Package 'PolarMetrics'

September 27, 2018

Type Package

Title Functions for analyzing seasonal patterns in environmental variables

Version 0.1.0

Author Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

Maintainer Bjorn J. Brooks <bjorn@geobabble.org>

Depends R (>= 2.10),
zoo

Suggests plotrix,
xts

Description Polar coordinate based functions useful for analyzing seasonal patterns in environmental variables, which reduces the multitude of input values to a smaller set of seasonal metrics (e.g., length of season, seasonality) that quantify the features of each annual (or diel) cycle. Functions are generalized to work on any time series with cycles of any length (daily, annual, interannual), but were developed for improved interannual analysis of land surface phenology data (NDVI).

License GPL (>=3)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

R topics documented:

`PolarMetrics-package` *Functions for analyzing seasonal patterns*

Description

This package can be used on any uniformly sampled environmental time series containing a periodic cycle (i.e., temperature, CO2, NDVI, stream flow). These functions transform and reproject time series on to a polar plane and calculate various measures of seasonal timing, asymmetry and amplitude. Timing metrics are expressed either on the day-of-year calendar or as days on an empirically defined calendar. Although developed to improve interpretation and analysis for NDVI time series these functions work on other types of environmental data subject to daily, annual or interannual cycles.

Details

`PolarMetrics` is a family of primarily trigonometric functions written for extracting polar measures from uniformly sampled time series. These functions ultimately output a set of timing and magnitude metrics that describe attributes of each year (cycle), such as seasonality (deciduousness). The metrics are useful for subsequent analysis such as clustering and factor analysis and allow the analysis to be tailored to a specific season of interest. The `window_idx` function controls the interval over which measures are calculated. Timing metrics can be output in day-of-year calendar (1-365 corresponding to Jan. 1-Dec. 31), or the empirical-year of the time series (days 1-365 starting from the average point of least activity). Although uncommon, note that if the empirical-year and calendar years are synchronized the polar metrics output will have the same number of years as the input data. However, in most cases the empirical-year in the data is offset and thus the output measures will have one less year of data than in the input, because the values preceeding the first offset are discarded as well as the values in the final year following the last complete year.

Table 1. Example output of derived polar metrics for MODIS NDVI from 2000-2002 for a single pixel covering the Coweeta Hydrologic Lab eddy flux tower in North Carolina. (See list of variable definitions below)

Year	ES	EMS	MS	LMS	LS	Sintv	Savg	Ssd	Smag	EMSma	LMSma
2000	98	162	210	258	306	208	0.702	0.126	0.397	0.536	0.699
2001	98	170	218	258	306	208	0.704	0.120	0.390	0.522	0.730
2002	106	170	218	266	314	208	0.726	0.111	0.406	0.571	0.709

Interpreting polar graphs: At first glance polar plots can be difficult to inspect, as compared to more familiar time series representations. However, polar graphs have several benefits over linear time traces. By wrapping data points radially around the center, deviations from the typical timing and progression of the seasonal cycle are easier to notice. For example the broad width of orbits in the polar plot in Fig. 1(b) indicate that there is significant difference among years, but most of this orbital width is constrained to December through February revealing that most of this interannual change is due to changes in the winter vegetation signature (evergreen component of the vegetation). Another visual advantage of polar plots is that changes in the timing of the season become more obvious. For example, during September the breadth in orbital path in Fig. 1(b) is constrained, indicating little interannual variability at that point in the year. However, during October the path is broader (more variable), revealing that October is the more variable month.

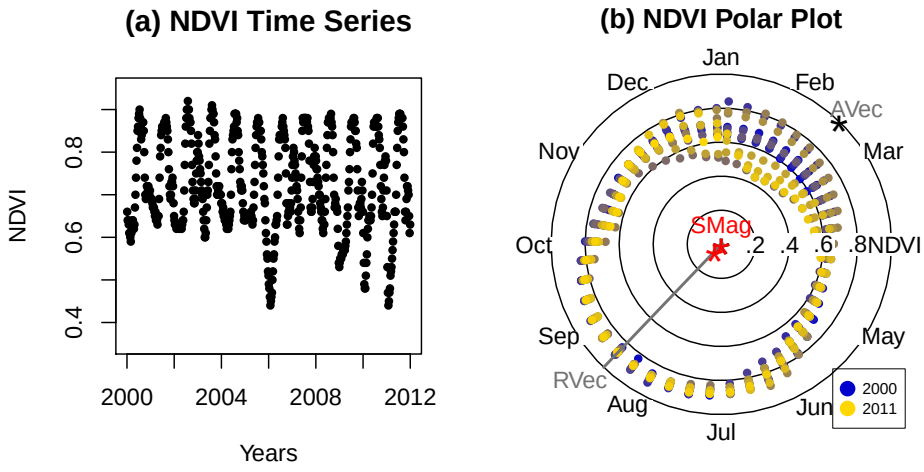


Figure 1. Time series and polar plot comparison. Data are from twelve years of MODIS NDVI measurements for one pixel (aprox. 250 x 250 m resolution) covering the Willow Creek AmeriFlux tower in northern Wisconsin. (a) shows a conventional time series. (b) shows the same data when graphed in polar coordinates. `RVec` (Resultant Vector, calculated by `vec_ang`) marks the angle of the average polar vector, which points toward mid-August. `AVec` is its opposite (calculated by `avec_ang`), which points at the period of least activity during the average cycle and is also the starting point for the average phenological year. `SMag` (calculated by `vec_mag`), shows the degree of seasonality (length of red line). The length of `SMag` is small because it is the average displacement of all vectors from the center of the polar graph, half of which have negative angles. (This figure produced by code in Example 3 below).

Interpreting polar metrics: Polar metrics are annual quantities that measure the timing and asymmetry of annual cycles. Consider, for example, an evergreen forest, as measured by NDVI. Its timing metrics will be more or less equally spaced and its seasonality will be small. On the other hand a landscape dominated by deciduous trees, because it has larger differences in greenness between winter and summer, will have a higher degree of seasonality as well as timing dates that are skewed closer to the middle of the growing season. This package calculates metrics that describe timing asymmetry and amplitude, which allows changes to be compared across years.

Calculating timing metrics: Threshold completion measures are calculated within each cycle (year) by starting from the point in the polar year corresponding to the shortest vector lengths (See `AVec` Fig. 1(b) and start of phenological year in Fig. 2). Values, in this case NDVI, are accumulated within each year (see `sum_cycle`) and each data point is assigned a corresponding cumulative percentage based on its proportional value within that year's total. User defined timing thresholds (e.g., 15% and 80%) are then assigned (see `window_idx`) as the time (day of year) when the data first crossed a given threshold. In these examples timing metrics are measured in days since the start of the calendar year or alternatively the offset/phenological year. (Phenological year days have the advantage of controlling for latitudinal, elevational, or other geographical bias in phenology timing.) Take for example the early season timing metric (ES) on the phenological year and assuming a 15% threshold. An `ES` value of 65 days means that 65 days passed until at least 15% of the year's total NDVI accumulated. The `window_idx` function is written such that timing metrics can occur on integer value days that were actually sampled, so the 15% threshold actually corresponds to the first sample day when 15% or more of the year's cumulative total NDVI was achieved. Fig. 2 provides an illustration of how the threshold completion measures were calculated for one example year.

