Package 'benchmarkMetrics'

December 2, 2015

Type Package

Title Benchmark Metrics

Version 2.0

Date 2015-11-16

Author Douglas Kelley

Maintainer Douglas Kelley <douglas.kelley@mq.edu.au>

Description Measures distances between modelled and obscured data, developed for the benchmarking of vegetation models.

License GPL-2

Imports raster, plotrix

R topics documented:

`benchmarkMetrics-package` *Benchmark Metrics*

Description

Specifically designed metrics quantify model performance against observations via “metric scores” and compare to scores to "null" model-benchmarks based on the temporal or spatial mean value of the observations and a “random” model produced by bootstrap resampling of the observations.

Details

Package:	benchmarkMetrics
Type:	Package
Version:	1.0
Date:	2014-05-30
License:	GPL 2
Website: https://bitbucket.org/teambcd/benchmarkmetrics	

Author(s)

Douglas Kelley `<douglas.i.kelley@gmail.com>`

References

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., & Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. Biogeosciences, 10(5), 3313-3340. doi:10.5194/bg-10-3313-2013

Examples

```
mnthN         = rep(1:12, length.out = dim(Seatbelts)[1])
yr            = seq(start(Seatbelts)[1], end(Seatbelts)[1]+11/12,by=1/12)
law           = Seatbelts[, "law"]
test          = law == 0

addCol2Ts <- function(dat,ndat) {
  cnames         = c(colnames(dat),colnames(ndat))
  dat            = cbind(dat,ndat)
  colnames(dat) = cnames
  return(dat)
}

SeatbeltsLaw   = addCol2Ts(Seatbelts,cbind(month=mnthN,year=yr))
SeatbeltsNoLaw = SeatbeltsLaw[test,]

form  = "drivers ~ kms * (a  + b*year + d*sin(e + month * 2*pi/12))"
start = c(a=0, b=-1, d=1, e=0)

performModel <- function(dat, dform = "", dstart =  c()) {
  start = c(start, dstart)
  form  = formula(paste(form, dform))
  mod   = nls(form, start = start, data=as.data.frame(dat))

  return(predict(mod,as.data.frame(SeatbeltsLaw)))
}

modNoLaw = performModel(SeatbeltsNoLaw)
modLaw   = performModel(SeatbeltsLaw, "+ f*law", c(f = 0))

SeatbeltsLaw = addCol2Ts(SeatbeltsLaw, cbind(modNoLaw,modLaw))


## Test interannual variability
ma           <- function(x, n = 12) filter(x, rep(1 / n, n), sides = 2)
ma.Seatbelts <- function(x) ma(SeatbeltsLaw[, x])
```

```
Deaths   = ma.Seatbelts('drivers')
modNoLaw = ma.Seatbelts('modNoLaw')
modLaw   = ma.Seatbelts('modLaw')

graphics.off()
par(mfcol=c(3,3),mar=c(4,3,1,0),oma=c(0,4,0,0))

plot (Deaths,ylab='Deaths',xpd=NA)
lines(yr, modNoLaw, col = 'red' )
lines(yr, modLaw  , col = 'blue')
legend(x = 'bottomleft', lty=1, col = c('black', 'red', 'blue'),
       legend = c('observed', 'Without Law', 'With Law'))

nme.NoLaw = NME(Deaths,modNoLaw)
nme.Law   = NME(Deaths,modLaw)


nmePlot <- function(nme,txt) {
  plot (nme, xlab = 'Observed Deaths', ylab = 'Predicted Deaths', xpd = NA)
  mtext(txt, side = 2, line = 4, adj = 0.99)
}

nmePlot(nme.NoLaw,"Model without Law")
nmePlot(nme.Law,"Model with Law")


## Test interannual variability in seasonality
plot(c(1, 12), range(SeatbeltsLaw[, 'drivers']),
     xaxt = 'n',xlab = 'Month',ylab = '', type = 'n')
axis(1, at=1:12, c('J','F','M','A','M','J','J','A','S','O','N','D'))

poly <- function(y, angle, col)
  polygon(c(1:12, 12:1), c(y + 100, rev(y) - 100), density = 25, angle = angle,
