	sample interval
KAL	Kalibrated response list
key	number of instrument
Calibnew	new instrument, complex vector or
waterlevel	waterlevel for low frequency division

## Details

To avoid problems with dividing by very small numbers, water level is set =1.e-8

## Value

deconvolved signal

**Note**

Calibnew(1)==3 then use a cos (hanning) taper

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PreSet.Instr, ReadSet.Instr, INSTresponse

**Examples**

```
Kal <- PreSet.Instr()
amp <- rnorm(1024)
Calibnew <- c(1,1.0, 0.0 )

dy <- deconinst(amp, 0.008, Kal,1, Calibnew, waterlevel=1.e-8)
```

---

deleteWPX

*Delete picks to WPX file*

---

**Description**

Delete pick to WPX file

**Usage**

```
deleteWPX(WPX, ind=1)
```

**Arguments**

WPX	WPX list
ind	integer, index to delete

**Details**

Deletes one pick to end of list.

**Value**

WPX list

**Note**

Uses, the last pick as a reference.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX, catWPX

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
s2 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(1))

s3 <- addWPX(s1, s2)

s4 <- deleteWPX(s3, ind=2:3)
```

---

detail.pick

*Detail Pick on 3-component seismogram*

---

**Description**

Pops up three components and prepares menu items for picking

**Usage**

```
detail.pick(y, ex, dt, TIT = "")
```

**Arguments**

y	signal amplitudes
ex	x-axis
dt	deltaT, sample rate, s
TIT	title

**Details**

Creates interactive session for picking seismograms. Is called from swig.

**Value**

KSAVE = list(x=xsave, y=ysave)

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig

**Examples**

```
data(CE1)
detail.pick(CE1$y, CE1$x, CE1$dt, TIT = "")
```

---

detrend                      *Remove trend from time series signal*

---

**Description**

Remove trend from time series signal

**Usage**

```
detrend(x)
```

**Arguments**

x                      vector

**Details**

Removes the trend from a signal.

**Value**

vector with linear trend removed.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

mean

**Examples**

```
dt <- 0.001

t <- seq(0, 6, by=0.001)

y <- 5*sin(2*pi*10*t)

plot(t,y, type='l')

y <- y + 3 * t
plot(t,y, type='l')

dy <- detrend(y)

plot(t,dy, type='l')
```

DISPLACE.SEISN

*Displacement seismogram***Description**

Removes seismic instrument response and integrates to displacement.

**Usage**

```
DISPLACE.SEISN(TH, sel = 1:length(TH$JSTR), inst = 1,
Kal = Kal,waterlevel = 1e-08, FILT = list(ON = FALSE,
fl = 1/30, fh = 7, type = "HP", proto = "BU",RM=FALSE, zp=TRUE))
```

**Arguments**

TH	list structure of seismic traces
sel	select which traces in list to deconvolve
inst	index to instrument in Kal list for calibration and instrument response
Kal	list of instrument responses
waterlevel	waterlevel for low frequency division
FILT	filter output, after instrumentation, see butfilt

**Details**

Instrument responses are lists of poles and zeros for each instrument defined.

**Value**

Same as input list with new traces representing displacement versus velocity



**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

VELOCITY.SEISN, deconinst, butfilt

**Examples**

```
data(KH)

Kal <- PreSet.Instr()

DH <- DISPLACE.SEISN(KH, sel = 1 , inst = 1,
Kal = Kal, FILT = list(ON = FALSE, fl = 1/200, fh = 7,
type = "BP", proto = "BU"))

if(interactive()){
  SOUT <- swig(DH, PADDLAB=c("CENTER", "fspread", "HALF", "PREV") )
}
```

---

distseisnXY

*Distances from an RSEIS list*

---

**Description**

Calculate euclidian distances from an RSEIS seismic data list, stations and event location.

**Usage**

```
distseisnXY(GH, sta=list(nam="", x=0 , y=0 , z=0) , LOC=list(x=0, y=0 , z=0))
```

**Arguments**

GH	Rseis list structure
sta	station list(x,y,z)
LOC	location list(x,y,z)

**Value**

d                      vector of distances in km, matching the stations in the RSEIS list.

**Note**

Locations of stations and source should be projected.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
data(GH)
### assume the lat lon in GH are x, y (projected)

staxy <- list(nam=GH$stafilename, x=GH$stafilelon,
             y=GH$stafilelat, z=GH$stafilez)
LOC <- list(x=GH$pickfile$LOC$lon, y=GH$pickfile$LOC$lat,
           z=GH$pickfile$LOC$z)
distseisnXY(GH, sta =staxy, LOC = LOC)
```

---

DISTxsec

*Distance Cross section*

---

**Description**

Plot time series vertically at specified distances. Produces a seismic cross section with correct spacing between traces.

**Usage**

```
DISTxsec(GH, dist, TIM.WIN = c(0, 3600), sel, trace.width = 10,
         col = "black", text.col = "blue", text.font = 2, text.size = 0.8,
         add = FALSE, plot = TRUE)
```

**Arguments**

GH	RSEIS seismic trace structure, output of prepSEIS used in swig
dist	distance for each station along x-axis
TIM.WIN	time window for cross section
sel	numeric, index of selected traces to plot.
trace.width	Width of each trace in plot. Should be in same units as x-axis
col	color for traces. If vector, each trace is plotted with assigned color.
text.col	color for text identifying each trace.
text.font	font for text identifying each trace.
text.size	size of text for identifying each trace.
add	logical, Whether to add traces, or just set up the figure
plot	logical, whether to plot the traces.

**Details**

Distances should be a vector for each trace in the RSEIS list.

**Value**

vector of x-y coordinates of the plot.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig, prepSEIS

**Examples**

```
#### example using data in the RSEIS package
data(GH)
#### get the source location
lat.org = GH$pickfile$LOC$lat
lon.org = GH$pickfile$LOC$lon
#### get the station locations
g1 =GH$stafile
#### find the distance to each station
gd = rdistaz(lat.org, lon.org, g1$lat, g1$lon )

##### optional, filter the data
sel= which( GH$COMPS == 'V')
### filter traces
Fdef <- list(ON=TRUE, fl=1, fh=1, type="HP", proto="BU", RM=TRUE, zp=TRUE )
KF <- FILT.SEISN(GH, FILT=Fdef)

### match the stations in GH to the station distances
m1 = match(GH$STNS , g1$name)
dist.GH = gd$dist[m1]
TIM.WIN = range(GH$ex)

##### prepare plot, but do not add traces
A = DISTxsec(KF, dist.GH, TIM.WIN, sel, trace.width = 0.5 , add=FALSE,
plot=FALSE )
##### add traces
B = DISTxsec(KF, dist.GH, TIM.WIN, sel, trace.width = 0.5 , add=TRUE,
plot=TRUE, col='black' , text.col='red', text.size=1 )
```

---

 DO.PMOT.ARR

*Particle Motion Analysis with arrows*


---

**Description**

Plot particle motion arrows

**Usage**

DO.PMOT.ARR(E, N)

**Arguments**

E	East component
N	East Component

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PMOT.drive

**Examples**

```

data(GH)
XLIM = c(1226, 1322 )

e = GH$JSTR[[1]][XLIM[1]:XLIM[2]]
n = GH$JSTR[[2]][XLIM[1]:XLIM[2]]

xx = range(e, na.rm =TRUE)
yy = range(n, na.rm =TRUE)
sx = range(c(xx, yy))

x = RPMG::RESCALE(e, 0, 1, sx[1], sx[2])
y = RPMG::RESCALE(n, 0, 1, sx[1], sx[2])

plot(range(x), range(y) , type='n')
lines(x, y, col=grey(0.8) )
DO.PMOT.ARR(x, y)

```

doGABOR.AR

*Gabor Transform with AR spectrum method***Description**

Gabor Transform with AR spectrum method

**Usage**

```
doGABOR.AR(Xamp, DT = 0.008, multi = 1, scale.def = 0, TWIN = 2, TSKIP =
0.2, PCTTAP = 0.05, pord=100, PLOT=TRUE)
```

**Arguments**

Xamp	signal
DT	sample rate interval (s)
multi	Multiples of time window estimate
scale.def	scaling flag for plotting (0=raw, 1=log, 2=sqrt)
TWIN	time for window
TSKIP	time for skip
PCTTAP	percent of taper to apply to individual windows
pord	order for the AR process (default=100)
PLOT	logical, TRUE=plot to device

**Details**

This is a spectrogram function similar to the Gabor Transform but uses the AR method for spectrum estimation.

**Value**

list	
sig	input signal
dt	deltat
numfreqs	Number of frequencies output
wpars	input parameters list(Nfft=numfreqs, Ns=Ns, Nov=Nov, fl=fl, fh=fh)
DSPEC	spectrum image
HIMAT	matrix with high values of F-test at 90 percent confidence
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Note**

The main difference between this and other similar calls is the way the windows are determined.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

Percival, Donald B., Walden, Andrew T. (1993): Spectral Analysis for Physical Applications, Cambridge University Press, Cambridge, 583p.

**See Also**

evolfft, evolMTM, MTM.drive, GETARAIC, doGABOR.AR, DOsgram, doGABOR.MTM

**Examples**

```
data(KH)
###  swig(KH)

Xamp <- KH$JSTR[[1]]
Xamp <- Xamp[57914:72989]

EV <- doGABOR.AR(Xamp, DT = KH$dt[1] , multi = 1, scale.def = 0,
TWIN = 2, TSKIP = 0.2, PCTTAP = 0.05)
```

---

doGABOR.MTM

*Evolutionary MTM Spectrum*


---

**Description**

Time varying Auto-Regressive Spectrum (Gabor Transform) using MTM. This is a driver for MT-Mgabor.

**Usage**

```
doGABOR.MTM(Xamp, DT = 0.008, ppoint=95 , multi = 1,
scale.def = 0, TWIN = 2, TSKIP = 0.2, PCTTAP = 0.05, PLOT=TRUE)
```

**Arguments**

Xamp	signal
DT	sample rate interval (s)
ppoint	percent confidence for F-test (default=95)
multi	Multiples of time window estimate
scale.def	scaling flag for plotting (0=raw, 1=log, 2=sqrt)
TWIN	time for window
TSKIP	time for skip
PCTTAP	percent of taper to apply to individual windows
PLOT	logical, TRUE=plot to device

**Details**

This is a spectrogram function similar to the Gabor Transform but uses the MTM (multi-taper method) for spectrum estimation. This is a non-interactive version of MTM.drive.

**Value**

list output of MTMgabor:

sig	input signal
dt	deltat
numfreqs	Number of frequencies output
wpars	input parameters list(Nfft=numfreqs, Ns=Ns, Nov=Nov, fl=fl, fh=fh)
DSPEC	spectrum image
HIMAT	matrix with high values of F-test at 90 percent confidence
DOFMAT	Matrix image of degrees of freedom
FVMAT	Matrix image of F-test values
kdof	test degrees of freedom=2*nwin-2
ppoint	percentage point for confidence bounds
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Note**

The main difference between this and other similar calls is the way the windows are determined.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

Percival, Donald B., Walden, Andrew T. (1993): Spectral Analysis for Physical Applications, Cambridge University Press, Cambridge, 583p.

**See Also**

MTMgabor, evolfft, evolMTM, MTM.drive, GETARAIC, doGABOR.AR, DOsgram

**Examples**

```
data(KH)
###  swig(KH)

Xamp = KH$JSTR[[1]]
Xamp = Xamp[57914:64914]

EV = doGABOR.MTM(Xamp, DT = KH$dt[1], multi = 1, scale.def = 0,
TWIN = 1, TSKIP = .1, PCTTAP = 0.05)
```

---

doMYBUTTS

*Dummy Button Function*


---

**Description**

This is a dummy button function showing how buttons can be created on the fly

**Usage**

```
doMYBUTTS(butt = "", clicks = NULL, x = NULL)
```

**Arguments**

butt	character vector
clicks	clicks
x	locations



---

DOsgram

*Gabor transform*


---

**Description**

Gabor transform with simple spectrum

**Usage**

```
DOsgram(Xamp, DT = 0.008, multi = 1, scale.def = 0,
        TWIN = 2, TSKIP = 0.2, PCTTAP = 0.05, PLOT=TRUE)
```

**Arguments**

Xamp	signal
DT	sample rate interval (s)
multi	Multiples of time window estimate
scale.def	scaling flag for plotting (0=raw, 1=log, 2=sqrt)
TWIN	time for window
TSKIP	time for skip
PCTTAP	percent of taper to apply to individual windows
PLOT	logical, TRUE=plot to device

**Details**

This is a non-interactive version of SPECT.drive.

**Value**

list	
sig	input signal
dt	deltat
numfreqs	Number of frequencies output
wpars	input parameters list(Nfft=numfreqs, Ns=Ns, Nov=Nov, fl=fl, fh=fh)
DSPEC	spectrum image
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

evolMTM, evolfft, evolAR, plotevol

**Examples**

```
data(KH)
### swig(KH)

Xamp <- KH$JSTR[[1]]
Xamp <- Xamp[57914:72989]

Nfft <- 1024   ### fft length
Ns <- 512      ### number of samples in a window
Nov <- 480     ### number of samples of overlap per window
fl <- 0        ### low frequency to return
fh <- 12       ### high frequency to return

EV <- DOsgram(Xamp, DT = 0.008, multi = 1, scale.def = 0,
              TWIN = 2, TSKIP = 0.2, PCTTAP = 0.05)
```

---

dowiggles

*Plot wiggles*

---

**Description**

Plot wiggles

**Usage**

```
dowiggles(AMAT, dt, dx)
```

**Arguments**

AMAT	Matrix of seismic time series
dt	time interval, sec
dx	x-spacing

**Value**

graphical side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

wiggleimage, matsquiggle

**Examples**

```
S1 = symshot1()
```

```
dowiggles(S1$smograms,S1$dt, S1$x)
```

---

downsample

*Interpolate time series at higher sample rate.*

---

**Description**

Interpolate a times series with a higher/lower sample rate for processes that are sensitive to low samples.

**Usage**

```
downsample(sig, dt=0.001, newdt=0.01, PLOT=FALSE )
```

**Arguments**

sig	time series vector
dt	sample rate s/sample
newdt	New, lower sample rate
PLOT	logical, plot both traces, default=FALSE

**Details**

Linear interpolation is performed between samples. If the newdt is an integer multiple of the old dt, The samples will not be modified.

**Value**

time series vector with new sample rate.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```

data(KH)
sig = KH$JSTR[[1]]
#### reduce samples from 125 (0.008) to 25Hz (0.04)
newdt = KH$dt[1]*5
sig2 = downsample(sig, dt = KH$dt[1], newdt = newdt )

L0 = length(sig)
L1 = length(sig2)

op <- par(no.readonly = TRUE)
par(mfrow=c(2,1) )
  plot.ts(ts(sig, deltat=KH$dt[1] ), xlab='s',
  ylab='Amplitude', main=paste('Original', L0) )
  grid()
  plot.ts(ts(sig2, deltat=newdt ), xlab='s',
  ylab='Amplitude', main=paste('Downsample', L1) )
  grid()
par(op)

```

---

editDB

*Edit Data Base*


---

**Description**

Edit, or remove items from an RSEIS data base after it has been read in.

**Usage**

```

editDB(DB, w)
pathDB(DB, path1="", path2="")

```

**Arguments**

DB	RSEIS data base
w	vector of index items to remove
path1	character for old path
path2	character for new path to replace old path

**Details**

The DB is a list. The program cycles through the elements of the list and removes all lines that correspond to the indices given in w.

**Value**

Returns a DB list

**Note**

A problem arises if the makeDB program reads in, or tries to read in files that have not data base header information. This program can eliminate these from the data base.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

makeDB, infoDB

**Examples**

```
##### create a data set and a DB
tdir = tempdir()
data(GH)

DD = data.frame(GH$info)
WV = which(GH$COMPS=='V')

L1 = length(WV)

#####
GIVE = vector(mode='list')

for(j in 1:L1)
{
  i = WV[j]
  AA = DD[i,]

  GIVE[[j]] = list(fn = AA$fn, sta =GH$STNS[i] , comp = GH$COMP[i],
                  dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
                  coords = NA, amp = GH$JSTR[[i]] )
}

##### save files in the tempdir
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM)
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}
```

```
LF = list.files(path=tdir,pattern='.RDS', full.names=TRUE)

##### make the database
cosoDB = FmakeDB(LF, kind=-1)

##### change the DB path:
path1<-tdir
path2<-"."

##### change the path name of the trace files
newDB <- pathDB(cosoDB, path1, path2 )
```

---

EmptyPickfile

*Create an empty RSEIS pickfile structure*

---

### **Description**

Creates a structure list with no data

### **Usage**

```
EmptyPickfile(GH)
```

### **Arguments**

GH                    RSEIS list structure

### **Value**

RSEIS pickfile list

### **Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

### **See Also**

EmptySEIS

### **Examples**

```
data(GH)
EmptyPickfile(GH)
```

---

`EmptySEIS`*Create an empty RSEIS structure*

---

**Description**

Creates a structure list with no data

**Usage**

```
EmptySEIS()
```

**Value**

RSEIS list

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EmptyPickfile

**Examples**

```
EmptySEIS()
```

---

`envelope`*Envelope Function with Hilbert Transform*

---

**Description**

Envelope Function with Hilbert Transform

**Usage**

```
envelope(x)
```

**Arguments**

x                    signal vector

**Details**

Uses the hilbert transform to get the envelope function.

**Value**

vector of the absolute of the hilbert transform

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data(CE1)
ev <- envelope(CE1$y)
plot(CE1$x, CE1$y, type='l')
lines(CE1$x, ev, col='red')
```

---

EPOCHday

*Epoch Day*

---

**Description**

Number of days since Origin Year

**Usage**

```
EPOCHday(yr, jd = 1, origyr = 1972)
```

**Arguments**

yr	year
jd	Julian Day
origyr	origin year, default=1972

**Details**

Either jd or mo, dom can be provided

**Value**

List:

jday	number of days since the start of origin year
origyr	origin year used

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>



**See Also**

EPOCHyear, recdate

**Examples**

```
tyears <- 1973:2009
E1 <- EPOCHday(tyears, jd=1, origyr=1972 )
EPOCHyear(E1$yday, origyr=1972 )
```

---

EPOCHyear

*Epoch Year*

---

**Description**

Get year and julian day given number of days since origin

**Usage**

```
EPOCHyear(iday, origyr = 1972)
```

**Arguments**

iday	Number of days since origin
origyr	origin year, default=1972

**Value**

List:

yr	Year
jd	Julian day in Year

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EPOCHday, recdate

**Examples**

```

tyears <- 1973:2009

E1 <- EPOCHday(tyears, jd=1, origyr=1972 )

EPOCHyear(E1$yday, origyr=1972 )

##### here is an example using year Month and day of month
### use March 19 for each year:
ii <- tojul(tyears, 3, 19)-tojul(tyears, 1, 1)

E1 <- EPOCHday(tyears, jd=ii, origyr=1972 )

EPOCHyear(E1$yday, origyr=1972 )

```

---

ETECTG

*Event Detection*


---

**Description**

Event Detection for a seismic section

**Usage**

```

ETECTG(GH, sel = sel, FRWD = 8, BKWD = 8, sbef = 1,
saft = 6, DFRWD = 0.5, DBKWD = 0.5, thresh = 2,
Tthresh2 = 7, stretch = 1000, flo = 0.1, fhi = 5,
PLOT = FALSE, Kmin = 7, perc = 0.05, kind = 1, DOARAIK = FALSE)

```

**Arguments**

GH	Seismic Structure
sel	select traces
FRWD	forward window, s
BKWD	backward window
sbef	seconds before
saft	seconds after
DFRWD	seconds before
DBKWD	seconds after
thresh	threshold 1
Tthresh2	threshold 2

stretch	stretch factor
flo	low frequency for BP filter
fhi	low frequency for BP filter
PLOT	logical, TRUE=plot diagnostics
Kmin	min number of picks per window
perc	percentage of Kmin allowed
kind	kind of picking
DOARAIIC	TRUE=do auto-regressive AIC method

### Details

Very complicated picking routine - designed for volcanic regions with emergent arrivals. Works with lots of tuning.

### Value

sel	input selection
JJ	index
PPTIM	p-arrivals
PP	all arrivals

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

---

evolAR	<i>Evolutionary Auto-Regressive Spectrum</i>
--------	--

---

### Description

Time varying Auto-Regressive Spectrum (Gabor Transform)

### Usage

```
evolAR(a, dt = 0, numf = 1024, pord = 100, Ns = 0, Nov = 0, fl = 0, fh = 10)
```

### Arguments

a	signal
dt	sample rate interval (s)
numf	Number of frequencies
pord	Order for Auto-regressive calculation
Ns	Number of sample in sub-window
Nov	Number of sample to overlap
fl	low frequency to display
fh	high frequency to display

**Details**

This is a spectrogram function similar to the Gabor Transform but uses the Auto-Regressive method for spectrum estimation.

**Value**

List

sig	input signal
dt	deltat
wpars	input parameters
DSPEC	spectrum image
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

evolfft, evolMTM, MTM.drive, GETARAIC

**Examples**

```
data(KH)
### swig(KH)

Xamp <- KH$JSTR[[1]]

dt <- KH$dt[1]
plot(seq(from=0, length=length(Xamp), by=dt), Xamp, type='l')
## limit the trace, somewhat
Xamp <- Xamp[12670:22669]
plot(seq(from=0, length=length(Xamp), by=dt), Xamp, type='l')

Nfft<-1024  ### fft length
Ns<-512     ### number of samples in a window
Nov<-480   ### number of samples of overlap per window
fl<-0      ### low frequency to return
fh<-12     ### high frequency to return

EV <- evolAR(Xamp, dt = dt, numf =Nfft , pord = 100, Ns = Ns,
             Nov = Nov, fl = fl, fh = fh)

PE <- plotevol(EV, log=1, fl=0.01, fh=fh,
              col=rainbow(100), ygrid=FALSE,
```

```
STAMP="", STYLE="ar")
```

---

 evolvefft

*Spectrogram fft*


---

### Description

Spectrogram using simple fft (Gabor Transform)

### Usage

```
evolvefft(a, dt = 0, Nfft = 0, Ns = 0, Nov = 0, fl = 0, fh = 10, pcttap =
0.05, adjust=TRUE )
```

### Arguments

a	signal
dt	sample rate interval (s)
Nfft	Number of points in fft
Ns	NUmber of sample in sub-window
Nov	number of sample to overlap
fl	low frequency to display
fh	high frequency to display
pcttap	Percent cosine taper for each window
adjust	logical, if TRUE adjust the parameters so the plot looks good (DEFAULT). If FALSE, keep user parameters.

### Details

This is a duplication of the spectrogram function in matlab which applies Welch's Method. Each mini-window is tapered with a cosine window.

### Value

List	
sig	input signal
dt	deltat
wpars	input parameters
DSPEC	spectrum image
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Note**

Parameter adjust is by default TRUE so that the choice of Ns, Nov, and kcol will be optimized, more or less. Set this logical to FALSE to force the function to use user input parameters.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

evolMTM, evolAR, MTM.drive

**Examples**

```
data(CE1)

#### plot signals
plot(CE1$x, CE1$y, type='l')

### set parameters
Nfft<-1024  ### fft length
Ns<-250    ### number of samples in a window
Nov<-240   ### number of samples of overlap per window
fl<-0      ### low frequency to return
fh<-1/(2*CE1$dt)  ### high frequency to return

##### calculate the evolutive fft (Gabor Transform)
EV <- evolfft(CE1$y, dt =CE1$dt , Nfft = Nfft, Ns =Ns , Nov =Nov , fl =fl
, fh = 25)

### plot image, but it does not look too interesting
image(EV$DSPEC)

### plot Gabor transform with special function
PE <- plotevol(EV, log=0, fl=0.01, fh=100, col=rainbow(100), ygrid=FALSE,
STAMP="", STYLE="fft")
```

---

evolMTM

*Evolutive Multi-taper Spectrum*

---

**Description**

Time varying Multi-taper Spectrum (Gabor Transform)

**Usage**

```
evolMTM(a, dt = 0, numf = 1024, Ns = 0, Nov = 0, fl = 0, fh = 10)
```

**Arguments**

a	Signal
dt	Sample rate interval (s)
numf	Number of points in fft
Ns	Number of sample in sub-window
Nov	Number of sample to overlap
fl	low frequency to display
fh	high frequency to display

**Details**

This is a spectrogram function similar to the Gabor Transform but uses the MTM method for spectrum estimation.

**Value**

List	
sig	input signal
dt	deltat
wpars	input parameters
DSPEC	spectrum image
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

evolfft, MTM.drive

**Examples**

```

data(KH)
###  swig(KH)

Xamp <- KH$JSTR[[1]]

dt <- KH$dt[1]
plot(seq(from=0, length=length(Xamp), by=dt), Xamp, type='l')
## limit the trace, somewhat
Xamp <- Xamp[12670:22669]
plot(seq(from=0, length=length(Xamp), by=dt), Xamp, type='l')

Nfft<-4096  ###  fft length
Ns<-512     ###  number of samples in a window
Nov<-480    ###  number of samples of overlap per window
fl<-0       ###  low frequency to return
fh<-12      ###  high frequency to return

EV <- evolMTM(Xamp, dt = dt, numf = Nfft, Ns = Ns, Nov = Nov, fl = fl, fh
= fh)

PE <- plotevol(EV, log=1, fl=0.01, fh=fh, col=rainbow(100), ygrid=FALSE,
STAMP="", STYLE="ar")

##  compare with:
## EVf <- evolfft(Xamp, dt = dt, Nfft =Nfft , Ns =Ns , Nov =Nov , fl =fl, fh = fh)

## PE <- plotevol(EVf, log=1, fl=f1, fh=fh, col=rainbow(100), ygrid=FALSE,STAMP="", STYLE="fft")

```

---

FAKEDATA

*Fake Data for Examples.*


---

**Description**

Create a list of artificial seismic traces to illustrate examples that require a database or long sequences.

**Usage**

```

FAKEDATA(amp, OLDdt = 0.01, newdt = 0.1, yr = 2000,
JD = 5, mi = 0, sec = 0, Ntraces = 48, seed = 200,
noise.est = c(1, 100), verbose = FALSE)

```



**Arguments**

amp	vector, some signal that will be repeated
OLDdt	Original sample rate
newdt	New sample rate, usually less than the original
yr	year
JD	starting Julian day
mi	starting minute
sec	starting second
Ntraces	number of traces
seed	random seed
noise.est	2-vector, starting and ending sample to estimate noise level of trace
verbose	logical, message feed back

**Details**

The input signal can be any time series, or even a made up signal. This is just to give the look of the result something like real data. The noise level is extracted from the mean and std of the real data at the samples indicated by noise.est.

The sampling rate (dt, sec/sample ) is increased mainly for speed and plotting. This may be skipped for certain functions involving spectrum analysis.

The signal is distributed randomly in each hour along the total span of the requested period, i.e. each hour has one instance of the signal.

The date is arbitrary, of course.

**Value**

List of data in a format similar to the output of GET.seis.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

GET.seis

**Examples**

```
##### get a time series
data(KH)

amp = KH$JSTR[[1]]
OLDdt = KH$dt[1]
##### downsample to:
newdt = 0.1
```

```

JK = FAKEDATA(amp, OLDdt=OLDdt, newdt = 0.1, yr = 2000,
              JD = 4, mi = 12, sec = 0, Ntraces = 3,
              seed=200, noise.est=c(1, 100) , verbose=TRUE )

op <- par(no.readonly = TRUE)
par(mfrow=c(length(JK), 1) )
for(i in 1:length(JK) )
{
  DATTIM = paste(c(unlist(JK[[i]]$DATTIM), JK[[i]]$N), collapse=' ')

  plotGH( JK[[i]] )
  mtext(DATTIM, side=3, at=JK[[i]]$DATTIM$t2/2)
}
par(op)

```

---

filedatetime

---

*Create a character string from a date*


---

### Description

Create a character string from a date for naming unique output files.

### Usage

```
filedatetime(orgtim, tims=0, datesep="-", timesep="_", secsep="_")
```

### Arguments

orgtim	time vector of length 5: c(yr, jd, hr, mi, sec)
tims	seconds to add to orgtim, default=0
datesep	character, seperater for the date
timesep	character, seperator for the time
secsep	character, seperator for the seconds

### Value

filename	character string
----------	------------------

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```

data(GH)

g1 <- getGHtime(GH)
g2 <- unlist(g1)

filedatetime(g2, 1)

```

---

FILT.SEISN

*Filter Traces*


---

**Description**

Filter Traces in a seismic structure

**Usage**

```

FILT.SEISN(TH, sel = 1:length(TH$JSTR),
FILT = list(ON = TRUE, fl = 0.5, fh = 7, type = "HP",
proto = "BU", RM=FALSE, zp=TRUE ), TAPER = 0.1, POSTTAPER = 0.1, AUGMENT=FALSE)

```

**Arguments**

TH	Seismic structure
sel	selection of traces
FILT	filter definition
TAPER	filter taper
POSTTAPER	taper after filter
AUGMENT	Logical, FALSE

**Details**

RSEIS Seismic structure is filtered, trace by trace. If AUGMENT is TRUE, traces are augmented at beginning and end, filtered and then truncated to suppress edge effects. In that case no tapering is applied post filter.

**Value**

RSEIS Seismic structure, traces are filtered and a proc is added to the trace history.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

butfilt

**Examples**

```
## Fdef = choosfilt()
Fdef <- list(ON=FALSE, fl=0.5, fh=7.0, type="BP", proto="BU", RM=FALSE, zp=TRUE )
data("GH")
sel <- which(GH$COMPS=="V")

sel <- 1:3
KF <- FILT.SEISN(GH, sel = sel, FILT=Fdef)
swig(KF, sel=sel, SHOWONLY=0)
```

---

FILT.spread

---

*Filter trace with a spread of filters*


---

**Description**

Show a time series and a spread of user defined filters to show signal at a variety of bandwidths.

**Usage**

```
FILT.spread(x, y, dt, fl = fl, fh = fh, sfact = 1,
  WIN = NULL, PLOT = TRUE, TIT = NULL, TAPER = 0.05,
  POSTTAPER=0.05, RM=FALSE, zp=TRUE )
```

**Arguments**

x	x-axis
y	y-amplitude
dt	delta-t, sec
fl	vector of low frequency cut offs
fh	vector of high frequency cut offs
sfact	scale factor, 0,1
WIN	xlimits to constrain plotting
PLOT	logical, plotting
TIT	title
TAPER	taper data prior to filter, percent cosine, default=NULL
POSTTAPER	taper output after filter, percent cosine, default=0.05
RM	Remove mean value from trace, default=FALSE
zp	zero phase filter, default=TRUE

**Details**

Use the TAPER and POSTTAPER to reduce the edge effects prior to and after filtering.

**Value**

list:

FMAT	matrix of time series filtered
Notes	Notes for filter of each element of FMAT

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

butfilt, PLOT.MATN

**Examples**

```
data(KH)
dt <- KH$dt[1]

y <- KH$JSTR[[1]]

x <- seq(from=0, by=dt, length=length(y))

f1 <- rep(1/100, 5)
fh <- 1/c(1,2,5,10,20)

FILT.spread(x, y, dt, f1 = f1, fh = fh, sfact = 1,
            WIN = NULL, PLOT = TRUE, TIT = NULL, TAPER = 0.05)
```

---

filterstamp

*Make Filter Stamp*

---

**Description**

Create an text stamp describing a filter

**Usage**

```
filterstamp(f1=1/2, fh=10, type="BP")
```

**Arguments**

f1	vector, low frequency
fh	vector, high frequency
type	vector, type of filter

**Details**

If the frequency is less than 1, the period is displayed. For now only 3 digits are displayed. If the first argument, f1, is a list the parameters are extracted from the list and the other arguments are ignored.

**Value**

stamps            text strings

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

FILT.spread

**Examples**

```
f1 <- c(0.01, 2)
fh <- c(10, 20)
type <- "BP"
filterstamp(f1, fh, type)
```

```
FILT<-list(ON=TRUE, f1=1/2, fh=12, type="HP", proto="BU")
filterstamp(FILT)
```

```
FILT<-list(ON=TRUE, f1=1/2, fh=12, type="BP", proto="BU")
filterstamp(FILT)
```

```
FILT<-list(ON=TRUE, f1=1/2, fh=12, type="LP", proto="BU")
filterstamp(FILT)
```

---

finteg	<i>Integration in Frequency Domain</i>
--------	--

---

**Description**

Integration of seismic signal in Frequency Domain. Used for converting velocity seismogram to displacement.

**Usage**

```
finteg(data, dt)
```

**Arguments**

data	time series
dt	sample interval

**Value**

Integrated time series signal

**Note**

To avoid problems with dividing by very small numbers, water level is set =1.e-8

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
## waterlevel=1.e-8

dfor5 <- rnorm(1000)

idfor5 <- finteg(dfor5, 0.008)
```

---

fixcompname	<i>Fix component names for uniformity</i>
-------------	---

---

**Description**

Fix component names for uniformity

**Usage**

```
fixcompname(comp)
```

**Arguments**

comp           4, "SHV"

**Details**

Translate the component names to something uniform that can be used for sorting and other functions.