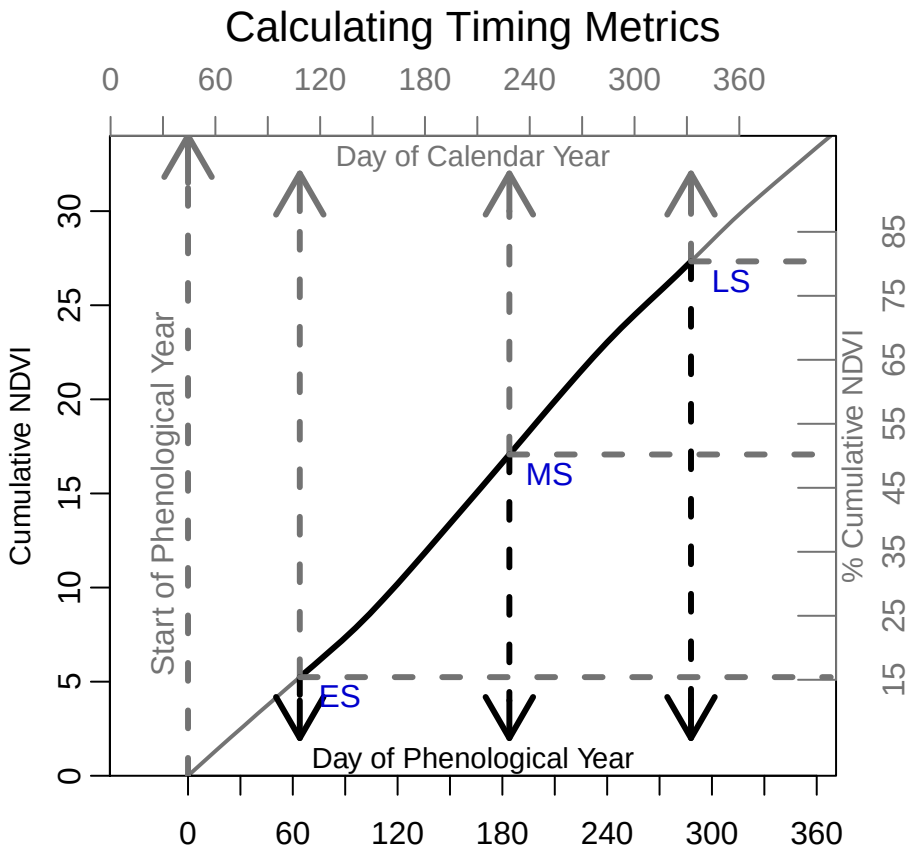


Figure 2. Timing metrics. Data correspond to year two (2001) of NDVI at Willow creek. Starting from the day corresponding to the average low point in the seasonal cycle, NDVI is cumulatively summed within the phenological year. The early season metric (ES) marks the first day when at least 15% of cumulative NDVI was reached (day 107 of calendar year or 65 on phenological year. See example code for `window_idx` function). (This figure produced by code in Example 4 below)

`AVec` **(start/end of phenological yr)** Number of days (offset) between Jan. 1 and the point of least activity (shortest vectors in the polar plot)

`Early season (ES)` Number of days to cross first threshold in cumulative annual NDVI (e.g., 15%).

`Early-mid season (EMS)` Number of days to cross the midpoint (e.g., 32.5%) between first threshold (e.g., 15%) and midpoint (50%).

`Mid-season (MS)` Number of days required to cross 50% threshold

`Mid-late season (LMS)` Number of days to cross the midpoint (e.g., 67.5%) between mid-season (50%) and midpoint (e.g., 80%).

`Late season (LS)` Number of days to cross late season threshold (e.g., 80%)

`Length of season (LOS)` Number of days between early and late season thresholds

Calculating asymmetry and amplitude metrics: These power measures are calculated from the values falling within the `ES` and `LS` thresholds. If `window_idx` is set to search the entire year

(e.g., by setting `p1=0` and `p2=1`) then asymmetry and amplitude measures will be based on the entire phenological year of data. The list of these measures is as follows:

`Savg` The average NDVI value between early and late thresholds (e.g., 15-80%)

`Ssd` The standard deviation of NDVI values between early and late thresholds

`Smag` **(seasonality/deciduousness)** The length (in units of input data) of the average vector between ES and LS timing points

`EMSmag` **("spring" seasonal magnitude)** The length of the average vector between ES and MS timing points

`LMSmag` **("fall" seasonal magnitude)** The length of the average vector between MS and LS timing points

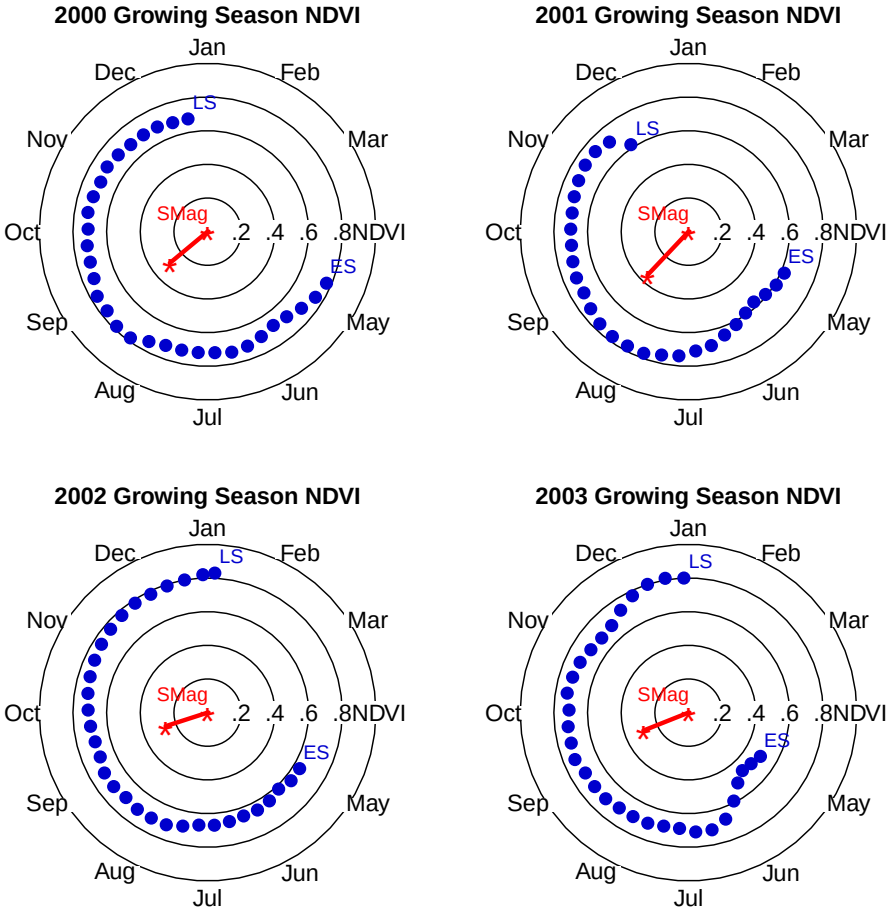


Figure 3. Polar plots of timing and asymmetry metrics for four years at Fraser Experimental Forest in Colorado. Each panels indicates annual changes in timing and asymmetry metrics. Note that the angle of `SMag` corresponds to the timing of mid season (MS), but the length of `SMag`, seasonality, is actually a measure of the degree to which data are massed in the direction of the `MS` date. These years in particular represent the early stages of mortality in lodgepole pines at FEF caused by bark beetles. Early season NDVI (April) appeared to start a variable decline in 2001, meanwhile summer NDVI remained fairly constant (perhaps due to compensatory greening from understory vegetation). Note that this area had been under significant drought stress and we do not intend to suggest this as a system for diagnosing specific disturbances, only that this technique is capable of measuring changes caused by environmental shifts. (This figure produced by code in Example 5 below)