          col = col, border = NA)

addSeasonal <- function(nm, col, angle)
  apply(ts2matrix(SeatbeltsLaw[, nm]), 1, poly, col, angle)

addSeasonal('drivers' ,  45, '#000000')
addSeasonal('modNoLaw', -45, '#FF0000')
addSeasonal('modLaw'  ,  00, '#0000FF')

mpd.NoLaw = MPD(SeatbeltsLaw[,'drivers'],SeatbeltsLaw[,'modNoLaw'])
mpd.Law   = MPD(SeatbeltsLaw[,'drivers'],SeatbeltsLaw[,'modLaw'  ])

mpdPlot <- function(mpd) {
  plot(mpd.NoLaw,labels=c('Jan','Jul'),
       radial.lim=c(0,0.08),radial.labels='')
  axis(1,pos=0,at=c(0,0.08))
}
mpdPlot(mpd.NoLaw)
mpdPlot(mpd.Law)


## Test law model explonation of front:rear ratio killed
tot = SeatbeltsLaw[, 'front'] + SeatbeltsLaw[, 'rear']
```

```
SeatbeltsLaw[, 'front'] = SeatbeltsLaw[, 'front'] / tot
SeatbeltsLaw[, 'rear' ] = SeatbeltsLaw[, 'rear' ] / tot
SeatbeltsNoLaw          = SeatbeltsLaw[test,]

form  = "front ~ kms * (a  + b*year + d*sin(e + month * 2*pi/12))"
modFrontNoLaw = performModel(SeatbeltsNoLaw)
modFrontLaw   = performModel(SeatbeltsLaw, "+ f*law", c(f = 0))

SeatbeltsLaw = addCol2Ts(SeatbeltsLaw,
      cbind(modFrontNoLaw, modRearNoLaw = 1 - modFrontNoLaw,
            modFrontLaw  , modRearLaw   = 1 - modFrontLaw  ))

returnDensity <- function(x) {
  out=hist(SeatbeltsLaw[, x], 1000, plot = FALSE)[c('mids','density')]
  out[[2]]=ma(out[[2]],100)
  return(out)
}

Obs   = returnDensity('front'        )
NoLaw = returnDensity('modFrontNoLaw')
Law   = returnDensity('modFrontLaw'  )

plot(range(Obs[[1]]), range(c(Obs[[2]], NoLaw[[2]], Law[[2]]), na.rm = TRUE),
     type = 'n', xlab = '', ylab = 'Count')

addDensityLines <- function(x, col) lines(x[[1]], x[[2]], col = col)

addDensityLines(Obs,   'black')
addDensityLines(NoLaw, 'red')
addDensityLines(Law,   'blue')


mm.NoLaw = MM(SeatbeltsLaw[,c('front'        , 'rear'        )],
              SeatbeltsLaw[,c('modFrontNoLaw', 'modRearNoLaw')])
mm.Law   = MM(SeatbeltsLaw[,c('front'        , 'rear'        )],
              SeatbeltsLaw[,c('modFrontLaw'  , 'modRearLaw'  )])

plot(mm.NoLaw)
plot(mm.Law  )

## Find nulls models
nmeNull = null.NME(SeatbeltsLaw[,'drivers'])
mpdNull = null.MPD(SeatbeltsLaw[,'drivers'])
mmNull  = null.MM (SeatbeltsLaw[,'drivers'])

## Compare metrics to null models

par(mfcol=c(2,2))
cols = cbind(Law =   c("#0000FF","#0000DD","#0000DD"),
             NoLaw = c("#DDDD00","#AAAA00","#550000"))


addRsltLine <- function(score  ,col ) lines(c(score,score),c(0,9E9),col=col)

addStepLine <- function(scores ,cols)
  mapply(addRsltLine,score(scores),cols)
```

```
plot(nmeNull, main = "NME Model Comparison",
     xlim = c(0.4 , 1.5), legend = FALSE)
addStepLine(nme.Law  , cols[, 1])
addStepLine(nme.NoLaw, cols[, 2])

plot(mmNull,  main = " MM Model Comparison ",
     xlim = c(0.08, 0.2), legend = FALSE)
addRsltLine(score(mm.Law)   , cols[1, 1])
addRsltLine(score(mm.NoLaw), cols[1, 2])

plot(mpdNull, main = "MPD Model Comparison" , legend = FALSE)
```

---

atans *Performing a full circle arc-tangent*

Description

Performs to atan2, but allows conversion into different angle measures and allows raster inputs. The arc-tangent of two vectors, x and y, returns the angle between the x-axis and the vector from the origin to (x, y).