**Value**

one of "V", "N", "E"

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
fixcompname("SHV")  
fixcompname("SHE")
```

---

fixcomps	<i>Fix Station Component Names</i>
----------	------------------------------------

---

**Description**

Convert components to common names: V N E

**Usage**

```
fixcomps(oldcomps, SEGY = FALSE)
```



**Arguments**

oldcomps            vector of compnets  
SEGY                logical, TRUE= segy data with components 4,5,6 or 1,2,3

**Details**

Attempts to convert irregular component names to common format for later processing.

**Value**

character vector

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

fixcompname

**Examples**

```
comp <- c("v", "e")  
fixcomps(comp)
```

---

fixNA

*Fix NA values.*

---

**Description**

Replace NA values in a time series with mean values between end points of missing segments, or first and last real values in case the NA's are at the beginning or ends of traces.

**Usage**

```
fixNA(y)
```

**Arguments**

y                    numeric vector

**Details**

fixNA searches for stretches of NA 's in a time series and replaces the NA values with numeric values based on the two end points of each section.

**Value**

numeric vector with no NA values.

**Note**

function is used primarily in filter applications.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

butfilt

**Examples**

```
## source("~/Site/TA_DATA/CODE/fixNA.R")

### last samples are NA
zig = rnorm(25)
zig[10:15] = NA

noNA = fixNA(zig)

### first samples are NA
zig = rnorm(25)
zig[1:5] = NA
noNA = fixNA(zig)

zig = rnorm(25)
zig[1:5] = NA
zig[21:25] = NA

noNA = fixNA(zig)

zig = rnorm(25)
zig[1] = NA
zig[21:25] = NA
zig[10:12] = NA

noNA = fixNA(zig)
cbind(zig, noNA)
```

---

fixUWstasLL	<i>fixUWstasLL</i>
-------------	--------------------

---

**Description**

Matches station locations to pickfile stations

**Usage**

```
fixUWstasLL(STAS, stafile)
```

**Arguments**

STAS	structure of station lat, lon, z
stafile	station file

**Details**

Matches station locations to pickfile stations

**Value**

structure of station lat, lon, z

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

fromjul	<i>given julian day and year get month/day</i>
---------	--

---

**Description**

given julian day and year get month/day

**Usage**

```
fromjul(jul, yy)
```

**Arguments**

jul	Julian Day
yy	year

**Value**

list(mo=mm, dom=dd)

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

tojul

**Examples**

```
iyear <- 2001
jul <- 233
inine <- tojul(iyear,1,1);
ijul <- inine + jul - 1;
fromjul( ijul, iyear);
```

---

FRWDft

*Forward fourier Transform*

---

**Description**

Forward fourier Transform

**Usage**

```
FRWDft(g, n, tstart, dt)
```

**Arguments**

g	input signal
n	number of points
tstart	start of trace
dt	sample interval, s

**Value**

G	fourier components
f	frequency vector
t	time vector

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

INVRft

**Examples**

```
zil <- rnorm(300)
fss <- FRWDft( zil, length(zil), 0, 0.004)
```

---

gaddtix

*add tic marks*

---

**Description**

Add tic marks to plot

**Usage**

```
gaddtix(side = 3, pos = 0, tck = 0.005, at = c(0, 1),
labels = NULL, col = 2, addline = FALSE, ...)
```

**Arguments**

side	side = 1, 2, 3, 4
pos	relative to axis
tck	tic length
at	vector of positions
labels	vector of labels
col	color for plotting
addline	add lines
...	graphical parameters from par

**Value**

Graphical side effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

par

**Examples**

```
plot(c(0,1), c(0,1), type='n', ann=FALSE, axes=FALSE)

gaddtix(side=1, pos=0, tck=-0.01, at=seq(from=0, to=.5, by=.2) ,
labels=seq(from=0, to=.5, by=.2), col=1)
```

GAZI

*Get azimuthal particle motion***Description**

Do particle motion analysis

**Usage**

```
GAZI(ADAT, dt = 1, ex = seq(0, 100),
comp = c(4, 5, 6), sta = "ZZZ", az = 0,
len = 50, shift = 10, prev = 1, fileid = "", picks = NA, labs = NA)
```

**Arguments**

ADAT	Matrix of 3 component seismic data
dt	delta T (s)
ex	x-axis vector
comp	component names
sta	station name
az	azimuth of station orientation
len	length of time series
shift	amount to shift per window
prev	length of buffer at beginning of trace
fileid	character string to put on plot
picks	arrival times for annotation
labs	labels for arrival times for annotation

**Value**

```
list(aex=aex[1:jall], rateig=rateig[1:jall], aaz=aaz[1:jall], ai=ai[1:jall], figaz=figaz, azpar=azpar, in-
cpar=incpar )
```

**Examples**

```
data("GH")

temp <- cbind(GH$JSTR[[4]], GH$JSTR[[5]], GH$JSTR[[6]])

pmolabs <- c("Vertical", "North", "East")

G <- GAZI(temp, dt =GH$dt[4] , comp = pmolabs, sta = GH$STNS[4] ,
az = 0, len =75, shift = 10, prev = 1)
```

---

genrick

*Ricker Wavelet*

---

**Description**

Generate a ricker wavelet of a specified frequency and length

**Usage**

```
genrick(freq, dt, nw)
```

**Arguments**

freq	frequency of ricker wavelet
dt	Time sample rate (s)
nw	length of wavelet.

**Value**

ricker wavelet as a vector.

**Note**

Original code by Leonard Lisapaly (leonardl@fisika.ui.ac.id), converted to R by J.M. Lees.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
dt <- 0.01
freq <- 16
nlen <- 35

G <- genrick(freq, dt, nlen)

tee <- seq(from=0, by=dt, length=length(G))

plot(tee, G, type='l')
```

---

`get.corner`*Get Corner Frequency: Linear Model*

---

**Description**

Search for low frequency asymptote, corner frequency, and fall off slope of seismic spectrum.

**Usage**

```
get.corner(INfreq, INspec, dt, f1, f2, PLOT = FALSE, VERBOSE = FALSE)
```

**Arguments**

INfreq	frequency vector
INspec	spectrum
dt	deltaT
f1	low frequency for modeling, Hz
f2	High frequency for modeling, Hz
PLOT	logical, TRUE=plot
VERBOSE	TRUE=diagnostics

**Details**

This routine does not assume any particular mathematical model. It searches for a three parameters that describe two lines that mimic the displacement spectrum. The search is done via least squares.

**Value**

Model of 3 parameters, best fit.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>



**See Also**

brune.doom

**Examples**

```

data(CE1)

## set frequency range for modeling for this high frequency data
## we use f2 = 50, but for volcano data should be f2<15

f1 <- 0.01
f2 <- 50.0

## set up data and parameters
amp <- CE1$y
len2 <- 2*next2(length(amp))
a <- list(y=amp, dt=CE1$dt)

Spec <- MTMdisp(a, f1=f1, f2=f2, len2=len2, PLOT=FALSE )

lspec <- Spec$displ

### get initial estimate of parameters
xc <- get.corner( Spec$f , lspec, CE1$dt, f1, f2, PLOT=FALSE)

```

---

GET.seis

*Reads various seismic file formats*


---

**Description**

This function calls binary routines to read in 'segy', 'sac'.

**Usage**

```
GET.seis(fnames, kind = 1, Iendian=1, BIGLONG=FALSE ,
HEADONLY=FALSE, PLOT = -1, RAW=FALSE)
```

```
JGET.seis(fnames, kind = 1, Iendian=1, BIGLONG=FALSE ,
HEADONLY=FALSE, PLOT = -1, RAW=FALSE)
```

**Arguments**

fnames            list of file names.  
kind                an integer -1, 0, 1, 2 ; 0="RDATA" , -1="RDS", 0="RDATA", 1 = "segy", 2 =  
"sac", see notes below

Iendian	vector, Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE=long=8 bytes
HEADONLY	logical, TRUE= header information only; not seismic trace will be returned (runs a little faster).
PLOT	integer, <0 no plot; 0 interactive; >0 number of seconds to sleep
RAW	logical, default=FALSE(convert to volts) , TRUE (return counts instead of volts)

### Details

"kind" can be numeric or character: options are 'RDS', 'RDATA', 'SEGY', 'SAC', corresponding to (-1, 0, 1, 2).

Uses readBin to extract data in SAC/SEGY format. User must know what kind of machine the data was created on for I/O purposes.

If data was created on a little endian machine but is being read on big endian machine, need to call the endian "swap" for swapping.

Iendian can be a vector if input files have different endian-ness.

If data was created on a machine with LONG=4 bytes, be sure to call the program with BIGLONG=FALSE.

The data returned is a list of lists, each element is one trace not necessarily related to the other traces in the list.

Once the data is read in, use prepSEIS to reformat the data into a list more amenable to further analysis in RSEIS.

See examples below for different cases.

### Value

List containing the seismic data and header information. Each trace consists of a list with:

fn	original file name
sta	station name
comp	component
dt	delta t in seconds
DATTIM	time list
yr	year
jd	julian day
mo	month
dom	day of month
hr	hour
mi	minute
sec	sec
msec	milliseconds
dt	delta t in seconds

t1	time start of trace
t2	time end of trace
off	off-set
N	number of points in trace
units	units
amp	vector of trace values

**Note**

The easiest way to process data is to convert the data to an R-format type, using either `save` (`kind=0`) or `saveRDS` (`kind=-1`). If these are used then I/O is simple.

**OLDER:**

Information in the file names is ignored, so be sure to modify headers prior to using this method of extracting meta-data. (Or modify the meta data from the file names after reading in the data.)

For SEGY files, in LINUX-UNIX, use: `rename`, `segymod` (PASSCAL) to modify the headers

JGET.seis extracts digital seismic data from binary files stored in the file system. The program uses `readBin` for I/O and passes data back to R. Currently SAC, SEGY formats are installed but it is easy to extend. AH format is available for LINUX systems, but there were problems compiling in WINDOWS and MACOS so this feature was removed.

A filter for mseed format is currently being developed. Could use package 'IRISSeismic'

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**See Also**

`plotJGET`, `JSAC.seis`, `prepSEIS`, `Mine.seis`

**Examples**

```
data(GH)

DD = data.frame(GH$info)

#### get only vertical traces
WV = which( GH$COMPS=='V' )
L1 = length(WV)

GIVE = vector(mode='list')

for(j in 1:L1 )
{
  i = WV[j]
  AA = DD[i,]
  GIVE[[j]] = list(fn = AA$fn, sta =GH$STNS[i] , comp = GH$COMP[i],
                  dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
```

```

        coords = NA, amp = GH$JSTR[[i]] )
    }
    #### par(mfrow=c(length(GIVE) , 1) )
    # for(i in 1:length(GIVE) ) { plotGH(GIVE[[i]]) }
    tdir = tempdir()
    for(i in 1:length(GIVE) )
    {
        sig = GIVE[[i]]
        d1 = dateStamp(sig$DATTIM, sep='_')
        nam1 = paste(d1,sig$sta, sig$comp, sep='_')
        nam2 = paste0(nam1, '.RDS')
        nam3 = paste(tdir, nam2, sep='/')
        saveRDS(file=nam3, sig)
    }
    ##### Now read files and make the DataBase:
    LF = list.files(path=tdir,pattern='.RDS', full.names=TRUE)

    Gseis = GET.seis(LF, kind = -1, Iendian=1, BIGLONG=FALSE ,
    HEADONLY=FALSE, PLOT = -1, RAW=FALSE)

    zed <- prepSEIS(Gseis)

    #### plot the data, and interact with the data
    swig(zed, sel=which(zed$COMPS=='V'), SHOWONLY=0)

    if(interactive()){ plotJGET(Gseis) }

    ### for data created on UNIX (SUN) but read on linux:
    ### S1 <- GET.seis(Lname, kind = 1, Iendian="swap", BIGLONG=FALSE, PLOT = -1)

    ### for data created on linux (32 bit) but read on linux 64 bit:
    ### S1 <- GET.seis(Lname, kind = 1, Iendian="little", BIGLONG=FALSE, PLOT = -1)

    ### for SEG Y data created on linux (64 bit) but read on linux 32 bit:
    ### S1 <- GET.seis(Lname, kind = 1, Iendian="little", BIGLONG=TRUE, PLOT = -1)

    ### for SAC data created on MAC-OS (64 bit) but read on linux 32 bit:
    ### S1 <- GET.seis(Lname, kind = 2, Iendian="swap", BIGLONG=TRUE, PLOT = -1)

```

---

get.slepians

*Get Slepian Tapers*


---

## Description

Return a matrix of Slepian tapers

**Usage**

```
get.slepians(npoints = 900, nwin = 5, np_i = 3)
```

**Arguments**

<code>npoints</code>	Number of points to return
<code>nwin</code>	Number of windows (default =5)
<code>np_i</code>	Pi-Prolate number (3)

**Details**

This function only returns the tapers for inspection. To apply the tapers use the function `mtapspec`.

**Value**

Matrix: `nwin` vectors of `npoints` Slepian tapers

**Author(s)**

Jonathan M. Lees<[jonathan.lees.edu](mailto:jonathan.lees.edu)>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

`mtapspec`

**Examples**

```
nwin <- 5
np_i <- 3
npoints <- 900
sleps <- get.slepians(npoints, nwin, np_i)

matplot(sleps, type='l', xlab="Index", ylab="Taper Amplitude")
legend('topleft', legend=1:nwin, lty=1:nwin, col=1:nwin)
```

---

Get1Dvel                      *Read 1D velocity model*

---

**Description**

Read in a velocity model

**Usage**

```
Get1Dvel(infile, PLOT = TRUE)
```

**Arguments**

infile	Path to ascii-text model
PLOT	logical, TRUE=plot

**Details**

Reads Velocity model from a text file

**Value**

LIST:

zp	vector of Tops of Layers, P-wave, (km)
vp	vector of velocities of Layers, P-wave,(km/s)
ep	errors for velocities, P-wave,(km/s)
zs	vector of Tops of Layers, S-wave, (km)
vs	vector of velocities of Layers, S-wave,(km/s)
es	errors for velocities, S-wave,(km/s)
name	character, name of model
descriptor	character vector description of model

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

Plot1Dvel, Comp1Dvel, Comp1Dvels, travel.time1D

**Examples**

```
data(VELMOD1D)
```

```
Get1Dvel(VELMOD1D, PLOT=TRUE)
```

---

`GETARAIC`*Auto-Regressive AIC estimate of arrival time*

---

**Description**

Auto-Regressive AIC for arrival estimate, signal detection

**Usage**

```
GETARAIC(z4, DT = 0.008, Mar = 8, O1 = 2, O2 = 0.2, WW = 2, T1 = 1, PLOT = FALSE)
```

**Arguments**

<code>z4</code>	signal time series
<code>DT</code>	sample rate,s
<code>Mar</code>	AR Model Order
<code>O1</code>	window before, s
<code>O2</code>	window after, s
<code>WW</code>	window length, s
<code>T1</code>	initial guess, number of samples from beginning of trace
<code>PLOT</code>	logical, TRUE =plot

**Details**

Method of Sleeman for automatic phase determination.

**Value**

<code>Taic</code>	Arrival time of wave
-------------------	----------------------

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Sleeman

**See Also**

PSTLTcurve

**Examples**

```

data(CE1)
plot(CE1$x, CE1$y, type='l')

Xamp = CE1$y[CE1$x>4.443754 & CE1$x<6.615951]
Mar=8
z4 = Xamp
DT = CE1$dt
T1 = 50

O1 = 10*DT
O2 = 10*DT
WW = 10*DT
Nz4 = length(z4)

araict = GETARAIC(Xamp, DT=CE1$dt, Mar=8, T1=T1, O1=O1, O2=O2, WW=WW, PLOT=TRUE)

```

---

getb1b2

*Event Detection*


---

**Description**

Used for event detection

**Usage**

```
getb1b2(J, L, zwin, maxx, max2)
```

**Arguments**

J	Thresh.J
L	Thresh.J
zwin	maximum of forwd and bakwrđ windows
maxx	max number of points
max2	all points

**Value**

vector c(b1,b2)

**Note**

Used for thresholding on event detection.



**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**See Also**

Thresh.J, ETECTG

---

 getEcard

---

*Error Card*


---

**Description**

Location Error Card

**Usage**

getEcard(ECARD)

**Arguments**

ECARD	error card from Lquake
-------	------------------------

**Value**

LOC	character, location
rms	root mean square error
meanres	mean residual
sdres	standard deviation of residuals
sdmean	standard error of mean
sswres	sum squares
ndf	number degrees of freedom
fixflgs	flags for inversion
sterrx	error in x-direction
sterry	error in y-direction
sterrz	error in z-direction
sterrt	error in origin time
mag	mag
sterrmag	error for mag

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

---

getFcard	<i>Parse UW F Card</i>
----------	------------------------

---

**Description**

get F-card information

**Usage**

getFcard(FCARD)

**Arguments**

FCARD	Error Ellipsoid card
-------	----------------------

**Value**

List:

azim1	angle, degrees
plunge1	angle, degrees
val1	value
azim2	angle, degrees
plunge2	angle, degrees
val2	value
azim3	angle, degrees
plunge3	angle, degrees
val3	value
herr	error
verr	vertical error

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

`getGHtime`*Get Seismic reference time*

---

**Description**

Extract the times of all traces relative to a reference trace on a seismic RSEIS list.

**Usage**

```
getGHtime(GH, wi = 1, pix = NULL)
```

**Arguments**

GH	RSEIS seismic data list
wi	which event to use as a reference baseline
pix	list of time to difference

**Value**

list: times relative to reference time:

yr	year
jd	julian day
hr	hour
mi	minute
sec	second
spix	seconds after reference

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

secdifL, secdif

**Examples**

```
data(GH)
```

```
getGHtime(GH)
```

---

`getHcard`*Parse UW Hires location Card*

---

**Description**

Extract High resolution information from H-card

**Usage**

```
getHcard(hcard)
```

**Arguments**

<code>hcard</code>	ascii h-card
--------------------	--------------

**Value**

List:

<code>yr</code>	Year
<code>mo</code>	Month
<code>dom</code>	Day of Month
<code>hr</code>	Hour
<code>mi</code>	minute
<code>sec</code>	second
<code>lat</code>	latitude
<code>lon</code>	longitude
<code>z</code>	depth
<code>mag</code>	magnitude

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EmptyPickfile

---

getIRIS                      *get Hypocenters from IRIS web site*

---

### Description

Convert hypocenters from the IRIS website and prepare for plotting in GEOmap

### Usage

```
getIRIS(fn, skip=0)
getANSS(fn, skip=2)
```

### Arguments

fn	character, file path name
skip	numeric, number of lines to skip (e.g. for the header)

### Details

Reads in a file dumped out by the website selection box.

### Value

list:

yr	vector year
dom	vector, day of month
mo	vector, mo
hr	vector, hour
mi	vector, minute
sec	vector, sec
lat	vector, latitude
lon	vector, longitude
z	vector, depth
mag	vector, magnitude

### Note

Be careful about headers and lines that need to be skipped.

for IRIS: <http://www.iris.washington.edu/data/event/eventsearch.htm>

For ANSS: <http://www.quake.geo.berkeley.edu/annc/catalog-search.html>

For NEIC (yet to be added) [http://earthquake.usgs.gov/earthquakes/eqarchives/epic/epic\\_global.php](http://earthquake.usgs.gov/earthquakes/eqarchives/epic/epic_global.php)

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees@unc.edu&gt;

**See Also**

getjul

**Examples**

```
fn <- tempfile()
K = c(
'Date      Time          Lat      Lon Depth  Mag Magt Nst Gap Clo RMS SRC  Event ID',
'-----',
'1994/09/06 09:37:36.48 40.1330 144.6240 33.40 4.60 Mb 28      1.22 NEI 199409064025',
'1994/09/06 10:00:02.97 36.4840 140.5730 66.60 4.90 Mb 39      0.88 NEI 199409064028',
'1994/09/06 10:07:16.53 40.1700 144.5890 33.00 4.70 Mb 49      1.09 NEI 199409064029',
'1994/09/06 17:31:52.27 42.6220 142.7000 33.00 5.00 Mb 13      0.54 NEI 199409064042')

cat(file=fn, K, sep='\n')

### check: z = scan(file=fn, what='', sep='\n')

g <- getANSS(fn, skip=2)
g$jd <- getjul(g$yr, g$mo, g$dom)
```

getjul

*Get Julian day***Description**

Get Julian day

**Usage**

getjul(year, month, day)

**Arguments**

year	year
month	month
day	day of month

**Value**

Julian Day

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

getmoday

**Examples**

getjul(2003, 11, 13)

---

getmoday

*Get Month Day*

---

**Description**

Get month day from julian day and year

**Usage**

getmoday(jul, iyear)

**Arguments**

jul	julian day
iyear	Year

**Value**

mo	Month
dom	day of month

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

getmoday(234, 2005)

**getNcard***Parse Name Card*

---

**Description**

extract name from N-card

**Usage**

```
getNcard(ncard)
```

**Arguments**

ncard            ncard from UW-pickfile

**Value**

Ncard

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EmptyPickfile

---

**getPDEcsv***Unpack PDE file*

---

**Description**

Unpack PDE file as CSV file or ascii screen dump

**Usage**

```
getPDEcsv(pde = 'filename')  
getPDEscreen(pde = 'filename' )
```

**Arguments**

pde            character, file name



**Details**

Download pde from: <http://neic.usgs.gov/neis/epic/epic.html>. csv version uses comma separated values. screen versions uses the screen dump and a parser

**Value**

list of locations, times and magnitude

**Note**

if using screen dump, may need to clean up file a bit first.

**Author(s)**

Jonathan M. Lees<[jonathan.lees@unc.edu](mailto:jonathan.lees@unc.edu)>

**References**

<http://neic.usgs.gov/neis/epic/epic.html>

**Examples**

```
##### copy/paste from the screen dump at the NEIC web site

fn <- tempfile()
K = c(
' PDE-Q 2008 12 31 053408.80 40.11 -77.00 1 2.4 LgGS ... .. ',
' PDE-Q 2008 12 31 084757.50 46.75 154.41 14 4.9 mbGS ... .. ',
' PDE-Q 2008 12 31 090228 44.53 -110.36 4 3.6 MLSLC ... .. ',
' PDE-Q 2008 12 31 110505 33.94 -118.78 14 3.1 MLPAS 2F. .... ',
' PDE-Q 2008 12 31 113957.56 4.91 127.43 77 5.4 MwGS ..M ..... ',
' PDE-Q 2008 12 31 140227.55 -25.35 -177.61 154 5.3 MwGS ..M ..... ')

cat(file=fn, K, sep='\n')

### check: z = scan(file=fn, what='', sep='\n')

g <- getPDEscreen(pde = fn)
```

---

getpfile

*Get Pick File*

---

**Description**

Read Pick File to R

**Usage**

```
getpfile(uwpickfile, stafile = NULL)
```

**Arguments**

uwpickfile	pick file
stafile	station file

**Details**

University of washington Format pickfiles are used. See EmptyPickfile for the structure stored.

**Value**

pickfile structure

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EmptyPickfile

---

getphaselag2

*Phase Lag*

---

**Description**

Use MTM spectrum to estimate phase lag between two signals.

**Usage**

```
getphaselag2(y1, y2, DT = 0.008, frange = c(0, 20),  
PLOT = FALSE, PLOT1 = FALSE, PLOT2 = FALSE)
```

**Arguments**

y1	vector times series one
y2	vector times series two
DT	deltaT sample rate, s
frange	vector, frequency bounds for analysis
PLOT	logical, TRUE=diagnostic plot
PLOT1	logical, TRUE=diagnostic plot
PLOT2	logical, TRUE=diagnostic plot

**Details**

uses the slope of the cross spectrum to estimate the phase lag.

**Value**

phase lag, seconds

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

mtapspec

**Examples**

```
data("GH")

Xamp1<-GH$JSTR[[1]]
Xamp1<-Xamp1[1123:2000]

Xamp2<- GH$JSTR[[4]]
Xamp2<-Xamp2[1123:2000]
plot(Xamp1,type='l')
lines(Xamp2,type='l',col='red')

pshift <- getphaselag2(Xamp1, Xamp2, DT=GH$info$dt[1],
frange=c(5, 15), PLOT=TRUE)
```

---

getrdpix

*get read picks*

---

**Description**

get read picks

**Usage**

```
getrdpix(zloc, zenclick, sel, NH)
```

**Arguments**

zloc	location list
zenclick	number of picks
sel	sel vector in swig
NH	RSEIS list

**Details**

Used internally in swig

**Value**

list: rd: date/times of picks for stations and comps

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

---

getseis24

*Get 24 Hours of Seismic Data*

---

**Description**

Get 24 Hours of Seismic Data

**Usage**

```
getseis24(DB, iyear = 2009, iday = 1, usta = "",
  acomp = "", kind = 1, Iendian=1, BIGLONG=FALSE)
```

**Arguments**

DB	Data base of meta-data about the seismic trace files
iyear	Year for extraction
iday	Julian day for extraction
usta	station to show
acomp	component to show
kind	kind of data, default=1, 0="RDATA" , -1="RDS", 0="RDATA", 1 = "segy", 2 = "sac"
Iendian	Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE=long=8 bytes

**Details**

The DB file consists of a list of information on where to find the data and what times are covered.  
DB is

**fn** full path to file

**yr** year

**jd** julian day

**hr** hour

**mi** minute

**sec** second

**dur** duration, seconds

**origyr** origin time for epoch calculations

**Value**

yr	start year
jd	start julian day
t1	start t1 (with epoch)
t2	start t2 (with epoch day)
ed	epoch day
hr	start hour
mi	start minute
sec	start seconds
gamp	Amplitude of each trace
gdt	delta-t, sample interval, in seconds
gnam	station name
gfile	file information
sigs	List of time series
zna	List of NA values in each time series

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

setupDB, plotseis24

**Examples**

```

data(KH)

amp = KH$JSTR[[1]]
OLDdt = KH$dt[1]
newdt = 0.1
yr = 2000
GIVE = FAKEDATA(amp, OLDdt=0.01, newdt = 0.1, yr = 2000,
                JD = 4, mi = 12, sec = 0, Ntraces = 24*3,
                seed=200, noise.est=c(1, 100) , verbose=TRUE )
#### each trace in a separate file
tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)

DB = FmakeDB(LF, kind=-1)

IDB = infoDB(DB)

START = list(yr =yr , jd= 5 , hr= 0 , mi= 0 ,sec= 0)

END = list(yr =yr , jd= 7 , hr= 0 , mi= 0 ,sec= 0)

h = getseis24(DB, iyear = 2000, iday = 5, usta = IDB$usta,
              acomp = IDB$ucomp, kind = -1, Iendian=1, BIGLONG=FALSE)

pjj <- plotseis24(h, dy=1/18, FIX=24, SCALE=1,
                 FILT=list(ON=FALSE, fl=0.05 , fh=20.0, type="BP", proto="BU"),
                 RCOLS=c(rgb(0.2, .2, 1), rgb(.2, .2, .2)) )

```

**Description**

Uses a Pickfile and the Waveform file, and creates a vector ordering the waveforms by P-wave arrival.

**Usage**

```
getvertsorder(P, GU)
```

**Arguments**

P	Pickfile Structure
GU	Waveform structure

**Details**

Waveforms structure may already have pickfile, but this is overridden by input pickfile P.

**Value**

list:	
sel	index of traces in order of first P-wave arrival
win	vector, c(1,2), time window from the first arrival to the last

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig

**Examples**

```
data(GH)

vertord <- getvertsorder(GH$pickfile, GH)

swig(GH, sel=vertord$sel, WIN=vertord$win, SHOWONLY=TRUE)
```

GH

*Earthquake Seismic Data***Description**

Example of seismic data structure. Geothermal Earthquake.

**Usage**

data(GH)

**Format**

List, consisting of:

**JSTR** list of digital seismic data traces

**STNS** vector of stations

**dir** directory

**ifile** original file names

**COMPS** Component names, V N E, e.g.

**OCOMPS** Old Component names

**dt** vector of delta-t, sampling time intervals

**KNOTES** Notes for plotting on panels

**info** List, detailed information about traces, including

**dat** not used

**nn** Number of traces

**ex** time axis for plotting

**pcol** colors for plotting

**ok** which traces are okay

**wintim** window span time, seconds

**ftime** alphanumeric time stamp

**pickfile** pickfile, see below

**velfile** velocity model list

**stafle** station information list including lat, lon, z

**aname** source name for loading

**UWFILEID** event ID number

The info list consists of:

**fn** file name

**name** identification name



**yr** start year  
**jd** start julianday  
**mo** month  
**dom** day of month  
**hr** hour  
**mi** minute  
**sec** second  
**msec** millisecond  
**dt** delta-t  
**t1** time 1  
**t2** time 2  
**off** offset  
**n1** number of samples  
**n2** not used  
**n3** not used  
**n** number of samples

The pickfile consists of:

**LOC** list(yr, jd, mo, dom, hr, mi, sec, lat, lon, z, mag, gap, delta , rms, hozerr)  
**MC** list(az1, dip1, az2, dip2, dir, rake1, dipaz1, rake2, dipaz2, F=list(az, dip), G=list(az, dip),  
 U=list(az, dip), V=list(az, dip), P=list(az, dip), T=list(az,dip),sense,M=list( az1, d1, az2,  
 d2, uaz, ud, vaz, vd, paz, pd , taz, td), UP=TRUE, icol=1, ileg, fcol='red', CNVRG, LIM  
 =c(0,0,0,0))  
**STAS** list(tag, name, comp, c3, phase, sec, err, pol, flg , res)  
**LIP** vector, length=6  
**H** list(yr,mo,dom,hr,mi,sec,lat,lon,z,mag)  
**N** name card  
**E** list(rms,meanres,sdres,sdmean, sswres,ndf,fixflgs,sterrx,sterry,sterrz,sterrt,mag,stermmag)  
**filename** file name  
**PICKER** Name of Picker  
**UWFILEID** numeric ID  
**winID1** win format ID  
**comments** Vector of comments  
**OSTAS** Old station names

## References

Lees, J.M., 2004. Scattering from a fault interface in the Coso geothermal field. *Journal of Volcanology and Geothermal Research*, 130(1-2): 61-75.

## Examples

data(GH)

---

`ghstamp`*Identification stamp for RSEIS data*

---

**Description**

Prepare a character string stamp for identification of plots of signals in swig.

**Usage**

```
ghstamp(GH, sel, WIN = c(485, 600))
```

**Arguments**

GH	RSEIS list structure
sel	numeric index vector, selection of traces
WIN	time window within a trace

**Details**

The character string can be used as a stamp on plots for unique identification. Uses the info list in the RSEIS list. This function combines Zdate with the window time information.

**Value**

character array for each component in the sel vector.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

Zdate, MTM.drive, plotwlet

**Examples**

```
data(KH)
ghstamp(KH)
```

```
data(GH)
ghstamp(GH, sel=1:3)
```

---

`GLUE.GET.seis`*GLUE.GET.seis*

---

**Description**

Once a database has been mined this program re-arranges the seismograms and creates a structure used in other programs.