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

Maintainer: Bjorn J. Brooks <bjorn@geobabble.org>

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (accepted). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
### Example 1, (Table 1 above) Use easy function to calculate polar vectors
library(PolarMetrics)
data(mndvi)               # Load data
lb <- 0.15                # Cutoff point, as a proportion of 1 (0-0.5), to
                          # remove from each years cumulative total.
                          # e.g., 0.1 will remove from 0 to 0.1.
ub <- 0.8                 # Cutoff point, as a proportion of 1 (0.5-1), to
                          # remove at the end of each years cumulative total.
                          # e.g., 0.9 will remove 0.9 to 1.
s_p_yr <- 46              # Number of MODIS values per year (8-day NDVI)
# Calculate polar metrics for one MODIS NDVI pixel within Coweeta Hydrologic Lab
pmets <- calc_metrics(mndvi$chl, t=mndvi$day, yr_type='cal_yr',
  spc=s_p_yr, lcut=lb, hcut=ub, return_vecs=FALSE, sin_cos=FALSE)

### Example 2, Long version of Example 1
library(PolarMetrics)
pmets <- data.frame(yr=NA, es=NA, ems=NA, ms=NA, lms=NA, ls=NA,
                        s_intv=NA, s_avg=NA, s_sd=NA, s_mag=NA,
                        ems_mag=NA, lms_mag=NA) # Initialize output
dpy <- 365                # Days/year
lb <- 0.15                # Lower cutoff for cumulative annual NDVI
ub <- 0.8                 # Upper cutoff for cumulative annual NDVI
c <- 12                   # Num. of years/cycles
cxs <- 1                  # Text scaling in plots
spy <- 46                 # Number of samples in one cycle (yr)
data(mndvi)               # Load data
t <- as.vector(mndvi$day) # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)    # Transform days of year to radians
v <- as.vector(mndvi$chl) # MODIS NDVI over eddy flux tower at Coweeta Hyd. Lab
VX <- vec.x(r,v)          # Avg horizontal vector
VY <- vec.y(r,v)          # Avg vertical vector
vx <- mean(VX, na.rm=TRUE) # Avg horizontal vector
vy <- mean(VY, na.rm=TRUE) # Avg vertical vector
rv <- vec_mag(vx,vy)      # Magnitude (length) of average vector
rv_ang <- vec_ang(vx,vy)  # Angle of the avg vector (phenological median)
avec_ang <- avec_ang(rv_ang)   # Vert opposite of med (avg NDVI min/pheno yr start)
pyStartDay <- rad2d(avec_ang, dpc=dpy) # Cal dy (1-365 beg at Jan 1) marking NDVI min
pyStartIdx <- rad2idx(avec_ang, spc=spy) # Index marking point of least activity
CumNDVI <- sum_cycle(v,pyStartIdx,spy)$cumsum # Cumulative NDVI within each pheno yr
  # Note, output is re-centered, and accumulates starting from the annual NDVI
  # minimum (PyStartIdx) and has c-1 yrs of data due to re-centering.
c.CumNDVI <- c-1          # Number of years in CumNDVI
# Calculate phenologicalc param's for year 2000
cy <- 1                       # The first yr of data (which is 2000 here)
es.idx <- window_idx(CumNDVI,c.CumNDVI,cy,lb,ub)[1] # earlySsn CumNDVI index
```

```
ems.idx <- window_idx(CumNDVI,c.CumNDVI,cy,lb,ub)[2] # earlySsn CumNDVI index
ms.idx <- window_idx(CumNDVI,c.CumNDVI,cy,lb,ub)[3] # midSsn CumNDVI index
lms.idx <- window_idx(CumNDVI,c.CumNDVI,cy,lb,ub)[4] # midSsn CumNDVI index
ls.idx <- window_idx(CumNDVI,c.CumNDVI,cy,lb,ub)[5] # lateSsn CumNDVI index
# 3 variables below are days calculated relative to the phenological year, i.e.
# calculated starting from avg NDVI minimum. To get the day relative to the
# calendar year (e.g. 1=Jan 1, 2=Jan 2...) add pyStartDay.
pmets$yr <- cy + 1999
pmets$es <- round(t[es.idx] + pyStartDay)  # Day of calendar yr marking early season milestone
pmets$ems <- round(t[ems.idx] + pyStartDay) # Day of calendar yr marking early-mid season
pmets$ms <- round(t[ms.idx] + pyStartDay)  # Day of calendar yr marking mid season
pmets$lms <- round(t[lms.idx] + pyStartDay) # Day of calendar yr marking late-mid season
pmets$ls <- round(t[ls.idx] + pyStartDay)  # Day of calendar yr marking late season
pmets$s_intv <- pmets$ls - pmets$es # Length of the season from ES to LS
pmets$s_avg <- mean(v[es.idx:ls.idx], na.rm=TRUE) # Mean NDVI for Ssn
pmets$s_sd <- sd(v[es.idx:ls.idx]) # Standard deviation of NDVI for Ssn
pmets$s_mag <- vec_mag(mean(VX[es.idx:ls.idx], na.rm=TRUE),
  mean(VY[es.idx:ls.idx], na.rm=TRUE)) # Magnitude (length) of average vector
    # Magnitude of avg vec between ES & MS thresholds
pmets$ems_mag <- vec_mag(mean(VX[es.idx:(ms.idx-1)], na.rm=TRUE),
  mean(VY[es.idx:(ms.idx-1)], na.rm=TRUE)) # Magnitude of avg vec between ES & MS
pmets$lms_mag <- vec_mag(mean(VX[ms.idx:(ls.idx-1)], na.rm=TRUE),
  mean(VY[ms.idx:(ls.idx-1)], na.rm=TRUE)) # Magnitude of avg vec between MS & LS
print(pmets)