Usage

```
atans(x, y, units = "months")
```

Arguments

`x, y`	numeric or complex vectors or rasters
`units`	How the resultant angle should be measured. See value for more information

Value

The angle between the origin and points for each element in x and y. Expressed in units as described by 'units'. units='months' is the default, and returns fractional months between 0 and 12 (0 representing the x-axis). 'radians' returns radians. 'degrees' returns degrees.

Author(s)

Douglas Kelley <douglas.i.kelley@gmail.com>

See Also

atan2 raster

Examples

```
x <- array(1:24,dim=c(2,3,4))
y <- array(rev(1:24),dim=c(2,3,4))
phase <- atans(x,y)
```

`MM` *Manhattan Metric/ Square Chord Distance*

Description

Performs the Manhattan Metric (MM) and Square Chord Distance (SCD) comparison between modelled and observed data on fractional-type items.

Usage

```
MM (x, y, w = NULL, allowRegridding = TRUE)
SCD(x, y, w = NULL, allowRegridding = TRUE)
```

Arguments

`x, y`	matrix, data.frame, raster or raster stack where y is the approximation to x. The last dimension represents the fractional items, whilst all other dimenstions represent different realisations of these items
`w`	vector or raster of weights. If NULL, all items are considered equally. If a vector, must by the same length as first dimension of x and represents the weight of each point. If matrix or raster, than weight for each point and item.
`allowRegridding`	Logical arguement used if x and y are both raster. If TRUE and x and y are on different grids, then x and y are cropped to the smallest shared extent and lowest reslution of either x and y. If w is raster as well, this will also be regridded to smallest extent and lowest resolution of x or y.

Details

MM measure the absolute distamce between items of simulated and observed items.

$$MM = \Sigma|x_{i,j} - y_{i,j}|/n$$

where i is the fractrual measurement of item j

SCD is similar and is often employed as an enhanced way of measuring “signal-to-noise”.

$$SCD = \Sigma(x_{i,j}^{0}.5 - y_{i,j}^{0}.5)^2/n$$

They both take the value of 0 for perfect agreement, and 2 for complete disagreement.

If x and y are 2-dimensional, rows represent ’points’, columns represent items. Items for each ’point’ are normalised before metric comparison is made.

Value

MM and SCD returns an object of class "MM".

The `print` print function returns the call information and score

The function `summary` can be used to obtain information on individual item means and variances for both x and y, as well as the ratio of x and ys mean and variance.

The `plot` function plots the items in a scatter plot.

Author(s)

Douglas Kelley <douglas.i.kelley@gmail.com>

References

Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. City, 1(2), 1.

Gavin, D. G., Oswald, W. W., Wahl, E. R., & Williams, J. W. (2003). A statistical approach to evaluating distance metrics and analog assignments for pollen records. Quaternary Research, 60(3), 356-367.

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., & Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. Biogeosciences, 10(5), 3313-3340. doi:10.5194/bg-10-3313-2013

See Also

print.MM, summary.MM, plot.MM

Examples

```
require("raster")
## NPP allocation observations from Schuur & Matson (2001)
Obs =cbind(canopy = c(4.25 ,5.0  ,4.25 ,4.5  ,3.0  ,2.0 ),
           stem   = c(0.37 ,0.12 ,0.62 ,0.12 ,0.75 ,0.05),
           wood   = c(0.62 ,0.21 ,1.03 ,0.21 ,1.24 ,0.08))

## Allocacion fractions from GDAY fixed allocation (Mcmurtrie & Comins 1996)
Mod = cbind(canopy = rep(0.2,6),
            stem   = rep(0.6,6),
            wood   = rep(0.2,6))

## Weight are total NPP of each measure
weights = apply(Obs,1,sum)

## Perform and display comparison
m = MM(Obs,Mod,weights)
m
summary(m)
plot(m)

m = SCD(Obs,Mod,weights)
summary(m)

## Items from raster brick, taken from ?brick
## 3-item brick
b = brick(system.file("external/rlogo.grd", package="raster"))

## randomizd 3-item brick
br   = b
br[] = b[sample(1:ncell(b))]

## metric comparison
m = MM(b,br)

## Display results
```

```
summary(m)
plot(m)

## Schuur E. A. G., Matson P. A. 2001 Net primary productivity and nutrient cycling acr
## Mcmurtrie, R. E., & Comins, H. N. (1996). The temporal response of forest ecosystems
```

MPD *Mean Phase and Concentration Difference*

Description

MPD measures model to observed error for pedictable periodic patterns (ie weasonal variablity) characterised in terms of differences in concentration (i.e inverse of season length) and phase (i.e timing of season).