**Usage**`GLUE.GET.seis(GG)`**Arguments**

GG                    list of seismograms with headers

**Value**

structure of seismograms glued together

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

Mine.seis

---

`GLUEseisMAT`*GLUEseisMAT*

---

**Description**

Find duplicated stations in a matrix and fill in the traces that are continuations, return the new matrix and the vector duplicates

**Usage**`GLUEseisMAT(GFIL)`**Arguments**

GFIL                    list of data and headers, with duplicated stations glued

**Value**

New List of data and headers with same sensors/components glued together

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

Mine.seis

---

gpoly

*Convert Poles and Zeros to Polynomial*

---

**Description**

Get Polynomial from Poles and Zeros

**Usage**

gpoly(x)

**Arguments**

x                    complex vector of poles or zeros

**Value**

vector of coefficients

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
K <- PreSet.Instr()
## convert zeros to polynomial coefficients
gpoly(K[[1]]$zeros)
```

---

GreatDist	<i>Distance Along Great Circle Arc</i>
-----------	--

---

**Description**

Distance Along Great Circle Arc in degrees, kilometers

**Usage**

```
GreatDist(LON1, LAT1, LON2, LAT2, EARTH RAD= 6371)
```

**Arguments**

LON1	Longitude, point1
LAT1	Latitude, point1
LON2	Longitude, point2
LAT2	Latitude, point2
EARTH RAD	optional earth radius, default = 6371

**Value**

LIST:

d rad	distance in radians
d deg	distance in degrees
d km	distance in kilometers

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**Examples**

```
### get distance between London, England and Santiago, Chile
london <- c(51.53333, -0.08333333)
santiago <- c(-33.46667, -70.75)

GreatDist(london[2], london[1], santiago[2], santiago[1])
```

---

`grotseis`*Get seismic rotation matrix*

---

**Description**

Set up a rotation matrix for a seismic trace. Rotation matrix is 3D, although this rotation only creates a rotation for conversion to radial-transverse orientation.

**Usage**

```
grotseis(ang, flip = FALSE)
```

**Arguments**

<code>ang</code>	Angle to rotate horizontal components, degrees from North
<code>flip</code>	Logical, TRUE=flip the vertical axis, default=FALSE

**Details**

Returns a 3 by 3 matrix used for rotating a 3-component seismic record, usually stored as an N by 3 matrix.

Only the N-E components are rotated, although the vertical component can be flipped.

It is important to note the order components are introduced in the rotation matrix. Here we assume East is X (to the right), and North is Y (to the top).

For data that has (V,N,E) as (1,2,3) need to switch components (1,3,2)

For data with (V,E,N) use the normal (1,2,3)

If Back-Azimuth is used, radial is directed towards the source. If azimuth is used, radial is directed away from the source.

**Value**

3 by 3 rotation matrix.

**Note**

Positive radial is away from the source (direction of wave propagation). Positive transverse is to the right when facing the direction of wave propagation.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

`rdistaz`

**Examples**

```
#### simple case:

vecs <- rbind(c(0,0,1), c(0,1,0))
rbaz <- grotseis(21.76, flip=FALSE)
bvec <- vecs %*% rbaz

plot(c(-2,2) , c(-2,2) , asp=1, xaxs="r" , yaxs="r" , type='n' )

  arrows(0, 0, 0+bvec[,2], 0+bvec[,3],
         col=c("red", "blue"), length=.08)

  arrows(0, 0, vecs[,2], vecs[,3],
         col=c("red", "blue"), length=.08, lty=2)

text(0+bvec[1,2], 0+bvec[1,3], labels='radial', pos=3)
text(0+bvec[2,2], 0+bvec[2,3], labels='transverse', pos=4)

text(0+vecs[1,2], 0+vecs[1,3], labels='North', pos=3)
text(0+vecs[2,2], 0+vecs[2,3], labels='East', pos=4)

#### realistic case:
STAXY<-list()

STAXY$x'<-c(-2.9162198461534,-2.49599248511068,
-2.85909405321704,-1.96135073099434,
-6.50413342506259,2.64026676599765,
-3.95701139503518,-2.84082134537436,
-0.0457817300378462,-2.74214190991955)
STAXY$y'<-c(-7.83435541676815,-4.46180337254565,
-6.46036190991833,-5.01212763828746,
-2.56091416028758,
5.31173503708142,2.10545324503380,-0.87490923667824,
-0.172422188354707,-1.52055218789877)

STAXY$'lat'<-c(14.685621984127,14.7159182222222,
14.6979647030651,14.710975070028,
14.7329873333333,14.8037143111518
,14.7749104943935,14.7481391460905,
14.7544511215933,14.7423394025875)

STAXY$'lon'<-c(268.420918730159,268.424817925926,
268.421447725096,268.429783940243,268.387586722222,
268.472531954619,268.41123843527,268.421611351166,
268.447574716981,268.422528671994)

STAXY$'z'<-c(0.92522857142857,1.48225333333333,
1.14740517241379,1.4423781512605,1.51148,
2.53268681318681,2.70014678899083,2.04094444444444,
2.90827547169811,2.31817123287671)
```

```

STAXY$'cen'<-c(14.756,-91.552)

STAXY$name<-c('OBS','CAR','MAR','CAS','MTB','STA','STE','MOT','SUM','DOM')
sguitoXY<-list()
sguitoXY$x<-c(-1.78551922571555)
sguitoXY$y<-c(-1.80850340813817)
sguitoXY$'lat'<-c(14.7397535236)
sguitoXY$'lon'<-c(268.4314147874)
sguitoXY$'z'<-c(2.501)

DAZ <- rdistaz( sguitoXY$lat, sguitoXY$lon ,      STAXY$lat, STAXY$lon)

STAXY$az <- DAZ$baz

#### plotting
plot(STAXY$x, STAXY$y, asp=1, xaxs="r" , yaxs="r" )
text(STAXY$x, STAXY$y,STAXY$name, pos=3)
points(0,0, pch=3)
points(sguitoXY$x,sguitoXY$y , pch=8)
segments(sguitoXY$x, sguitoXY$y, STAXY$x, STAXY$y, col="green", lty=2)

#### be aware of the convention used: (V-N-E) or (V-E-N)
### here first vector is east, second vector is north
###      if you use the V-N-E convention
vecs <- rbind( c(0,1,0), c(0,0,1))

for( i in 1:length(STAXY$x))
{
rbaz <- grotseis(STAXY$az[i], flip=FALSE)
bvec <- vecs %*% rbaz
##### red is north, blue east
##### red is radial positive away or toward source, blue is transverse
##### blue is positive rotated to the right of red
##
arrows(STAXY$x[i],STAXY$y[i], STAXY$x[i]+bvec[,2], STAXY$y[i]+bvec[,3],
col=c("red", "blue"), length=.08)
}

```

---

hilbert

*Hilbert Transform*

---

### Description

Hilbert transform



**Usage**

```
hilbert(x)
```

**Arguments**

x                    time series vector

**Details**

Returns the hilbert transform. Used for calculating the envelope function.

**Value**

vector

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

fft, envelope

**Examples**

```
x <- rnorm(100)
y <- hilbert(x)
```

---

hilow

*Find Maxima and Minima*

---

**Description**

Search for Extrema along time series

**Usage**

```
hilow(y)
```

**Arguments**

y                    time series

**Value**

LIST:

hi                    indexes to peaks

lo                    indexes to valleys

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

peaks

**Examples**

```
ex <- seq(from=0, to=4*pi, length = 200)

y <- sin(ex)
plot(ex, y, type='l')

peakval <- hilow(y)

abline(v=ex[peakval$hi], col='green')
abline(v=ex[peakval$lo], col='red')
```

---

hodogram

*HodoGram Plot*

---

**Description**

HodoGram Plot

**Usage**

```
hodogram(nbaz, dt = dt, labs = c("Vertical", "North", "East"),
COL =rainbow(140)[1:100] , STAMP = "")
```

**Arguments**

nbaz	n by 3 matrix
dt	time sample rate
labs	labels for the components
COL	color palette
STAMP	character stamp for identification

**Value**

sx = list graphical side effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data("GH")

temp <- cbind(GH$JSTR[[1]][1168:1500], GH$JSTR[[2]][1168:1500],
             GH$JSTR[[3]][1168:1500])

pmolabs <- c("Vertical", "North", "East")

sx <- hodogram(temp, dt=GH$dt[1] ,labs=pmolabs,
              STAMP="Example", COL=rainbow(100) )
```

---

hypot

*Hypot*

---

**Description**

length of line connecting two points in a plane

**Usage**

```
hypot(x1, y1, x2, y2)
```

**Arguments**

x1	x-location point 1
y1	y-location point 1
x2	x-location point 2
y2	y-location point 2

**Details**

Euclidean distance

**Value**

numeric distance

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
hypot(34, 12, 56, 89)
```

---

idpoints.hodo                    *ID points on Hodogram*

---

### Description

Identification of points on a hodogram

### Usage

```
idpoints.hodo(nbaz, sx, X, Y)
```

### Arguments

nbaz	matrix 3 by n
sx	x vector
X	x-coordinates to id
Y	y-coordinates to id

### Details

Used in conjunction with other interactive plots.

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

### See Also

PMOT.drive

### Examples

```
data("GH")
sel<- which(GH$STNS == "CE1")

temp <- cbind(GH$JSTR[[sel[1]]][1168:1500],
  GH$JSTR[[sel[2]]][1168:1500], GH$JSTR[[sel[3]]][1168:1500])
dt <- GH$dt[ sel[1] ]
STAMP <- "GH"

PMOT.drive(temp, dt,
  pmolabs = c("Vertical", "North", "East"), STAMP = STAMP)

## ids <- idpoints.hodo(temp, sx, zloc$x[sn1], zloc$y[sn1])
```

---

`info.seis`*Information on a Seismic record*

---

**Description**

Retrieve information on a seismic record

**Usage**

```
info.seis(GH)
```

**Arguments**

GH                    RSEIS seismic record list

**Details**

Prints summary information on the traces in the seismic record

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
data(KH)
info.seis(KH)
```

---

`infoDB`*Print information about the seismic database*

---

**Description**

Print information about the seismic database

**Usage**

```
infoDB(DB, verbose=TRUE)
```

**Arguments**

DB	Database list
verbose	logical, print information to screen, default=TRUE

**Value**

list(	
usta	Unique station names
ucomp	Unique component names
start	starting date
end	ending date

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

makeDB

**Examples**

```
##### to illustrate, we make a set of individual seismograms
data(GH)
L1 = length(GH$JSTR)
DD = data.frame(GH$info)

GIVE = vector(mode='list')

for(i in 1:L1)
{
  AA = DD[i,]
  GIVE[[i]] = list(fn = AA$fn, sta =GH$STNS[i] , comp = GH$COMP[i],
                  dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
                  coords = NA, amp = GH$JSTR[[i]] )
}

##### save the seismic data in a temporary directory
#### each trace in a separate file
tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}
```

```
##### Now read files and make the DataBase:  
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)  
DB = FmakeDB(LF, kind=-1)  
IDB = infoDB(DB)
```

---

insertNAs

*Insert NA in a vector at given break points*

---

### Description

Insert NA in a vector at given break points

### Usage

```
insertNAs(v, w)
```

### Arguments

v	original vector
w	break points

### Details

Used for plotting lines that wrap around.

### Value

vector with NA inserted

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### Examples

```
a <- 1:20  
b <- insertNAs(a, c(5, 12))  
b
```

---

**INSTFREQS***Instrument Frequencies*

---

**Description**

Vector of frequencies

**Usage**

```
INSTFREQS(b, a, w)
```

**Arguments**

b	numerator, zeros
a	denominator, poles
w	frequency

**Details**
$$h = \text{jpolyval}(b,s) / \text{jpolyval}(a,s)$$
**Value**
$$h = \text{jpolyval}(b,s) / \text{jpolyval}(a,s)$$
**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
K <- PreSet.Instr()
b <- K[[1]]$zeros
a <- K[[1]]$poles
INSTFREQS(b, a, 2*pi*12)
```



---

INSTresponse	<i>Instrument Response Function</i>
--------------	-------------------------------------

---

**Description**

Extract Instrument Response from Poles and Zeros

**Usage**

```
INSTresponse(Kal, key, ff, tt = tt, plotkey = NULL)
```

**Arguments**

Kal	Calibration
key	index to list of instruments
ff	frequency vector
tt	time vector
plotkey	TRUE = plot

**Details**

response is fourier transform of delta function run through the filter

**Value**

List:

transfer	transfer function
aa	a coefficients
bb	b coefficients
resp	real part of response

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Adapted from Ken Creager's Matseis

**See Also**

deconinst

**Examples**

```
##### set list of possible instruments:
Kal <- PreSet.Instr()
### get instrument reponse for first in list:
resp1 <- INSTresponse(Kal, 1, c(0,100) , tt=c(1,0.008), plotkey=TRUE)
### plots amplitude and phase
```

---

<i>integ1</i>	<i>Integrate seismogram</i>
---------------	-----------------------------

---

**Description**

integrate under the curve of a pulse

**Usage**

```
integ1(x, y, dm = -Inf, hm = +Inf)
```

**Arguments**

x	x-axis vector
y	y-axis vector
dm	lower bound
hm	upper bound

**Value**

vector: c(osum,cista) one with the bottom triangle included one without

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

**INVRft** *Inverse Fourier Transform*

---

**Description**

Inverse Fourier Transform

**Usage**

```
INVRft(G, n, tstart, dt)
```

**Arguments**

G	Input fourier transform
n	length of time vector
tstart	time series starts at tstart
dt	Delta t, sample rate

**Details**

G is a vector spectrum evaluated at positive and negative frequencies as defined by makefreq. tstart, dt and n define the output time vector as described above.

g is the Inverse Fourier Transform of G scaled by dt. time shift theorem has been used to account for time not starting at t=0.

**Value**

g	truncate time vector to N points
f	frequencies
t	times

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

makefreq, FRWDft, INSTresponse

**Examples**

```
zil <- rnorm(300)
fss <- FRWDft( zil, length(zil), 0, 0.004)
INVRft(fss$G, length(zil), 0, 0.004)
```

---

`j2posix`*Convert RSEIS date list to Posix*

---

**Description**

Convert RSEIS date list to a compatible date/time for calculating dates and times with base R codes.

**Usage**

```
j2posix(timeinput)
```

**Arguments**

`timeinput`      RSEIS date-time list

**Details**

Code here converts to posix, but works only down to the second, i.e. fractions of a second are dropped.

**Value**

POSIX compatible date time structure.

**Note**

If you need to preserve the fractional seconds (as we do in seismology) it is recommended to cut them off and add them later.

**Author(s)**

Jonathan M. Lees<[jonathan.lees@unc.edu](mailto:jonathan.lees@unc.edu)>

**See Also**

`recdate`, `recdate1`, `dateList`, `dateStamp`, `filedatetime`, `rangedatetime`, `yeardate`, `Zdate`, `as.POSIXct`

**Examples**

```
yr = 2014
j = 233.1234
A = convertATT(j, yr)
j2posix(A)
### note fractional seconds are truncated.
```

---

jadjust.length	<i>Zero Padding</i>
----------------	---------------------

---

**Description**

Add zeros to the end of the data if necessary so that its length is a power of 2. It returns the data with zeros added if necessary and the length of the adjusted data.

**Usage**

```
jadjust.length(inputdata)
```

**Arguments**

inputdata      either a text file or an S object containing data

**Value**

Zero-padded 1D array.

**References**

See discussions in the text of "Practical Time-Frequency Analysis".

---

JBLACK	<i>Gray scale Color Palette</i>
--------	---------------------------------

---

**Description**

generate a gray scale color palette

**Usage**

```
JBLACK(n, acol=rgb(0,0,0))
```

**Arguments**

n                      number of colors to produce  
acol                    RGB color

**Details**

Creates a black color palette suitable for replacing rainbow for B/W color plots. This is inserted in case user needs to completely eliminate color from a plot that uses color palettes for fixing colors.

**Value**

n characters used for color palette

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

shade.col, rainbow, colors

**Examples**

```
pal <- JBLACK(100)
```

---

JGRAY

*Gray scale Color Palette*

---

**Description**

generate a gray scale color palette

**Usage**

```
JGRAY(n)
```

**Arguments**

n                    number of colors to produce

**Details**

Creates a grey scale color palette suitable for replacing rainbow for grey shade plots.

**Value**

n characters used for color palette

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

shade.col, rainbow, colors

**Examples**

```
pal <- JGRAY(100)
data(volcano)
image(volcano, col=pal)
```

---

`jitter.lab`*Jitter a set of labels*

---

**Description**

Jitter a set of labels so they do not overlap

**Usage**

```
jitter.lab(x, w)
```

**Arguments**

x	X-positions
w	widths of the labels

**Details**

New label positions are computed such that they do not overlap. They are shifted up or down. Works only on horizontal labels.

**Value**

vector of integer shifts.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu> Jake Anderson<ajakef@gmail.com>

**See Also**

textrect

**Examples**

```
APAL <-
c('tan2', 'red2', 'lightpink3', 'chocolate4', 'blue3', 'thistle4', 'lightcyan4',
'orangered1', 'purple4', 'darkred', 'dodgerblue1', 'gold3', 'chartreuse', 'sienna4',
'aquamarine3', 'mistyrose4', 'sienna1', 'darkkhaki', 'darkgoldenrod4', 'magenta4',
'pink3', 'orangered', 'darkslategray4', 'red3', 'goldenrod3', 'palegreen4', 'deepskyblue3',
'turquoise3', 'seagreen4', 'springgreen4', 'gold4', 'lightsalmon4', 'limegreen', 'orchid4',
'darkseagreen4', 'chartreuse3', 'goldenrod4', 'salmon2', 'deeppink3', 'forestgreen',
```

```

'lightskyblue4','mediumorchid3','deepskyblue2','chocolate2','violetred4','blue1',
'honeydew4','darkgreen','royalblue1','lightseagreen')

s <- sort(sample.int(100,25))
plot(c(1,110),c(0,8),col='white') ##### set up plot area

PplusPHASE <- c("P-up","P","Pdiff","PKP","PKiKP","PcP",
"pP","pPdiff","pPKP","pPKiKP","sP","sPdiff","sPKP","sPKiKP")
SplusPHASE <- c("S-up","S","Sdiff","SKS","sS",
"sSdiff","sSKS","pS","pSdiff","pSKS")
basic1 <- c("ScP","SKP","PKKP","SKKP","PP","PKPPKP")
basicPHASE <- c(PplusPHASE,SplusPHASE,basic1)
PHS <- basicPHASE[1:25]

x <- s
y <- rep(0, length(x))

RMAT <- RPMG::textrect(x,y, PHS, xpd=TRUE, add=FALSE, font=1, cex=.8 )

newjitz <- jitter.lab(RMAT[,1] , RMAT[,3]-RMAT[,1])
y <- y+newjitz*(RMAT[,4]-RMAT[,2])

MCOL <- length(PHS)

PASTCOL <- APAL[1:MCOL]
RMAT <- RPMG::textrect(x,y, PHS, xpd=TRUE,
add=TRUE, textcol=PASTCOL, font=1, cex=.8 )

```

---

jlegend

*plot a legend*


---

### Description

Add legend to side of figure

### Details

Rewrite of the legend function for easier manipulation.

### Value

See legend() for details on input

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>



**See Also**

legend

**Examples**

```
plot(c(0,1), c(0,1))
u <- par('usr')
LEG <- jlegend( u[1], u[4], c("Vp", "Vs"),
               lwd=2, col=c(4,3), plot=FALSE )
```

---

jpolyval

*Polynomial Value*

---

**Description**

Polynomial value

**Usage**

jpolyval(p, x)

**Arguments**

p	coefficients
x	input value

**Value**Sum of polynomial:  $p_1 + p_2 * x^1 + p_3 * x^2 \dots$ **Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**Examples**

jpolyval(c(2,3,5), 7)

---

 JSAC.seis

*JSAC.seis*


---

**Description**

Read SEGY/SAC format binary data

**Usage**

```
JSAC.seis(fnames, Iendian = 1 , HEADONLY=FALSE,
BIGLONG=FALSE, PLOT = -1, RAW=FALSE)
JSEGY.seis(fnames, Iendian = 1 , HEADONLY=FALSE,
BIGLONG=FALSE, PLOT = -1, RAW=FALSE)
```

**Arguments**

fnames	vector of file names to be extracted and converted.
Iendian	vector, Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
HEADONLY	logical, TRUE= header information only
BIGLONG	logical, TRUE=long=8 bytes
PLOT	integer, <0 no plot; 0 interactive; >0 number of seconds to sleep
RAW	logical, default=FALSE(convert to volts) , TRUE (return counts instead of volts)

**Details**

Uses readBin to extract data in SAC format. user must know what kind of machine the data was created on for I/O purposes.

For SEGY data the program is the same, although SEGY data does not have the problem of the BIGLONG so that is ignored.

For either code, a full header is returned, although the header for each format may be different.

**Value**

List containing the seismic data and header information. Each trace consists of a list with:

fn	original file name
sta	station name
comp	component
dt	delta t in seconds
DATTIM	time list
yr	year
jd	julian day
mo	month

dom	day of month
hr	hour
mi	minute
sec	sec
msec	milliseconds
dt	delta t in seconds
t1	time start of trace
t2	time end of trace
off	off-set
N	number of points in trace
units	units
amp	vector of trace values
HEAD	Full header as a data-frame of values (mixture of float and character strings)
N	Number of samples in trace
units	Units of samples, possibly: counts, volts, s, m/s, Pa, etc
IO	list: kind, Iendian, BIGLONG flags for I/O

**Note**

SAC created on PC (windows) or LINUX machines typically will be in little endian format. SAC created on a SUN will be in big endian format. If you want to swap endian-ness , choose swap.

MAC uses different convention.

Iendian can be a vector if input files have different endian-ness.

SAC inserts -12345 for no data.

There are other issues regarding the size of long.

The units are often questionable and depend on the processing. The user should be careful and check to see that the proper conversions and multipliers have been applied.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

Mine.seis, rseis2sac

**Examples**

```
##### make some SAC files, then read them in
data(GH)
apath = tempdir()
## setwd(apath)
## apath = 'TEMP'
J = rseis2sac(GH, sel =1:5, path = apath, BIGLONG =FALSE )
```

```
##### next read them in
Lname <- list.files(path=J , pattern='SAC', full.names=TRUE)

S1 <- JSAC.seis(Lname, Iendian = .Platform$endian, BIGLONG =FALSE , PLOT = -1)

#### check just the first one
i = 1
plotGH(S1[[i]])
```

---

jstats

*statistics of a vector*

---

### Description

returns relevant stats

### Usage

```
jstats(d)
```

### Arguments

d                    vector

### Details

Program calls R routines to gather important statistics for later use.

### Value

list:

mean	mean value
std	standard deviation
med	median
qdist	quartile distance
bstats	boxplot quantiles
mstats	vector of mean and std
N	number of points

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

boxplot, mean, median

**Examples**

```
x <- rnorm(100, m=43)
jstats(x)
```

---

Jtim *Decimal Julian Day*

---

**Description**

convert JD, HR, MIN SEC to Decimal Julian Day

**Usage**

```
Jtim(jj, hr = hr, mi = mi, sec = sec, yr=NULL, origyr=NULL)
JtimL(j)
```

**Arguments**

jj	Julian day
hr	Hour
mi	Minute
sec	Second
yr	year, default = NULL
origyr	default = NULL
or	
j	list of the above

**Details**

Using a NULL value for yr gives the fractional julian day in a year. If yr is a legitimate year, and the origyr is provided, then the EPOCH number of days from origyr are added onto the fractional julian day. The default for origyr is 1972 for most of seismology.

If the dates span a new year, sometimes it is useful to use the earliest year as the origyr.

**Value**

Julian day

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

secdif

**Examples**

```
Jtim( 9 , hr= 14 , mi= 53 ,sec= 16.7807606880087 )
```

```
Jtim( 9 , hr= 14 , mi= 53 ,sec= 16.7807606880087, yr=2019, origyr=1972 )
```

```
##### or,
j = list(jd=9 , hr= 14 , mi= 53 ,sec= 16.7807606880087)
```

```
JtimL(j)
```

---

 KH

*Volcano Seismic Data*


---

**Description**

Seismic data from erupting Reventador Volcano. Vertical component only.

**Usage**

```
data(KH)
```

**Format**

```
KH = list( LOC=list(yr=0, jd=0, mo=0, dom=0, hr=0, mi=0, sec=0, lat=0, lon=0, z=0, mag=0,
gap=0, delta=0 , rms=0, hozerr=0), MC=list(az1=0, dip1=0, az2=0, dip2=0, dir=0, rake1=0, di-
paz1=0, rake2=0, dipaz2=0, F=list(az=0, dip=0), G=list(az=0, dip=0), U=list(az=0, dip=0), V=list(az=0,
dip=0), P=list(az=0, dip=0), T=list(az=0, dip=0),sense=0,M=list( az1=0, d1=0, az2=0, d2=0, uaz=0,
ud=0, vaz=0, vd=0, paz=0, pd =0, taz=0, td=0), UP=TRUE, icol=1, ileg="", fcol='red', CN-
VRG="", LIM =c(0,0,0,0) ),
```

```
STAS=list(tag="", name="", comp="", c3="", phase="", sec=0, err=0, pol="", flg=0 , res=0),
```

```
LIP=vector(length=6),
```

```
H=list(yr=0,mo=0,dom=0,hr=0,mi=0,sec=0,lat=0,lon=0,z=0,mag=0),
```

```
N=list(name=""),
```

```
E=list(rms=0,meanres=0,sdres=0,sdmean=0,sswres=0,ndf=0,fixflgs=0, sterrx=0,sterry=0,sterz=0,stertr=0,mag=0,stermag=
filename="",
```

```
PICKER="", UWFILEID="",winID1="",comments="", OSTAS="")
```

## References

Lees, J. M., J. B. Johnson, M. Ruiz, L. Troncoso, M. Welsh, Reventador Volcano 2005: Eruptive Activity Inferred from Seismo-Acoustic Observation *Journal of Volcanology and Geothermal Research* in Press, 2007.

## Examples

```
data(KH)
##### set SHOWONLY=FALSE for interactive
swig(KH, SHOWONLY=0)
```

---

lagplot	<i>Plot phase lags</i>
---------	------------------------

---

## Description

Shift a times series by a specified phase lag.

## Usage

```
lagplot(y1, dt, lag, PLOT = FALSE)
```

## Arguments

y1	seismic signal
dt	DeltaT, s
lag	lag, s
PLOT	logical, TRUE=plot

## Value

Graphical Side Effects.

## Author(s)

Jonathan M. Lees<jonathan.lees.edu>

## See Also

getphaselag2

**Examples**

```
data(KH)

ts1 = KH$JSTR[[1]]

lagplot(ts1, KH$dt[1], 300, PLOT=TRUE )
```

---

leests

*Time Series Structure*

---

**Description**

return time series structure

**Usage**

```
leests(a, dt = 0.008)
```

**Arguments**

a	vector signal
dt	sample rate

**Value**

```
list(y=y, dt=dt)
```

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
x <- rnorm(10)

leests(x, dt = 0.01)
```



---

legitpix	<i>Legitimate picks in swig</i>
----------	---------------------------------

---

**Description**

Legitimate picks in swig (used internally)

**Usage**

```
legitpix(sel, zloc, zenclick)
```

**Arguments**

sel	seleceted traces in swig
zloc	location list
zenclick	number of legitimate picks

**Value**

list: ypick, ppick

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

---

letter.it	<i>Add letters to the corners of plots in multiple figures</i>
-----------	--

---

**Description**

Add letters to the corners of plots in multiple figures

**Usage**

```
letter.it(a, corn = 1)
```

**Arguments**

a	character letter for marking figure
corn	corner to put letter in

**Details**

Can use uppercase or lower case letters, or roman numerals.

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
par(mfrow=c(2,2))
for(i in 1:4)
{
  x <- 1:10
  y <- rnorm(10)
  plot(x,y)
  letter.it(letters[i], 2)
}
```

---

LocalUnwrap

*Unwrap spectrum phase*

---

**Description**

unwrap the phase spectrum so it does not wrap around

**Usage**

```
LocalUnwrap(p, cutoff = cutoff)
```

**Arguments**

p	phase spectrum
cutoff	cut off angle = pi

**Value**

Unwrapped spectrum

**Note**

Algorithm minimizes the incremental phase variation by constraining it to the range  $[-\pi, \pi]$

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
x <- 1:512
amp <- sin(1*2*pi*x/16) + sin(2*2*pi*x/16) + sin(3*2*pi*x/16)

spc <- fft(amp)

plot(Mod(spc), type='l')

angle <- Arg(spc)

plot(angle, type='l')

unang <- LocalUnwrap(angle, cutoff =pi )
plot(unang, type='l')
```

---

logspace

*Logarithm*

---

**Description**

Logarithmically spaced vector

**Usage**

```
logspace(d1, d2, n = n)
```

**Arguments**

d1	lower frequency
d2	upper frequency
n	number of frequencies

**Details**

generates a row vector of n logarithmically equally spaced points between decades  $10^{X1}$  and  $10^{X2}$

**Value**

vector

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
f <- logspace(1, 25)
```

---

longfft

*Long FFT Spectrogram*


---

**Description**

Creates hourly spectrograms, either alternating seismic and infrasound data or sequences of one component.

**Usage**

```
longfft(DB, DAYS = c(233, 234), HRS = 1:24, sta = "KR1", comp = c("V",
"I"), NPP = 6, CSCALE = FALSE, pal = rainbow(100), PS = FALSE, kind = 1,
Iendian = 1, BIGLONG = FALSE)
```

```
longreset(NPP, PS)
```

```
longpstart(NPP = 6, asta = "", acomp = "", theday = 1, hr = 0)
```

**Arguments**

DB	RSEIS Data base
DAYS	vector of Days to display
HRS	vector of hours to display
sta	stations to extract
comp	component to extract
NPP	Number of plot strips per page, default = 6
CSCALE	scaling
pal	palettes to use (given two will alternate these)
PS	logical, TRUE postscript output
kind	data type, an integer -1, 0, 1, 2 ; 0=R(DAT) , -1=RDS, 0=RDATA, 1 = segy, 2 = sac
Iendian	Endian-ness of binary data
BIGLONG	logical, TRUE=long is 8 bytes
asta	character, one station
acomp	character, one component
theday	one day
hr	one hour

**Details**

Extracts data from the DB data base and plots strips of spectrograms for perusal.

longpstart, longreset are auxiliary codes used to set up the postscript files and initialize the plotting.

**Value**

Graphical Side effects

**Note**

Program is set for data being ready from external sources in binary (SAC, SEGY) format. If data is in R-format already, the code may not work.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

SPECT.drive

**Examples**

```
if(interactive()){
  ##### get a time series
  data(KH)

  amp = KH$JSTR[[1]]
  OLDdt = KH$dt[1]
  ##### downsample to:
  newdt = 0.1

  JK = FAKEDATA(amp, OLDdt=OLDdt, newdt = 0.1, yr = 2000,
               JD = 4, mi = 12, sec = 0, Ntraces = 24,
               seed=200, noise.est=c(1, 100) , verbose=TRUE )

  tdir = tempdir()
  for(i in 1:length(JK) )
  {
    sig = JK[[i]]
    d1 = dateStamp(sig$DATTIM, sep='_')
    nam1 = paste(d1,sig$sta, sig$comp, sep='_')
    nam2 = paste0(nam1, '.RDS')
    nam3 = paste(tdir, nam2, sep='/')
    saveRDS(file=nam3, sig)
  }

  LF = list.files(path=tdir,pattern='.RDS', full.names=TRUE)
  DB = FmakeDB(LF, kind=-1)
  IDB = infoDB(DB)
```

```

p1 <- RPMG::Gcols(plow=5, phi=0, N=100, pal="topo.colors", mingray=0.8)
p2 <- RPMG::Gcols(plow=5, phi=0, N=100, pal="rainbow", mingray=0.8)

longfft(DB, DAYS=5 , HRS=1:24 ,
  sta=IDB$usta, comp=IDB$ucomp , NPP=6 , CSCALE=FALSE,
  pal = list(p1=p1, p2=p2), PS=FALSE , kind = -1,
  Iendian=1, BIGLONG=TRUE )

}

```

---

makeDB

---

*Create a seismic Waveform Database*


---

### Description

Create a seismic Waveform Database

### Usage

```

makeDB(path=".", pattern="R", dirs="", kind = 1,
  Iendian=1, BIGLONG=FALSE)
FmakeDB(LF2, kind =1, Iendian=1, BIGLONG=FALSE)

```

### Arguments

path	character, Path to directory where files and directories exist
pattern	character, pattern for listing of files
dirs	character, vector of directories to be scanned
kind	kind of data: RDS=-1, R(DAT)=0, segy=1; sac=2
Iendian	default=1, Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE means long=8 bytes
LF2	list of files

### Details

The files are typically located in a directory structure created by programs like ref2segy, a PASS-CAL program for downloading data in the field. Each file contains one seismogram, with a header. makeDB reads in all the headers and creates a list of meta-data for later use in RSEIS.

"kind" can be numeric or character: options are 'RDS', 'RDATA', 'SEGY', 'SAC', corresponding to (-1, 0, 1, 2).

Uses readBin to extract data in SAC format. user must know what kind of machine the data was created on for I/O purposes.

If data was created on a little endian machine but is being read on big endian machine, need to call the endian "swap" for swapping.

If data was created on a machine with LONG=4 bytes, be sure to call the program with BIG-LONG=FALSE.

If the base directory, or the subdirectories, contain files that are not seismic data then care must be taken. Perhaps use FmakeDB to explicitly names the files for the DataBase.

If using FmakeDB a simple vector of files (full path names) should be provided.

The origin year, used for getting the Epoch year, is stored as attribute origyr.

### Value

list:

fn	file name
yr	year
jd	julian day
hr	hour
mi	minute
sec	second
dur	duration, seconds
t1	time 1 in Epoch days
t2	time 2 in Epoch days
sta	station name
comp	component name
dt	sample rate, seconds

### Note

Epoch times are used to accomodate problems where julian days cross year end boundaries, so that day 366 comes before day 1 of the next year.

The origyr, kind , Iendian, BIGLONG are stored as attributes in the Database.

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

setupDB, Mine.seis , getseis24, plotseis24, EPOCHday, swig

**Examples**

```
##### to illustrate, we make a set of individual seismograms
data(GH)
L1 = length(GH$JSTR)
DD = data.frame(GH$info)

GIVE = vector(mode='list')

for(i in 1:L1)
{
AA = DD[i,]
GIVE[[i]] = list(fn = AA$fn, sta = GH$STNS[i] , comp = GH$COMP[i],
                dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
                coords = NA, amp = GH$JSTR[[i]] )
}

##### save the seismic data in a temporary directory
#### each trace in a separate file
tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir,pattern='.RDS', full.names=TRUE)
DB = FmakeDB(LF, kind=-1)
```

---

**makefreq***Make Frequency*

---

**Description**

Create a frequency value for integration and differentiation

**Usage**

```
makefreq(n, dt)
```

**Arguments**

n	number of freqs
dt	deltat



**Value**

vector of frequencies

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

INVRft

**Examples**

```
N <- 256
dt <- 0.008
f <- makefreq(N,dt)
```

---

markseis24

*Mark 24 hour seismic display*

---

**Description**

Mark a 24 hour seismic display

**Usage**

```
markseis24(pjj, pix = list(yr = 2009, jd = 1, hr = 0, mi = 0, sec = 0,
dur = 0), col = "red", LEGON = 3, BARON = TRUE, ARROWS = TRUE, lwd=1)
```

**Arguments**

pjj	Output information from plotseis24 (x,y, yr, jd)
pix	list: date list consisting of: yr, jd, hr, mi, sec, dur)
col	Color, specified as color index, character string or rgb
LEGON	plotting flag for legs: 0=no legs, 1=left leg, 2=right leg, 3=both legs(def ault)
BARON	logical:plotting flag for bar
ARROWS	logical: plot arrows FALSE=no arrows
lwd	numeric, graphical parameter, line width

**Details**

the LEGON parameter controls the small marks at the ends: Either left(1) right(2) both(3) or no legs(0) are plotted. window bars should wrap around the ends of the hour to the next hour below. The durations of the windows are supplied in seconds. If no duration is supplied, it is set to 0. If one duration is supplied it is copied to all other windows.

**Value**

Graphical Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

winmark, getseis24, plotseis24

**Examples**

```

data(KH)

amp = KH$JSTR[[1]]
OLDdt = KH$dt[1]
newdt = 0.1
yr = 2000
GIVE = FAKEDATA(amp, OLDdt=0.01, newdt = 0.1, yr = 2000,
                JD = 4, mi = 12, sec = 0, Ntraces = 24*3,
                seed=200, noise.est=c(1, 100) , verbose=TRUE )

tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)

DB = FmakeDB(LF, kind=-1)

IDB = infoDB(DB)

START = list(yr =yr , jd= 5 , hr= 0 , mi= 0 ,sec= 0)

END = list(yr =yr , jd= 7 , hr= 0 , mi= 0 ,sec= 0)

h = getseis24(DB, iyear = 2000, iday = 5, usta = IDB$usta,
              acomp = IDB$ucomp, kind = -1, Iendian=1, BIGLONG=FALSE)

pjj <- plotseis24(h, dy=1/18, FIX=24, SCALE=1,

```

```

    FILT=list(ON=FALSE, fl=0.05 , fh=20.0, type="BP", proto="BU"),
    RCOLS=c(rgb(0.2, .2, 1), rgb(.2, .2, .2)) )

### set up pix
WINS2 <- list(hr = c(12.5, 12.7) )

Apix <- WINS2$hr[seq(from=1, to=length(WINS2$hr), by=2) ]
dur <- (WINS2$hr[seq(from=2, to=length(WINS2$hr), by=2) ]-Apix)*3600

## dur <- rep(0, times=length(Apix))

## mark the 24 hour plot

pix =list(yr=rep(pjj$yr, length(Apix)),
  jd=rep(pjj$jd, length(Apix)) , hr=Apix, mi=rep(0, length(Apix)),
  sec=rep(0, length(Apix)), dur=dur)

markseis24(pjj, pix=pix, col='red', ARROWS=TRUE )

```

---

matsquiggle

*Matrix Seismic Record*


---

## Description

Plot a matrix of time series as a var-squiggle display (filled in half traces)

## Usage

```

matsquiggle(XMAT, dt1, dist = NULL, thick = 1,
  FLIP = FALSE, filcol='blue', tracecol="black", add=FALSE, PLOT=TRUE, xpd=TRUE, plotdir=1 )

```

## Arguments

XMAT	matrix of traces
dt1	sample interval, s
dist	distance for each trace in the matrix
thick	thickness for each trace to be plotted
FLIP	logical, FALSE (default) plot horizontal, TRUE=plot vertical
filcol	color for shading
tracecol	color for trace
add	add traces to existing plot
PLOT	whether to create a new plotting region
xpd	logical, set xpd parameter (see par)
plotdir	1=left to right, 0=right to left (default=1)

**Details**

see varsquiggle for more details

**Value**

side effects.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

varsquiggle, varsquig

**Examples**