### Example 3, (see Fig. 1 above) Transform into polar coordinates and compare
library(PolarMetrics)
library(plotrix)           # Load plotrix for plotting
dev.new()
dpy <- 365                 # Days/year
c <- 12                    # Num. of years/cycles
data(mndvi)                # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)     # Transform days of year to radians
v <- as.vector(mndvi$wc)   # MODIS NDVI for Willow Creek flux tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
rv <- vec_mag(vx,vy)       # Magnitude (length) of average vector
rv_ang <- vec_ang(vx,vy)   # Angle of the avg vector (phenological median)
avec_ang <- avec_ang(rv_ang)   # Vert opposite of med (avg NDVI min/pheno yr start)
dev.new(width=12,height=6) # New plot window
par(mfrow=c(1,2))          # Make this a multi-pane plot
cxs <- 1                   # Text scaling in plots
# Time series plot of Willow Creek
plot(2000+(t/dpy), v, pch=20, col='black', xlab='Years',
  ylab='NDVI', main='(a) NDVI Time Series',
  ylim=c(0.35,0.95), cex=cxs, cex.axis=cxs, cex.lab=cxs, cex.main=cxs*1.25)
# Polar plot phenology variables
ndvilabs <- c('','.2','.4','.6','.8','') # labels for radial plot
s.pos <- pi/2 # Radial position to start plotting from
lab.pos <- c(seq(from=0, to=2*pi-(2*pi/12), by=(2*pi)/12))[-4]
rad.labs <- c(month.abb[seq(from=1, to=12)])[-4]
clrs <- colorRampPalette(c('blue3', 'gold'))(length(r)) # Color ramp for plot
# Polar plot of Willow Creek
radial.plot(v,r,clockwise=TRUE,start=s.pos,
  label.pos=lab.pos,labels=rad.labs,radial.labels=ndvilabs,
  rp.type='s',point.symbols=20,point.col=clrs,radial.lim=c(0,1),
```

```
show.radial.grid=FALSE,main='(b) NDVI Polar Plot',
grid.col='black',grid.unit='NDVI')
radial.plot(c(0,1),c(0,rv_ang),
clockwise=TRUE,start=s.pos,rp.type='r',
lwd=2,line.col='gray45',add=TRUE) # rv_ang, Angle of avg vec
radial.plot(c(0,1),c(0,avec_ang),
clockwise=TRUE,start=s.pos,rp.type='s',
point.symbols='*',cex=cxs*2,
add=TRUE) # avec_ang, opposite angle of rv_ang
radial.plot(c(0,rv),c(rv_ang,rv_ang),
clockwise=TRUE,start=s.pos,rp.type='r',
lwd=3,line.col='gray45',add=TRUE) # rv, Magnitude of avg vec
radial.plot(c(0,rv),c(rv_ang,rv_ang),
clockwise=TRUE,start=s.pos,rp.type='s',
point.symbols='*',cex=cxs*2,
point.col='red',add=TRUE)   # rv, Magnitude of avg vec
text(sin(avec_ang)*1.15, cos(avec_ang)*1.1,
'AVec', col='gray45', cex=cxs) # Add text label
text(sin(rv_ang)*1.2, cos(rv_ang)*1.1,
'RVec', col='gray45', cex=cxs) # Add text label
text(0,0.1,'SMag',col='red',cex=cxs) # Add text label
legend('bottomright',c('2000','2011'),col=c('blue3','gold'),
  pch=20,cex=0.7,pt.cex=2,box.lwd=0)
par(xpd=FALSE)
print('Note that 1 year of data is removed, because polar transformation')
print('shifts and centers the data on the phenological year.')


### Example 4, (see Fig. 2 above) Plot cum. NDVI and specify completion milestones
library(PolarMetrics)
dev.new()
dpy <- 365                 # Days/year
c <- 12                    # Num. of years/cycles
spy <- 46                  # Samples/year (or samples/cycle)
data(mndvi)                # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)     # Transform days of year to radians
v <- as.vector(mndvi$wc)   # MODIS NDVI for Willow Creek flux tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
rv_ang <- vec_ang(vx,vy)   # Angle of the avg vector (phenological median)
avec_ang <- avec_ang(rv_ang)   # Vert opposite of med (avg NDVI min/pheno yr start)
pyStartDay <- rad2d(avec_ang, dpc=dpy) # Cal dy (1-365 beg at Jan 1) marking NDVI min
pyStartIdx <- rad2idx(avec_ang, spc=spy) # Index marking point of least activity
CumNDVI <- sum_cycle(v,pyStartIdx,spy)$cumsum # Cumulative NDVI within each pheno yr
  # Note, output is re-centered, and accumulates starting from the annual NDVI
  # minimum (PyStartIdx) and has c-1 yrs of data due to re-centering.
c.CumNDVI <- c-1           # Number of years in CumNDVI
cy <- 2 # Year within which to examine and plot cumulative NDVI
es <- window_idx(CumNDVI,c.CumNDVI,cy,0.15,0.8)[1] # Indx of CumNDVI marking ES
ms <- window_idx(CumNDVI,c.CumNDVI,cy,0.15,0.8)[3] # Indx of CumNDVI marking MS
ls <- window_idx(CumNDVI,c.CumNDVI,cy,0.15,0.8)[5] # Indx of CumNDVI marking LS
c.CumNDVI <- c-1           # Number of years in CumNDVI
t2 <- seq(pyStartDay,by=8,length.out=47)
y <- c(0,CumNDVI[(spy+1):(2*spy)])
dev.new(width=4,height=4) # New plot window
cxs <- 1                   # Text scaling in plots
```

```
plot(t2,y,type='l', xlim=c(15,400),
xaxt='n', ylim=c(0,max(y)),yaxs='i', yaxt='n', xlab='', ylab='',
col='gray45', lwd=cxs*2,cex=cxs,cex.axis=cxs,cex.lab=cxs,cex.main=cxs)
mtext('Calculating Timing Metrics', col='black',
side=3, line=2.3, cex=cxs*1.35) # Main title
y1 <- seq(0,30,by=5)
y2 <- seq(15,85,by=10)
x1 <- seq(0,360,by=30)
x2 <- seq(0,360,by=30)
axis(side=2, labels=y1, at=y1, padj=0.7, col='black', col.axis='black',
cex.axis=cxs) # Plot left y axis
mtext('Cumulative NDVI', col='black',
side=2, line=1.9, cex=cxs*0.9)
## x1 axis
axis(side=1, labels=x1, at=x1+pyStartDay, col='black', col.axis='black',
cex.axis=cxs) # Plot right y axis
## x2 axis
par(tcl = -0.5) # Switch back to outward facing tick marks
axis(side=3, labels=x2, at=x2, col='gray45', col.axis='gray45',
cex.axis=cxs) # Plot right y axis
par(tcl = 1) # Switch to inward facing tick marks
mtext('Day of Calendar Year',side=3,line=-1,
col='gray45',cex=cxs*0.9)
mtext('Day of Phenological Year',side=1,line=-1,
col='black',cex=cxs*0.9)
##
axis(side=4, labels=y2, at=y2/(100/max(y)), col='gray45', col.axis='gray45',
cex.axis=cxs) # Plot right y axis
mtext('% Cumulative NDVI',side=4,line=-0.1,
col='gray45',cex=cxs*0.9)
lines(t2[(es-spy+1):(ls-spy+1)],
y[(es-spy+1):(ls-spy+1)], col='black', lwd=cxs*3)
#
segments(pyStartDay,0,pyStartDay,max(y),
col='gray45',lwd=cxs*3, lty=2) # Phen yr offset dahsed line
arrows(pyStartDay,31.5,pyStartDay,max(y),
col='gray45',lwd=cxs*3) # Phen yr offset arrow head
text(pyStartDay-13,15,srt=90,'Start of Phenological Year',cex=cxs,col='gray45')
#
segments(t2[es-spy+1],y[(es-spy+1)],t2[es-spy+1],30,
col='gray45',lwd=cxs*3, lty=2) # ES vertical dashed line (calendar yr)
arrows(t2[es-spy+1],30,t2[es-spy+1],32,
col='gray45',lwd=cxs*3) # ES arrow head (calendar yr)
segments(t2[es-spy+1],y[(es-spy+1)],t2[es-spy+1],4,
col='black',lwd=cxs*3, lty=2) # ES vertical dashed line (phen yr)
arrows(t2[es-spy+1],4,t2[es-spy+1],2,
col='black',lwd=cxs*3) # ES arrow head (phen yr)
segments(t2[es-spy+1],y[(es-spy+1)],max(t2),y[(es-spy+1)],
col='gray45',lwd=cxs*3,lty=2) # ES horizontal dashed line
text(t2[es-spy+1]+23,y[(es-spy+1)]-1,srt=0,'ES',cex=cxs,col='blue3')
#
segments(t2[ms-spy+1],y[(ms-spy+1)],t2[ms-spy+1],30,
col='gray45',lwd=cxs*3, lty=2) # MS vertical dashed line (calendar yr)
arrows(t2[ms-spy+1],30,t2[ms-spy+1],32,
col='gray45',lwd=cxs*3) # MS arrow head (calendar yr)
segments(t2[ms-spy+1],y[(ms-spy+1)],t2[ms-spy+1],4,
col='black',lwd=cxs*3, lty=2) # MS vertical dashed line (phen yr)
```