Usage

```
MPD(x, y, w = NULL, , allowRegridding = TRUE, ...)
```

Arguments

`x, y`	matrix, data.frame, raster or raster stacks. Where y is the approximation to x
`w`	vector or raster of weights or same size as x. If NULL, all items are considered equally.
`allowRegridding`	Logical arguement used if x and y are both raster. If TRUE and x and y are on different grids, then x and y are cropped to the smallest shared extent and lowest reslution of either x and y. If w is raster as well, this will also be regridded to smallest extent and lowest resolution of x or y.
`...`	arguments passed to `NME`

Details

Each simulated or observed timestep (e.g month) with in the period (i.e year) is represented by a vector in the complex plane, which is split into phase and concontration components as descibed in `PolarConcentrationAndPhase`. If the variable is concentrated all in one point within the polar coordinates, seasonal concentration is equal to 1 and the phase corresponds to that month. If the variable is evenly spread over all coordinates, then concentration is equal to zero and phase is undefined. If either modelled or observed values have zero values for all months in a given cell or site, then that cell/site is not included in the comparisons. Concentration comparisons are performed using `NME`. Modelled and observed phase are compared using mean phase difference:

$$mpd = (1/\pi).acos[cos(\omega - \phi)/n]$$

where ω is the phase from input x, and ϕ is from input y.

Value

Length comparisons return values as outlines in `NME`. Phase comparisons represent a fration of maximum possible timing error (i.e 6 months for a year).

Author(s)

Douglas Kelley `<douglas.i.kelley@gmail.com>`

References

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., & Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. Biogeosciences, 10(5), 3313-3340. doi:10.5194/bg-10-3313-2013

Examples

```
## World Data Center-C1 For Sunspot Index Royal Observatory of Belgium, Av. Circulair
## see ?sunspot.month

## Seasonal cycle of sunspots
full  = t(matrix(sunspot.month,nrow=12))
clim  = apply(full,2,mean)

testAndPlot <- function(mod) {
    m     = MPD(full,mod) # Full model test
    print(summary(m))     # Prints scores and obs/mod information

    # Perform and plot example for first 10 only
    m10   = MPD(full[1:10,],mod)
    plot(m10,lwd=2,labels=c('Jan','Jul'),radial.labels='',cex=2)
}
testAndPlot(clim)


## test seasonal climatology model
mnthN= rep(1:12,length.out=length(full))*pi/6
fullTS=as.vector(t(full))

climMod = predict(nls(fullTS ~ a*sin(b+mnthN) + c ,
                      start=c(a=1,b=1,c=1)))[1:12]

testAndPlot(climMod)


## test inter-annual sunspot cycle
fullTS=fullTS[1:2904] #2904 = 22 11-year cycles

#11-year cycle phase position
yearP = seq(0,11,by=1/12)*2*pi/11
yearP = rep(yearP,length.out=length(fullTS))

IAVMod= nls(fullTS ~ a*cos(b+yearP) + c, start=c(a=50,b=1,c=1))
IAVMod= predict(IAVMod)[1:(11*12)]  ## Repreated cycle so only consider fiirst 11 yea

Yr11Cycle <- function(d) t(matrix(d,nrow=(11*12)))
full11=Yr11Cycle(fullTS) # Transform observations into matri of 11-year cycles

m     = MPD(full11,IAVMod)
print(summary(m))
plot(m,lwd=2,labels=c('0','6.5'),radial.labels='',cex=2)
```

```
## Trying to find an even long cycle
longCycle <- function(n) { # n is number of months in longer cycle
    IAVMod= nls(fullTS ~ a*cos(b+yearP) +
                c*rep(1:n,length.out=length(fullTS)) + e ,
                start=c(a=50,b=1,c=1,e=1))
    IAVMod=predict(IAVMod)
    return(Yr11Cycle(IAVMod))
}

# Test length of cycle in stages of 1 yar between 11 and 100 years
cycleLength=seq(11*12,1200,by=12)
mods=lapply(cycleLength,longCycle)

# Find scores and transform into range of 0 to 1
scores=sapply(mods,function(i) score(MPD(full11,i)))
range01 <- function(x){(x-min(x))/(max(x)-min(x))}
scores = apply(scores,1,range01)


# Plot Phase and concentration scores
par(mfcol=c(2,1))
plot(range(cycleLength),c(0,1),type='n',yaxt='n',
     xlab='Cycle Length',ylab='score range')
axis(labels=c('min','max'),at=c(0,1),side=2)

for (i in 1:2) {
    y=scores[,i]
    col=rainbow(2)[i]
    lines(cycleLength,y,col=col)
    j=which.min(y)

    lines(rep(cycleLength[j],2),c(-1,2),cex=3,xpd=FALSE)
    lines(rep(cycleLength[j],2),c(-1,2),col=col,cex=3,lty=2,xpd=FALSE)
}
plot.new()
legend('top',colnames(scores)[1:2],col=rainbow(2),bty='n',ncol=2,lty=1)

legend('bottom',colnames(scores)[1:2],col='black',lty=1,bty='n',ncol=2)
legend('bottom',paste("min. score",colnames(scores)[1:2]),col=rainbow(2),lty=2,bty='r
```