```

data(GH)
m <- match( GH$STNS,    GH$stafilename)
LATS <- GH$stafile$lat[m]
LONS <- GH$stafile$lon[m]
dees <- rdistaz( GH$pickfile$LOC$lat, GH$pickfile$LOC$lon, LATS, LONS)

sel <- which(GH$COMPS=="V")
sel <- sel[order(dees$dist[sel])]

### plot normal way:
### swig(GH, sel=sel, WIN=c(5,10), SHOWONLY=TRUE)

### plot with varsquiggle
### varsquiggle(GH, sel=sel, WIN=c(5,10))

ex <- seq(from=0, by=GH$dt[sel[1]], length=length(GH$JSTR[[sel[1]]]))
wx <- ex>=5 & ex<=10
XMAT <- matrix(ncol=length(sel), nrow=length(which(wx)))

for(i in 1:length(sel))
{
  XMAT[,i] <- GH$JSTR[[sel[i]]][wx]
}

matsquiggle(XMAT, GH$dt[sel[1]] , dist = dees$dist[sel] , thick = 1,
  FLIP = FALSE)

axis(1)
axis(2)
title(xlab="Time, s", ylab="Distance, km")

```

---

 Mine.seis

---

*Mine a seismic data base to extract sections of time limited data*


---

### Description

Mine a seismic data base to extract sections of time limited data

### Usage

```
Mine.seis(at1, at2, DB, grepsta, grepcomp, kind = 1, Iendian=1,
BIGLONG=FALSE, CHOP=TRUE, verbose=FALSE, chtoken=NULL, statoken=NULL, RAW=FALSE)
```

### Arguments

at1	time 1 in julian days
at2	time 2 in julian days
DB	data base structure to search through that provides the files where data is to extracted from
grepsta	which stations to extract
grepcomp	which components to extract
kind	kind of data, -1="RDS", 0="RDATA", 1 = "segy", 2 = "sac"
Iendian	Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE=long=8 bytes
CHOP	cut the data to a window using CHOP.SEISN
verbose	print out intermediate information for debugging
chtoken	channel token for selecting channels (NULL)
statoken	station token for selecting stations (NULL)
RAW	logical, default=FALSE(convert to volts) , TRUE (return counts instead of volts)

### Details

The data base is a list or dataframe containing the files names, the beginning time (t1) and ending time (t2) for each file in the data base. Mine.seis uses grep on the file names to extract specific files from the DB list.

Mine.seis needs to know what format the data was created in: little/big endian and the size of the LONG.

If data was created on a little endian machine but is being read on big endian machine, need to call the endian "swap" for swapping.

If data was created on a machine with LONG=4 bytes, be sure to call the program with BIGLONG=FALSE.

Use sysinfo to findout the system parameters for the local system. You need to know, however, what machine the binary files were created on.

In some situation the chanel name and the station name are not embedded in the file headers - in that case use the token from the file name.

**Value**

List of seismograms cut from the database

**Note**

The headers in the digital (segy or SAC) data files may not necessarily match the file names. Note that program JGET.seis extracts the station name and component name from the digital header and does not use the file name. It may be prudent to force the file names and header files to match prior to using Mine.seis. For SEGY files, in LINUX-UNIX, use: rename, segymod (PASSCAL) to modify the headers.

For SAC files, use sac software.

For R-based codes save the files in a format that has the relevant information (DAT format).

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

makeDB, GLUEseisMAT, JGET.seis, JSAC.seis, JSEGY.seis, sysinfo

**Examples**

```
data(GH)

DD = data.frame(GH$info)

#### get only vertical traces
WV = which( GH$COMPS=='V' )
L1 = length(WV)

GIVE = vector(mode='list')

for(j in 1:L1 )
{
  i = WV[j]
  AA = DD[i,]
  GIVE[[j]] = list(fn = AA$fn, sta =GH$STNS[i] , comp = GH$COMP[i],
                 dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
                 coords = NA, amp = GH$JSTR[[i]] )
}
#### par(mfrow=c(length(GIVE) , 1) )
# for(i in 1:length(GIVE) ) { plotGH(GIVE[[i]]) }
tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
```

```

        nam3 = paste(tdir, nam2, sep='/')
        saveRDS(file=nam3, sig)
    }
##### Now read files and make the DataBase:
LF = list.files(path=tdir,pattern='.RDS', full.names=TRUE)
DB = FmakeDB(LF, kind=-1)
IDB = infoDB(DB)

SAMPseis <- Mine.seis(IDB$at1, IDB$at2, DB, IDB$usta[1:3], IDB$ucomp[1], kind = -1 )

w <- swig(SAMPseis, SHOWONLY=0)

```

---

mirror.matrix

*mirror matrix*

---

## Description

mirrored representation of image matrix

## Usage

```
mirror.matrix(x)
```

## Arguments

x                    matrix

## Details

Used for flipping the output of the wavelet transform for more convenient plotting.

## Value

matrix

## Author(s)

Jonathan M. Lees<jonathan.lees.edu>

## References

Rwave, plotwlet, wlet.do, wlet.drive

**Examples**

```
xy <- matrix(rnorm(100), ncol=10)
mirror.matrix(xy)
```

---

Mmorlet

*Morlet Wavelet*

---

**Description**

Make Morlet Wavelet

**Usage**

```
Mmorlet(UB = -4, LB = 4, N = 256, plot = FALSE)
```

**Arguments**

UB	upper bound
LB	lower bound
N	number of points
plot	logical, TRUE=plot

**Details**

create a morlet function based on the matlab style routines

**Value**

time series list:

xval	x-output
mor1	y-output

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

scal2freqs, Rwave

**Examples**

```
mm <- Mmorlet(-8, 8, 256)
```



---

mtapspec	<i>MTM spectrum</i>
----------	---------------------

---

**Description**

Multi-tape Method Spectrum

**Usage**

```
mtapspec(a, dt, klen = length(a), MTP = NULL)
```

**Arguments**

a	vector time series
dt	sample rate
klen	length of fft
MTP	MTM parameters, list: <b>kind</b> kind of taper average <b>nwin</b> number of windows <b>npi</b> number of Pi-prolate functions <b>inorm</b> normalization flag

**Details**

MTP represent parameters that control the multi-tape pi-prolate functions used by mtapspec. See reference for details.

**Value**

LIST	
dat	input data
dt	sample rate
spec	Estimated power spectrum
dof	degrees of freedom for each frequency
Fv	F-values for each frequency
Rspec	real part of complex spectrum
Ispec	imaginary part of complex spectrum
freq	frequencies
df	delta frequency
numfreqs	number of frequencies
klen	length used in fft
mtm	input MTM parameters, see above

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

fft

**Examples**

```
data(CE1)
a <- list(y=CE1$y[CE1$x>5.443754 & CE1$x<5.615951], dt=CE1$dt)

Mspec <- mtapspec(a$y,a$dt, klen=4096,
                  MTP=list(kind=2,nwin=5, npi=3,inorm=0) )
```

---

MTM.drive

*Interactive MTM driver*

---

**Description**

MTM analysis of signals

**Usage**

```
MTM.drive(a, f1 = f1, f2 = f2, len2 = 1024, COL = 2, PLOT = FALSE,
          PADDLAB = NULL, GUI = TRUE)
```

**Arguments**

a	list(y=time series amp, dt=delta-ts, stamps=text stamps)
f1	low frequency
f2	high frequency
len2	power of two length
COL	colors
PLOT	logical PLOT=TRUE
PADDLAB	vector of buttons
GUI	Whether to be in GUI (interactive) mode

**Value**

Graphical Side effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

plt.MTM0

**Examples**

```
data("GH")
sel <- which(GH$COMPS=="V")

amp <- list()
dees <- list()
stamps <- list()

for( i in 1:3)
{
amp[[i]] <- GH$JSTR[[sel[i]]]
dees[i] <- GH$dt[sel[i]]
stamps[i] <- paste(GH$STNS[sel[i]], GH$COMPS[sel[i]])
}

a <- list(y=amp, dt=dees, stamps=stamps)

f1 <- 0.1
f2 <- floor(0.33*(1/a$dt[[1]]))

speccol <- c('red', 'blue', 'purple')

MTM.drive(a, f1, f2, COL=speccol, PLOT=TRUE)
```

---

MTMdisp

*MTMdisp*

---

**Description**

Display MTM displacement spectrum.

**Usage**

```
MTMdisp(a, f1 = f1, f2 = f2, len2 = 1024, PLOT = FALSE)
```

**Arguments**

a	seismic velocity trace, as a ts structure (list(y=trace, dt=sample rate))
f1	low frequency
f2	high frequency
len2	length of fft
PLOT	logical, TRUE=plot

**Details**

Uses Multi-taper estimate of spectrum and divides the spectrum by  $1/(2*\pi*f)$  to get integration of velocity seismogram.

**Value**

Returns displacement spectrum. Graphical Side effect.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

mtapspec

**Examples**

```
data(CE1)
xvel <- list(y=CE1$y[CE1$x>5.443754 & CE1$x<5.615951], dt=CE1$dt)

len2 <- next2(length(xvel$y))
Spec <- MTMdisp(xvel, f1=.01, f2=25, len2=len2, PLOT=FALSE )
```

---

 MTMgabor

*Evolutionary MTM Spectrum*


---

**Description**

Time varying Auto-Regressive Spectrum (Gabor Transform) using MTM

**Usage**

```
MTMgabor(a, dt = 0, ppoint=95 , numf = 1024, Ns = 0, Nov = 0, fl = 0, fh = 10)
```

**Arguments**

a	signal
dt	sample rate interval (s)
ppoint	percent confidence for F-test (default=95)
numf	Number of frequencies
Ns	Number of sample in sub-window
Nov	Number of sample to overlap
fl	low frequency to display
fh	high frequency to display

**Details**

This is a spectrogram function similar to the Gabor Transform but uses the MTM (multi-taper method) for spectrum estimation. This is a non-interactive version of MTM.drive.

**Value**

List	
sig	input signal
dt	deltat
numfreqs	Number of frequencies output
wpars	input parameters list(Nfft=numfreqs, Ns=Ns, Nov=Nov, fl=fl, fh=fh)
DSPEC	spectrum image
HIMAT	matrix with high values of F-test at 90 percent confidence
DOFMAT	Matrix image of degrees of freedom
FVMAT	Matrix image of F-test values
kdof	test degrees of freedom=2*nwin-2
ppoint	percentage point for confidence bounds
freqs	output frequencies (y axis)
tims	output times (x-axis)

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Percival and Walden;

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

Percival, Donald B., Walden, Andrew T. (1993): Spectral Analysis for Physical Applications, Cambridge University Press, Cambridge, 583p.

**See Also**

evolfft, evolMTM, MTM.drive, GETARAIC, doGABOR.AR, DOsgram, doGABOR.MTM

**Examples**

```
data(KH)
###  swig(KH)

Xamp <- KH$JSTR[[1]]
Nfft <- 1024  ###  fft length
Ns <- 512    ###  number of samples in a window
Nov <- 480   ###  number of samples of overlap per window
fl <- 0      ###  low frequency to return
fh <- 12     ###  high frequency to return
dt <- KH$dt[1]

#### shorten the signal here, just for speed on the example:
sig = Xamp[37501:75001]

EV <-  MTMgabor(sig, dt = dt, numf =Nfft , Ns = Ns, Nov = Nov, fl = fl, fh= fh)

PE  <-  plotevol(EV, log=1, fl=0.01, fh=fh, col=rainbow(100),
                ygrid=FALSE, STAMP="", STYLE="ar")
```

---

MTMplot

*Plot Multi-taper Spectrum*

---

**Description**

Plots output of MTM spectrum

**Usage**

```
MTMplot(a, f1 = f1, f2 = f2, len2 = 1024, PLOT = FALSE, PADDLAB = NULL, GUI = TRUE)
```

**Arguments**

a	signal
f1	lower frequency
f2	upper frequency
len2	number of points in spectrum
PLOT	logical, TRUE=plot
PADDLAB	Labels for buttons
GUI	use a GUI to display for other interactions

**Details**

Uses Lees' MTM code.

**Value**

```
list(len2=len2, f=f, f1=f1, f2=f2, displ=displ, ampsp=amp, flag=flag )
```

len2	next power of 2 for fft calculation
f	frequencies
f1	lower freq
f2	upper freq
displ	kind of display
ampsp	amplitude spectrum
flag	

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

MTM.drive, MTMdisp, plt.MTM0

NEW.getUWSTAS            *get UW station file*

---

**Description**

Match Picks with stations and return station structure

**Usage**

NEW.getUWSTAS(PICS)

**Arguments**

PICS                      Picks in pickfile

**Details**

matches Picks with stations

**Value**

STAS structure

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

NEWPLOT.WPX            *Plot Window Picks (WPX)*

---

**Description**

adds picks to existing seismic section

**Usage**

NEWPLOT.WPX(t0, STNS, COMPS, YPX, FILL = FALSE, FORCE = TRUE, cex = cex, srt = srt)



**Arguments**

t0	starting time for window
STNS	stations to match
COMPS	components to match
YPX	list of picks
FILL	fill color
FORCE	logical, TRUE=plot picks on all traces
cex	character expansion
srt	string rotation angle

**Details**

Used in conjunction with swig program

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig

**Examples**

```
##### no example available now
```

---

next2

*Next Power of Two*

---

**Description**

Return next power of two greater than n

**Usage**

```
next2(x)
```

**Arguments**

x	length of vector
---	------------------

**Value**

integer value

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
k <- 1236
next2(k)
```

---

OH

*Delta-O18 isotope record*

---

**Description**

Data from Delta-O18 Isotope record of climate change. Periodicities of this data show the Milankovic cycles.

**Usage**

data(OH)

**Format**

```
OH = list( LOC=list(yr=0, jd=0, mo=0, dom=0, hr=0, mi=0, sec=0, lat=0, lon=0, z=0, mag=0,
gap=0, delta=0, rms=0, hozerr=0), MC=list(az1=0, dip1=0, az2=0, dip2=0, dir=0, rake1=0, dipaz1=0, rake2=0, dipaz2=0, F=list(az=0, dip=0), G=list(az=0, dip=0), U=list(az=0, dip=0), V=list(az=0, dip=0), P=list(az=0, dip=0), T=list(az=0, dip=0), sense=0, M=list( az1=0, d1=0, az2=0, d2=0, uaz=0, ud=0, vaz=0, vd=0, paz=0, pd =0, taz=0, td=0), UP=TRUE, icol=1, ileg="", fcol='red', CN-VRG="", LIM =c(0,0,0,0) ),
```

```
STAS=list(tag="", name="", comp="", c3="", phase="", sec=0, err=0, pol="", flg=0, res=0),
```

```
LIP=vector(length=6),
```

```
H=list(yr=0, mo=0, dom=0, hr=0, mi=0, sec=0, lat=0, lon=0, z=0, mag=0),
```

```
N=list(name=""),
```

```
E=list(rms=0, meanres=0, sdres=0, sdmean=0, sswres=0, ndf=0, fixflgs=0, sterrx=0, sterry=0, sterrz=0, sterrt=0, mag=0, sterrmag=0, filename=""),
```

```
PICKER="", UWFILEID="", winID1="", comments="", OSTAS="")
```

**Note**

The sample unit here is set to 0.3 which is 10000 times the correct sample rat.

**References**

Lees, J. M. and J. Park (1995): Multiple-taper spectral analysis: A stand-alone C-subroutine: Computers & Geology: 21, 199-236.

**Examples**

```
data(OH)
xx <- swig( OH, sel=which(OH$COMPS == "V"), SHOWONLY=0)
```

---

one	<i>one plotting region</i>
-----	----------------------------

---

**Description**

change from multiple R-screens to one

**Usage**

```
one()
```

**Examples**

```
par(mfrow=c(2,1))
plot(rnorm(10), rnorm(10) )
plot(rnorm(10), rnorm(10) )

one()
plot(rnorm(10), rnorm(10) )
```

---

P2GH	<i>XTR button to RSEIS</i>
------	----------------------------

---

**Description**

Convert output of XTR button to RSEIS list.

**Usage**

```
P2GH(P1)
```

**Arguments**

P1                    Output of swig after clicking XTR

**Details**

Running swig out after a selection of a window and the XTR button, one can create an RSEIS structure for further use in swig.

**Value**

RSEIS list

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig, prepSEIS

**Examples**

```
if(interactive()){
  data(GH)

  ##### click twice and select the XTR button
  P1 <- swig(GH)

  LH <- P2GH(P1)
  L1 <- swig(LH)

}
```

---

parse.pde

*Parse PDE file*

---

**Description**

Parse and Extact information from a screen dump of PDE (preliminary earthquake estimates) from the internet,

**Usage**

parse.pde(card)

**Arguments**

card                    character, one line from the PDE file

**Details**

Parsing is done by column specification. Uses screen dump format. see <http://neic.usgs.gov/neis/epic/epic.html>

**Value**

Time, Location and Magnitude: list(yr, jd, mo, dom, hr, mi, sec, lat, lon, depth, z, mag)

**Note**

May try using the CSV version of the dump.

**Author(s)**

Jonathan M. Lees<[jonathan.lees@unc.edu](mailto:jonathan.lees@unc.edu)>

**References**

<http://neic.usgs.gov/neis/epic/epic.html>

**See Also**

getPDEcsv, getPDEscreen

**Examples**

##### copy/paste from the screen dump at the NEIC web site

```
K = c(
' PDE-Q 2008 12 31 053408.80 40.11 -77.00 1 2.4 LgGS ... .. ',
' PDE-Q 2008 12 31 084757.50 46.75 154.41 14 4.9 mbGS ... .. ')
```

```
G = parse.pde(K[1])
```

---

parseFN2STA

*get station from file name*

---

**Description**

station and component are assumed to be the last elements of a file name - this function returns a list with these text strings.

**Usage**

```
parseFN2STA(fn, ista, icomp, sep="\\. ", dir=0 )
```

**Arguments**

fn	text file name
ista	index of station name counting from the end of the file name
icomp	index of station name counting from the end of the file name
sep	separator token in file name
dir	integer, default=0, direction for counting. see details

**Details**

Some seismic data formats store the station in the file name rather than the seismic header. The default (dir=0) assumes that the station name and the component name are the last items on the file name separated by a period. So ista and icomp are computed from the end of the file name, i.e. ista=1 and icomp=0. If (dir=1) the counting is from the beginning of the string and the count starts at 1. Remember to count double tokens, they return a blank.

**Value**

list(sta='text station name', comp='compname')

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
parseFN2STA('/data/wadati/bourbon/GUATEMALA/SEGY/R009.01/07.009.22.25.34.CAS.E')
fn <- "2011-11-06-0637-21S.SI01__003_SI01__SH_N_SAC"

parseFN2STA(fn, 4, 1, sep="_" )
### or:
parseFN2STA(fn, 4, 7, sep="_", dir=1 )
```

---

partmotnet

*Particle Motion on Stereonet*

---

**Description**

Show Particle Motion on Stereonet

**Usage**

```
partmotnet(temp, LINES = FALSE, STAMP = STAMP, COL = rainbow(100))
```

**Arguments**

temp	matrix of 3-component seismic data
LINES	logical, TRUE=draw lines
STAMP	identification stamp
COL	color palette

**Details**

Show seismic particle motion on a sphere color coded by time.

**Value**

graphical side effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data("GH")

temp = list(x=GH$JSTR[[1]][1168:1500],
           y=GH$JSTR[[2]][1168:1500], z=GH$JSTR[[3]][1168:1500])

sx = partmotnet(temp, STAMP="Example",
               LINES=TRUE, COL=rainbow(100) )
```

---

PDE2list

*Convert PDEs to List*

---

**Description**

Convert a list of individual PDE events to a list of lat, lon, z...etc

**Usage**

```
PDE2list(PDF)
```

**Arguments**

PDF	list of individual events
-----	---------------------------

**Details**

uses getmem

**Value**

list

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

getmem, getPDEcsv, parse.pde, getPDEscreen

---

peaks

*Peaks*

---

**Description**

Find peak amplitudes in a time series signal.

**Usage**

```
peaks(series, span = 3, do.pad = TRUE)
```

**Arguments**

series	signal
span	span for window
do.pad	padding

**Details**

This function originated in a note from Brian Ripley.

**Value**

vector of peak indexes

**Author(s)**

Brian Ripley



**Examples**

```
data(CE1)
plot(CE1$x, CE1$y, type='l')

pp <- seq(from=53, to=80, by=1)

plot(CE1$x[pp], CE1$y[pp], type='l')

aa <- peaks(CE1$y[pp], span=3)

abline(v=CE1$x[pp[aa]], col='red')
```

---

PICK.DOC

*Documentation for swig*

---

**Description**

Prints brief documentation for buttons in swig

**Usage**

```
PICK.DOC(w)
```

**Arguments**

w                    vector of buttons needed

**Details**

Buttons are defined in advance

**Value**

printed side effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig

**Examples**

```
if(interactive() ) PICK.DOC(6:23)
```

---

pickgeninfo	<i>print swig information</i>
-------------	-------------------------------

---

**Description**

print swig information to screen

**Usage**

```
pickgeninfo()
```

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

**Examples**

```
pickgeninfo()
```

---

pickhandler	<i>Handle Pick in RSEIS</i>
-------------	-----------------------------

---

**Description**

Update the WPX (pick data frame) list with a new pick.

**Usage**

```
pickhandler(i1 = 1, ppick = 0, kzap = "Y", err = NA, res=0, ycol =  
rgb(0, 0, 1), pol=0, flg=0, onoff=1, NPX = 1, WPX = WPX, NH)
```

**Arguments**

i1	Index of trace
ppick	time for pick in seconds
kzap	character label of pick
err	error for pick
res	residual(or duration)
ycol	color for pick
pol	polarity of pick
flg	flag for pick
onoff	turn or off for pick
NPX	index of pick in WPX
WPX	Pick data frame
NH	List of traces

**Value**

Returns WPX data frame with new pick added (or replaced).

**Note**

If WPX is missing, it is created. If NH is missing (no seismic traces) program returns NULL.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig, YPIX, WPIX, NOPIX, REPIX, PickWin, pADDPIX, Ppic, POLSWITCH, Pup

---

pickit

*Automatic Picking Algorithm*

---

**Description**

Automatic Picking Algorithm

**Usage**

```
pickit(ay, deltat = 0.008, MED = 225, FRWD = 8, BKWD = 8,  
      sbef = 1, saft = 6, thresh = 2, Tthresh2 = 7,  
      stretch = 1000, flo = 0.1, fhi = 5, Kmin = 7,  
      dthresh = 0.01, threshbot = 1.01)
```

**Arguments**

ay	signal
deltat	sample rate
MED	use median smoothing?
FRWD	forward window, s
BKWD	backward window
sbef	seconds before
saft	seconds after
thresh	threshold 1
Tthresh2	threshold 2
stretch	stretch factor
flo	low frequency for BP filter
fhi	low frequency for BP filter
Kmin	min number of picks per window
dthresh	delta threshold
threshbot	threshold bottom limit

**Details**

used internally. This code uses several methods for getting best pick.

**Value**

list(RAT=A\$rat, x=x, ay=ay, fy=fy, deltat=deltat, J=J\$J, Z=Z, a1=a1, a2=a2, thresh=thresh, Tthresh2=Tthresh2, Kmin=Kmin)

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

ETECTG

---

pickseis24                      *Pick zooms on 24 hour display*

---

### Description

Pick zooms on 24 hour display.

### Usage

```
pickseis24(w, DB, usta, ucomp, kind=-1, Iendian=1,
           BIGLONG=FALSE)
```

### Arguments

w	picking windows from output of plotseis24 and winseis24
DB	Database of seismic trace meta data
usta	stations to extract
ucomp	components to extract
kind	an integer -1, 0, 1, 2 ; 0="RDATA" , -1="RDS", 0="RDATA", 1 = "segy", 2 = "sac", see notes below
Iendian	vector, Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE=long=8 bytes

### Details

Use sequence of 2 clicks per zoom window on the plotseis24 display.

### Value

Graphical Side effects. Program starts swig

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

swig, winseis24 , plotseis24 , getseis24

**Examples**

```

if(interactive())
{
data(KH)

amp = KH$JSTR[[1]]
OLDdt = KH$dt[1]
newdt = 0.1
yr = 2000
GIVE = FAKEDATA(amp, OLDdt=0.01, newdt = 0.1, yr = 2000,
                JD = 4, mi = 12, sec = 0, Ntraces = 24*3,
                seed=200, noise.est=c(1, 100) , verbose=TRUE )

tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)

DB = FmakeDB(LF, kind=-1)

IDB = infoDB(DB)

START = list(yr =yr , jd= 5 , hr= 0 , mi= 0 ,sec= 0)

END = list(yr =yr , jd= 7 , hr= 0 , mi= 0 ,sec= 0)

h = getseis24(DB, iyear = 2000, iday = 5, usta = IDB$usta,
              acomp = IDB$ucomp, kind = -1, Iendian=1, BIGLONG=FALSE)

pjj <- plotseis24(h, dy=1/18, FIX=24, SCALE=1,
                 FILT=list(ON=FALSE, f1=0.05 , fh=20.0, type="BP", proto="BU"),
                 RCOLS=c(rgb(0.2, .2, 1), rgb(.2, .2, .2)) )

w = winseis24(pjj)

dev.new()

pickseis24(w, DB, IDB$usta[1], IDB$ucomp[1] )

}

```

---

plocator                      *Specialized Locator function*

---

### Description

Locator function with set parameters

### Usage

```
plocator(COL = 1, NUM = FALSE, YN = NULL, style = 0)
```

### Arguments

COL	color
NUM	number of points
YN	number of windows to span for lines
style	0,1,2 for differnt style of plotting vertical lines

### Details

if the window is divided into YN horizontal regions, style =2 will plot segments only within regions based on y-value of locator().

### Value

list:

x	x-locations
y	y-locations
n	number of points

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

### See Also

locator

### Examples

```
plot(c(0,1), c(0,1), type='n')
for(i in 1:5) { abline(h=i/6) }
```

```
if(interactive()) plocator(COL = 1, NUM = 4, YN = 6, style = 2)
```

PLOT.ALLPX                    *plot all phase arrival picks*

---

**Description**

plot all phase arrival picks

**Usage**

```
PLOT.ALLPX(t0, STNS, COMPS, YPX, PHASE = NULL, POLS = TRUE,  
          FILL = FALSE, FORCE = TRUE, cex = cex, srt = srt)
```

**Arguments**

t0	time for start of window, s
STNS	station names to plot
COMPS	components to plot
YPX	y-picks (times)
PHASE	Phases to plot
POLS	polaritiy information (up, down)
FILL	fill color
FORCE	logical, force all phases plotted on all traces
cex	character expansion
srt	string rotation angle, degrees

**Details**

for use in conjunction with PLOT.SEISN program

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PLOT.SEISN, swig



**Examples**

```

data(GH)

WPX = data.frame(GH$pickfile$STAS)
T0 = data.frame(GH$info)[1,]

sel = which(GH$COMPS=='V')
PLOT.SEISN(GH, sel=sel)

PLOT.ALLPX(T0, GH$STNS, GH$COMPS, WPX, PHASE='P', FORCE=TRUE)

```

---

PLOT.MATN

*plot a matrix of several seismograms*


---

**Description**

Matrix of several seismograms

**Usage**

```

PLOT.MATN(ascd, tim=1, dt=1, T1=0, WIN=c(0,1), labs="",
notes=notes, sfact=1, ampboost=0, shift=NULL, LOG="",
COL='red', add=1, AXES=1, units=NULL, VS=FALSE)

```

**Arguments**

ascd	N by K matrix of seismograms where
tim	time values fo x-axis
dt	sample interval, seconds
T1	Time for starting sample (default=0)
WIN	vector, time window for zoom
labs	vector of labels for each panel
notes	vector of notes for each panel
sfact	scaling factor, 1=window, 2=trace
ampboost	increase each amplitude by this multiplier
shift	vector, shift each trace by these time
LOG	log x-axis
COL	vector of colors or indexes to colors

add	numeric, to existing plot. add = 1,2,3 if add=1 plot and add traces, add =2 plot, but no traces, add = 3 no plot, but add traces. DEFAULT=1
AXES	numeric, 0,1,2,3,4; default=1
units	label for units of Y-axis
VS	var-squiggle display

### Details

Plots a matrix of seismograms that each have the same starting time. For the AXES argument, 0 = no axes, AXES=1 plot scale for largest amplitude band and a multiplier for all others, AXES=2 left side, AXES=3 right side, AXES=4 alternate sides

### Value

Graphical side effects and,

n	n
windiv	matrix of n rows, with columns=(window Y min, window Y max, user Y min, user Y max)

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

swig, matsquiggle, dowiggles, varsquiggle

### Examples

```
dt <- 0.001

t <- seq(0, 6, by=0.001)

thefreqs <- seq(from=10, to=100, by=10)
theamps <- runif(length(thefreqs))

# sample rate is 1000 Hz, 0.001 seconds 601 samples
x <- NULL

for(i in 1:length(thefreqs))
{
x <- cbind(x, theamps[i]*sin(2*pi*thefreqs[i]*t))
}

PLOT.MATN(x, dt = dt)
```

PLOT.SEISN

*Plot Seismic Section***Description**

Seismic traces are plotted on a panel horizontally.

**Usage**

```
PLOT.SEISN(GH, tim = 1, dt = 1, sel =c(1:4) , WIN =c(1,0) ,
labs=c("CE1") ,
notes = "CE1.V", subnotes=NA, tags ="CE1.V" ,
sfact = 1, LOG = "", COL = 'red', add = 1, pts = FALSE,
YAX = 1, TIT = NULL, SHIFT = NULL, COLLAPSE=FALSE, rm.mean = TRUE, UNITS = "volts",
MARK = TRUE, xtickfactor = 1, vertline=NA )
```

**Arguments**

GH	RSEIS data structure
tim	tim axis vector, seconds
dt	deltaT, sample rate
sel	select which traces from GH
WIN	initial time window for plot
labs	character string vector, labels for units on y-axes, depends on YAX
notes	character string vector, labels on upper right of each panel
subnotes	character string vector, labels on lower-right of each panel
tags	character string vector, labels next to right end of trace (usually numbers)
sfact	scaling flag, 1=scale individually(DEFAULT), 2 = scale by window
LOG	log for x-axis
COL	color vector for plotting traces
add	integer: add to plot=1,2,3, add=1 plot and add traces, add =2 plot, but no traces, add = 3 no plot, but add traces
pts	add points
YAX	type of Yaxis label, 1,2,3 DEFAULT=1 only one y-axis others scaled; 2=all y-axes are plotted on left; 3=all y-axes plotted, alternating left and right
TIT	title
SHIFT	vector, shift each trace along x-axis by associated moveout time
COLLAPSE	logical, Collapse all traces onto one panel, default=FALSE
,	
rm.mean	remove mean from traces

UNITS	character, units of traces (see labs)
MARK	character marking for earthquake
xtickfactor	Factor for multiplying the x-axis tick markers (default=1; for minutes=60, hrs=3600, days=24*3600)
vertline	time list (yr, jd, hr, mi sec) for plotting vertical lines on window. Default=NA

### Details

panel of N traces are plotted. For YAX, default is YAX=1, plot an axis with no units label and scale all the traces to

### Value

Graphical Side effect. list(n=nn, dy=dy, minS=minS, maxS=maxS, meanS=meanS, DX=range(tim[tflag]) )

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

### See Also

swig

### Examples

```
data("GH")
m <- match( GH$STNS, GH$stafilename)
LATS <- GH$stafilename$lat[m]
LONS <- GH$stafilename$lon[m]
dees <- rdistanz( GH$pickfile$LOC$lat, GH$pickfile$LOC$lon, LATS, LONS)

sel <- which(GH$COMPS=="V")
sel <- sel[order(dees$dist[sel])]

### set up good colors
pcols <- seiscols(GH)

### select only vertical components

PLOT.SEISN(GH, sel=sel)

GH$units <- rep("m/s", times=length(GH$KNOTES))
GH$pcols <- pcols

##### simple plot of GH structure
YN <- PLOT.SEISN(GH, WIN=c(5,12))

##### a color must be provided for all traces.
```

```
##### simple plot of GH structure, with selection and colors

YN <- PLOT.SEISN(GH, WIN=c(5,12), sel=sel, COL=rainbow(length(sel)) )

#### alternating Y axes
  YN <- PLOT.SEISN(GH, WIN=c(5,12) , dt=GH$dt[sel], sel=sel, sfact=1 ,
notes=GH$KNOTES[sel], YAX =3, UNITS = TRUE ,labs = GH$units[sel],
COL=pcols , TIT="test")

#### Y axes on same side
  YN <- PLOT.SEISN(GH, WIN=c(5,12) , dt=GH$dt[sel], sel=sel, sfact=1 ,
notes=GH$KNOTES[sel], YAX =2, UNITS = TRUE ,labs = GH$units[sel],
COL=pcols , TIT="test")
```

---

PLOT.TTCURVE

---

*Plot Seismic Section, travel time curve*


---

### Description

Seismic traces are plotted on a panel horizontally, with spacing according to distance from source.

### Usage

```
PLOT.TTCURVE(GH, STAXY = NULL, DIST = c(0, 10), DY = 0.1,
tim = 1, dt = 1, sel = c(1:4), WIN = c(1, 0), labs = c("CE1"),
notes = "CE1.V", tags = "CE1.V", sfact = 1, COL = "red",
add = 1, pts = FALSE, YAX = FALSE, TIT = NULL, SHIFT = NULL,
rm.mean = TRUE, UNITS = "volts", MARK = TRUE)
```

### Arguments

GH	Seismic data Structure
STAXY	Station Locations and distances in KM
DIST	Distance range, km
DY	height of each wiggle
tim	time span for plotting
dt	sample interval, seconds

sel	select which traces to plot
WIN	vector, time window for zoom
labs	vector of labels for each panel
notes	vector of notes for each panel
tags	character string vector, labels
sfact	scaling flag
COL	col vector
add	add to plot
pts	add points
YAX	Yaxis label
TIT	title
SHIFT	shift traces
rm.mean	remove mean from traces
UNITS	character, units of traces
MARK	character marking for earthquake

**Value**

Graphical Side effect.

list(n=nn, dy=dy, minS=minS, maxS=maxS, meanS=meanS, DX=range(tim[tflag]), DY=DY, DIST=DIST  
)

**Note**

This program is similar to PLOT.SEISN but traces are plotting with increasing distance from a set point. The distances are calculated prior to execution and passed as a vector or structure.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

PLOT.SEISN

---

Plot1Dvel

*Plot 1D Velocity Model*


---

**Description**

Plot 1D velocity model showing P-wave and S-wave layered models.