```
arrows(t2[ms-spy+1],4,t2[ms-spy+1],2,
col='black',lwd=cxs*3) # MS arrow head (phen yr)
segments(t2[ms-spy+1],y[(ms-spy+1)],max(t2),y[(ms-spy+1)],
col='gray45',lwd=cxs*3,lty=2) # MS horizontal dashed line
text(t2[ms-spy+1]+23,y[(ms-spy+1)]-1,srt=0,'MS',cex=cxs,col='blue3')
#
segments(t2[ls-spy+1],y[(ls-spy+1)],t2[ls-spy+1],30,
col='gray45',lwd=cxs*3, lty=2) # LS vertical dashed line (calendar yr)
arrows(t2[ls-spy+1],30,t2[ls-spy+1],32,
col='gray45',lwd=cxs*3) # LS arrow head (calendar yr)
segments(t2[ls-spy+1],y[(ls-spy+1)],t2[ls-spy+1],4,
col='black',lwd=cxs*3, lty=2) # LS vertical dashed line (phen yr)
arrows(t2[ls-spy+1],4,t2[ls-spy+1],2,
col='black',lwd=cxs*3) # LS arrow head (phen yr)
segments(t2[ls-spy+1],y[(ls-spy+1)],max(t2),y[(ls-spy+1)],
col='gray45',lwd=cxs*3,lty=2) # LS horiztontal dashed line
text(t2[ls-spy+1]+23,y[(ls-spy+1)]-1,srt=0,'LS',cex=cxs,col='blue3')
#
par(tcl = -0.5) # Switch back to outward facing tick marks


### Example 5, (see Fig. 3 above) Plot 4 pheno yrs & show phen params of grow.seas.
library(PolarMetrics)
library(plotrix)           # Load plotrix for plotting
dev.new()
lb <- 0.15                 # Lower cumulative sum threshold for window_idx
ub <- 0.8                  # Upper cumulative sum threshold for window_idx
dpy <- 365                 # Days/year
c <- 12                    # Num. of years/cycles
spy <- 46                  # Number of samples in one cycle (yr)
data(mndvi)                # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)     # Transform days of year to radians
v <- as.vector(mndvi$fef)  # MODIS NDVI for pixel covering Fraser Exp. Forest
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
rv_ang <- vec_ang(vx,vy)   # Angle of the avg vector (phenological median)
avec_ang <- avec_ang(rv_ang)   # Vert opposite of med (avg NDVI min/pheno yr start)
pyStartIdx <- rad2idx(avec_ang, spc=spy) # Index marking point of least activity
CumNDVI <- sum_cycle(v,pyStartIdx,spy)$cumsum # Cumulative NDVI within each pheno yr
  # Note, output is re-centered, and accumulates starting from the annual NDVI
  # minimum (PyStartIdx) and has c-1 yrs of data due to re-centering.
c.CumNDVI <- c-1           # Number of years in CumNDVI
# Next remove data prior to first NDVI minimum and after last NDVI min
idx=sum_cycle(v,pyStartIdx,spy)$vidx # Indices of v from start of first pheno yr
  # (NDVI min) to end of last complete phenological yr
v1.gs=v[idx]
r1.gs=r[idx]
# Polar plot phenology variables
cxs <- 1                   # Text scaling in plots
ndvilabs <- c('','.2','.4','.6','.8','') # labels for radial plot
s.pos <- pi/2 # Radial position to start plotting from
lab.pos <- c(seq(from=0, to=2*pi-(2*pi/12), by=(2*pi)/12))[-4]
rad.labs <- c(month.abb[seq(from=1, to=12)])[-4]
# Calculate threshold completion milestones for 4 consecutive years
for (I in 1:(c-1)) {
  es.ms.ls=window_idx(
```

```
    CumNDVI,c.CumNDVI,I,lb,ub) # Indices for early, mid, late seas.
  idx2=es.ms.ls[1]:es.ms.ls[5] # Indices within this yrs season
  if (I==1) {
    df=cbind(r1.gs[idx2],v1.gs[idx2],I)
  } else {
    df=rbind(df,cbind(r1.gs[idx2],v1.gs[idx2],I))
  }
}
# Combine the calculated polar measures into a data frame
df1=data.frame(df)
colnames(df1)=c('radians','ndvi','yr')
# Plot the 4 years of polar measures on 4 polar plots
dev.new(width=6,height=12)
par(mfrow=c(2,2))
for (I in 1:4) {
  df1.tmp=df1[df1$yr==I,]
  vx2=mean(vec.x(df1.tmp$radians,df1.tmp$ndvi), na.rm=TRUE)
  vy2=mean(vec.y(df1.tmp$radians,df1.tmp$ndvi), na.rm=TRUE)
  rv2=vec_mag(vx2,vy2)
  rv_ang2=vec_ang(vx2,vy2)
  radial.plot(df1.tmp$ndvi,df1.tmp$radians,clockwise=TRUE,start=s.pos,
    label.pos=c(seq(from=0, to=2*pi-(2*pi/12), by=(2*pi)/12)),
    labels=c(month.abb[seq(from=1, to=12)]),
    rp.type='s',point.symbols=20,radial.lim=c(0,1),
    radial.labels=ndvilabs,
    show.radial.grid=FALSE,
    main=paste(I+1999,' Growing Season NDVI',sep=''),
    grid.col='black',point.col='blue3',
    grid.unit='NDVI',cex=cxs*2,cex.axis=cxs,cex.lab=cxs)
  radial.plot(c(0,rv2),c(rv_ang2,rv_ang2),
clockwise=TRUE,start=s.pos,rp.type='r',
lwd=3,line.col='red',add=TRUE,cex=cxs) # rv, Magnitude of avg vec
  radial.plot(c(0,rv2),c(rv_ang2,rv_ang2),
clockwise=TRUE,start=s.pos,rp.type='s',
point.symbols='*',point.col='red',
add=TRUE,cex=cxs*2)                        # rv endpoints
  text(-0.15, 0.1,
    'SMag', col='red', cex=cxs) # Vector magnitude
  text(sin(df1.tmp$radians[1])*df1.tmp$ndvi[1]+0.1,
    cos(df1.tmp$radians[1])*df1.tmp$ndvi[1]+0.1,
  'ES',col='blue3',cex=cxs) # Early season
  text(sin(df1.tmp$radians[nrow(df1.tmp)])*df1.tmp$ndvi[nrow(df1.tmp)]+0.1,
    cos(df1.tmp$radians[nrow(df1.tmp)])*df1.tmp$ndvi[nrow(df1.tmp)]+0.1,
  'LS',col='blue3',cex=cxs) # Late season
}
```

---

avec_ang *Calculate the opposites of the input radial angle(s)*

Description

Calculate the vertical opposite of the input radian value(s).

Usage