NME *Normalised Mean (Squared) Error*

Description

Performs the Normalised Mean Error (NME) and Normalised Mean Squared Error (NMSE) comparison between model and observed.

Usage

```
NME (x, y, w = NULL, allowRegridding = TRUE)
NMSE(x, y, w = NULL, allowRegridding = TRUE)
```

Arguments

`x, y`	matrix, data.frame, raster or raster stacks. Where y is the approximation to x
`w`	vector or raster of weights or same size as x. If NULL, all items are considered equally.
`allowRegridding`	Logical arguement used if x and y are both raster. If TRUE and x and y are on different grids, then x and y are cropped to the smallest shared extent and lowest reslution of either x and y. If w is raster as well, this will also be regridded to smallest extent and lowest resolution of x or y.

Details

NME/NMSE measures the distance (or error) between model and observations.

$$NME = \Sigma|y_i - x_i|/\Sigma|x_i - mean(x)|$$

$$NMSE = \Sigma(y_i - x_i)^2/\Sigma(x_i - mean(x))^2$$

where y_i is the modelled value of variable x in grid cell (or at site) i, x_i the corresponding observed value, and $mean(x)$ the mean observed value across all grid cells or sites.

They both take the value of 0 for perfect agreement. A value of 1 is equivalent to a comparison between observations and the observtations mean value. Large numbers donote larger errors and therefore worse model performance.

The functions provide 3 measures: 1) straight NME/NMSE comparisons; 2) comparisons with the influcance of the mean removed (i.e testing pattern and variablity); 3) comparisons with the influance of the mean and variance removed (i.e testing the pattern only).

Value

NME and NMSE returns an object of class "NME".

The `print` print function returns the call information and scores for each step.

The function `summary` can be used to obtain information means and variances for x and y, as well as the ratio of x and ys mean and variance.

The `plot` function plots the steps in a scatter plot.

Author(s)

Douglas Kelley `<douglas.i.kelley@gmail.com>`

References

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., & Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. Biogeosciences, 10(5), 3313-3340. doi:10.5194/bg-10-3313-2013

Nash, Je., & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I-A discussion of principles. Journal of hydrology, 10(3), 282-290.