**Usage**

```
Plot1Dvel(v, tit = NULL, col=c('blue', 'brown'), ...)
```

**Arguments**

v	Velocity models
tit	Title for plot (character)
col	2-colors for P and swave
...	other graphical parameters (e.g. lty, lwd)

**Details**

Velocity model consists of a list of P and S depths and layer velocity values. See example below.

**Value**

Graphical Side effect

**Note**

Errors are not required, although future versions may include the plotting of error bars.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

Get1Dvel, Comp1Dvel, Comp1Dvels, travel.time1D

**Examples**

```
VEL <- list()
VEL$'zp' <- c(0,0.25,0.5,0.75,1,2,4,5,10,12)
VEL$'vp' <- c(1.1,2.15,3.2,4.25,5.3,6.25,6.7,6.9,7,7.2)
VEL$'ep' <- c(0,0,0,0,0,0,0,0,0,0)
VEL$'zs' <- c(0,0.25,0.5,0.75,1,2,4,5,10,12)
VEL$'vs' <- c(0.62,1.21,1.8,2.39,2.98,3.51,3.76,3.88,3.93,4.04)
VEL$'es' <- c(0,0,0,0,0,0,0,0,0,0)
```

```
VEL$name' <- '/data/wadati/lees/Site/Hengil/krafla.vel'  
Plot1Dvel(VEL, tit = 'This is an Example' )
```

---

plotarrivals

*plot theoretical arrival times for a seismic section*

---

### Description

plot theoretical arrival times for a seismic section

### Usage

```
plotarrivals(x, THEORY, add = FALSE)
```

### Arguments

x	matrix of wiggles
THEORY	theoretical arrivals
add	logical, if TRUE=Add to existing plot

### Details

plots go from top of page down

### Value

graphical side effect

### Note

Used for adding information to wiggle plots.

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

symshot1, wiggleimage



**Examples**

```
S1 <- symshot1()

wiggimage(S1$smograms , dt=(-S1$dt), dx=S1$dx)

plotarrivals(S1$x, S1$THEORY, add = TRUE)
```

---

plotDB

*Plot a time line of a DB set in RSEIS*


---

**Description**

makes a plot of the data base files stored on disk.

**Usage**

```
plotDB(DB)
```

**Arguments**

DB                    List, Data Base created by makeDB or setupDB

**Value**

Graphical Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

makeDB, setupDB

**Examples**

```
##### to illustrate, we make a set of individual seismograms
data(GH)
L1 = length(GH$JSTR)
DD = data.frame(GH$info)

GIVE = vector(mode='list')

for(i in 1:L1)
{
```

```

AA = DD[i,]
GIVE[[i]] = list(fn = AA$fn, sta =GH$STNS[i] , comp = GH$COMP[i],
               dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
               coords = NA, amp = GH$JSTR[[i]] )
}

##### save the seismic data in a temporary directory
#### each trace in a separate file
tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='RDS', full.names=TRUE)
DB = FmakeDB(LF, kind=-1)
## IDB = infoDB(DB)

plotDB(DB)

```

---

plotevol

*Plot Spectrogram*

---

## Description

Plot Spectrogram

## Usage

```
plotevol(DEVOL, log = 0, fl = 0, fh = 10, col = col, ylog = FALSE, ygrid
= FALSE, AXE = c(1, 2, 3, 4), CSCALE = FALSE, WUNITS = "Volts", STAMP =
NULL, STYLE = "fft")
```

```
plotevol2(DEVOL, log = 0, fl = 0, fh = 10, col = col, ylog = FALSE, ygrid
= FALSE, AXE = c(1, 2, 3, 4), CSCALE = FALSE, WUNITS = "Volts", STAMP =
NULL, STYLE = "fft", add=FALSE, IMAGE=TRUE, WIG=TRUE )
```

```
blankevol(DEVOL, log=0, fl=0, fh=10 , col=col, ylog=FALSE, ygrid=FALSE,
AXE=c(1,2,3,4),
CSCALE=FALSE, WUNITS="Volts", STAMP=NULL, STYLE="fft", WIG=TRUE )
```

**Arguments**

DEVOL	spectrogram structure
log	scale by logarithm
f1	low frequency
fh	high frequency
col	color palette
ylog	scale Y-axis by log
ygrid	logical, TRUE=add grid
AXE	sides to add axis
CSCALE	logical, TRUE=add color scale
WUNITS	character string for units
STAMP	character string for identification
STYLE	Plotting style. Default, "fft"=plot half the spectrum image , else plot whole spectrum
add	logical, add to existing plot, default=FALSE
IMAGE	logical, whether to plot the image or not
WIG	logical, whether to plot the wiggle or not

**Details**

Plot Spectrogram. Because the fft function returns positive and negative frequencies, if STYLE="fft" then the image matrix is reduced  $IMAT = t(DSPEC[1:(numfreqs/2),])$  otherwise  $IMAT = t(DSPEC)$ .

plotevol2 is used to add secondary spectra to ones already plotted, or to manage graphical paramters, or create other plots that match the graphical presentation of the spectrogram (plots of frequency versus time, but not images)

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

evolfft

**Examples**

```

data(CE1)

Xamp <- CE1$y

DT <- CE1$dt

tsecs <- DT*(length(Xamp)*.02)
  multi <- 2
scale.def <- 1
  TWOSEC <- tsecs*(1/DT)

NS <- floor(multi*TWOSEC)
NOV <- floor(multi*(TWOSEC-.2*TWOSEC))

Nfft<-4096

pal <- rainbow(100)

fl <- 0
fh <- 1/(2*DT)

flshow <- .5
fhshow <- 120

DEV <- evolfft(Xamp,DT , Nfft=Nfft, Ns=NS , Nov=NOV, fl=fl, fh=fh )

PE <- plotevol(DEV, log=scale.def, fl=flshow, fh=fhshow,
              col=pal, ygrid=FALSE, STAMP="HITHERE", STYLE="fft")

```

---

plotGH

*Plot a seismic trace.*


---

**Description**

Quick and dirty plot of a seismic trace as recorded and save using stream2GHnosens or other RSEIS savers.

**Usage**

```
plotGH(h)
```

**Arguments**

h This is a standard GH object as defined in RSEIS

**Details**

The input is a list that has, as a minimum the following items: 'amp', 'dt', 'sta', 'comp', 'DATTIM'. Item 'amp', a time series vector is converted to a ts object.

**Value**

Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

RSEIS::prepSEIS, RSEIS::prep1wig, RSEIS::PLOT.SEISN, RSEIS::swig

**Examples**

```
data(GH)
L1 = length(GH$JSTR)
DD = data.frame(GH$info)
#### convert to individual traces,
### here just use the first one:
i = 1
AA = DD[i,]
zh = list(fn = AA$fn, sta = GH$STNS[i], comp = GH$COMP[i],
         dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
         coords = NA, amp = GH$JSTR[[i]] )
##### plot
plotGH(zh)
```

---

plotJGET

*Plot JGET output*

---

**Description**

Plot JGET output using interactive swig

**Usage**

```
plotJGET(J, SHOWONLY = FALSE)
```

**Arguments**

J                    list, output of JGETseis  
 SHOWONLY        logical, if SHOWONLY== TRUE, no interaction

**Details**

Program combines prepSEIS and swig

**Value**

GH list ready for use in other RSEIS programs. See prepSEIS for details

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

JGET.seis, prepSEIS, swig

**Examples**

```
data(GH)
  Iendian = .Platform$endian
  apath = tempdir()
  ## setwd(apath)
  ##
  Iendian = .Platform$endian
  ## apath = './TEMP'
  ### dir.create(apath)

J = rseis2sac(GH, sel = 1:5, path = apath, BIGLONG =FALSE )

Lname <- list.files(path=J , pattern='SAC', full.names=TRUE)

J <- JGET.seis(Lname,kind=2,BIGLONG=FALSE,HEADONLY=FALSE,Iendian=Iendian,PLOT=0)

if(interactive()) { plotJGET(J) }
```

---

plotseis24

*Plot 24 hours of seismic data*

---

**Description**

Plot 24 hours of seismic data using output of getseis24.

**Usage**

```
plotseis24(JJ, dy = 1/18, FIX = 24, SCALE = 0, FILT = list(ON = FALSE,
f1 = 0.05, fh = 20, type = "BP", proto = "BU"), RCOLS = c(rgb(0.2, 0.2,
1), rgb(0.2, 0.2, 0.2)), add=FALSE )
```

**Arguments**

JJ	output list of getseis24
dy	Delta-y in percentage of trace
FIX	Fix 24 hour plot. If FIX is less than 24, the plot will show only that number of hours.
SCALE	scale, 0=scale each trace, 1=scale window
FILT	filter data
RCOLS	colors
add	logical, if TRUE, add to existing plot (i.e. do not issue a plot command)

**Details**

Plots full 24 hours of data. The list returned can be used by winseis24 to get picks and windows for zooming.

The FIX argument is currently not available.

**Value**

list:	
x	x-axis
y	y-axis
yr	year
jd	julian day

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

getseis24, winseis24

**Examples**

```
data(KH)

amp = KH$JSTR[[1]]
OLDdt = KH$dt[1]
newdt = 0.1
```

```

yr = 2000
GIVE = FAKEDATA(amp, OLDdt=0.01, newdt = 0.1, yr = 2000,
               JD = 4, mi = 12, sec = 0, Ntraces = 24*3,
               seed=200, noise.est=c(1, 100) , verbose=TRUE )

tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)

DB = FmakeDB(LF, kind=-1)

IDB = infoDB(DB)

START = list(yr =yr , jd= 5 , hr= 0 , mi= 0 ,sec= 0)

END = list(yr =yr , jd= 7 , hr= 0 , mi= 0 ,sec= 0)

h = getseis24(DB, iyear = 2000, iday = 5, usta = IDB$usta,
             acomp = IDB$ucomp, kind = -1, Iendian=1, BIGLONG=FALSE)

pjj <- plotseis24(h, dy=1/18, FIX=24, SCALE=1,
                 FILT=list(ON=FALSE, fl=0.05 , fh=20.0, type="BP", proto="BU"),
                 RCOLS=c(rgb(0.2, .2, 1), rgb(.2, .2, .2)) )

```

---

plotwlet

*Plot Wavelet Transform*


---

## Description

Plot Wavelet Transform

## Usage

```

plotwlet(baha, Ysig, dt, zscale = 1, zbound = NULL,
        col = rainbow(100), ygrid = FALSE,
        STAMP = "", xlab="Time, s" , units="", scaleloc=c(0.4,0.95))

```



**Arguments**

baha	Output of wlet.do
Ysig	signal processed
dt	sample rate
zscale	scale of image
zbound	limits on scale
col	color palette
ygrid	add grid
STAMP	character string for identification
xlab	character, label for the x-axis
units	character, units on signal
scaleloc	2-vector, percentatge of bottom margin for the color scale

**Details**

This function plots the wavelet transform in a way that is similar to the spectrogram plots.

**Value**

list( y=, why=why, yBounds=c(0,perc), x=x, yat=raxspec)

y	input signal
why	scaled image
yBounds	vector of boundaries
x	x axis
yat	y axis tic marks

Graphical side effects.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

cwt, pwlet2freqs, wlet.do, wlet.drive

---

`plt.MTM0`*Plot MTM structure*

---

**Description**

Plot MTM structure

**Usage**`plt.MTM0(frange, prange, plxy, M, freqs, amp, a, dof = dof, Fv = Fv, COL = 2)`**Arguments**

<code>frange</code>	frequency range
<code>prange</code>	point range
<code>plxy</code>	log x,y axes
<code>M</code>	structure from MTM
<code>freqs</code>	frequencies
<code>amp</code>	amplitude
<code>a</code>	list(y=original data, dt=deltat)
<code>dof</code>	degrees of freedom
<code>Fv</code>	F-values
<code>COL</code>	color

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**References**

Lees, J. M. and Park, J., 1995: Multiple-taper spectral analysis: A stand-alone C-subroutine, *Computers and Geology*, 21(2), 199-236.

**See Also**

MTM.drive

**Examples**

```

data(CE1)
plot(CE1$x, CE1$y, type='l')

len <- length(CE1$y)
len2 <- 2*next2(len)
Mspec <- mtapspec(CE1$y, CE1$dt, klen=len2, MTP=list(kind=1,nwin=5,
              npi=3,inorm=0) )

f<-Mspec$freq
M <- 1
f1 <- 0.01
f2 <- 100
plxym <- ''
flag <- f>=f1 & f <= f2;
  freqs <- list(f[flag])
mydof <- NULL
myFv <- NULL
amp <- Mspec$spec[1:length(f)]

  amp <- list(amp[flag])

a <- list(y=CE1$y, dt=CE1$dt)
frange <- range(freqs, na.rm = TRUE)
prange <- range(amp , na.rm = TRUE)

### plot(freqs[[1]], amp[[1]])

plt.MTM0(frange, prange, plxy, M, freqs, amp, a,
        dof=mydof, Fv=myFv, COL=4)

```

---

 PLTpicks

*Plot picks on seismic record*


---

**Description**

Add lines at phase arrival times

**Usage**

```
PLTpicks(picks, labs = NA, cols = NA)
```

**Arguments**

picks	vector of times relative to the start of the plot
labs	labels for picks
cols	colors for picks

**Details**

picks = vector of times relative to the start of the plot (seismogram)

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
ex <- seq(from=0, to=4*pi, length = 200)

y <- sin(ex)
picks = c(0.5*pi, 2.3*pi)
plot(ex, y, type='l')

PLTpicks(picks, labs =c("P","P") , cols =c('red','green') )

PLTpicks(picks+2, labs =c("S","PKIKP") , cols = 'blue' )
```

---

PMOT.drive

*Interactive Particle Motion Plot*

---

**Description**

Plot Hodogram and show seismic particle motion

**Usage**

```
PMOT.drive(temp, dt, pmlabs = c("Vertical", "North", "East"), STAMP = "", baz = 0)
```

**Arguments**

temp	matrix of 3-component seismic signal
dt	sample interval (delta-T, seconds)
pmlabs	labels for traces
STAMP	Character string Identification stamp
baz	Back Azimuth, degrees

**Details**

Input matrix should V, N, E. Baz is not implemented yet.

**Value**

Graphical Side Effect.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data("GH")
sel <- which(GH$STNS == "CE1")

YMAT <- cbind(GH$JSTR[[sel[1]]][1168:1500],
              GH$JSTR[[sel[2]]][1168:1500],
              GH$JSTR[[sel[3]]][1168:1500])

dt <- GH$dt[ sel[1] ]
ftime <- Zdate(GH$info, sel[1], 1)

if(interactive()){
  PMOT.drive(YMAT, dt, pmolabs = c("Vertical", "North", "East"),
             STAMP =ftime )
}
```

---

posix2RSEIS

*Posix to RSEIS DATE/TIME*

---

**Description**

Reformat posix time stamp to RSEIS list

**Usage**

```
posix2RSEIS(p)
```

**Arguments**

p                    posix time, either lt or ct

**Value**

returns a list of data/time in format RSEIS understands

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

j2posix

**Examples**

```
### make up a time:
P1 = as.POSIXct(Sys.time(), "America/New_York") # in New York
R1 = posix2RSEIS(P1)
## also
unlist( as.POSIXlt(P1))
```

---

PPIX

*P-picking*

---

**Description**

Add Pick Marks and Labels

**Usage**

```
PPIX(zloc, YN = NULL, col = 1, lab = "")
```

**Arguments**

zloc	locator output
YN	number of panels
col	color for picks
lab	labels for picks

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

 prep1wig

 Prepare structure for RSEIS
 

---

### Description

Takes list of traces and prepares new list for analysis in RSEIS

### Usage

```
prep1wig(wig=vector(), dt=1, sta="STA", comp="CMP",
units="UNITS", starttime=list(yr=0, jd=1, mo=1, dom=1,
hr=1, mi=1, sec=0) )
```

### Arguments

wig	vector of time series
dt	sample interval
sta	character, station name
comp	character, component name
units	character, units of signal
starttime	list(yr=1972, jd=1, mo=1, dom=1, hr=1, mi=1, sec=0)

### Details

prep1wig is offered to reformat a time series  
for input to program swig()

### Value

Rnac output list

amp	amplitude
dt	sample rate
nzyear	year
nzhour	hour
nzmin	minutes
nzsec	seconds
nzmsec	msec
b	sac stuff
e	sac stuff
o	sac stuff
fn	character, file name

sta	character
comp	character
DATTIM	list of date and time
N	number of points
units	character

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig, prepSEIS

**Examples**

```

data(sunspots)
AA <- attributes(sunspots)
starttime<-list(yr=AA$tsp[1], jd=1,mo=1,dm=1,hr=0,mi=0,sec=0)
ES <- prep1wig(wig=sunspots, dt=1/12, sta="STA", comp="CMP",
units="UNITS", starttime=starttime )

EH<-prepSEIS(ES)

STDLAB <- c("DONE", "zoom out", "refresh", "restore",
"XTR", "SPEC", "SGRAM", "WLET")

##### set SHOWONLY=FALSE for interactive
xx <- swig( EH, STDLAB = STDLAB, SHOWONLY=0)

#####
#####
##### example with multiple signals

dt <- 0.001
t <- seq(0, 6, by=0.001)
##### sample rate = 1000 Hz, 0.001 seconds 601 samples

### set up the frequencies and amplitudes for signals that have 2 frequencies
afreqs1 <- c(50, 40,10, 5 )
amps1 <- c(6, 2,3, 2 )
####
afreqs2 <- c(120,30,20, 30 )
amps2 <- c(10,5, 9, 2 )

x <- cbind( amps1[1]*sin(2*pi*afreqs1[1]*t) +
amps2[1]* sin(2*pi*afreqs2[1]*t),
amps1[2]*sin(2*pi*afreqs1[2]*t) + amps2[2]* sin(2*pi*afreqs2[2]*t),
amps1[3]*sin(2*pi*afreqs1[3]*t) + amps2[3]* sin(2*pi*afreqs2[3]*t),
amps1[4]*sin(2*pi*afreqs1[4]*t) + amps2[4]* sin(2*pi*afreqs2[4]*t))

```



```

d <- dim(x)

##### names of signals
mysta<-c("R1", "R2", "R3", "R4")

MYLIST <- list()
starttime <- list(yr=2008, jd=1,mo=1,dm=1,hr=0,mi=0,sec=0)
##### set up the initial list of wiggles
for(i in 1:d[2])
{

A <- prep1wig(wig =x[,i], sta=mysta[i], dt=dt, comp="D0",
units= "amp", starttime=starttime)

A[[1]]$DATTIM$yr <- 2000
MYLIST <- c(MYLIST, A)

}

### reorganize into RSEIS format:
PH1 <- prepSEIS(MYLIST)

STDLAB <- c("DONE", "zoom out", "refresh", "restore",
"XTR", "SPEC", "SGRAM", "WLET")

swig(PH1, STDLAB = STDLAB)

```

---

```
prepSEIS
```

```
Prepare structure for RSEIS
```

---

### Description

Takes list of traces and prepares new list for analysis in RSEIS

### Usage

```
prepSEIS(GG)
```

### Arguments

GG                      Output list of Rsac function GET.seis

### Details

prepSEIS is offered to reformat the output of a list of seismic traces (or other time series) for input to program swig()

**Value**

RSEIS list

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig, JGET.seis, GET.seis(package="Rsac"), Package:Rsac

**Examples**

```
data(sunspots)

ES <- prep1wig(wig=sunspots, dt=1/12, sta="STA",
              comp="CMP", units="UNITS" )

EH <- prepSEIS(ES)

STDLAB <- c("DONE", "zoom out", "refresh", "restore",
           "XTR", "SPEC", "SGRAM", "WLET")

xx <- swig( EH, STDLAB = STDLAB)
#####
#####
```

---

PreSet.Instr

*Set up Standard Instrument Responses*

---

**Description**

A set of standard known instrument responses.

**Usage**

PreSet.Instr()

**Value**

List of instrument responses. Each is a list:

np	Number of poles
poles	complex vector of poles
nz	number of zeros
zeros	complex vector of zeros
Knorm	normalization factor
Sense	sensitivity factor

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

**See Also**

ReadSet.Instr

**Examples**

```
MYset <- PreSet.Instr()
MYset[[1]]
```

---

PSTLTcurve

*Short Term/Long Term curve*


---

**Description**

ST/LT ratio curve for sutomated picking routines

**Usage**

```
PSTLTcurve(y, dt = 0.008, fwlen = 125, bwlen = 125,
  perc = 0.05, stretch = 1000, MED = 255, PLOT = FALSE)
```

**Arguments**

y	signal
dt	deltaT (s)
fwlen	forward window
bwlen	backward window
perc	percent cut-off
stretch	stretch curve
MED	Median smoothing parameter
PLOT	logical, TRUE=PLOT

**Value**

list(flag=1, ind=ix, eye=eye, mix=mix, SNR=SNR, s2=s2, rat=therat)

flag	flag on success
ind	index of pick estimate 1
eye	index of pick estimate 2
mix	index of pick estimate 3
SNR	Signal/Noise ratio
s2	sum squared
rat	ratio curve

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
require(stats)

data(CE1)
plot(CE1$x, CE1$y, type='l')

z <- CE1$y[ CE1$x>5.352622 & CE1$x<5.589836]
x <- CE1$x[ CE1$x>5.352622 & CE1$x<5.589836]

G <- PSTLTcurve(z, dt = CE1$dt, fwlen = 10,
               bwlen = 10, perc = 0.05,
               stretch = 10, MED = 11, PLOT = FALSE)

### get time from beginning of trace
tpick <- x[G$ind]
abline(v=x[G$ind], col='red', lty=2)
```

---

Put1Dvel

*Dump a velocity model to an ascii file*

---

**Description**

Dump a velocity model to an ascii file

**Usage**

```
Put1Dvel(vel, outfile)
```

**Arguments**

vel	Velocity Model Structure
outfile	File name

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

Get1Dvel, travel.time1D

---

pwlet2freqs

*Convert Wavelet Axis to Frequency*

---

## Description

Convert Wavelet Axis to Frequency

## Usage

```
pwlet2freqs(noctave, nvoice, dt, flip = TRUE,  
tab.FREQ, plot = FALSE, perc = 0.85)
```

## Arguments

noctave	number of octaves
nvoice	number of voices
dt	sample rate (s)
flip	logical, whether to flip the orientation
tab.FREQ	vector of frequencies
plot	logical, TRUE=add to plot
perc	percent of range to consider

## Details

This function is used to add a y-axis to a wavelet transform plot.

## Value

list:

why	y-axis coordinate on wavelet transform
Iat	location
efs	frequencies

## Author(s)

Jonathan M. Lees<jonathan.lees.edu>

## See Also

wlet.do

**Examples**

```
pfreqs <- c(0.5, 1, 2,3,4,5, 10, 14)

zp <- pwlet2freqs(noctave= 6, nvoice= 20, 0.004,
  flip = TRUE, pfreqs, plot = FALSE, perc = 0.85)
```

---

rangedatetime	<i>Range of Date Time</i>
---------------	---------------------------

---

**Description**

Return the range of dates and times for any list with a date/time list

**Usage**

```
rangedatetime(D)
```

**Arguments**

D	info list from RSEIS seismic data list
---	--

**Value**

min	date time list
max	date time list

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
data(GH)

rangedatetime(GH$info)
```

---

Ray.time1D                      *Seismic 1D Travel Time and raypath*

---

**Description**

Travel time and raypath from source to receiver in 1D local model.

**Usage**

```
Ray.time1D(indelta, inhpz, instaz, inlay, ztop, vel)
```

**Arguments**

indelta	distance in KM
inhpz	depth of hypocenter, km
instaz	elevation of station
inlay	number of layers
ztop	vector, tops of layers
vel	vector, velocities in layers

**Details**

Uses local 1D velocity model, not appropriate for spherical earth.

**Value**

list:

dt dr	derivative of t w.r.t. horizontal distance
dt dz	derivative of t w.r.t. z, depth
angle	incidence angle, degrees
tt	travel time, s
nnod	number of nodes
znod	node depths, km
rnod	node offset distances, km

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

travel.time1D, Get1Dvel

**Examples**

```

data(VELMOD1D)

v <- VELMOD1D
indelta=23.;
  inhpz=7.;
  instaz=0.;
nz = length(v$zp)

tees <- travel.time1D(indelta, inhpz, instaz, nz , v$zp , v$vp)
rays <- Ray.time1D(indelta, inhpz, instaz, nz , v$zp , v$vp)

plot(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod],type="n",
     xlab="distance, km" , ylab="Depth, km")

abline(h=-v$zp, lty=2, col=grey(0.80) )
points(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod], pch=8, col='green')
lines(rays$rnod[1:rays$nnod] , -rays$znod[1:rays$nnod])
points(rays$rnod[rays$nnod] , -rays$znod[rays$nnod], pch=6, col='red', cex=2)
##### to coordinate this in space, need to rotate about
##### the line between source and receiver locations

```

---

rdistaz

*Distance and Azimuth from two points*


---

**Description**

Calculate distance, Azimuth and Back-Azimuth from two points on Globe.

**Usage**

```
rdistaz(olat, olon, tlat, tlon)
```

**Arguments**

olat	origin latitude, degrees
olon	origin longitude, degrees
tlat	target latitude, degrees
tlon	target longitude, degrees



**Details**

The azimuth is returned in degrees from North.

Program is set up for one origin (olat, olon) pair and many target (tlat, tlon) pairs given as vectors.

If multiple olat and olon are given, the program returns a list of outputs for each.

If olat or any tlat is greater than 90 or less than -90, NA is returned and error flag is 0.

If any tlat and tlon is equal to olat and olon, the points are coincident. In that case the distances are set to zero, but the az and baz are NA, and the error flag is set to 0.

**Value**

List:

del	Delta, angle in degrees
az	Azimuth, angle in degrees
baz	Back Azimuth, angle in degrees from target to origin
dist	Distance in km
err	0 or 1, error flag. 0=error, 1=no error, see details

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

along.great, getgreatarc

**Examples**

```
#### one point
d <- rdistaz(12, 23, -32, -65)
d

#### many random target points
org <- c(80.222, -100.940)
targ <- cbind(runif(10, 10, 50), runif(10, 20, 100))

rdistaz(org[1], org[2], targ[,1], targ[,2])

##### if origin and target are identical
##### the distance is zero, but the az and baz are not defined
rdistaz(80.222, -100.940, 80.222, -100.940)

##### set one of the targets equal to the origin
targ[7,1] <- org[1]
targ[7,2] <- org[2]
```

```

rdistaz(org[1], org[2], targ[,1], targ[,2])

#### put in erroneous latitude data

targ[3,1] <- -91.3

rdistaz(org[1], org[2], targ[,1], targ[,2])
#####
### New York and Chapel Hill
NY =list(lat=40.6698, lon=286.0562)
CH = list(lat=35.92761, lon=280.9594)
## h = GEOMap::distaz(CH$lat, CH$lon, NY$lat, NY$lon)
h = rdistaz(CH$lat, CH$lon, NY$lat, NY$lon)

##### get great circle ray path
RAY = GEOMap::getgreatarc(CH$lat, CH$lon, NY$lat, NY$lon, 100)
#### get great circle through north pole
Nor1 = GEOMap::getgreatarc(CH$lat, CH$lon, 90, CH$lon, 100)
PROJ = GEOMap::setPROJ(2, CH$lat, CH$lon)
RAY.XY = GEOMap::GLOB.XY(RAY$lat, RAY$lon, PROJ)
Nor1.XY = GEOMap::GLOB.XY(Nor1$lat, Nor1$lon, PROJ)
VEE1 = c(Nor1.XY$x[2]-Nor1.XY$x[1], Nor1.XY$y[2]-Nor1.XY$y[1])
VEE2 = c(RAY.XY$x[2]-RAY.XY$x[1], RAY.XY$y[2]-RAY.XY$y[1])
VEE1 = VEE1/sqrt(sum(VEE1^2))
VEE2 = VEE2/sqrt(sum(VEE2^2))
##### get angle from north:
ANG = acos( sum(VEE1*VEE2) ) *180/pi
#### compare with h

print(paste(h$az, ANG, h$az-ANG) )

```

---

rDUMPLOC

*DUMP vectors to screen in list format*


---

### Description

For saving vectors to a file after the locator function has been executed.

### Usage

```
rDUMPLOC(zloc, dig = 12)
```

### Arguments

zloc	x,y list of locator positions
dig	number of digits in output

**Value**

Side effects: print to screen

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
G <- list()
G$x <- c(-1.0960,-0.9942,-0.8909,-0.7846,-0.6738,-0.5570,-0.4657,-0.3709,
-0.2734,-0.1740,-0.0734, 0.0246, 0.1218, 0.2169, 0.3086, 0.3956, 0.4641,
0.5293, 0.5919, 0.6530, 0.7131)
G$y <- c(-0.72392,-0.62145,-0.52135,-0.42599,-0.33774,-0.25896,-0.20759,
-0.16160,-0.11981,-0.08105,-0.04414,-0.00885, 0.02774, 0.06759, 0.11262,
0.16480, 0.21487, 0.27001, 0.32895, 0.39044, 0.45319)

g <- G
rDUMPLoc(g, dig = 5)
```

---

read1segy

*Read one SEGYSAC file*

---

**Description**

Read one SEGYSAC file

**Usage**

```
read1segy(fname, Iendian = 1, HEADONLY = FALSE, BIGLONG = FALSE)
read1sac(fname, Iendian = 1, HEADONLY = FALSE, BIGLONG = FALSE )
```

**Arguments**

fname	character, file name
Iendian	Endian of the input file name
HEADONLY	logical, TRUE=return only header (default=FALSE)
BIGLONG	logical, indicating whether long is 8 or 4 bytes.

**Details**

Segy format files are in integer format. The time series usually represents counts recorded in a data acquisition system. The header includes meta-data and other identifying information.

SAC data is stored as floats, typically volts.

**Value**

list of header and times series

**Note**

The Endian-ness of the input files is set by the system that created them. If the read1segy or read1sac does not make sense, try a different endian or BIGLONG setting.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

write1sac, write1segy, sac2rseis, segy2rseis, prepSEIS

**Examples**

```
data(GH)
theENDIAN = .Platform$endian

apath = tempdir()
J = rseis2segy(GH, sel=1:5, path=apath , BIGLONG=FALSE )

Lname <- list.files(path=J , pattern='SEGY', full.names=TRUE)

zed = read1segy(Lname[1], Iendian = theENDIAN,
  HEADONLY = FALSE, BIGLONG = FALSE)
```

---

ReadInstr

*Read Instrument Response in IRIS SEED format*

---

**Description**

Read Instrument Response, poles and zeros, in IRIS SEED format.

**Usage**

```
ReadInstr(fn)
```

**Arguments**

fn                    File name with Poles and Zeros

**Details**

RSEIS currently has a function (ReadSet.Instr) to read pole/zero files, but it seems to expect a format different from what one gets from IRIS. This one is compatible with pole/zero files produced by rdseed when converting seed files from the DMC to SAC files.

**Value**

List of poles and zeros compatible for swig decon

**Author(s)**

Jake Anderson<ajakef@gmail.com>

**See Also**

ReadSet.Instr

**Examples**

```
##### create a SAC format response file:
temp.file= tempfile("PZ")
cat(file=temp.file, c(
  "ZEROS 4",
  "-999.0260 0.0000",
  "POLES 6",
  "-0.1480 0.1480",
  "-0.1480 -0.1480",
  "-314.1600 0.0000",
  "-9904.8000 3786.0000",
  "-9904.8000 -3786.0000",
  "-12507.0000 0.0000",
  "CONSTANT 4.540182e+20"), sep='\n')

RESP <- ReadInstr(temp.file)
```

---

ReadSet.Instr

*Read Instrument Response file*

---

**Description**

Read in an instrument response file, or

**Usage**

```
ReadSet.Instr(file)
```

**Arguments**

file                    name of file to read, or vector of character strings from the file

**Details**

If file is a path to a file it is read in and processed. If file is a vector of character strings from a file that has already been read in, the file is processed directly. The tag names (ZEROS, POLES, SENSE, CONSTANT) can be upper, lower or mixed case. Alternative to SENSE = sensitivity, and CONSTANT=norm or knorm.

**Value**

list:

np	Number of poles
poles	complex vector of poles
nz	number of zeros
zeros	complex vector of zeros
Knorm	normalization factor
Sense	sensitivity factor

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
### in this case a file has already been read in:
CMG <- c(
  "ZEROS 2",
  "0.0000E+00 0.0000E+00",
  "0.0000E+00 0.0000E+00",
  "POLES 3",
  "-0.1480E+00 0.1480E+00",
  "-0.1480E+00 -0.1480E+00",
  "-50.0 0.0",
  "CONSTANT 1.0",
  "SENSE 800")

ReadSet.Instr(CMG)
```

---

readUW.OSTAS

*Parse UW O-Cards*

---

**Description**

PARse out UW O-cards from Pickfile

**Usage**

readUW.OSTAS(OS1)

**Arguments**

OS1                    cards starting with O

**Value**

vector of station names not picked

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

redate	<i>Rectify Date</i>
--------	---------------------

---

**Description**

Rectify a date that may be out of wack.

**Usage**

```
redate(jd=0, hr=0, mi=0, sec=0, yr=0)
redate1(X)
```

**Arguments**

jd	Julian Day
hr	hours
mi	minutes
sec	seconds
yr	year
or	
X	list of date

**Details**

Returns date with correct numbers. So if number of seconds is greater than 60, will add to minutes...

**Value**

jd	Julian Day
hr	hours
mi	minutes
sec	seconds
yr	year

**Note**

Default value for jd is 1, the rest are 0. This function now should successfully span year breaks. Leap years are correctly accounted for too.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

showdatetime, DAYSPerYEAR, fromjul, getjul, tojul, getmoday

**Examples**

```
reccdate(76, 23, 22, yr=2000)

##### example spanning leap year
## start on Day 1, but subtract 36 hours and proceed to plus 36 hours
hrs <- seq(from=-36, to=36, by=2)
rd <- reccdate(jd=1, hr=hrs, mi=34,
              sec=23+runif(n=length(hrs), 0, 59) , yr=2009)
write.table(data.frame(rd))

##### example spanning non-leap year
rd2 <- reccdate(jd=1, hr=hrs, mi=34,
               sec=23+runif(n=length(hrs), 0, 59) , yr=2008)
write.table(data.frame(rd2))
```

---

repairWPX

*Repair WPX*

---

**Description**

Repair a WPX list that may be deficient in one or more of its components.