```
avec_ang(r)
```

Arguments

r	A numeric vector of values of radial coordinates.

Details

avec_ang returns the vertical opposite(s) of the input angles.

Value

Returns the vertical opposite of radial angle(s).

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/yr
data(mndvi)               # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t,dpy)          # Transform days of year to radians
v <- as.vector(mndvi$wc)   # MODIS NDVI for Willow Creek tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
vm <- vec_mag(vx,vy)       # Magnitude of average vector
rv_ang <- vec_ang(vx,vy)   # Angle of resultant vec (point of max activity)
av_ang <- avec_ang(rv_ang)  # Angle marking point of least activity
```

---

calc_metrics *Easy function for calculating all polar meatrics.*

Description

A basic, but fast function for calculating all polar metrics from arrays or xts objects.

Usage

```
calc_metrics(input, t = NULL, yr_type, spc, lcut, hcut, return_vecs, sin_cos)
```

Arguments

input	Either a vector of values or a 1-column xts object indexed with time stamps. If input is an xts object then omit argument t. Else if input is a vector of values then t should be set to a corresponding vector of time values.
t	An optional vector of time values (e.g., days) corresponding to the input vector sampling points. Do not use this argument if input is an xts object.
yr_type	Argument specifying either 'cal_yr' for output (of timing variables) given in days starting from Jan. 1, or 'rot_yr' for output in days starting from the average seasonal minimum.
spc	Integer value specifying the number of samples per cycle (measurements per year) in input.
lcut	Numeric value in the range [0,0.5] passed to window_idx function. Indicates the percentile to truncate from the first half of each cycle. For example, 0.15 will result in remove the interval corresponding to [0%,15%] representing a window that begins after the 15th-percentile is crossed.
hcut	Numeric value in the range (0.5,1] passed to window_idx function. Indicates the percentile to truncate from the last half of each cycle. For example, 0.85 will result in remove the interval corresponding to [85%,100%] representing a window that begins after the 85th-percentile is crossed.
return_vecs	logical argument specifying whether or not to include all of the horizontal and vertical component vectors in output.
sin_cos	logical argument. If TRUE then each timing metric (es, ms, etc.) is returned as its sine and cosine components, es is returned as es_sin and es_cos.

Details

calc_metrics runs through the entire polar transformation process and conveniently outputs the final polar metrics for all years included in the input.

Value

Returns a list with all of the derived polar metrics (e.g., early season, mid season, etc.). Timing variables in output can be returned relative to the standard calendar year or rotated to a relative year using yr_type argument. Sine and cosine components can also be returned instead of days since start using sin_cos argument. The optional argument return_vecs can be used to add the actual vecturs used in deriving the polar metrics to the returned list. This will also return the overall average (all years) statistics of the resultant vector (rv) and its opposite the anti-vector (av). Table below indicates the variable components of each object within the returned list. metrics (data frame): year, es (or es_sin, es_cos), # Early ssn DOY ems (or ems_sin, ems_cos), # Early-mid ssn DOY ms (or ms_sin, ms_cos), # Mid ssn DOY lms (or lms_sin, lms_cos), # Late-mid ssn DOY ls (or ls_sin, ls_cos), # Late ssn DOY s_intv, # Season length(days) s_avg, # Avg data val in ssn s_sd, # StDev of data dur ssn s_mag, # Mag of avg vec in ssn ems_mag, # Mag early-mid ssn vec lms_mag, # Mag late-mid ssn vec a_avg # Avg data val of yr component_vectors (data frame): VX, VY # Hrz, vert vec comp. average_vectors (data frame): rv_idx, rv_ang, # Resultant vector rv_doy, rv_mag, # attributes av_idx, av_ang, av_doy # Anti-vector attr.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
library(PolarMetrics)
library(xts)
input <- xts(mndvi$fef, as.Date(mndvi$date))
### Calculate polar measures relative to calendar year
calc_metrics(input, yr_type='cal_yr', spc=46, lcut=0.15, hcut=0.8,
             return_vecs=FALSE, sin_cos=FALSE)
### Calculate as above and return sine, cosine components of timing metrics
calc_metrics(input, yr_type='cal_yr', spc=46, lcut=0.15, hcut=0.8,
             return_vecs=FALSE, sin_cos=TRUE)
### Calculate & return the average vectors for the entire time series
calc_metrics(input, yr_type='cal_yr', spc=46, lcut=0.15, hcut=0.8,
             return_vecs=TRUE, sin_cos=FALSE)$avg_vectors
### Calculate & return the horizontal and vertical vector components
head(calc_metrics(input, yr_type='cal_yr', spc=46, lcut=0.15, hcut=0.8,
             return_vecs=TRUE, sin_cos=FALSE)$vectors)
```

---

mndvi *MODIS NDVI data from select National Forest sites*

Description

A data set of Normalized Difference Vegetation Index values (NDVI) from MODIS for a select number of locations. NDVI was sampled each 8 days (46 per year) for a total of 552 samples spanning years 2000 through 2011. MODIS sensors have a spatial resolution of approximately 250 square meters, thus each column represents a 250 x 250 meter area.

Usage

```
data("mndvi")
```

Format

`mndvi` is a data frame with 552 rows and 5 columns. Each column corresponds to the variables as follows:

`date` Time stamp

`day` a numeric vector of sequential days since January 1, 2000

`chl` a numeric vector of NDVI values for the area surrounding the Coweeta Hydrologic Lab (North Carolina) eddy covariance tower at 83.4252 west longitude and 35.0587 north latitude

`fef` a numeric vector of NDVI values for the area surrounding the Fraser Experimental Forest (Colorado) CO2 tower (run by NCAR) at 105.885417 west longitude, 39.905028 north latitude

`wc` a numeric vector of NDVI values for the area surrounding the Willow Creek (Wisconsin) eddy covariance tower at 90.0798 west longitude, 45.8060 north latitude

Source

http://dx.doi.org/10.3334/ORNLDAAC/1299

Examples