See Also

`print.NME`, `summary.NME`, `plot.NME`

Examples

```
## C. I. Bliss (1952) The Statistics of Bioassay. Academic Press.
## In McNeil, D. R. (1977) Interactive Data Analysis. New York: Wiley.
## see ?ToothGrowth

## Load data
Obs   = ToothGrowth[,'len' ]

## Construct Model
supp  = ToothGrowth[,'supp']
dose  = ToothGrowth[,'dose']
toothFairy = sample(c(TRUE,FALSE), length(Obs),replace=TRUE)
Mod   = predict(lm(Obs ~  supp + dose + toothFairy))/2 + 10

## Compare model errors using metrics
nme = NME (Obs, Mod)
nmse= NMSE(Obs, Mod)

## View metric info
nme
nmse
summary(nme)

## Plot metric info
par(mfrow=c(2,1))
plot(nme)
mtext("NME")
plot(nmse)
mtext("NMSE")


## Compare Models with and without toothfariy
with    = predict(lm(Obs ~  supp + dose + toothFairy))
without = predict(lm(Obs ~  supp + dose))

## Perform metrics
with    = NME (Obs,with)
without = NMSE(Obs,without)

## Plot with and without
par(mfrow=c(2,1))
plot(with)
mtext("with")
plot(without)
mtext("without")


## Example taken from ?brick
require(raster)
b = brick(system.file("external/rlogo.grd", package="raster"))

m2 = NME(b[[1]],b[[2]])
m3 = NME(b[[1]],b[[3]])

par(mfrow=c(2,1))
plot (m2)
```

```
mtext("2")
plot (m3)
mtext("3")
```

`null.FUN` *Null Benchmark Models*

Description

Performs mean and random-resampling null-models for given metric.

Usage

```
null.FUN (x, FUN, w = NULL, n = 1000, ...)
null.NME (x, ...)
null.NMSE(x, ...)
null.MM   (x, ...)
null.SCD (x, ...)
null.MPD (x, ...)
```

Arguments

`x`	matrix, data.frame, raster or raster stacks, matching the format descibed by the corrisonponding metric
`FUN`	A function to perform null models on. Calling e.g. `null.FUN(x, NME)` is equivalent to calling `null.NME(x)`
`w`	vector or raster of weights of the size relavent to the corrisponding metric. If NULL, all items are considered equally.
`n`	number of resampled bootstraps to be performed for the randomly-resampled null model.
`...`	Additional arguements used by the corrisponding metric and by `null.FUN`

Details

To facilitate interpretation of the scores, `null.FUN` compares a benchmark dataset `x` to a dataset of the same size, filled with: 1) the mean of the observations; and 2) a “randomized” datasets. Ramdomizing of datasets is performed using a bootstrapping procedure (Efron, 1979; Efron and Tibshirani, 1994), whereby a constructed dataset of the same dimensions as `x` is filled by randomly resampling the data in `x` with replacement 100 times to estimate a probability density function of “random-resampled” scores.

Value

null.FUN returns an object of class “nullModel”.

The `print` print function returns the score for all runs.

The function `summary` returns the mean model score(s) and the mean and `sd` of the randomly-resampled model.

The `plot` function plots a histagram of the probability density function (pdf) of radomly-reampled models, and indicated on this plot the `summary` information.

Author(s)

Douglas Kelley <douglas.i.kelley@gmail.com>

References

Efron, B. (1979). Bootstrap methods: another look at the jackknife. The annals of Statistics, 1-26.

Efron, B., & Tibshirani, R. J. (1994). An Introduction to the Bootstrap (Chapman & Hall/CRC Monographs on Statistics & Applied Probability).

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., & Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. Biogeosciences, 10(5), 3313-3340. doi:10.5194/bg-10-3313-2013