**Usage**

```
repairWPX(wpx)
```

**Arguments**

wpx                      Pick information, dataframe

**Details**

Program checks a few of the elements and tries to fix potential problems.



**Value**

WPX dataframe

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX, catWPX, checkWPX, cleanWPX, clusterWPX, saveWPX, setWPX

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))  
s1$col <- NULL  
s2 <- repairWPX(s1)
```

---

replaceWPX

*Replace picks in WPX file*

---

**Description**

Replace pick in WPX file

**Usage**

```
replaceWPX(WPX, onepx , ind=1)
```

**Arguments**

WPX	WPX list
onepx	WPX list with one pick
ind	integer, index to replace

**Details**

Replaces one pick at index provided.

**Value**

WPX list

**Note**

Replaces in the location provided. No test is made to determine if there is a pick already there. Maybe future versions will allow multiple replacements.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX, catWPX, deleteWPX, selWPX

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
s2 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(1))
```

```
s4 <- replaceWPX(s1,s2, ind=4)
```

---

rseis2segy

---

*Convert RSEIS to SEGYSAC format*


---

**Description**

Convert RSEIS to SEGYSAC format

**Usage**

```
rseis2segy(GH, sel = 1, win = c(0, 1), path = ".", BIGLONG = FALSE)
rseis2sac(GH, sel = 1, win = c(0, 1), path = ".", BIGLONG = FALSE)
```

**Arguments**

GH	RSEIS format list
sel	select traces to convert
win	vector, t1 and t2 window each trace
path	path to directory where files are created
BIGLONG	logical, indicating whether long is 8 or 4 bytes.

**Details**

This is the converse of the segy2rseis routine.

Segy format files are in integer format. The time series usually represents counts recorded in a data acquisition system. The header includes meta-data and other identifying information.

**Value**

Side effects in file system

**Note**

The Endian-ness of the output file will be the native endian-ness of the system.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

write1segy, write1sac, read1sac, read1segy, sac2rseis, segy2rseis

**Examples**

```
data(KH)
apath = tempdir()

J = rseis2segy(KH, sel=1, path=apath, BIGLONG=FALSE )
L = list.files(path=J, full.names=TRUE)
Z = read1segy(L[1], Iendian = 1, HEADONLY = FALSE, BIGLONG = FALSE)

# data(KH)
# apath = tempdir()
J = rseis2sac(KH, sel = 1, win = c(0, 1), path = apath, BIGLONG = FALSE)

L = list.files(path=J, full.names=TRUE)
Z = read1sac(L[1], Iendian = 1, HEADONLY = FALSE, BIGLONG = FALSE)
```

---

rseis2ts

---

*Convert RSEIS to TS*


---

**Description**

Convert one trace from an RSEIS seismic list to a ts time-series object.

**Usage**

```
rseis2ts(GH, sel = 1, notes = "")
```

**Arguments**

GH	List structure of seismic traces from RSEIS
sel	numeric index of one trace.
notes	character string of notes

**Details**

Function extracts one trace and associated information from an RSEIS structure and returns a ts, time-series, object.

**Value**

ts object

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
data(GH)
H = rseis2ts(GH, 1, notes='Coso Trace 1')
plot(H)
title(main=attr(H, 'info')$notes)
```

---

rsspec.taper

*Taper spectrum*

---

**Description**

Taper function for spectrum analysis

**Usage**

```
rsspec.taper(x, p = 0.1)
```

**Arguments**

x	time series trace
p	percent taper

**Details**

Cosine taper at ends of trace.

**Value**

tapered trace is returned.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data(CE1)
Xamp <- CE1$y[CE1$x > 5.443754 & CE1$x<5.615951]
### 10% cosine taper:
xtap <- rsspec.taper(Xamp, p = 0.1)
```

---

ruler

*Column Ruler*

---

**Description**

Column Ruler for determining columns to read.

**Usage**

```
ruler(a = "")
```

**Arguments**

a                    character string, optional

**Details**

This routine is set up to help get the columns for specific column oriented data. The ruler is dumped out below the character string for comparison. If no string is provided, just the rule is dumped. Use routine substr to extract the data from the columns.

**Value**

Side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

substr

**Examples**

```
aa <- paste(runif(n=5), collapse='-')
aa = substr(aa, 1, 72)
ruler(aa)
```

---

`save.wpix`*Save WPIX from swig output*

---

**Description**

Save WPIX from swig output

**Usage**

```
save.wpix(KOUT, fn = "wpix.out")
```

**Arguments**

KOUT	List output from swig
fn	file name for saving.

**Details**

Takes the output list from swig, specifically the WPX component and writes a table to the file system. This function is embedded in view.seis.

**Value**

Side effects: file is created and appended to.

**Note**

User must have write permission to the file.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

view.seis, swig

---

saveWPX	<i>Save WPX</i>
---------	-----------------

---

**Description**

Save a WPX list to a file on the local file system.

**Usage**

```
saveWPX(twpx, destdir = ".")
```

**Arguments**

twpx	WPX list
destdir	character, destination directory, default=getwd()

**Details**

Creates a file with the list as in native binary format. This file can be loaded with the standard load function in R. The name of the file is created by using the minimum time extracted from the WPX list. The suffix on the file name is RDATA. When reading in, the object created is named "twpx" for further processing.

**Value**

Side effects on file system. The name of the output file is returned.

**Note**

User must have write access to the destination directory.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX, catWPX, checkWPX, cleanWPX, clusterWPX, repairWPX, setWPX

**Examples**

```
tdir = tempdir()
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
hh <- saveWPX(s1, destdir = tdir )

### read in the data

load(hh)
```

```
data.frame(twpx)
```

---

scal2freqs	<i>Wavelet Frequency Scale</i>
------------	--------------------------------

---

**Description**

Get frequencies associated with the wavelet transform.

**Usage**

```
scal2freqs(octs, dt, plot = FALSE)
```

**Arguments**

octs	number of octaves
dt	sample rate, s
plot	logical, TRUE=plot

**Details**

Use morelet wavelet to estimate frequency from wavelet transform.

**Value**

frequency values

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

Mmorlet, fft

**Examples**

```
noctave <- 6
nvoice <- 20
dt <- 0.004
i1 <- sort(rep(c(1:noctave), times=nvoice))
jj <- rep(c(0:(nvoice-1)), times=noctave)

sa <- 2^(i1+jj/nvoice)

efs <- scal2freqs(sa, dt)
```



---

screens

*screens*

---

**Description**

Open  $n$  devices for plotting.

**Usage**

screens( $n$ )

**Arguments**

$n$                     number of devices required

**Details**

If  $k$  screens are open and  $k \geq n$ , nothing is done.

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

devices

**Examples**

```
if(interactive() ) screens(2)
```

SEARCHPIX

*Search Pix*

---

**Description**

Search through pick structure to select phase arrivals

**Usage**

```
SEARCHPIX(KPX, IPX, tol = 0.5)
```

**Arguments**

KPX	user locator pix
IPX	set of pix in memory
tol	tolerance, s

**Details**

returns index vector of picks that satisfy:  $w_n = \text{which}(\text{abs}(t_2 - t_1) < \text{tol})$

**Value**

index vector

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data(GH, package='RSEIS')
IPX = data.frame( uwpfile2ypx(GH$pickfile ) )

##### take for example on pick
KPX = IPX[6, ]

SEARCHPIX(KPX, IPX, tol = 0.5)
```

---

secdif	<i>Return difference in seconds</i>
--------	-------------------------------------

---

**Description**

Difference between two Date/Times (Julian Day)

**Usage**

```
secdif(jd1, hr1, mi1, sec1, jd2, hr2, mi2, sec2)
```

**Arguments**

jd1	Julian Day
hr1	hour
mi1	minute
sec1	second
jd2	Julian Day
hr2	hour
mi2	minute
sec2	second

**Details**

Returns T2-T1. Year is not included.

**Value**

numeric	seconds
---------	---------

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

secdifL

**Examples**

```
T1 <- list(jd=12, hr=13, mi=23, sec=21)
T2 <- list(jd=14, hr=23, mi=23, sec=2)
secdif(T1$jd, T1$hr, T1$mi, T1$sec, T2$jd, T2$hr, T2$mi, T2$sec)
```

---

secdifL                      *Seconds Difference*

---

**Description**

Given two date/time lists, return seconds difference

**Usage**

```
secdifL(T1, T2)
```

**Arguments**

T1                      list(jd, hr, mi, sec)

T2                      list(jd, hr, mi, sec)

**Details**

Year is not included in this calculation.

**Value**

numeric                seconds

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

secdif

**Examples**

```
T1 <- list(jd=12, hr=13, mi=23, sec=21)
T2 <- list(jd=14, hr=23, mi=23, sec=2)
secdifL(T1, T2)
```

---

secdifv	<i>Seconds Difference</i>
---------	---------------------------

---

**Description**

Given two date/time vectors, return seconds difference

**Usage**

```
secdifv(T1, T2)
```

**Arguments**

T1                   c(jd, hr, mi, sec)

T2                   c(jd, hr, mi, sec)

**Details**

Year is not included in this calculation.

**Value**

numeric           seconds

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

secdif

**Examples**

```
T1 <- c(12, 13, 23, 21)
T2 <- c(14, 23, 23, 2)
secdifv(T1, T2)
```

---

 segy2rseis

*Read in multiple segy files.*


---

**Description**

Read in multiple segy files, and create a list of seismic traces.

**Usage**

```
segy2rseis(fnames, Iendian = 1, HEADONLY = FALSE, BIGLONG = FALSE, PLOT
= -1, RAW = FALSE)
sac2rseis(fnames, Iendian = 1, HEADONLY = FALSE,
BIGLONG = FALSE, PLOT = -1, RAW = FALSE)
```

**Arguments**

fnames	character vector of file names.
Iendian	Endian-ness of the files
HEADONLY	logical, TRUE=read only the header information. default=FALSE
BIGLONG	logical, indicating whether long is 8 or 4 bytes.
PLOT	logical, TRUE = plot traces
RAW	logical, TRUE=do not convert data to volts

**Details**

Segy format files are in integer format. The time series ususally represents counts recorded in a data acquisition system. The header includes meta-data and other identifying information.

**Value**

List of seismic traces.

**Note**

The Endian-ness of the input files is set by the system that created them. If the read1segy or read1sac does not make sense, try a different endian or BIGLONG setting.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

read1sac, read1segy, sac2rseis, prepSEIS

**Examples**

```
##### make some SAC files, then read them in
data(GH)
apath = tempdir()
## setwd(apath)
## apath = 'TEMP'
J = rseis2sac(GH, sel =1:5, path = apath, BIGLONG =FALSE )
Iendian = .Platform$endian
##### next read them in
Lname <- list.files(path=J , pattern='SAC', full.names=TRUE)

H = sac2rseis(Lname , Iendian =Iendian , HEADONLY = FALSE,
BIGLONG = FALSE, PLOT = -1, RAW = FALSE)

#### should have 5 traces, look at elements of the first one:
names(H[[1]])

plotGH(H[[1]])
```

---

**SEIS2list***Convert a SEIS list to a list of seismograms*

---

**Description**

Convert a SEIS list to a list of seismograms each independent.

**Usage**

```
SEIS2list(GH)
```

**Arguments**

GH                    SEIS list (swig input)

**Details**

The list returned is useful for editing or modifying the seismic data prior to swig.

**Value**

List of seismograms.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

plotGH, swig

**Examples**

```
data(GH)
gg = SEIS2list(GH)
## for(i in 1:length(gg) )
i = 1

{
plotGH(gg[[i]]); Sys.sleep(0.2)
}
```

---

seiscols

*Set colors for seismic display*

---

**Description**

Given an RSEIS list of seismic data return a set of colors associated with the structure that colors each trace and its components the same color.

**Usage**

```
seiscols(GH, acols="black", M="STNS")
```

**Arguments**

GH	Seismic RSEIS list
acols	vector of colors to choose from
M	character, "STNS" = stations, "COMPS" = components

**Value**

colors	alpha/numeric vector of colors
--------	--------------------------------

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>



**Examples**

```

data(GH)

GH$pcol <- seiscols(GH)
swig(GH, sel=which(GH$COMPS=="V"), WIN=c(3, 10), SHOWONLY=TRUE)

xcol <- seiscols(GH, acols=c("black", "darkmagenta", "forestgreen") )

GH$pcol <- xcol

swig(GH, sel=which(GH$COMPS=="V"), , SHOWONLY=TRUE)

```

---

SEISNtime	<i>Minimum time in an RSEIS list</i>
-----------	--------------------------------------

---

**Description**

Return date/time of trace with earliest date/time.

**Usage**

```
SEISNtime(GH)
```

**Arguments**

GH	RSEIS seismic list
----	--------------------

**Value**

yr	year
jd	julian day
hr	hour
mi	minute
sec	second
w1	which one, index to GH

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```

data(GH)
SEISNtime(GH)

```

---

seisorder	<i>Order seismic traces</i>
-----------	-----------------------------

---

**Description**

Use RSEIS structure to get convenient ordering of seismic data

**Usage**

```
seisorder(GH, ORD, VNE = c("V", "N", "E"))
```

**Arguments**

GH	RSEIS list
ORD	predetermined ordering, list(name, dist)
VNE	Order, for components, default=c("V", "N", "E")

**Details**

Uses information about the location of the stations to determine appropriate order. Order can be determined from the location of the stations, or from the travel times.

**Value**

Vector of indices of GH in correct order

**Note**

If ORD is provided from travel times, it uses this instead

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

JGET.seis

**Examples**

```
data(GH)
staf <- GH$staf

##### get the distances from the source to the stations
d1 <- GreatDist(GH$pickfile$LOC$lon, GH$pickfile$LOC$lat,
               staf$lon, staf$lat)

### staf has the names of the stations already, so insert the order via
```

```
###                                dist
staf$dist <- d1$dkm

sorder <- seisorder(GH, staf, VNE= c("V", "N", "E"))

if(interactive()){
  swig(GH, sel=sorder)
}
```

---

selAPX

*Select Picks*

---

### Description

select a subset of picks from a larger data base

### Usage

```
selAPX(APX, ista = NULL, icomp = c("V", "N", "E"))
selWPX(APX, ind=NULL, ista = NULL, icomp = c("V", "N", "E"))
```

### Arguments

APX	Pick Data Frame
ista	vector of stations to select
icomp	vector of components
ind	index of picks to select (negative values imply omission)

### Value

returns subset list

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

---

SELBUT

*Select Buttons*

---

### Description

Select buttons interactively.

### Usage

```
SELBUT(OPTS, onoff = 1, ocols = "white", default = "opt")
```

### Arguments

OPTS	character list of buttons
onoff	which buttons are active
ocols	colors for plotting
default	default list of buttons

### Details

Used in swig. Options can be added, subtracted, deleted, or completely filled out based on interactive choice.

### Value

character list of chosen options.

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

swig

### Examples

```
if(interactive()){
  STDLAB <- c("DONE", "QUIT", "zoom.out", "zoom.in", "SELBUT",
    "FILT", "UNFILT", "PSEL", "SGRAM", "WLET", "SPEC", "XTR" )
  onoff = rep(0, length(STDLAB))
  onoff[1:5] <- 1
  SELBUT(STDLAB, onoff=onoff)
}
```

---

selpgen	<i>Pick stations and components interactively</i>
---------	---

---

**Description**

Pick stations and components interactively. This is a routine used in swig.

**Usage**

```
selpgen(MH, newdev = TRUE, STAY = FALSE)
```

**Arguments**

MH	RSEIS list
newdev	logical, whether to create a new device.
STAY	logical, whether to keep device active.

**Value**

vector of index to list of stations and components

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

---

SELSTA	<i>Pick stations and components interactively</i>
--------	---

---

**Description**

Pick stations and components interactively. This is a routine used in swig.

**Usage**

```
SELSTA(GH, sel=1, newdev = TRUE, STAY = FALSE)
```

**Arguments**

GH	RSEIS list
sel	vector of index to selected traces
newdev	logical, whether to create a new device.
STAY	logical, whether to keep device active.

**Value**

vector of index to list of stations and components

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

**Examples**

```
data(GH)
```

```
SELSTA(GH, sel=1:7 , newdev = TRUE, STAY = FALSE)
```

---

selstas

*Select Stations*

---

**Description**

Extract a set of stations from a longer station file.

**Usage**

```
selstas(sta, ind)
```

**Arguments**

sta                    station list (name, lat, lon, z)

ind                    index to station list = positive is select, negative is remove

**Value**

station list with those indeces either removed or save.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

---

SENSORsensitivity      *Sensor Sensitivity from a known set of seismo/acoustic sensor*

---

**Description**

From published sensitivities of seismic and acoustic sensors.

**Usage**

SENSORsensitivity(K = 1)

**Arguments**

K                      number of sensor from list

**Value**

Sensitivity

**Note**

Current choices are: c("40T", "3T", "L28", "LD", "EL", "MC", "EL(SANGAY)")

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**References**

Johnson, J.B., R.C. Aster, M.C. Ruiz, S.D. Malone, P.J. McChesney, J.M. Lees, and P.R. Kyle, Interpretation and utility of infrasonic records from erupting volcanoes, *J. Volc. Geoth. Res.*, 121 (1-2), 15-63, 2003.

**Examples**

```
SENSORsensitivity(3)
SENSORsensitivity(5)
```

---

setPrePix                      *Set list of arrival times for swig.*

---

**Description**

Prepare a set of arrival picks for swig plotting.

**Usage**

```
setPrePix(R1, tt, name, flag = "K", col = "blue")
```

**Arguments**

R1	Location and time of event source. (list)
tt	Vector of travel times, seconds.
name	Station names
flag	Phase Identifier, character
col	Color

**Value**

List of picks suitable for swig plotting.

**Note**

R1 should have yr, jp, hr, mi, sec at the least.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

setWPX

**Examples**

```
T1 = as.POSIXct("2020-08-20 06:30:17.15 UTC", "UTC")
R1 = posix2RSEIS(T1)

name = c("MERT", "KRN", "KUA")
tt = c(1,2,3)
wpx = setPrePix(R1, tt, name, flag = "K", col = "blue")
```



---

setstas	<i>Set Station information</i>
---------	--------------------------------

---

**Description**

Read station information and set in list

**Usage**

```
setstas(stafile)
```

**Arguments**

stafile            character, station file name path

**Details**

reads in ASCII data file.

**Value**

LIST

name	character, station name
lat	numeric, decimal degrees
lon	numeric, decimal degrees
z	numeric, decimal degrees

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data(GH)

tsta = GH$stafile

tfile = tempfile()

write.table(file=tfile, tsta, row.names=FALSE, col.names=FALSE )

sta <- setstas(tfile)
```

---

setupDB	<i>Set up a seismic data base</i>
---------	-----------------------------------

---

**Description**

Set up a data base storing the location and times for a set of seismic data.

**Usage**

```
setupDB(DB, token = TRUE, split = "\\.")
```

**Arguments**

DB	<b>fn</b> full path to file <b>yr</b> year <b>jd</b> julian day <b>hr</b> hour <b>mi</b> minute <b>sec</b> second <b>dur</b> duration, seconds <b>origyr</b> origin time for epoch calculations
token	logical, use tokens in the file names of the fn's to extract station and component names for selection. default=TRUE
split	character string to split if using token, default is a period.

**Details**

If token is FALSE, then the station name and component are selected using substr, i.e. by column number.

**Value**

DB with epoch time and station information appended,

t1	epoch start time
t2	epoch end time = t1+nsamps*sample rate n seconds
sta	station
comp	component

**Note**

Program attaches station identification used for grepping.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

EPOCHday, T12.pix, Mine.seis

**Examples**

```
##### to illustrate, we make a set of individual seismograms
data(GH)
L1 = length(GH$JSTR)
DD = data.frame(GH$info)

GIVE = vector(mode='list')

for(i in 1:L1)
{
  AA = DD[i,]
  GIVE[[i]] = list(fn = AA$fn, sta =GH$STNS[i] , comp = GH$COMP[i],
                 dt = AA$dt, DATTIM = AA, N = AA$n1, units = NA,
                 coords = NA, amp = GH$JSTR[[i]] )
}

##### save the seismic data in a temporary directory
#### each trace in a separate file
tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM)
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)
DB = FmakeDB(LF, kind=-1)
## IDB = infoDB(DB)

plotDB(DB)
```

setwelch

*Set up Matrix of fft for Welch method***Description**

Prepares a matrix for estimation of power spectrum via Welch's method. Also, is can be used for spectrogram.

**Usage**

```
setwelch(X, win = min(80, floor(length(X)/10)),  
inc = min(24, floor(length(X)/30)), coef = 64, wintaper=0.05)
```

**Arguments**

X	Time series vector
win	window length
inc	increment
coef	coefficient for fft
wintaper	percent taper window taper

**Value**

List:

values	Matrix of fft's staggered along the trace
window size	window length used
increment	increment used
wintaper	percent taper window taper

**Author(s)**

originally written by Andreas Weingessel, modified Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Welch, P.D. (1967) The use of Fast Fourier Transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms IEEE Trans. Audio Electroacoustics 15, 70-73.

**See Also**

stft

**Examples**

```
dt <- 0.001  
  
t <- seq(0, 6, by=dt)  
x <- 6*sin(2*pi*50*t) + 10* sin(2*pi*120*t)  
y <- x + rnorm(length(x), mean=0, sd=10)  
  
plot(t,y, type='l')  
  
title('sin(2*pi*50*t) + sin(2*pi*120*t)+ rnorm')
```

```

Y <- fft(y)
Pyy <- Y * Conj(Y)
N <- length(y)
n <- length(Pyy)/2
Syy <- (Mod(Pyy[1:n])^2)/N
fn <- 1/(2*dt)

f <- (0:(length(Syy)-1))*fn/length(Syy)
plot(f, Syy, type='l', log='y', xlim=c(0, 150));
abline(v=c(50, 120),col='blue', lty=2)

plot(f, Syy, type='l', log='y', xlim=c(0, 150));
abline(v=c(50, 120),col='blue', lty=2)

win <- 1024
inc <- min(24, floor(length(y)/30))
coef <- 2048

w <- setwelch(y, win=win, inc=inc, coef=coef, wintaper=0.2)

KK <- apply(w$values, 2, FUN="mean")

fw <- seq(from=0, to=0.5, length=coef)/(dt)
plot(fw, KK^2, log='', type='l', xlim=c(0, 150)) ;
abline(v=c(50, 120), col='blue', lty=2)

Wyy <- (KK^2)/w$windowsize
plot(f, Syy, type='l', log='y', xlim=c(0, 150))
lines(fw,Wyy , col='red')

DBSYY <- 20*log10(Syy/max(Syy))
DBKK <- 20*log10(Wyy/max(Wyy))

plot(f, DBSYY, type='l', xlim=c(0, 150), ylab="Db", xlab="Hz")

lines(fw, DBKK, col='red')
title("Compare simple periodogram with Welch's Method")

```

---

`setwpix`*Set Window Pix for swig*

---

**Description**

Create list of windows picks suitable for plotting in swig.

**Usage**

```
setwpix(phase = NULL, col = NULL, yr = NULL, jd = NULL,  
hr = NULL, mi = NULL, sec = NULL, dur = NULL, name = NULL,  
comp = NULL, dispcomp = NULL)
```

**Arguments**

phase	phase name
col	color for plotting
yr	year
jd	julian day
hr	hour
mi	minute
sec	second
dur	duration
name	name of station
comp	component
dispcomp	display on which component

**Details**

Some phases should be displayed on only certain components of a station.

**Value**

list of window picks

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

**Examples**

```

data(KH)

orgtim <- c( 2005,214,7,1,10.7313152551651 )
tims <- c( 0,46.7119,102.438451,113.092049,123.54077 )
psecs <- NULL
nam <- NULL

aphases <- NULL
sta <- "9024"

for(j in 1:length(tims))
{
psecs <- c(psecs, tims[j]+orgtim[5])
nam <- c(nam, sta)
aphases <- c(aphases, paste(sep="", "K", j) )
}

pp <- setwpix(phase=aphases , col="blue", yr=orgtim[1], jd=orgtim[2],
hr=orgtim[3], mi=orgtim[4], sec=psecs, dur=0, name=nam , comp="V")

W <- secdifL(KH$info, pp)

win <- c(min(W)-5, max(W)+5 )
swig(KH, APIX=pp, WIN=win , SHOWONLY=TRUE)

```

---

setWPX

*Set WPX*


---

**Description**

Create a WPX list from vector input or relevant parameters.

**Usage**

```

setWPX(phase = NULL, col = NULL, yr = NULL, jd = NULL,
hr = NULL, mi = NULL, sec = NULL, dur = NULL, name = NULL,
comp = NULL, dispcomp = NULL, onoff = NULL)

```

**Arguments**

phase            character, phase names

col	character, colors
yr	numeric, year
jd	numeric, julian day
hr	numeric, hour
mi	numeric, minute
sec	numeric, second
dur	numeric, duration(s)
name	character, station name
comp	character, component
dispcomp	character, display string
onoff	numeric, flag for turning pick on or off

**Details**

Utility for setting up a WPX list for further processing.

**Value**

WPX list.

**Note**

Used internally.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

addWPX, catWPX, checkWPX, cleanWPX, clusterWPX, repairWPX, saveWPX

**Examples**

```
s1 <- setWPX(name="HI", yr=2011, jd=231, hr=4, mi=3, sec = runif(5))
```



---

setypx                      *Create an empty window pick list*

---

**Description**

Create an empty window pick list. This is used primarily internally.

**Usage**

setypx()

**Value**

List:

tag	tag for identification of station and component
name	station name
comp	component name
c3	component name with secondary tags
phase	phase
err	error
pol	polarity
flg	flag
res	residual
dur	duration
yr	year
mo	month
dom	day of month
jd	julian day
hr	hour
mi	minute
sec	second
col	color
onoff	logical, ON or OFF for plotting

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

setwpix

**Examples**

```
a <- setypx()
print(a)
```

---

showdatetime	<i>Print Date/TIME</i>
--------------	------------------------

---

**Description**

Print Date and Time as yyyy-mo-do hr:mi:se msec

**Usage**

```
showdatetime(rd, AMPM = FALSE, verbose=TRUE)
```

**Arguments**

rd	date time list, jd hr mi sec yr
AMPM	24 hour time (AMPM=FALSE) or 12 hour clock (AMPM=TRUE)
verbose	logical, print information to screen, default=TRUE

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
hrs <- seq(from=-36, to=36, by=2)
rd <- recdate(jd=1, hr=hrs, mi=34,
             sec=23+runif(n=length(hrs), 0, 59) , yr=2009)
showdatetime(rd)
showdatetime(rd, AMPM=TRUE)
```

---

sigconv	<i>Convolve spikes with wavelets</i>
---------	--------------------------------------

---

**Description**

Convolve spikes with wavelets

**Usage**

```
sigconv(wigmat, wavepulse)
```

**Arguments**

wigmat	matrix, spikes
wavepulse	wavelet for convolution

**Details**

Convolution is done in Frequency domain on each trace

**Value**

Matrix, waveforms

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

wiggleimage, symshot1, genrick

**Examples**

```
S1 <- symshot1()

##### S1$THEORY$treflex

d <- dim(S1$smograms)
G1 <- matrix( rep(0, length=d[1]*d[2]), ncol=d[2], nrow=d[1])

##### set up the spike set for reflexions
for(i in 1:3){
  p <- round( S1$THEORY$treflex[i,]/S1$dt );

  G1[cbind(p , 1:d[2]) ] <- 1
}
```

```
#### plot the spikes
wiggimage(0.1*G1, dt = -S1$dt, dx = S1$x, col = "black")

### make a ricker wavelet
wavelet <- genrick(25,S1$dt,35)
klem <- 11
###
nwave <- RPMG::RESCALE(wavelet, 0, 1, wavelet[1], max(wavelet))

##### convolve the wavelet with the set of spikes
H1 <- sigconv(G1, nwave)

##### plot
wiggimage(0.1*H1, dt = -S1$dt, dx = S1$x, col = "black")
```

---

sigconvGR

*convolve for Ground roll*


---

### Description

convolve a set of spikes for extended ground roll. This is a special case of sigconv.

### Usage

```
sigconvGR(wigmat, wavepulse, dt)
```

### Arguments

wigmat	matrix of traces with spikes
wavepulse	wavelet
dt	sampling interval

### Details

This is similar to the sigconv program but it assumes that the ground roll is extended in time and space as the wave expands.

### Value

Matrix, waveforms

### Note

the program spreads the sinusoidal wavelet along a band to simulate ground-roll head wave noise.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

wiggleimage, symshot1, genrick, sigconv

**Examples**

```
S1 <- symshot1()
dt <- S1$dt
##### these are the reflections S1$GRrec

d <- dim(S1$smograms)
G1 <- matrix( rep(0, length=d[1]*d[2]), ncol=d[2], nrow=d[1])

### these are the refractions S1$THEORY$trefrac
p <- round( S1$THEORY$trefrac[1,]/S1$dt );
G1[cbind(p , 1:d[2]) ] <- 1

#### plot the spikes
wiggleimage(0.1*G1, dt = -S1$dt, dx = S1$x, col = "black")

grlen <- floor(.6/dt)
fgr <- 10
tape <- applytaper( rep(1, grlen), p = 0.2)
tgr <- seq(from=0, by=dt, length=grlen)
siggr <- tape*sin(2*pi*fgr*tgr)

##### convolve the wavelet with the set of spikes
H1 <- sigconvGR(G1, siggr, dt)

##### plot
wiggleimage(0.1*H1, dt = -S1$dt, dx = S1$x, col = "black")
```

---

SNET.drive

*stereonet representation of particle motion*

---

**Description**

stereonet representation of particle motion

**Usage**

```
SNET.drive(intempmat, pmolabs = c("Vertical", "North", "East"), STAMP = "")
```

**Arguments**

intempmat	matrix of 3-component seismogram
pmolabs	labels for components
STAMP	Identification stamp

**Details**

Interactive driver for partmotnet.

**Value**

Graphical Side effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

partmotnet

**Examples**

```
data("GH")  
  
temp <- cbind(GH$JSTR[[1]], GH$JSTR[[2]], GH$JSTR[[3]])  
  
atemp <- temp[1168:1500, ]  
SNET.drive(atemp, pmolabs = c("Vertical", "North", "East"), STAMP = "")
```

---

SPECT.drive

*Interactive Spectrogram Driver*

---

**Description**

Interactive Spectrogram Driver

**Usage**

```
SPECT.drive(Xamp, DT = 0.008, NEW = TRUE, STAMP = NULL ,  
  freqlim=c(0, 20, 0, 20), winparams=c(4096,256, 204 ))
```

**Arguments**

Xamp	signal trace
DT	deltaT sample interval, s
NEW	logical, TRUE=recalculate spectrum
STAMP	character stamp for identification
freqlim	vector of 4 frequency limits: min max for calculations, min max for display. Default=see below
winparams	vector of 3 window parameters: Number of points for FFT, number of time samples for window, number of overlap samples: default=see below

**Details**

Interactive buttons are set internally. The parameters freqlim and winparams can be changed - these are simply the starting parameters for the initial display.

For winparams, the parameters are set to be appropriate for sample rates of typical seismic data, 100-125 samples per second. The number of points in the FFT are initially set to 4096 and the time window is set to 256. The overlap is calculated by subtracting 20 percent of the time window, so the overlap is 80 percent. Of course, the number of samples in a window must be less than the length of input time series.

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

plotevol, RPMG

**Examples**

```
data(CE1)

##### Xamp = CE1$y[CE1$x>5.443754 & CE1$x<5.615951]

Xamp = CE1$y
plot(Xamp, type='l')

DT = CE1$dt

if(interactive() ) {
  SPECT.drive(Xamp, DT = DT, NEW = TRUE, STAMP = NULL) }
```

Spectrum

*Calculate Different Spectrum Types in Physical Units***Description**

Spectrum is a wrapper function for `stats::fft` and `RSEIS::mtapspec`. For a given method (multi-taper spectrum or fft spectrum) and spectrum type (power, energy, amplitude, or phase), it returns the spectrum in physical units (obeying Parseval's theorem) and the corresponding frequency axis.

**Usage**

```
Spectrum(x, dt, one_sided = TRUE, type = 1, method = 1)
```

**Arguments**

<code>x</code>	Time series for which a spectrum is to be calculated (assumed to be in volts)
<code>dt</code>	Sample interval for <code>x</code> (assumed to be in seconds)
<code>one_sided</code>	Logical: should the spectrum be a function of positive frequencies only ( $f < \text{nyquist frequency}$ ) and spectral density doubled to be consistent with that (TRUE, default), or should the spectrum be provided for all frequencies, positive and negative?
<code>type</code>	Type of spectrum: 1 (default) is power spectrum; 2 is energy spectrum; 3 is amplitude spectrum; 4 is phase spectrum
<code>method</code>	Method used to calculate spectrum. 1 (default) is fft; 2 is multi-taper.

**Details**

Phase spectrum is currently enabled only for `method = 1` (fft). All possible energy and power spectra obey Parseval's relation ( $\sum(s) \cdot df \approx \text{mean}(x^2)$  for power;  $\sum(s) \cdot df \approx \sum(x^2) \cdot dt$  for energy). Parseval's relation may not be exact due to approximations used in making the spectrum one-sided or in the multi-taper method.

Input units are assumed to be volts and seconds; if other input units are used, adjust output units accordingly.

**Value**

List with following elements.

<code>f</code>	frequency axis (Hz; cycles per second, not radians per second)
<code>df</code>	interval for frequency axis (Hz)
<code>spectrum</code>	spectral values corresponding to <code>f</code>
<code>type</code>	spectrum type: Power, Energy, Amplitude, or Phase
<code>units</code>	Units of spectrum (assuming that input units are volts and seconds)



**Author(s)**

Jake Anderson

**See Also**

RSEIS::mtapspec stats::fft

**Examples**

```
## example time series
x = rnorm(1000)
dt = 0.01

## power spectrum, multi-taper method, one-sided
S = Spectrum(x, dt, type = 1, method = 2, one_sided = TRUE)
sum(S$spectrum) * S$df ## frequency-domain power
mean(x^2) ## time-domain power

## energy spectrum, fft method, two-sided
S = Spectrum(x, dt, type = 2, method = 1, one_sided = FALSE)
sum(S$spectrum) * S$df ## frequency-domain energy
sum(x^2) * dt ## time-domain energy
```

---

STALTA

*Short term, long term average ratio*

---

**Description**

Calculate the short term, long term average ratios of the squared amplitude in a time series.