```
data(mndvi)
  ## Not run:
  data(mndvi) # Load data
  head(mndvi) # Inspect first few lines of data

## End(Not run)
```

`rad2d` *Transform from radians to time units*

Description

Transform radians to dates in time units (e.g., day of year).

Usage

```
rad2d(r, dpc)
```

Arguments

`r`	A vector of numeric values of angles in radians (representing times/dates) to be converted back to time units.
`dpc`	A numeric value representing the number of divisions per cycle (e.g., hours per day, days per year, etc.)

Details

`rad2d` transforms input values of `r` from radians to dates in the interval [0,dpc+1).

Value

Returns a vector of cyclic dates in the interval [0,dpc+1) having the same length as input vector r.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/year
data(mndvi)               # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)     # Transform days of year to radians
days_from_rads <- rad2d(r, dpc=dpy) # Transform radians back to days of yr
head(cbind(t, days_from_rads)) # Compare results. Cols should be equiv.
```

---

rad2idx *Transform radians to sequential index values*

Description

Transform radian values to the nearest corresponding indices of the original data.

Usage

```
rad2idx(r, spc)
```

Arguments

r	A vector of numeric values representing dates in radians.
spc	A numeric value indicating the number of samples per cycle (e.g., samples per year).

Details

rad2idx finds the corresponding indices for each element of r.

Value

Returns the index values corresponding to r.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/year
spy <- 46                 # Samples/year (or samples/cycle)
data(mndvi)               # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)     # Transform days of year to radians
r_idx <- rad2idx(r, spc=spy) # Indices corresponding to radial angles in r
head(cbind(t,r,r_idx))     # Compare results
```

`sum_cycle` *Calculate cumulative sums within each cycle in a series*

Description

Calculate the cumulative sums within each cycle in a uniformly sampled series. This function is useful where the phenological (or variable-defined) year is offset from the calendar year. Note that 1) This function assumes every cycle contains the same number of elements, and 2) This function truncates the values before the start of the first cycle and after the end of the last cycle, which for a series of n cycles results in an output series with $n - 1$ cycles.

Usage

```
sum_cycle(v, b, spc)
```

Arguments

`v`	A vector of numeric values representing a uniformly sampled series of data that oscillate over multiple cycles.
`b`	A positive integer indicating the index (i.e., 1 -> *spc*) in the average cycle corresponding to the beginning (or period of least activity) of the average cycle. Within polar space, `b` lies along a ray that intersects the origin and evenly bisects the summed values of average cycle. `b` can be obtained using `avec_ang`.
`spc`	A numeric value indicating the number of samples per cycle (e.g., samples per year).

Details

`sum_cycle` calculates the cumulative sums within cycles beginning from a specified offset (i.e., argument $b > 1$). Let b be the index for the beginning of cycle c and e be the index for the ending index for the cycle $\{v_b, v_{b+1}, \ldots, v_e\}$. Within each cycle the sum at each index j is calculated as the sum of all values leading up to and including j following:

$$c_j = \sum_{i=b}^{j} v_i, v_{i+1}, \ldots v_j$$

This is repeated over $c - 1$ cycles in series v to obtain a new series of cumulative sums.

Value

Returns a vector of values (and their corresponding indices from the input vector) representing the cumulative sums within each cycle based on a specified beginning point. If `b` > 1 then the output vector will have `spc` fewer values, which were truncated due to the offset. Output vector will begin at `b`, the average minimum not at the beginning of the time series, unless the average min is the beginning (`b` = 1).

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/yr
spy <- 46                 # Number of samples in one cycle (yr)
data(mndvi)               # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)     # Transform days of year to radians
v <- as.vector(mndvi$wc)   # MODIS NDVI for Willow Creek tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
rv_ang <- vec_ang(vx,vy)  # Angle of the resultant vector (the direction
  # that the average vector points)
av_ang <- avec_ang(rv_ang)  # Angle marking point of least activity
av_idx <- rad2idx(av_ang, spc=spy) # Index (1-spc) marking avg start of yr
ann_cum <- sum_cycle(v,av_idx,spc=spy)$cumsum # Accum. vals within each yr
```

---

t2rad *Transform dates to radians*

Description

Transform dates to radians.

Usage

```
t2rad(t, dpc)
```

Arguments

t	A vector of numeric dates to be transformed to polar coordinates.
dpc	A numeric value representing the number of divisions per cycle (e.g., hours per day, days per year, etc.)

Details

t2rad transforms the sequence t of dates to radians using:

$$\frac{(t-1) \bmod d}{d} * 2\pi$$

where d indicates the number of days per year in sequence t.

Value

Returns a vector of polar coordinates the same length as input vector t.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/year
data(mndvi)               # Load data
t <- as.vector(mndvi$day) # Days since January 1, 2000
r <- t2rad(t, dpc=dpy)    # Transform days of year to radians
head(cbind(t,r))          # Compare results
```

---

vec.x *Calculate horizontal component vectors*

Description

Calculate the horizontal vector component for each pair of polar coordinates.

Usage

```
vec.x(r, v)
```

Arguments

r	A vector of numeric values representing dates in radians.
v	A vector of numeric values representing a uniformly sampled series of data that vary over multiple cycles.

Details

vec.x calculates the length of the horizontal vector component of each polar coordinate pair. Given an angle r in radians and a corresponding amplitude v the horizontal vector component, VX, is calculated by:

$$VX(r, v) = v * \cos(r)$$

.

Value

Returns the length of the horizontal vector component for each polar coordinate pair.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/yr
data(mndvi)               # Load data
t <- as.vector(mndvi$day) # Days since January 1, 2000
r <- t2rad(t,dpy)         # Transform days of year to radians
v <- as.vector(mndvi$wc)  # MODIS NDVI for Willow Creek tower, WI
vx <- vec.x(r,v)          # The horizontal vector components
```

`vec.y` *Calculate vertical component vectors*

Description

Calculate the vertical vector component for each pair of polar coordinates.

Usage

```
vec.y(r, v)
```

Arguments

`r`	A vector of numeric values representing dates in radians.
`v`	A vector of numeric values representing a uniformly sampled series of data that vary over multiple cycles.

Details

vec.y calculates the length of the vertical vector component of each polar coordinate pair. Given an angle r in radians and a corresponding amplitude v the vertical vector component, Vy, is calculated by:

$$VY(r, v) = v * \sin(r)$$

.