See Also

print.NME, summary.NME, plot.NME

Examples

```
######################################################################
## NME Null model                                                   ##
######################################################################
## C. I. Bliss (1952) The Statistics of Bioassay. Academic Press.
## In McNeil, D. R. (1977) Interactive Data Analysis. New York: Wiley.
## see ?ToothGrowth

Obs  = ToothGrowth[,'len' ]
nNME = null.NME(Obs)

summary(nNME)

plot(nNME)

## Plotting the affect of different sample sizes
summary4N <- function(n) {
  nNME = null.NME(Obs, n)
  nNME = unlist(summary(nNME))[c(1,4,5)]
  return( c(nNME[1:2], nNME[2] - nNME[3], nNME[2] + nNME[3]) )
}

# Set sample size
ns = 1:10
ns = c(ns, ns*10, ns*100, 1000)

# plot randomly-resampled mean and sd for each sample size
nTest = sapply(ns,summary4N)
plot(range(ns), range(nTest, na.rm = TRUE), type = 'n', xlab = "Sample size",
     ylab = "Score", log = "x")

lines(ns, nTest[1,], col = 'red' )
lines(ns, nTest[2,], col = 'blue')
polygon(c(ns, rev(ns)), c(nTest[3,], rev(nTest[4,])), col = "#0000FF55")

legend('topright', legend = c("Mean", "Randomly-Resampled"),
       col = c('red', 'blue'     ), lwd = 1)
legend('topright', legend = c("Mean", "Randomly-Resampled"),
        col = c('red', '#0000FF55'), lwd = c(1, 10))

######################################################################
## MM Null model                                                    ##
######################################################################
## NPP allocation observations from Schuur & Matson (2001)
Obs =cbind(canopy = c(4.25 ,5.0  ,4.25 ,4.5  ,3.0  ,2.0 ),
           stem   = c(0.37 ,0.12 ,0.62 ,0.12 ,0.75 ,0.05),
           wood   = c(0.62 ,0.21 ,1.03 ,0.21 ,1.24 ,0.08))

nMM = null.MM(Obs)

summary(nMM)
plot(nMM)

######################################################################
## MPD Null model                                                   ##
######################################################################
## World Data Center-C1 For Sunspot Index Royal Observatory of Belgium, Av. Circulaire,
## see ?sunspot.month

## Seasonal cycle of sunspots
Sunspots  = t(matrix(sunspot.month,nrow=12))
nMPD = null.MPD(Sunspots)

plot(nMPD, legend = TRUE)
```

`PolarConcentrationAndPhase` *Polar Concentration And Phase*

Description

Calculates the concentration and phase of a polar data

Usage

```
PolarConcentrationAndPhase(cdata, phase_units = "radians", n = 12,
                           disag = FALSE, justPhase = FALSE)
```

Arguments

`cdata`	Either vector or matrix. If vector, each element reprents equally spaced points along a polar coordinate system. If matrix, each column represnets equally spaced position along a polar coordinate system and each row a new series of measurments.
`phase_units`	Units of the phase outputs. Default is `"radians"` ($-\pi$ to π), but there is also a choice of `"degrees"` (-180 to 180) and `"months"` (-6 months to 6 months)
`n`	length of cycle phase and concentration is calculated over. Default number of columns for matrix or 12 (i.e, 12 in a year) for raster stack or brick. If dat is longer (i.e, if \cidencol(dat) for matrix or number of layers for raster objects is > n), a a 'climatology' is calculated at base n.

`disagFact`	Only used if dat is a raster brick or stack. Disaggregatation factor used by `disaggregate` to smooth raster outputs. Useful fi neater plotting. Default is NaN which does not perform disaggregatation.
`justPhase`	Logical only used if dat is a raster brick or stack. . If `TRUE`, just returns the phase metric.

Details

Each simulated or observed timestep (e.g month) with in the period (i.e year) is represented by a vector in the complex plane, the length of the vector corresponding to the magnitude of the variable for each period and the directions of the vector corresponding to the timeing within the period. It is assumed each timestep is uniformly distributed:

$$\theta_t = 2.\pi(t-1)/n$$

where n is the number of timesteps t in the period.

A mean vector L is calculated by averaging the real and imaginary parts of the n vectors, x.

$$Lx = \Sigma x cos(\theta)$$

and

$$Ly = \Sigma x sin(\theta)$$

The length of the mean vector divided by the annual value stands for seasonal concentration, C; its direction stands for phase, P:

$$C = (Lx^2 + Ly^2)/\Sigma x$$

$$P = atan(Lx/Ly)$$

Thus, if the variable is concentrated all in one point within the polar coordinates, seasonal concentration is equal to 1 and the phase corresponds to that month. If the variable is evenly spread over all coordinates, then concentration is equal to zero and phase is undefined.