**Usage**

```
STALTA(y, fwlen = 125, bwlen = 125)
```

**Arguments**

y	vector, or time series
fwlen	forward number of samples
bwlen	backward number of samples

**Details**

Calculates the ratio of the forward/backard mean square sum.

**Value**

vector of ratios

**Note**

All filtering or pre and post analysis should be done outside of ratio curve estimate.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

STLTcurve, PSTLTcurve

**Examples**

```
### easy example find P and S-wave arrivals, low noise
data(GH)
i = 6
z = GH$JSTR[[i]]

z.curve = STALTA(z, fwlen = 10, bwlen = 325)

ex = seq(from=0, length=length(z), by=GH$dt[i])
par(mfrow=c(2, 1) )
plot(ex, z, type='l')
plot(ex, z.curve, type = 'l' )

aa = peaks(z.curve, span = 11, do.pad = TRUE)
wa = which( aa & z.curve>50 )

abline(v=wa*GH$dt[i] , col='red')
par(mfg=c(1,1) )
abline(v=wa*GH$dt[i] , col='red')
```

---

STLTcurve

*Short-term/Long-term Average curve*

---

**Description**

Get short-term average long-term verage ratio curve for picking

**Usage**

```
STLTcurve(y, dt = 0.008, fwlen = 125, bwlen = 125,
stretch = 1000, MED = 255, PLOT = FALSE)
```

**Arguments**

y	signal
dt	sample rate
fwlen	forward window, number of samples
bwlen	back window length, number of samples
stretch	stretch multiplier
MED	median smoother
PLOT	logical, TRUE=plot diagnostics

**Details**

Uses C-code and fast tanking algorithm written at UW

**Value**

sample to significant change in ratio curve

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PSTLTcurve

**Examples**

```
data(CE1)

y = CE1$y

DT = CE1$dt

sy = STLTcurve(y, dt=DT, fwlen = 25, bwlen = 25,
stretch=1000, MED=255, PLOT=FALSE)

par(mfrow=c(2,1))

plot(CE1$x, CE1$y, type='l')
plot(CE1$x, sy$rat, type='l')
```

swig

*Seismic Wiggle Analysis***Description**

Main Interactive Program for plotting and analyzing seismic waveform data.

**Usage**

```
swig(GH, sel = 1:length(GH$dt), ORD = NULL, WIN = NULL, APIX = NULL,
PHASE = NULL,
STDLAB = NULL, PADDLAB = NULL, TEMPBUT=NULL,
SHOWONLY = FALSE, CHOP = FALSE, TIT = "",
pts = FALSE, forcepix = FALSE, pcex=0.7, SCALE = 1, ilocstyle=1,
velfile = "", stafile = "", LOC = NULL,
prefilt=list(fl=0.2, fh=15, type="HP", proto="BU"), filters=NULL,
YAX = 1 , xtickfactor = 1, vertline=NA, destdir='.')
```

**Arguments**

GH	Seismic data structure
sel	selection of traces from structure
ORD	order to plot traces
WIN	vector c(t1, t2) for window of traces to be shown
APIX	structure of arrival time picks
PHASE	phase to display, "P", "S", etc
STDLAB	label of buttons
PADDLAB	label of phase-pick buttons
TEMPBUT	temporary, user defined buttons
SHOWONLY	logical, TRUE=non-interactive
CHOP	whether to chop the signal
TIT	title for the top of plot
pts	whether to plot specific points on the plot
forcepix	logical, force all phase picks to be shown on all traces
pcex	Pick label size expansion (cex), default=0.7
SCALE	flag, 1,2= scale according to window or trace (default=1, scale by trace)
ilocstyle	integer, style of click graphic, one of -1, 0, 1, 2, 3, indicating: points, abline, segs, segs+abline, segs+long-abline , default=1
velfile	velocity structure or file name
stafile	station structure or file name
LOC	source location structure (lat, lon, depth)

prefilt	default filter definition list(fl=.2, fh=15, type="HP", proto="BU")
filters	a list of filters for choosfilt, the list consists of 3 vectors: flo, fhi and type defining the filter choices.
YAX	type of Yaxis label, 1,2,3 DEFAULT=1 only one y-axis others scaled; 2=all y-axes are plotted on left; 3=all y-axes plotted, alternating left and right
xtickfactor	Factor for multiplying the x-axis tick markers (default=1; for minutes=60, hrs=3600, days=24*3600)
vertline	time list (yr, jd, hr, mi sec) for plotting vertical lines on window. Default=NA
destdir	Destination directory(folder) for writing output to disk, default = current directory

### Details

This is the main program that drives the other analysis in RSEIS. GH is a list consisting of header (meta-data) and time series information. See documentation on GH to get complete description.

A set of filters can be defined by the user, see choosfilt

Default Buttons, can be created by: STDLAB = c("DONE", "QUIT", "zoom out", "zoom in", "Left", "Right", "restore", "Pinfo", "WINFO", "XTR", "SPEC", "SGRAM", "WLET", "FILT", "UNFILT", "SCALE", "Postscript")

If the user has defined STDLAB.DEFAULT and PADDLAB.DEFAULT in the .Rprofile or .First commands, these will override the default in the function definition.

### Value

Various structures are returned based on interactive selections of the user.

However, the default return list:

but	last button pushed
sloc	location of last set of clicks
WPX	set of saved WPIX (window picks
BRUNINFO	Brune Model information
DETLINFO	Detailed information about traces
mark	mark (MARK button was pressed
PUSHED	list of all buttons pressed prior to exit

### Note

If using the filters for button FILT, it is useful to have a "None" in case no filter is desired (i.e. user changes mind).

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PICK.DOC, GH, RPGM, choosfilt

**Examples**

```

data("GH")
### This loads a structure

STDLAB <- c("DONE", "QUIT", "zoom out", "zoom in", "Left",
"Right", "restore", "Pinfo", "WINFO",
"XTR", "SPEC", "SGRAM", "WLET",
"FILT", "UNFILT", "SCALE", "Postscript")

sel <- GH$COMPS=="V"
if(interactive() ) { p <- swig(GH, sel=sel, STDLAB=STDLAB)
print(p)
}
if(interactive()) {
p <- swig(GH, sel=sel, WIN=c(4,14) , STDLAB=c("DONE", "LAME", "DAME") )

print(p)
}

##### example with filter
data(KH)

thefilts <-
list(flo=
c(0.02, 0.02, 0.02, 0.02, 0.02, 0.02,
0.02, 0.02, 0.02, 0.02, 0.02, 0.02,
0.02,
1/2, 1/50,1/100, 1/100,
1/100,1/100,1/100,1,1,
0.2, 15, 5, 2,1,
100),
fhi=
c(1/10, 1/6, 1/5, 1/4, 1/3, 1/2,
0.2, 0.5, 1.0, 2.0, 3.0, 4.0,
7.0,
8, 1/2.0,1/5.0,1/10.0,
1/20, 1/30,1/40,10,5,
7.0, 100, 100, 100,10,
100),
type =
c("LP", "LP", "LP", "LP", "LP", "LP",
"LP", "LP", "LP", "LP", "LP", "LP",
"LP",
"BP", "BP", "BP", "BP", "BP", "BP",
"BP", "BP", "BP",
"HP", "HP", "HP", "HP", "HP",
"None"))

```

```
if(interactive()) {  
  swig(KH, filters=thefilts)  
}else{  
  swig(KH, filters=thefilts, SHOWONLY=TRUE )  
}
```

---

swig.ALLPX

*plot all phase arrival picks*

---

### Description

plot all phase arrival picks

### Usage

```
swig.ALLPX(t0, STNS, COMPS, YPX, PHASE = NULL, POLS = TRUE,  
  FILL = FALSE, FORCE = TRUE, cex = cex, srt = srt)
```

### Arguments

t0	time for start of window, s
STNS	station names to plot
COMPS	components to plot
YPX	y-picks (times)
PHASE	Phases to plot
POLS	polaritiy information (up, down)
FILL	fill color
FORCE	logical, force all phases plotted on all traces
cex	character expansion
srt	string rotation angle, degrees

### Details

for use in conjunction with PLOT.SEISN program

### Value

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

PLOT.SEISN, swig

**Examples**

```
##### this example needs some work:
data(GH)
WPX = uwpfile2ypx(GH$pickfile)

swig(GH, SHOWONLY=TRUE )

swig.ALLPX(GH$pickfile$LOC , GH$STNS, GH$COMPS, WPX, PHASE='P',
FORCE=TRUE)
```

---

symshot1

*Simulate a seismic shot*

---

**Description**

Simulate an exploration style seismic shot with ground roll, air wave, refractions and reflections.

**Usage**

```
symshot1(PLOT = FALSE)
```

**Arguments**

PLOT                   logical, TRUE=plot the wiggles. DEFAULT=FALSE

**Details**

Arrivals are calculated based on geometric considerations with a 1D layered model.

**Value**

smograms	Matrix: columns are individual traces
dt	sample interval in time, s
x	x locations
dx	spacing in X-direction



REFL	reflection information
REFR	refraction image
GRrec	ground roll image
AIRrec	air wave image
THEORY	List of theoretical values
trefrac	refraction arrival times
treflex	reflection arrival times
tair	Air arrival times
velair	velocity for the air wave
mod	Layered Model

**Note**

MOdel is relatively simple:

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**References**

Sherrif

**See Also**

wiggleimage, symshot

**Examples**

```
S1 <- symshot1()
wiggleimage(S1$smograms, dt = -S1$dt, dx = S1$x, col = "black")
```

---

sysinfo

*System Information*

---

**Description**

Extract OS system information

**Usage**

```
sysinfo()
```

**Details**

Returns parts of the output of variables .Machine and .Platform.

Endian Problem

these should be used for reading binary data when crossing platforms. If binary files are created on a little-endian platform, but are being read on a big-endian platform, then one should use "swap".

SizeOf Problem

Many older machines use 4 bytes for LONG. Newer 64 bit machines use 8 bytes for LONG = so this is a big problem.

**Value**

A=.Machine, B=.Platform

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

.Machine, .Platform

**Examples**

```
sysinfo()
```

---

T12.pix

*Get T1, T2*

---

**Description**

Modify opick data frame and add T2=T1+dur

**Usage**

```
T12.pix(A)
```

**Arguments**

A                   pick data.frame

**Details**

Given t1 and duration, returns to structure, t2=t1+dur.

**Value**

pick data.frame with t2 as a member.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

TAPER.SEISN

*Taper Traces*

---

**Description**

Taper traces in a seismic structure using a cosine function on the ends.

**Usage**

```
TAPER.SEISN(TH, sel = 1:length(TH$JSTR), TAPER = 0.1 )
```

**Arguments**

TH	Seismic structure
sel	selection of traces
TAPER	filter taper, percent cosine taper

**Details**

Seismic structure

**Value**

Seismic structure

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

butfilt

**Examples**

```
data("GH")
sel <- which(GH$COMPS=="V")

sel <- 1:3
KF <- TAPER.SEISN(GH, sel = sel, TAPER=0.1)
swig(KF, sel=sel, SHOWONLY=0)
```

---

Thresh.J

*Threshold Adjuster*

---

**Description**

determine cut off for ratio curve

**Usage**

Thresh.J(y, thresh)

**Arguments**

y	signal
thresh	inital threshold

**Details**

Attempts to automatically optimize the threshold for automated picking. Used deep in picking algorithm.

**Value**

list(J=J, L=L)

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

TOCART                      *Convert to Cartesian coordinates*

---

**Description**

Convert to cartesian coordinates

**Usage**

TOCART(az, nadir)

**Arguments**

az	degrees, azimuth
nadir	degrees, dip

**Value**

LIST

x	x-coordinate
y	y-coordinate
z	z-coordinate
az	degrees, azimuth
nadir	degrees, dip

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**See Also**

tocartL

**Examples**

TOCART(132, 69)

tojul

*Julian Day*

---

**Description**

Convert to Julian Day. Used for calculations.

**Usage**

```
tojul(year, month, day)
```

**Arguments**

year	year
month	month
day	day

**Value**

Julian Days

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
tojul(1953, 3, 19)
```

---

tomo.colors

*Tomography Colors*

---

**Description**

Color Palette ranging from red to blue through black.

**Usage**

```
tomo.colors(n, alpha = 1)
```

**Arguments**

n	number of colors
alpha	hsv color parameter

**Value**

color palette

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

rainbow, colors, hsv

**Examples**

```
tomo.colors(25, alpha = 1)
```

---

trapz

*Integrate using trapezoidal rule*

---

**Description**

Integrate using trapezoidal rule

**Usage**

```
trapz(y, dt, rm.mean=TRUE)
```

**Arguments**

y	Input signal
dt	sample interval time, seconds
rm.mean	logical, whether to remove the mean prior to integration (TRUE)

**Value**

vector: Integrated signal

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
x <- rnorm(100)
trapz(x, 0.01)
```

---

 travel.time1D

*Seismic Travel Time 1D*


---

### Description

Travel time from source to receiver in 1D local model.

### Usage

```
travel.time1D(indelta, inhpz, instaz, inlay, ztop, vel)
many.time1D(indelta, inhpz, instaz, inlay, ztop, vel)
```

### Arguments

indelta	distance in KM
inhpz	depth of hypocenter, km
instaz	elevation of station
inlay	number of layers
ztop	vector, tops of layers
vel	vector, velocities in layers

### Details

Uses local 1D velocity model, not appropriate for spherical earth. The many.time1D version will take a vector of distances (indelta) and either one station elevation or a vector.

The station elevation should be referenced to the top of the velocity model, not necessarily sea level. Usually this is set to zero and a station correction is used to take into account the topographic and other geologic effects.

### Value

list:	
dt dr	derivative of t w.r.t. horizontal distance
dt dz	derivative of t w.r.t. z, depth
angle	incidence angle, degrees
tt	travel time, s

### Author(s)

Jonathan M. Lees<jonathan.lees.edu>



**See Also**

Ray.time1D, Get1Dvel

**Examples**

```

data(VELMOD1D)

v <- VELMOD1D

tees <- travel.time1D(23, 7, 0, length(v$zs) , v$zp , v$vp)

print(tees)

```

---

tung.pulse

*Volcanic Pulse Analysis*


---

**Description**

Given a series of pulses, do analysis on each one

**Usage**

```
tung.pulse(r, q, dt)
```

**Arguments**

r	x-coordinates
q	y-coordinates
dt	deltat, sample interval

**Details**

Calculates, min, max of edges and center, then models the pulse with a triangular pulse and integrates.

**Value**

vector=c(Ex[1], Ex[2], Ey[1], Ey[2], Cx, Cy, ar2, DefInt[1], DefInt[2], sum0) where:

Ex	left minimum
Ey	right minimum
Cx, Cy	center (max?)
ar2	area of triangle
DefInt[1]	integral under curve
DefInt[2]	integral under curve ( bottom triangle removed)
sum0	RMS amplitude

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

peaks

**Examples**

```
if(interactive()){
  data(CE1)

  ex <- CE1$x[CE1$x>5.453291 &CE1$x< 5.507338]
  why <- CE1$y[CE1$x>5.453291 &CE1$x< 5.507338]
  plot(ex, why, type='l')

  tung.pulse(ex, why, CE1$dt)

}
```

---

unpackAcard

*Parse Acard from UW-format pickfile*

---

**Description**

Parse Acard from UW-format pickfile

**Usage**

```
unpackAcard(AC)
```

**Arguments**

AC                   ascii acard

**Details**

Reads and Parses A-cards from UW foprformatted data.

**Value**

List:

yr	Year
mo	Month
dom	Day of Month
hr	Hour
mi	minute
sec	second
lat	latitude
lon	longitude
z	depth
mag	magnitude
gap	gap in station coverage
delta	distance to nearest station
rms	root mean square residual
hozerr	horizontal error

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

---

`uwpxfile2ypx`*UW pickfile to pphase pick data.frame*

---

**Description**

Read in ASCII version of pickfile. This is the output list used to plot picks on swig, often called WPX or YPX in other functions.

**Usage**`uwpxfile2ypx(P)`**Arguments**

P            pickfile

**Value**

list:

STAS	input structure
yr	year
mo	month
dom	day of month
jd	julian day
hr	hour
mi	minute
sec	second
col	color
onoff	logical, TRUE plot trace

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
data("GH")
WW = RSEIS::uwpfile2ypx(GH$pickfile)
vertord <- getvertsorder(GH$pickfile, GH)
R1 = rangedatetime(WW)
R2 = rangedatetime(GH$info)
S1 = secdifL(R2$min, R1$min)

swig(GH, sel=vertord$sel, APIX=WW, WIN=c(S1-1, 15) , SHOWONLY=0)
```

---

varsquig

*Var-Squiggle plot*

---

**Description**

Plot one seismogram in Var-Squiggle mode - like on an exploration record section with half the wiggled shaded.

**Usage**

```
varsquig(x, y, L = locator(2), FLIP = FALSE, filcol="blue",
tracecol="red", var = 0, xpd=TRUE )
```

**Arguments**

x	X (time axis) coordinates
y	Y amplitudes
L	rectangular region on plot where plotting occurs
FLIP	logical - whether to flip the amplitudes by -1
filcol	color for shading
tracecol	color for trace
var	logical, whether to shade
xpd	logical, set xpd parameter (see par)

**Details**

A set of traces can be plotted after the plotting region has been set.

**Value**

Graphical Side Effects

**Note**

varsquig is meant to be used within other program not as a stand alone routine. The plotting region must be set up prior to plotting. The time series is scaled to fit in the rectangular region defined by L.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

varsquiggle

**Examples**

```
data(KH)

x <- KH$ex[KH$ex>95& KH$ex<125]
y <- KH$JSTR[[1]][KH$ex>95& KH$ex<125]

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

u <- par('usr')
L <- list(x=c(u[1], u[2]), y = c(u[3], u[4]))

plot(L$x, L$y, type='n')
varsquig(x, y, L=L , FLIP=FALSE, filcol="blue", tracecol="blue", var=TRUE)
```

```
plot(L$x, L$y, type='n')
varsquig(x, y, L=L , FLIP=FALSE, filcol="red", tracecol="blue", var=FALSE)
```

---

varsquiggle

*Var-Squiggle Plot*


---

### Description

Plot A seismic section using Var-Squiggle, like an exploration seismic record.

### Usage

```
varsquiggle(GH, sel = c(1, 2), WIN = c(0, 1), dist=NULL, thick=1 ,
FLIP=FALSE, filcol='blue', tracecol='blue', xpd=TRUE, plotdir=1 )
```

### Arguments

GH	Seismic List
sel	selection of seismic traces
WIN	time window
dist	distance from the source
thick	thickness of plotting region per trace
FLIP	logical, whether to plot vertical or horizontal, default FALSE, TRUE = vertical
filcol	color for shading
tracecol	color for trace
xpd	logical, set xpd parameter (see par)
plotdir	1=left to right, 0=right to left (default=1)

### Details

Traces are plotted and scaled each with its own window. The distance vector provides the location on the seismic record.

### Value

Graphical Side effects

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

matsquiggle, varsquig

**Examples**

```
data(GH)
m <- match( GH$STNS,    GH$stafilename)
LATS <- GH$stafile$lat[m]
LONS <- GH$stafile$lon[m]
dees <- rdistaz( GH$pickfile$LOC$lat, GH$pickfile$LOC$lon, LATS, LONS)

sel <- which(GH$COMPS=="V")
sel <- sel[order(dees$dist[sel])]

### plot normal way:
swig(GH, sel=sel, WIN=c(5,10), SHOWONLY=TRUE)

### plot with varsquiggle
varsquiggle(GH, sel=sel, WIN=c(5,10))
```

---

 VELMOD1D

*Sample Velocity Model*


---

**Description**

Seismic Velocity Model for Coso California

**Usage**

```
data(VELMOD1D)
```

**Format**

LIST:

**zp** vector of Tops of Layers, P-wave, (km)  
**vp** vector of velocities of Layers, P-wave,(km/s)  
**ep** errors for velocities, P-wave,(km/s)  
**zs** vector of Tops of Layers, S-wave, (km)  
**vs** vector of velocities of Layers, S-wave,(km/s)  
**es** errors for velocities, S-wave,(km/s)  
**name** character, name of model  
**descriptor** character vector description of model

**Details**

Velocity model from a text file

**References**

Wu, H., and J. M. Lees (1999), Three-dimensional P- and S-wave velocity structures of the Coso Geothermal Area, California, from microseismic travelttime data, *J. Geophys. Res.* 104, 13,217-13,233.

**Examples**

```
data(VELMOD1D)
Get1Dvel(VELMOD1D, PLOT=TRUE)
```

---

VELOCITY.SEISN

*Velocity Seismogram*

---

**Description**

Removes seismic instrument response and corrects for sensitivity of seismoc instrument, returning units of m/s rather than volts.

**Usage**

```
VELOCITY.SEISN(TH, sel = 1:length(TH$JSTR), inst = 1,
Kal = Kal,waterlevel = 1e-08, FILT = list(ON = FALSE,
fl = 1/30, fh = 7, type = "HP", proto = "BU"))
```

**Arguments**

TH	list structure of seismic traces
sel	select which traces in list to deconvolve
inst	index to instrument in Kal list for calibration and instrument response
Kal	list of instrument responses
waterlevel	waterlevel for low frequency division
FILT	filter output, after instrumentation

**Details**

Instrument responses are lists of poles and zeros for each instrument defined.

**Value**

Same as input list with new traces representing velocity versus volts



**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

DISPLACE.SEISN, deconinst

**Examples**

```
Kal <- PreSet.Instr()
data(KH)

inst <- rep(0, length(KH$STNS))

VH <- VELOCITY.SEISN(KH, sel = 1, inst = 1,
Kal = Kal, FILT = list(ON = FALSE, f1 = 1/30, fh = 7,
type = "HP", proto = "BU"))
```

---

view.seis

*View seismic data window*

---

**Description**

View seismic data (segy) window on an hourly basis.

**Usage**

```
view.seis(aday, ihour, inhour, SAVEFILE, days,
DB, usta, acomp,
STDLAB =c("QUIT", "NEXT", "PREV", "HALF"),
kind = -1, Iendian=1, BIGLONG=FALSE,
TZ=NULL)
```

**Arguments**

aday	index of which day to use in vector days
ihour	hour to start
inhour	increment in hours for viewing panel
SAVEFILE	file to save window picks in
days	vector of days to select from
DB	data base list of file names and start-times and durations
usta	stations to select
acomp	components to select

STDLAB	vector of buttons, DEFAULT = c("QUIT", "NEXT", "PREV", "HALF", "WPIX", "zoom out", "refresh", "restore", "SPEC", "SGRAM", "WLET", "FILT", "Pinfo", "WINFO")
kind	an integer -1, 0, 1, 2 ; 0="RDATA" , -1="RDS", 0="RDATA", 1 = "segy", 2 = "sac", see notes below
Iendian	vector, Endian-ness of the data: 1,2,3: "little", "big", "swap". Default = 1 (little)
BIGLONG	logical, TRUE=long=8 bytes
TZ	Number of hours to add to GMT to get local time

### Details

The program view.seis assumes the data is stored in files accessible by the user and that the DB list has been scanned in and parsed.

"kind" can be numeric or character: options are 'RDS', 'RDATA', 'SEGY', 'SAC', corresponding to (-1, 0, 1, 2)

### Value

Graphical side effects and save.wpix stores appended picks.

### Note

On LINUX systems I wrote these (non-R) programs to set up the data base for segy data:FLS.prl, segydatabase. To get these contact me directly. TZ is (-6) for Guatemala.

### Author(s)

Jonathan M. Lees<jonathan.lees@unc.edu>

### See Also

swig, save.wpix

### Examples

```
if(interactive() ) {

data(KH)

amp = KH$JSTR[[1]]
OLDdt = KH$dt[1]
newdt = 0.1
yr = 2000
GIVE = FAKEDATA(amp, OLDdt=0.01, newdt = 0.1, yr = 2000,
                JD = 4, mi = 12, sec = 0, Ntraces = 24*3,
                seed=200, noise.est=c(1, 100) , verbose=TRUE )

tdir = tempdir()
for(i in 1:length(GIVE) )
```

```

{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)

DB = FmakeDB(LF, kind=-1)

IDB = infoDB(DB)

pday <- 5
SAVEFILE <- tempfile()
ihour <- 15
inkhour <- .5

### days is a list of days (and associated years) that are in teh DB
days <- list(jd=c(4, 5, 6), yr=c(2000, 2000, 2000) )
aday = which(pday == days$jd)

#### aday refers to one of the days listed in the days structure

view.seis(aday, ihour, inhour, SAVEFILE, days, DB, IDB$usta, IDB$ucomp, TZ=(-6))

}

```

---

vlen

*vector length*


---

### Description

calculate euclidian vector length

### Usage

```
vlen(A1)
```

### Arguments

A1                      vector

**Value**

Euclidian Length

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
vlen(c(23, 43))
```

---

vline                      *vertical line on trace panel*

---

**Description**

add vertical line on trace panel

**Usage**

```
vline(x, per = 1, COL = 1, NUM = FALSE, LAB = 1:length(x), lwd = 0, lty = 1)
```

**Arguments**

x	vector of x-locations
per	percent of window
COL	color
NUM	number lines
LAB	character labels
lwd	line width
lty	line type

**Details**

adds vertical lines to plot

**Value**

Graphical side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

plocator

**Examples**

```
plot(c(0,1), c(0,1), type='n')  
vline(runif(4), COL = 'red')
```

---

wiggly.env

*Plot time series envelope*

---

**Description**

Gets an envelope and lpots on a time series

**Usage**

```
wiggly.env(x, y)
```

**Arguments**

x	x-coordinate
y	y-coordinate

**Details**

Uses Peaks and smooth.pline to estimate envelope

**Value**

list structure from smooth.spline

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

smooth.spline, peaks, hilbert

**Examples**

```
## data("CE1.Example.RDATA")
## load("CE1.Example.RDATA")
data(CE1)
plot(CE1$x, CE1$y, type='l')
wiggimage(CE1$x, CE1$y)
we = wiggimage(CE1$x, CE1$y)
lines(we$x, we$y, col='red')
```

---

wiggimage

*Seismic section*

---

**Description**

Plot a seismic section as shot record

**Usage**

```
wiggimage(Arot, dt = 1, dx = 1, col = "black")
```

**Arguments**

Arot	Matrix: columns are individual traces
dt	Sample rate, seconds
dx	spacing in x-direction. If a vector is given, it is used instead and dx is taken from the difference of the first to elements.
col	color for plotting wiggles

**Details**

Plot is arranged with time going down the page

**Value**

Graphical side effects

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

matsquiggle, varsquiggle

**Examples**

```
S1 = symshot1()
wiggieimage(S1$smograms, dt = -S1$dt, dx = S1$x, col = "black")
```

---

**WINGH***Window a GH structure and extract data*

---

**Description**

Window a time slice of seismic data and extract from a GH structure.

**Usage**

```
WINGH(GH, sel = 1, WIN = c(0,1) )
```

**Arguments**

GH	RSEIS seismic list
sel	Select which traces to extract
WIN	Time window to extract (seconds from the beginning of the first trace.)

**Details**

Preserves the data structure of the GH list. The purpose of this function is to extract a small subset of data from a larger data set (or longer time series) for subsequent processing.

**Value**

New GH structure.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

**Examples**

```

if(interactive()){
data(GH)

swig(GH, sel=which(GH$COMPS=="V" ))

jh = WINGH(GH, sel = which(GH$COMPS=="V" ), WIN = c(3.821281, 12.861820) )

swig(jh)
## compare with:
swig(GH, sel=which(GH$COMPS=="V" ), WIN = c(3.821281, 12.861820))

}

```

winmark

*Window Mark***Description**

Add Mark up to current seismic trace with a bar designating a window selection.

**Usage**

```

winmark(a1, a2, side = 1, bar = NULL,
leg = NULL, col = col, lwd = 1, lty = 1,
arrows = FALSE, alen = 0.1, leglen = 0.15,
LEGON = 3, BARON = TRUE)

```

**Arguments**

a1	x1-location
a2	x2-location
side	side where bar is drawn, as in axes: 1=bottom,2=left,3=top,4=right
bar	location of bar
leg	location of leg
col	color
lwd	line width
lty	line type
arrows	logical, add arrows to ends of legs
alen	length of arrow heads, inches, default=0.125
leglen	length of arrows as percent of usr("par"), default=0.125
LEGON	plotting flag for legs: 0=no legs, 1=left leg, 2=right leg, 3=both legs(default)
BARON	logical:plotting flag for bar



**Details**

Used for marking seismic traces. The window marker looks like a staple, three segments are drawn, a bar and two legs. The thickness of the legs are determined by bar and leg, unless these are missing. if they are missing parameter side is used to set the locations, and leglen determines the length of the legs. If either bar or leg are missing the parameters are derived from par("usr") and are applied to whole window. side switches the orientation of the staple mark, with the legs pointing according away from named the axis.

**Value**

Graphical Side Effect

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**Examples**

```
plot(c(0,1), c(0,1), type='n', xlab='', ylab='' )

winmark(.3, .7,      side=3, col='brown', arrows=TRUE, leglen=.4)
winmark(.3, .7,      side=1, col='blue', arrows=TRUE, leglen=.5)

winmark(.3, .7,      side=2, col='green',
arrows=TRUE, alen=.05, leglen=.4)

winmark(.3, .7,      leg=.65, bar=.6,
side=4, col='orange', arrows=TRUE, alen=.1, leglen=.125)

winmark(.3, .7,      bar=.65, leg=.6,
side=4, col='seagreen', arrows=TRUE, alen=.1, leglen=.125)
##### examples with different legs showing
plot(c(0,1), c(0,1), type='n', xlab='', ylab='' )

winmark(.3, .7,      side=3, col='brown',
arrows=TRUE, leglen=.4, LEGON=1)
winmark(.3, .4,      side=1, col='brown',
arrows=TRUE, leglen=.4, LEGON=2)
winmark(.7, .9,      side=1, col='blue',
arrows=TRUE, leglen=.4, LEGON=0)
```

---

`winseis24`*Locator for plotseis24*

---

**Description**

Locator for plotseis24

**Usage**

```
winseis24(pjj, pch = 3, col = "red")
```

**Arguments**

<code>pjj</code>	out put of plotseis24
<code>pch</code>	plotting character when clicking
<code>col</code>	color for plotting when clicking

**Details**

After extracting 24 hours and plotting with plotseis24, use winseis24 to click on the plot and return times for further analysis or zooming.

**Value**

list:

<code>hr</code>	hours picked
<code>yr</code>	year
<code>jd</code>	julian day

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

plotseis24, getseis24

**Examples**

```
if(interactive()){
  data(KH)

  amp = KH$JSTR[[1]]
  OLDdt = KH$dt[1]
  newdt = 0.1
  yr = 2000
  GIVE = FAKEDATA(amp, OLDdt=0.01, newdt = 0.1, yr = 2000,
```

```

        JD = 4, mi = 12, sec = 0, Ntraces = 24*3,
seed=200, noise.est=c(1, 100) , verbose=TRUE )

tdir = tempdir()
for(i in 1:length(GIVE) )
{
  sig = GIVE[[i]]
  d1 = dateStamp(sig$DATTIM, sep='_')
  nam1 = paste(d1,sig$sta, sig$comp, sep='_')
  nam2 = paste0(nam1, '.RDS')
  nam3 = paste(tdir, nam2, sep='/')
  saveRDS(file=nam3, sig)
}

##### Now read files and make the DataBase:
LF = list.files(path=tdir, pattern='.RDS', full.names=TRUE)

DB = FmakeDB(LF, kind=-1)

IDB = infoDB(DB)

START = list(yr =yr , jd= 5 , hr= 0 , mi= 0 ,sec= 0)

END = list(yr =yr , jd= 7 , hr= 0 , mi= 0 ,sec= 0)

h = getseis24(DB, iyear = 2000, iday = 5, usta = IDB$usta,
              acomp = IDB$ucomp, kind = -1, Iendian=1, BIGLONG=FALSE)

pjj <- plotseis24(h, dy=1/18, FIX=24, SCALE=1,
                 FILT=list(ON=FALSE, fl=0.05 , fh=20.0, type="BP", proto="BU"),
                 RCOLS=c(rgb(0.2, .2, 1), rgb(.2, .2, .2)) )

##### here is the picking:
wpicks = winseis24(pjj)

}

```

---

wlet.do

*Return Wavelet transform*


---

## Description

Wavelet transform

## Usage

```
wlet.do(why, dt, noctave = 6, nvoice = 20, w0=5,
```

```
flip = TRUE, ploty = TRUE, zscale = 1,
col = terrain.colors(100), STAMP = STAMP, units="", scaleloc=c(0.4,0.95))
```

### Arguments

why	signal
dt	sample rate (s)
noctave	number of octaves, default=6
nvoice	number of voices, nvoice = 20
w0	central frequency for morlet wavelet, default=5
flip	logical, whether to flip the orientation
ploty	logical, whether to plot y
zscale	scale of the image
col	color palette
STAMP	character stamp for identification
units	character, units to put on plot
scaleloc	2-vector, percentage of bottom margin for the color scale

### Details

This function uses the `cwt` (package:Rwave) code to calculate the continuous wavelet transform, but plots it differently. Morlet wavelet is used by default. The `cwt` produces an image, the modulus of the transform, which is passed on to `wlet.do` along with the number of octaves and the number of voices. Plotting parameters are passed to the function so that replotting can be accomplished (use `plotwlet`) without having to recalculate the transform.

Plotting parameters are passed on to the plotting function, `plotwlet`.

### Value

baha	list: wavelet transform image, noctave = number of octaves, nvoice = number of voices, w0= central freq, flip = logical, whether image is flipped (default=TRUE)
PE	plotting information list: why=y-axis, dt=time series sample, interval, zscale=(1,2,3) image scaling, col=color map, ygrid = logical(default=FALSE), STAMP = character string

### Author(s)

Jonathan M. Lees<[jonathan.lees.edu](mailto:jonathan.lees.edu)>

### See Also

Rwave, `cwt`, `plotwlet`, `contwlet`, `pwlet2freqs`, `wlet.drive`

**Examples**

```
data(CE1)

plot(CE1$x, CE1$y, type='l')

require(Rwave)

out <- wlet.do(CE1$y, CE1$dt, flip = FALSE, ploty = TRUE)
```

---

wlet.drive	<i>Interactive wavelet transform driver</i>
------------	---

---

**Description**

interactive wavelet transform driver

**Usage**

```
wlet.drive(Xamp, DT = 0.008, noctave = 6, nvoice = 20, w0=5, STAMP = NULL)
```

**Arguments**

Xamp	vector of signal
DT	sample interval (s)
noctave	number of octaves, default=6
nvoice	number of voices, nvoice = 20
w0	central frequency for morlet wavelet, default=5
STAMP	character string for identification

**Value**

Graphical Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

RPGM, plotwlet, wlet.do

**Examples**

```

data(CE1)
plot(CE1$x, CE1$y, type='l')

if(interactive() ) wlet.drive(CE1$y, CE1$dt, STAMP=CE1$name)

```

---

write1segy	<i>Write One segy/sac file</i>
------------	--------------------------------

---

**Description**

Write out one segy binary format file.