Value

Returns the length of the vertical vector component for each polar coordinate pair.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/yr
data(mndvi)               # Load data
t <- as.vector(mndvi$day) # Days since January 1, 2000
r <- t2rad(t,dpy)         # Transform days of year to radians
v <- as.vector(mndvi$wc)  # MODIS NDVI for Willow Creek tower, WI
vy <- vec.y(r,v)          # The vertical vector components
```

---

vec_ang *Calculate radial angles from horizontal and vertical component vectors*

Description

Calculate the radial angle for each horizontal and vertical component vector pair.

Usage

```
vec_ang(vx, vy)
```

Arguments

vx	A numeric vector of 1 or more values representing horizontal components of Euclidean vectors. vx can be obtained using vec.x.
vy	A numeric vector of 1 or more values representing vertical components of Euclidean vectors. vy can be obtained using vec.y.

Details

vec_ang returns the angle corresponding to the horizontal (vx) and vertical vector components (vy) according to:

$$atan2(vy, vx) + \pi$$

Value

Returns the polar coordinate angle corresponding to each horizontal and vertical component vector pair.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/yr
data(mndvi)               # Load data
t <- as.vector(mndvi$day) # Days since January 1, 2000
r <- t2rad(t,dpy)         # Transform days of year to radians
v <- as.vector(mndvi$wc)  # MODIS NDVI for Willow Creek tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
rv_ang <- vec_ang(vx,vy)    # Angle of resultant vec (point of max activity)
```

`vec_mag` *Calculate Euclidean magnitudes from horizontal and vertical component vectors*

Description

Calculate the magnitude (Euclidean distance) for each pair of horizontal and vertical component vectors.

Usage

```
vec_mag(vx, vy)
```

Arguments

`vx`	A numeric vector of 1 or more values representing horizontal components of Euclidean vectors. `vx` can be obtained using `vec.x`.
`vy`	A numeric vector of 1 or more values representing vertical components of Euclidean vectors. `vy` can be obtained using `vec.y`.

Details

`vec_mag` returns the Euclidean distance of each pair of horizontal and vertical component vectors according to:

$$\sqrt{vx^2 + vy^2}$$

Value

Returns the Euclidean distance corresponding to each horizontal and vertical component vector pair.

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                # Days/yr
data(mndvi)               # Load data
t <- as.vector(mndvi$day) # Days since January 1, 2000
r <- t2rad(t,dpy)         # Transform days of year to radians
v <- as.vector(mndvi$wc)  # MODIS NDVI for Willow Creek tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
vm <- vec_mag(vx,vy)      # Magnitude (length) of average vector
```

`window_idx` *Calculate index values marking truncation endpoints for a window within a series of cumulative sums*

Description

Calculate index values marking the truncation endpoints for a window within a series of cumulative sums. In a phenological context these endpoints define the beginning and end of a quantitatively specified growing season.

Usage

```
window_idx(s, c, cy, lb, ub)
```

Arguments

`s`	A vector of numeric values representing a uniformly sampled series of cumulative sums. `s` can be obtained using `sum_cycle`
`c`	A numeric value indicating the number of cycles in series `s`.
`cy`	A positive integer in the range [0,c] indicating the cycle (year) within which to calculate truncation endpoints.
`lb`	A numeric value in the range [0,0.5] indicating the percent of cumulative total to truncate from the first half of cycle number `cy`. For example, 0.15 will result in remove the interval corresponding to [0%,15%] representing a window that begins after 15% of cumulative annual total is crossed.
`ub`	A numeric value in the range (0.5,1] indicating the percent of cumulative total to truncate from the last half of cycle `cy`. For example, 0.8 will result in removing (80%,100%] of cumulative NDVI representing a window that begins after the 85% of cumulative annual total is crossed.

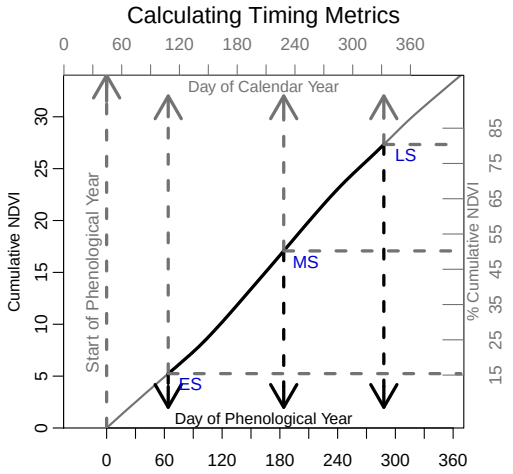
Details

`window_idx` returns a vector of two values indicating endpoints for a window within series S. Cumulative sums within these endpoints are used to place threshold crossings in the data such as growing season start and end points. Since a growing season can be based on crossing points of cumulative sums. Make S a series of cumulative sums and s a subset of growing season values within S. s can be defined on the basis of thresholds crossings of cumulative sums as follows. For a given threshold value, t, let $j(t)$ be the index j for S such that

$$\{i : t < S_i < (t-1)\}$$

Then $\min j(t)$ is the index of S marking the first value larger than t, which is also the beginning of the growing season. $\max j(t)$ marks the ending index.

Figure 2. This visualization shows how the subset of values representing the growing season are selected based on a cumulative NDVI threshold percentage of 15%. The growing season is defined to start after cumulative NDVI reaches 15% of the total for that phenology-centered year and ends at 85%.



Value

Returns a vector of the five indexes (from the input) marking the following timing metrics: early season (lb), early-mid season (midpoint of lb and 50 (50 cumulative total and ub), and late season (ub).

Author(s)

Bjorn J. Brooks, Danny C. Lee, William W. Hargrove, Lars Y. Pomara

References

Brooks, B.J., Lee, D.C., Desai, A.R., Pomara, L.Y., Hargrove, W.W. (2017). Quantifying seasonal patterns in disparate environmental variables using the PolarMetrics R package.

Examples

```
dpy <- 365                 # Days/yr
c <- 12                    # Num. of years/cycles
spc <- 46                  # Number of samples in one cycle (yr)
data(mndvi)                # Load data
t <- as.vector(mndvi$day)  # Days since January 1, 2000
r <- t2rad(t,dpy)          # Transform days of year to radians
v <- as.vector(mndvi$wc)   # MODIS NDVI for Willow Creek tower, WI
vx <- mean(vec.x(r,v), na.rm=TRUE) # Avg horizontal vector
vy <- mean(vec.y(r,v), na.rm=TRUE) # Avg vertical vector
rv_ang <- vec_ang(vx,vy)   # Angle of resultant vec (point of max activity)
av_ang <- avec_ang(rv_ang)  # Angle marking point of least activity
```

```
av_idx <- rad2idx(av_ang, spc=spc) # Index (1-spc) marking avg start of yr
ann_cum <- sum_cycle(v,av_idx,spc=spc)$cumsum # Accum. vals within each yr
# Find seasonal beg, end index for the 2nd yr using 15th pctile of cum NDVI
cy <- 2                    # The second yr of data (which is 2001 here)
es.idx <- window_idx(ann_cum,c-1,cy,0.15,0.8)[1] # Idx of ann_cum marking ES
ms.idx <- window_idx(ann_cum,c-1,cy,0.15,0.8)[2] # Idx of ann_cum marking MS
ls.idx <- window_idx(ann_cum,c-1,cy,0.15,0.8)[3] # Idx of ann_cum marking LS
es <- t[es.idx]          # Early growing season day of pheno yr
ems <- t[es.idx]         # Early-mid growing season day of pheno yr
ms <- t[ms.idx]          # Mid (50th %tile) growing season day
lms <- t[ls.idx]         # Late-mid growing season day of pheno yr
ls <- t[ls.idx]          # Late growing season day of pheno yr
Sintv <- ls-es           # Days in the growing season
ann_cum[es.idx:ls.idx]   # Show cumulative NDVI vals for growing season
```

Index