Value

Two compoments are returned, each of the length of the first dimension of cdata input, or length 1 if cdata is a vector

`phase`	the phase timing of each row of the inputs (see details above)
`conc`	the concentration around the phase of each row of the inputs (see details above)

Author(s)

Douglas Kelley `<douglas.i.kelley@gmail.com>`

References

Kelley, D. I., Prentice, I. C., Harrison, S. P., Wang, H., Simard, M., Fisher, J. B., & Willis, K. O. (2013). A comprehensive benchmarking system for evaluating global vegetation models. Biogeosciences, 10(5), 3313-3340. doi:10.5194/bg-10-3313-2013

See Also

`MPD`

Examples

```
require(plotrix)
################################################################################
## matrix                                                                     ##
################################################################################
## Average Monthly Temperatures at Nottingham, 1920-1939
## Anderson, O. D. (1976) Time Series Analysis and Forecasting: The
## Box-Jenkins approach. Butterworths. Series R.
## see ?nottem

## Load T
T       = t(matrix(nottem,nrow=12))

## Calculate seasonal climatology and angle of each month in degrees
climT      = apply(T, 2, mean)
climT[1:6] = climT[1:6]
periods    = head(seq(0, 360, length.out = 13), -1)

## Plot climatology
polar.plot(climT, periods,
           labels = c('J','F','M','A','M','J','J','A','S','O','N','D'),
           label.pos = periods, radial.labels = '', radial.lim = c(25,62),
           rp.type = 'p', poly.col = '#FF0000AA')

scaleConc <- function(i) min(climT) + i * diff(range(climT))

## Calculate phase and concentraion.
pc      = PolarConcentrationAndPhase(climT, phase_units = "degrees")
phase   = pc[[1]][1]

## Covert concentration to point on tempurature plot
conc    = scaleConc(pc[[2]][1])

## Plot climatology phase on concentration on plot
polar.plot(conc, phase, point.symbol = 4,radial.lim = c(25,62),
           rp.type = 'rs', cex = 2, lwd = 2, add = TRUE)

## same calculation and plot or each year.
pc      = PolarConcentrationAndPhase(T, phase_units = "degrees")
phase   = pc[[1]]
conc    = scaleConc(pc[[2]])

polar.plot(conc, phase, point.symbol = 16, radial.lim = c(25,62),
           rp.type = 'rs', cex = 1, add = TRUE, line.col = 'black')

################################################################################
## Raster                                                                     ##
################################################################################
require(raster)

b = brick(system.file("external/rlogo.grd", package = "raster"))
b = PolarConcentrationAndPhase(b)
dev.new()
plot(b)

b  = b[[2]]
```

```
b0 = b*2*pi
for (i in 1:12) {
    bi = cos(pi *i/12 + b0)
    b  = addLayer(b, bi)
}

b    = dropLayer(b, 1)
maxb = which.max(b)
phsb = PolarConcentrationAndPhase(b, phase_units = 'months', justPhase = TRUE)

dev.new()
par(mfrow = c(3, 1))
plot(maxb, main = 'max layer')
plot(phsb, main = 'phase')
plot(values(maxb), values(phsb), pch = 4)
```

ts2matrix *Time Series to Matrix*

Description

Comverts Time Series Class to Matrix

Usage

```
ts2matrix(ts)
```

Arguments

ts	An object of class ts

Value

The resultant matrix appears in the same shape with same values as the vector or matrix when using print(ts) with the same colnames and rownames as displayed by print(x). However, the resultant matrix is usable (not just displayable) as a matrix

Author(s)

Douglas Kelley <douglas.i.kelley@gmail.com>

See Also

ts

Examples

```
## Measurements of the annual flow of the river Nile at Ashwan 1871-1970.
## Durbin, J. and Koopman, S. J. (2001) _Time Series Analysis by State Space
## Methods._ Oxford University Press.  http://www.ssfpack.com/DKbook.html

print(Nile)
Nile  = ts2matrix(Nile)
```

```
print(Nile)

## Quarterly TS of UK gas consumption from 1960Q1 to 1986Q4, in millions of
## therms.
## Durbin, J. and Koopman, S. J. (2001) _Time Series Analysis by State Space
## Methods._ Oxford University Press. http://www.ssfpack.com/dkbook/

print(UKgas)
UKgas = ts2matrix(UKgas)
print(UKgas)

## Monthly TS of Atmospheric concentrations of CO2 are expressed in parts per
## million (ppm) and reported in the preliminary 1997 SIO manometric mole
## fraction scale.
## Keeling, C. D. and Whorf, T. P., Scripps Institution of Oceanography (SIO),
## University of California, La Jolla, California USA 92093-0220.
## ftp://cdiac.esd.ornl.gov/pub/maunaloa-co2/maunaloa.co2

co2
co2   = ts2matrix(co2)
print(co2)
```

Index