**Usage**

```

write1segy(alist, fn = NULL, BIGLONG = FALSE)
write1sac(alist, fn = NULL, BIGLONG = FALSE)

```

**Arguments**

alist	list of traces with segy/sac header and an integer/real format time series
fn	Output file name
BIGLONG	logical, indicating whether long is 8 or 4 bytes.

**Details**

Segy format files are in integer format. The time series ususally represents counts recorded in a data acquisition system. The header includes meta-data and other identifying information.

**Value**

Side effects in the file system.

**Note**

The Endian-ness of the output file will be the native endian-ness of the system.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

rseis2segy, read1sac, read1segy

**Examples**

```
## Not run:
  theENDIAN = .Platform$endian
  BIGLONG = FALSE
  ### write1segy is in rseis2segy
  data(KH)
  apath = tempdir()
  J = rseis2segy(KH, sel=1, path=apath, BIGLONG=BIGLONG )
  L = list.files(path=J, full.names=TRUE)

  Z = read1segy(L[1], Iendian = theENDIAN, HEADONLY = FALSE, BIGLONG = BIGLONG)
  plot(Z$amp, type='l')

##### same with SAC files:
  J = rseis2sac(KH, sel = 1, win = c(0, 1), path = apath, BIGLONG = BIGLONG)
  L = list.files(path=J, pattern='.SAC', full.names=TRUE)

  Z = read1sac(L[1], Iendian = theENDIAN, HEADONLY = FALSE, BIGLONG = BIGLONG)

  plot(Z$amp, type='l')

## End(Not run)
```

---

writeUW.Acard

*writeUW.Acard*

---

**Description**

write UW pickfile

**Usage**

```
writeUW.Acard(LOC)
```

**Arguments**

LOC                    location structure

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

writeUW.Commentcard    *writeUW.Commentcard*

---

**Description**

write UW pickfile

**Usage**

writeUW.Commentcard(comments)

**Arguments**

comments            comment vector

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

writeUW.DOTcard            *writeUW.DOTcard*

---

**Description**

write UW pickfile

**Usage**

writeUW.DOTcard(STAS)

**Arguments**

STAS                    station structure

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>



---

writeUW.Ecard

*writeUW.Ecard*

---

**Description**

write UW pickfile

**Usage**

writeUW.Ecard(E)

**Arguments**

E                    Ecard

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

writeUW.Fcard

*writeUW.Fcard*

---

**Description**

write UW pickfile

**Usage**

writeUW.Fcard(F)

**Arguments**

F                    F-card info

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

writeUW.Hcard

*writeUW.Hcard*

---

**Description**

write UW pickfile

**Usage**

writeUW.Hcard(H)

**Arguments**

H                    H-card

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

---

writeUW.Ncard*writeUW.Ncard*

---

**Description**

write UW pickfile

**Usage**

writeUW.Ncard(N)

**Arguments**

N                    Name

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees.edu&gt;

---

writeUW.OSTAScard	<i>writeUW.OSTAScard</i>
-------------------	--------------------------

---

**Description**

write UW pickfile

**Usage**

```
writeUW.OSTAScard(OSTAS)
```

**Arguments**

OSTAS	OSTAS
-------	-------

**Value**

Side Effects

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

---

writeUWpickfile	<i>UW formatted ascii pickfile</i>
-----------------	------------------------------------

---

**Description**

Write UW formatted ascii pickfile

**Usage**

```
writeUWpickfile(A, output = "")
```

**Arguments**

A	Pickfile structure
output	output file

**Value**

Side Effects. Used to save ASCII versions of pickfiles for other processing.

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

EmptyPickfile

---

X2RSEIS

*Extract data to RSEIS file*

---

**Description**

swig Button Extract seismic data in RSEIS and save in GH format for exchange.

**Usage**

X2RSEIS(nh, g)

**Arguments**

nh	RSEIS seismic data format
g	swig parameters

**Details**

This function is used internally in RSEIS as a button in swig. The program should be run in a directory that has write permission.

The data is saved as a GH list.

**Value**

No value, writes to disk

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

XTR, X2SAC, swig

**Examples**

```
if(interactive()){
### get data:
GH <- Mine.seis(at1, at2, DB, NULL , NULL,
               kind = 1, Iendian=1)
w <- swig(GH, PADDLAB=c("X2SAC", "X2RSEIS", "YPIX" ) )
}
```

---

X2SAC

*Extract Data to SAC format*

---

**Description**

swig Button Extract seismic data in RSEIS and save in SAC format for exchange.

**Usage**

X2SAC(nh, g)

**Arguments**

nh	RSEIS seismic data format
g	swig parameters

**Details**

This function is used internally in RSEIS as a button in swig. The program should be run in a directory that has write permission.

**Value**

No value, writes to disk

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

XTR, X2RSEIS, swig

**Examples**

```
if(interactive()){  
  ### get data:  
  GH <- Mine.seis(at1, at2, DB, NULL, NULL,  
    kind = 1, Iendian=1)  
  w <- swig(GH, PADDLAB=c("X2SAC", "X2RSEIS", "YPIX" ) )  
}
```

---

`xcor2`*Cross Correlation*

---

**Description**

Cross correlation of two signals

**Usage**

```
xcor2(a1, a2, DT, PLOT = FALSE, LAG = 100)
```

**Arguments**

<code>a1</code>	input signal 1
<code>a2</code>	input signal 1
<code>DT</code>	deltaT in seconds
<code>PLOT</code>	logical TRUE=plot
<code>LAG</code>	time lag for correlation function

**Details**

Illustrates the cross correlation of two time series.

**Value**

<code>ccf</code>	Return list from function <code>ccf</code>
<code>m1ag</code>	maximum lag in time
<code>mccx</code>	value of <code>ccf</code> at max lag <code>m1ag</code>
<code>m1ag2</code>	maximum absolute value lag
<code>mccx2</code>	value of <code>ccf</code> at <code>m1ag2</code>

**Author(s)**

Jonathan M. Lees<[jonathan.lees.edu](mailto:jonathan.lees.edu)>

**See Also**

`ccf`

**Examples**

```
data(CE1)

ts1 <- CE1$y[CE1$x>5.443754 & CE1$x<5.615951]

ts2 <- CE1$y[CE1$x>5.760959]
ts2 <- ts2[1:length(ts1)]

ts1 <- ts1-mean(ts1)
ts2 <- ts2-mean(ts2)

xc <- xcor2(ts1, ts2, CE1$dt , PLOT = TRUE)
```

---

xprod

*Vector Cross Product*

---

**Description**

Cross product of two vectors

**Usage**

```
xprod(A1, A2)
```

**Arguments**

A1	3 component vector of x,y,z
A2	3 component vector of x,y,z

**Value**

3 component vector of A1 cross A2

**Author(s)**

Jonathan M. Lees <jonathan.lees@unc.edu>

**Examples**

```
B1 <- c(4,9,2)
B2 <- c(2,-5,4)

xprod(B1, B2)
```

---

XTR

*Buttons for swig*

---

### **Description**

defining functions for swig

### **Usage**

XTR(nh, g)  
NEXT(nh, g)  
PREV(nh, g)  
HALF(nh, g)  
MARK(nh, g)  
DOC(nh, g)  
REFRESH(nh, g)  
RESTORE(nh, g)  
ZOOM.out(nh, g)  
ZOOM.in(nh, g)  
RIGHT(nh, g)  
LEFT(nh, g)  
SCALE(nh, g)  
PSEL(nh, g)  
FLIP(nh, g)  
PTS(nh, g)  
FILT(nh, g)  
UNFILT(nh, g)  
SPEC(nh, g)  
WWIN(nh, g)  
SGRAM(nh, g)  
WLET(nh, g)  
XTR(nh, g)  
Pinfo(nh, g)  
TSHIFT(nh, g)  
RMS(nh, g)  
LocStyle(nh, g)  
CENTER(nh, g)  
fspread(nh, g)  
Xwin(nh, g)

### **Arguments**

nh	waveform list for RSEIS
g	plotting parameter list for interactive program



**Details**

Buttons can be defined on the fly.

**Value**

The return value depends on the nature of the function as it is returned to the main code swig. Choices for returning to swig are: break, replot, revert, replace, donothing, exit.

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**See Also**

swig

**Examples**

```
if(interactive()){
  MYFUNC<-function(nh, g)
  {
    print("pressed MYFUNC")
    g$sel
    d <- data.frame(list(stations=nh$STNS[g$sel],
                        components=nh$COMPS[g$sel]))
    print(d)
    g$action <- "replot"
    invisible(list(global.vars=g))
  }

  STDLAB <- c("DONE", "QUIT", "SELBUT" , "PSEL", "MYFUNC" )
  data(GH)
  JJ <- swig(GH, sel=1:10, STDLAB=STDLAB)

}
```

---

xtract.trace

*Extract trace*


---

**Description**

Extract one time series trace from an RSEIS data list

**Usage**

```
xtract.trace(GH, sel = 1, WIN = c(0, 1))
```

**Arguments**

GH	RSEIS list
sel	select trace index
WIN	time window on trace, relative to start

**Details**

An attribute of dt (sample time interval) is attached to the time series for use in plotting.

**Value**

vector	amplitudes
--------	------------

**Author(s)**

Jonathan M. Lees<jonathan.lees@unc.edu>

**Examples**

```
data(GH)

x1 <- xtract.trace(GH, sel = 1, WIN = c(0, 1))
plot(x1, type='l')
```

---

yeardate	<i>time in decimal years</i>
----------	------------------------------

---

**Description**

contract a date to decimal years

**Usage**

```
yeardate(yr, jd, hr, mi, sec)
```

**Arguments**

yr	year
jd	julian day
hr	hour
mi	minute
sec	second

**Value**

decimal time

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

secdif

**Examples**

```
yeardate(2005, 98, 12, 16, 32)
```

---

YPIX

*PICK Buttons for swig*

---

**Description**

defining functions for swig

**Usage**

```
YPIX(nh, g)  
WPIX(nh, g)  
NOPIX(nh, g)  
REPIX(nh, g)  
DELpix(nh, g)  
PickWin(nh, g)  
pADDPPIX(nh, g, phase)  
Ppic(nh, g)  
Spic(nh, g)  
Apic(nh, g)  
POLSWITCH(nh, g, dir)  
Pup(nh, g)  
Pnil(nh, g)  
Pdown(nh, g)  
FILLPIX(nh, g)  
RIDPIX(nh, g)  
SEEPPIX(nh, g)  
ROT.RT(nh, g)  
JustV(nh, g)  
JustE(nh, g)  
JustN(nh, g)  
JustF(nh, g)  
SHOW3(nh, g)
```

**Arguments**

nh	waveform list for RSEIS
g	plotting parameter list for interactive program
phase	phase name (P, S, A, etc...)
dir	vertical up, down or nil

**Details**

Buttons can be defined on the fly.

**YPIX** Multiple picks on a panel

**WPIX** window picks (start and end)

**NOPIX** remove the picks

**REPIX** un-remove the picks

**DELpix** Delete pix near clicks

**PickWin** Pick window for 3 component picking

**pADDPIX** add picks

**Ppic** P-wave arrival (only one per station)

**Spic** S-wave arrival (only one per station)

**Apic** acoustic-wave arrival (only one per station)

**POLSWITCH** flip polarity

**Pup** Polarity Up

**Pnil** Polarity nil

**Pdown** Polarity down

**FILLPIX** Fill the pick from bottom to top of panel

**RIDPIX** remove pick

**SEEPix** print current picks to screen

**ROT.RT** Rotate to radial and transverse (need event and station locations)

**JustV** Display only vertical components

**JustE** Display only east components

**JustN** Display only north components

**JustF** Display only infrasound (F) components

**SHOW3** Display All 3 components

**iNEXT** Used internally in PickWin to move to next station

**Value**

The return value depends on the nature of the function as it is returned to the main code swig. Choices for returning to swig are: break, replot, revert, replace, donothing, exit.

**Author(s)**

Jonathan M. Lees&lt;jonathan.lees@unc.edu&gt;

**See Also**

swig, XTR

**Examples**

```

if(interactive()){

MYFUNC<-function(nh, g)
{
  print("pressed MYFUNC")
  d <- data.frame(list(stations=nh$STNS, components=nh$COMPS))
print(d)
  g$action <- "replot"
  invisible(list(global.vars=g))
}

STDLAB <- c("DONE", "QUIT", "SELBUT" , "MYFUNC" )
data(GH)
JJ <- swig(GH, sel=1:10, STDLAB=STDLAB)

}

```

---

YRsecdif

*Return difference in seconds*


---

**Description**

Difference between two Date/Times (Julian Day)

**Usage**

```
YRsecdif(jd1, hr1, mi1, sec1, jd2, hr2, mi2, sec2, yr1 = 0, yr2 = 0)
```

```
YRsecdifL(T1, T2)
```

**Arguments**

jd1	Julian Day
hr1	hour
mi1	minute

sec1	second
jd2	Julian Day
hr2	hour
mi2	minute
sec2	second
yr1	year 1
yr2	year 2
T1	list 1 with date time
T2	list 2 with date time

**Details**

Returns T2-T1, year is used.

**Value**

numeric          seconds

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

secdifL, secdif

**Examples**

```
T1 <- list(jd=12, hr=13, mi=23, sec=21, yr=1964 )
T2 <- list(jd=14, hr=23, mi=23, sec=2, yr=1976)

YRsecdif(T1$jd, T1$hr, T1$mi, T1$sec, T2$jd, T2$hr, T2$mi, T2$sec,
1964, 1976)

#### or

YRsecdifL(T1, T2)
```

---

Zdate

*Date functions*

---

### **Description**

Make character vector from dates

### **Usage**

```
Zdate(info, sel=1, t1=0, sep='_')  
dateList(datevec)  
dateStamp(datelist, sep='_')
```

### **Arguments**

info	info structure from trace structure
sel	selection of which ones to extract, default=1:length(info\$jd)
t1	time offset, seconds, default=0
sep	character for separating the components in the string, default=":"
datevec	vector with yr, jd, mo, day, hr, mi, sec
datelist	output of dateList

### **Details**

Format date stamp for plotting and identification. Used for STAMP.

### **Value**

character strings

### **Note**

If using Zdate to create a file name, be careful about the separator. A colon in the file name on PC and MAC systems can be confusing for the OS.

### **Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

### **See Also**

swig, dateStamp, ghstamp, filedatetime

**Examples**

```

data("GH")

sel <- which(GH$COMPS == "V")

ftime <- Zdate(GH$info, sel[1:5], 1)

dvec <- c(2009, 134, 5, 14, 10, 32, 24.5, 0)
A <- dateList(dvec)
dateStamp(A, sep=".")

dateStamp(A, sep="_")

```

---

zlocator

*zlocator*


---

**Description**

Locator function with set parameters

**Usage**

```
zlocator(COL = 1, ID = FALSE, NUM = FALSE, YN = NULL, style = 0)
```

**Arguments**

COL	color
ID	logical, identify points
NUM	number of points
YN	number of windows to span for lines
style	0,1,2 for differnt style of plotting vertical lines

**Details**

if the window is divided into YN horizontal regions, style =2 will plot segments only within regions based on y-value of locator().

**Value**

list:

x	x-locations
y	y-locations
n	number of points



**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

plocator, locator

**Examples**

```
plot(c(0,1), c(0,1), type='n')
for(i in 1:5) { abline(h=i/6) }

if(interactive() )zlocator(COL = 1, NUM = 4, YN = 6, style = 2)
```

---

ZOOM.SEISN

*ZOOM SEISMIC Panel*

---

**Description**

Zoom interactively on Seismic panel data.

**Usage**

```
ZOOM.SEISN(GH, sel = 1:length(GH$dt), WIN = NULL)
```

**Arguments**

GH	Seismic trace structure
sel	selection of traces
WIN	time window c(0,1)

**Value**

Seismic trace structure

**Author(s)**

Jonathan M. Lees<jonathan.lees.edu>

**See Also**

swig

**Examples**

```
data("GH")
sel <- which(GH$COMPS=="V")

KF <- ZOOM.SEISN(GH, sel=sel, WIN = c(0 , 5) )

if(interactive()){ swig(KF)
}
```

# Index

## \* MISC

P2GH, 171

## \* aplot

addpoints.hodo, 8  
addtix, 9  
comp.env, 33  
contwlet, 38  
DO.PMOT.ARR, 52  
gaddtix, 85  
jlegend, 136  
lagplot, 143  
letter.it, 145  
NEWPLOT.WPX, 168  
PLOT.ALLPX, 184  
PLTpicks, 203  
PPIX, 206  
pwlet2freqs, 213  
swig.ALLPX, 271  
vline, 292  
winmark, 296

## \* datasets

CE1, 25  
GH, 112  
KH, 142  
OH, 170  
VELMOD1D, 287

## \* hplot

autoreg, 16  
brune.doom, 17  
circ, 29  
Comp1Dvel, 34  
Comp1Dvels, 35  
complex.hodo, 36  
contwlet, 38  
FILT.spread, 76  
GAZI, 86  
get.corner, 88  
GETARAIC, 95  
hodogram, 122

INSTresponse, 129

matsquiggle, 155

MTMdisp, 163

MTMplot, 166

partmotnet, 174

PLOT.MATN, 185

PLOT.SEISN, 187

plotevol, 194

plotwlet, 200

plt.MTM0, 202

PSTLTcurve, 211

SNET.drive, 261

varsquig, 284

varsquiggle, 286

wlet.do, 299

ZOOM.SEISN, 321

## \* iplot

choosfilt, 26

detail.pick, 46

idpoints.hodo, 124

MTM.drive, 162

plocator, 183

plotJGET, 197

PMOT.drive, 204

SNET.drive, 261

SPECT.drive, 262

swig, 268

tung.pulse, 281

wlet.drive, 301

zlocator, 320

## \* math

Spectrum, 264

## \* misc

addpoints.hodo, 8

addWPX, 10

applytaper, 11

ASCII.SEISN, 12

attime12, 13

AUGMENTbutfilt, 14

autoreg, 16  
brune.doom, 17  
brune.func, 18  
brune.search, 19  
butfilt, 20  
BUTREPLOTT, 22  
catWPX, 24  
checkWPX, 25  
CHOP.SEISN, 28  
cleanpickfile, 29  
cleanWPX, 30  
colorwig, 31  
combineSEIS, 32  
complex.hodo, 36  
COMPorder, 37  
convert2Rseis, 39  
convertATT, 40  
correct.moveout, 41  
DAYSpERYEAR, 42  
DECIMATE.SEISN, 42  
deconinst, 44  
deleteWPX, 45  
detail.pick, 46  
detrend, 47  
DISPLACE.SEISN, 48  
distseisnXY, 49  
DISTxsec, 50  
DO.PMOT.ARR, 52  
doGABOR.AR, 53  
doGABOR.MTM, 54  
doMYBUTTS, 56  
DOsgram, 57  
dowiggles, 58  
downsample, 59  
editDB, 60  
EmptyPickfile, 62  
EmptySEIS, 63  
envelope, 63  
EPOCHday, 64  
EPOCHyear, 65  
ETECTG, 66  
evoIAR, 67  
evolfft, 69  
evoIMTM, 70  
FAKEDATA, 72  
filedatetime, 74  
FILT.SEISN, 75  
filterstamp, 77  
finteg, 79  
fixcompname, 80  
fixcomps, 80  
fixNA, 81  
fixUWstasLL, 83  
fromjul, 83  
FRWDft, 84  
gaddtix, 85  
genrick, 87  
get.corner, 88  
GET.seis, 89  
get.slepians, 92  
Get1Dvel, 94  
GETARAIC, 95  
getb1b2, 96  
getEcard, 97  
getFcard, 98  
getGHtime, 99  
getHcard, 100  
getIRIS, 101  
getjul, 102  
gettoday, 103  
getNcard, 104  
getPDEcsv, 104  
getpfile, 105  
getphaselag2, 106  
getrdpix, 107  
getseis24, 108  
getvertsorder, 110  
ghstamp, 114  
GLUE.GET.seis, 115  
GLUEseisMAT, 115  
gpoly, 116  
GreatDist, 117  
grotseis, 118  
hilbert, 120  
hilow, 121  
hodogram, 122  
hypot, 123  
idpoints.hodo, 124  
info.seis, 125  
infoDB, 125  
insertNAs, 127  
INSTFREQS, 128  
INSTresponse, 129  
integ1, 130  
INVRft, 131  
j2posix, 132

jadjust.length, 133  
JBLACK, 133  
JGRAY, 134  
jitter.lab, 135  
jpolyval, 137  
JSAC.seis, 138  
jstats, 140  
Jtim, 141  
leests, 144  
legitpix, 145  
LocalUnwrap, 146  
logspace, 147  
longfft, 148  
makeDB, 150  
makefreq, 152  
markseis24, 153  
Mine.seis, 157  
mirror.matrix, 159  
Mmorlet, 160  
mtapspec, 161  
MTM.drive, 162  
MTMgabor, 165  
MTMplot, 166  
NEW.getUWSTAS, 168  
NEWPLOT.WPX, 168  
next2, 169  
one, 171  
parse.pde, 172  
parseFN2STA, 173  
PDE2list, 175  
peaks, 176  
PICK.DOC, 177  
pickgeninfo, 178  
pickhandler, 178  
pickit, 179  
pickseis24, 181  
PLOT.TTCURVE, 189  
Plot1Dvel, 191  
plotarrivals, 192  
plotDB, 193  
plotGH, 196  
plotseis24, 198  
PMOT.drive, 204  
posix2RSEIS, 205  
prep1wig, 207  
prepSEIS, 209  
PreSet.Instr, 210  
PSTLTcurve, 211  
Put1Dvel, 212  
pwlet2freqs, 213  
rangedatetime, 214  
Ray.time1D, 215  
rdistaz, 216  
rDUMPLLOC, 218  
read1segy, 219  
ReadInstr, 220  
ReadSet.Instr, 221  
readUW.OSTAS, 222  
recdate, 223  
repairWPX, 224  
replaceWPX, 225  
rseis2segy, 226  
rseis2ts, 227  
rsspec.taper, 228  
ruler, 229  
save.wpix, 230  
saveWPX, 231  
scal2freqs, 232  
screens, 233  
SEARCHPIX, 234  
secdif, 235  
secdifL, 236  
secdifv, 237  
segy2rseis, 238  
SEIS2list, 239  
seiscols, 240  
SEISNtime, 241  
seisorder, 242  
selAPX, 243  
SELBUT, 244  
selpgen, 245  
SELSTA, 245  
selstas, 246  
SENSORsensitivity, 247  
setPrePix, 248  
setstas, 249  
setupDB, 250  
setwelch, 251  
setwpix, 254  
setWPX, 255  
setypx, 257  
showdatetime, 258  
sigconv, 259  
sigconvGR, 260  
STALTA, 265  
STLTcurve, 266

- symshot1, 272
- sysinfo, 273
- T12.pix, 274
- TAPER.SEISN, 275
- Thresh.J, 276
- TOCART, 277
- tojul, 278
- tomo.colors, 278
- trapz, 279
- travel.time1D, 280
- unpackAcard, 282
- uwpfile2ypx, 283
- VELOCITY.SEISN, 288
- view.seis, 289
- vlen, 291
- wiggle.env, 293
- wiggleimage, 294
- WINGH, 295
- winseis24, 298
- wlet.do, 299
- write1segy, 302
- writeUW.Acard, 303
- writeUW.Commentcard, 304
- writeUW.DOTcard, 304
- writeUW.Ecard, 305
- writeUW.Fcard, 305
- writeUW.Hcard, 306
- writeUW.Ncard, 306
- writeUW.OSTAScard, 307
- writeUWpickfile, 307
- X2RSEIS, 308
- X2SAC, 309
- xcor2, 310
- xprod, 311
- XTR, 312
- xtract.trace, 313
- yeardate, 314
- YPIX, 315
- YRsecdif, 317
- Zdate, 319
- \* package**
  - RSEIS-package, 7
- addpoints.hodo, 8
- addtix, 9
- addWPX, 10
- Apic (YPIX), 315
- applytaper, 11
- ASCII.SEISN, 12
- atime12, 13
- AUGMENTbutfilt, 14
- autoreg, 16
- blankevol (plotevol), 194
- brune.doom, 17
- brune.func, 18
- brune.search, 19
- butfilt, 20
- BUTREPLOT, 22
- catWPX, 24
- CE1, 25
- CENTER (XTR), 312
- checkWPX, 25
- choosfilt, 26
- CHOP.SEISN, 28
- circ, 29
- cleanpickfile, 29
- cleanWPX, 30
- colorwig, 31
- combineSEIS, 32
- comp.env, 33
- Comp1Dvel, 34
- Comp1Dvels, 35
- complex.hodo, 36
- COMPorder, 37
- contwlet, 38
- convert2Rseis, 39
- convertATT, 40
- correct.moveout, 41
- dateList (Zdate), 319
- dateStamp (Zdate), 319
- DAYSperYEAR, 42
- DECIMATE.SEISN, 42
- deconinst, 44
- deleteWPX, 45
- DELPix (YPIX), 315
- detail.pick, 46
- detrend, 47
- DISPLACE.SEISN, 48
- distseisnXY, 49
- DISTxsec, 50
- DO.PMOT.ARR, 52
- DOC (XTR), 312
- doGABOR.AR, 53
- doGABOR.MTM, 54
- doMYBUTTS, 56

- D0sgram, 57
- dowiggles, 58
- downsample, 59
  
- editDB, 60
- EmptyPickfile, 62
- EmptySEIS, 63
- envelope, 63
- EPOCHday, 64
- EPOCHyear, 65
- ETECTG, 66
- evolAR, 67
- evolfft, 69
- evolMTM, 70
  
- FAKEDATA, 72
- filedatetime, 74
- FILLPIX (YPIX), 315
- FILT (XTR), 312
- FILT.SEISN, 75
- FILT.spread, 76
- filterstamp, 77
- finteg, 79
- fixcompname, 80
- fixcomps, 80
- fixNA, 81
- fixUWstasLL, 83
- FLIP (XTR), 312
- FmakeDB (makeDB), 150
- fromjul, 83
- FRWDft, 84
- fspread (XTR), 312
  
- gaddtix, 85
- GAZI, 86
- genrick, 87
- get.corner, 88
- GET.seis, 89
- get.slepians, 92
- Get1Dvel, 94
- getANSS (getIRIS), 101
- GETARAIC, 95
- getb1b2, 96
- getEcard, 97
- getFcard, 98
- getGHtime, 99
- getHcard, 100
- getIRIS, 101
- getjul, 102
  
- getmoday, 103
- getNcard, 104
- getPDEcsv, 104
- getPDEscreen (getPDEcsv), 104
- getpfile, 105
- getphaselag2, 106
- getrdpix, 107
- getseis24, 108
- getvertsorder, 110
- GH, 112
- ghstamp, 114
- GLUE.GET.seis, 115
- GLUEseisMAT, 115
- gpoly, 116
- GreatDist, 117
- grotseis, 118
  
- HALF (XTR), 312
- hilbert, 120
- hilow, 121
- hodogram, 122
- hypot, 123
  
- idpoints.hodo, 124
- iNEXT (YPIX), 315
- info.seis, 125
- infoDB, 125
- insertNAs, 127
- INSTFREQS, 128
- INSTresponse, 129
- integ1, 130
- INVRft, 131
  
- j2posix, 132
- jadjust.length, 133
- JBLACK, 133
- JGET.seis (GET.seis), 89
- JGRAY, 134
- jitter.lab, 135
- jlegend, 136
- jpolyval, 137
- JSAC.seis, 138
- JSEGY.seis (JSAC.seis), 138
- jstats, 140
- Jtim, 141
- JtimL (Jtim), 141
- JustE (YPIX), 315
- JustF (YPIX), 315
- JustN (YPIX), 315

- JustV (YPIX), 315
- KH, 142
- lagplot, 143
- leests, 144
- LEFT (XTR), 312
- legitpix, 145
- letter.it, 145
- LocalUnwrap, 146
- LocStyle (XTR), 312
- logspace, 147
- longfft, 148
- longpstart (longfft), 148
- longreset (longfft), 148
- makeDB, 150
- makefreq, 152
- many.time1D (travel.time1D), 280
- MARK (XTR), 312
- markseis24, 153
- matsquiggle, 155
- Mine.seis, 157
- mirror.matrix, 159
- Mmorlet, 160
- mtapspec, 161
- MTM.drive, 162
- MTMdisp, 163
- MTMgabor, 165
- MTMplot, 166
- NEW.getUWSTAS, 168
- NEWPLOT.WPX, 168
- NEXT (XTR), 312
- next2, 169
- NOPIX (YPIX), 315
- noPS (YPIX), 315
- OH, 170
- one, 171
- P2GH, 171
- pADDPPIX (YPIX), 315
- parse.pde, 172
- parseFN2STA, 173
- partmotnet, 174
- pathDB (editDB), 60
- PDE2list, 175
- Pdown (YPIX), 315
- peaks, 176
- PICK.DOC, 177
- pickgeninfo, 178
- pickhandler, 178
- pickit, 179
- pickseis24, 181
- PickWin (YPIX), 315
- Pinfo (XTR), 312
- plocator, 183
- PLOT.ALLPX, 184
- PLOT.MATN, 185
- PLOT.SEISN, 187
- PLOT.TTCURVE, 189
- Plot1Dvel, 191
- plotarrivals, 192
- plotDB, 193
- plotevol, 194
- plotevol2 (plotevol), 194
- plotGH, 196
- plotJGET, 197
- plotseis24, 198
- plotwlet, 200
- plt.MTM0, 202
- PLTpicks, 203
- PMOT.drive, 204
- Pnil (YPIX), 315
- POLSWITCH (YPIX), 315
- posix2RSEIS, 205
- Ppic (YPIX), 315
- PPIX, 206
- prep1wig, 207
- prepSEIS, 209
- PreSet.Instr, 210
- PREV (XTR), 312
- PSEL (XTR), 312
- PSTLTcurve, 211
- PTS (XTR), 312
- Pup (YPIX), 315
- Put1Dvel, 212
- pwlet2freqs, 213
- rangedatetime, 214
- Ray.time1D, 215
- rdistaz, 216
- rDUMPLOC, 218
- read1sac (read1seg), 219
- read1seg, 219
- ReadInstr, 220
- ReadSet.Instr, 221
- readUW.OSTAS, 222



- recdate, 223
- recdate1 (recdate), 223
- REFRESH (XTR), 312
- repairWPX, 224
- REPIX (YPIX), 315
- replaceWPX, 225
- RESTORE (XTR), 312
- RIDPIX (YPIX), 315
- RIGHT (XTR), 312
- RMS (XTR), 312
- ROT.RT (YPIX), 315
- RSEIS (RSEIS-package), 7
- RSEIS-package, 7
- rseis2sac (rseis2segy), 226
- rseis2segy, 226
- rseis2ts, 227
- rsspec.taper, 228
- ruler, 229
  
- sac2rseis (segy2rseis), 238
- save.wpix, 230
- saveWPX, 231
- scal2freqs, 232
- SCALE (XTR), 312
- screens, 233
- SEARCHPIX, 234
- secdif, 235
- secdifL, 236
- secdifv, 237
- SEEPPIX (YPIX), 315
- segy2rseis, 238
- SEIS2list, 239
- seiscols, 240
- SEISNtime, 241
- seisorder, 242
- selAPX, 243
- SELBUT, 244
- selpgen, 245
- SELSTA, 245
- selstas, 246
- selWPX (selAPX), 243
- SENSORsensitivity, 247
- setPrePix, 248
- setstas, 249
- setupDB, 250
- setwelch, 251
- setwpix, 254
- setWPX, 255
- setypx, 257
  
- SGRAM (XTR), 312
- SHOW3 (YPIX), 315
- showdatetime, 258
- sigconv, 259
- sigconvGR, 260
- SNET.drive, 261
- SPEC (XTR), 312
- SPECT.drive, 262
- Spectrum, 264
- Spic (YPIX), 315
- STALTA, 265
- STLTcurve, 266
- swig, 268
- swig.ALLPX, 271
- symshot1, 272
- sysinfo, 273
  
- T12.pix, 274
- TAPER.SEISN, 275
- Thresh.J, 276
- TOCART, 277
- tojul, 278
- tomo.colors, 278
- trapz, 279
- travel.time1D, 280
- TSHIFT (XTR), 312
- tung.pulse, 281
  
- UNFILT (XTR), 312
- unpackAcard, 282
- uwpxfile2ypx, 283
  
- varsquig, 284
- varsquiggle, 286
- VELMOD1D, 287
- VELOCITY.SEISN, 288
- view.seis, 289
- vlen, 291
- vline, 292
  
- wiggle.env, 293
- wiggleimage, 294
- WINGH, 295
- winmark, 296
- winseis24, 298
- WLET (XTR), 312
- wlet.do, 299
- wlet.drive, 301
- WPIX (YPIX), 315

write1sac (write1segy), [302](#)  
write1segy, [302](#)  
writeUW.Acard, [303](#)  
writeUW.Commentcard, [304](#)  
writeUW.DOTcard, [304](#)  
writeUW.Ecard, [305](#)  
writeUW.Fcard, [305](#)  
writeUW.Hcard, [306](#)  
writeUW.Ncard, [306](#)  
writeUW.OSTAScard, [307](#)  
writeUWpickfile, [307](#)  
WWIN (XTR), [312](#)

X2RSEIS, [308](#)  
X2SAC, [309](#)  
xcor2, [310](#)  
xprod, [311](#)  
XTR, [312](#)  
xtract.trace, [313](#)  
Xwin (XTR), [312](#)

yeardate, [314](#)  
YPIX, [315](#)  
YRsecdif, [317](#)  
YRsecdifL (YRsecdif), [317](#)

Zdate, [319](#)  
zlocator, [320](#)  
ZOOM.in (XTR), [312](#)  
ZOOM.out (XTR), [312](#)  
ZOOM.SEISN, [321](#)