paired_annualT_signals

Tao Huang

2024-01-23

Annual Signal Functions

```r
library("lubridate")
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library("smwrBase")
```

```
##
## Attaching package: 'smwrBase'
##
## The following objects are masked from 'package:dplyr':
##
##     coalesce, pick, recode
```

```r
## ----------- Heed the Data Gaps -----------------##
## Assess the data inputs for missing data or incorrect data (values >100) ##
## Based on Johson 2021 analysis

data_gap_check <- function(df){
  #df <- Tem_df#bebuggg remove
  #df <- T.y
  df_1 <- df %>%
    dplyr::select(-one_of(c("flow", "bfi_daily")))%>% #one_of allows df without flow or bfi daily
    na.omit(df) %>%
    dplyr::filter(tavg_wat_C < 70) #removing weird values
  df_l <- lapply(unique(df$site_id), function(x){
                #
                df.x <- dplyr::filter(df, site_id == x)
                df.x$date <- as.Date(df.x$date)
                #calculate number of seq missing dates
                df.x$datediff <- difftime(df.x$date,lag(df.x$date),units=c("days"))

                val <- as.numeric(max(df.x$datediff, na.rm = TRUE))

                peryear <- as.numeric(nrow(df.x))

                df.y <- data.frame("site_id" = as.character(x),"max_conseq_missing_days" = val, "count"
              }

        )#end lapply


  df <- do.call(rbind.data.frame, df_l)
}
```

———— Calcuate radian date from date——

```
rad_day <- function(x, yr_type){ #input date vector

  print(head(x))
  if(missing(yr_type)){
    yr_type <- "water" #use water year unless calendar is specified
  }

  # #calendar year
  if(yr_type == "calendar"){
  d <- yday(as.POSIXct(x, format="%Y-%m-%d"))

  } else { #use water year

  wtr_yr <- as.numeric(as.character(waterYear(as.POSIXct(x, format="%Y-%m-%d")))) #to convert factor to
  d <- as.Date(x, format="%Y-%m-%d")
  d_df <- data.frame(wtr_yr, d)
               #https://stackoverflow.com/questions/48123049/create-day-index-based-on-water-year
  wtr_df <- d_df %>%
                 group_by(wtr_yr) %>%
                 mutate(wtr_day = as.numeric(difftime(d, ymd(paste0(wtr_yr - 1 ,'-09-30')), units = "da

  d <- wtr_df$wtr_day
  }

  rad_d <- 2*pi*d/365
  return(rad_d)
}
```

#TAS: Temperature Annual Signal #can be used for air temp and surface water temperature extraction of annual signal

```r
fit_TAS <- function(date, temp, yr_type){

  df <- as.data.frame(unlist(temp)) %>%
    cbind(., date) %>%#has to be done second to keep format (?)
    dplyr::rename("temp" = 1)

  #convert to radian date for sinsoidal extract
  df$rday <- rad_day(df$date, yr_type)

  #to convert back to Phase Days
  units_day <- 365

  #conduct linear fit to a sinsddial function
  Tfit.lm <- lm(temp ~ sin(rday) + cos(rday), data = df)

  #extract equation for the fit
  Tsin.lm <- coef(Tfit.lm)['sin(rday)']*sin(df$rday) +
              coef(Tfit.lm)['cos(rday)']*cos(df$rday) +
              coef(Tfit.lm)['(Intercept)']#TO or mean

  #Calculate Phase of the signal in days
  Phase <- (units_day/(2*pi))*((3*pi/2) -atan(coef(Tfit.lm)['cos(rday)']/
                                                   coef(Tfit.lm)['sin(rday)']))

  #Calculate Amplitude of the signal
  Amp <- sqrt((coef(Tfit.lm)['sin(rday)']^2) + (coef(Tfit.lm)['cos(rday)']^2))

  #remove names to make single values
  names(Phase) <- NULL; names(Amp) <- NULL
  #create dataframe output summary data
  lmStats <- data.frame(amplitude_C = Amp,
                          phase_d = Phase,
                        AdjRsqr=summary(Tfit.lm)$adj.r.squared,
                            RMSE=sqrt(mean(resid(Tfit.lm)^2)),
                            sinSlope=coef(Tfit.lm)['sin(rday)'],
                            cosSlope=coef(Tfit.lm)['cos(rday)'],
                            YInt=coef(Tfit.lm)['(Intercept)'])#; rownames(lmStats) <- "Air" #would like


return(lmStats)


  }
```

```r
#fit_TAS( date, temp, yr_type )
```

```r
#Thermal Metric Yearly Analysis.
TMy_output <- function(df, yr_type){

  if(missing(yr_type)){
    yr_type <- "water" #use water year unless calendar is specified
  }

  # #calendar year
  if(yr_type == "calendar"){
    T.y <- df
    T.y$year <- as.factor(year(T.y$date))
    T.yl <- lapply(levels(T.y$year), function(x){
      df.y <- T.y %>%
        filter(year == x)#%>%

      df.j <- left_join(therm_analysis(df.y), data_gap_check(df.y), by = "site_id")

      df.j$year <- x # add water year as a valuBe in table
      df.j$year_type <- yr_type

      df.j
    }) #end of lapply

  #Water Year
  } else {
      T.y <- add_waterYear(df)
      T.y$year_type <- yr_type
      T.yl <- lapply(levels(T.y$year_water), function(x){
                    df.y <- T.y %>%
                      filter(year_water == x)#%>%

                    df.j <- left_join(therm_analysis(df.y), data_gap_check(df.y), by = "site_id")
                    df.j$year <- x # add water year as a valuBe in table
                    df.j$year_type <- yr_type
                    df.j
  })
  }

  df <- do.call(rbind.data.frame, T.yl)%>% #
    mutate(AmpRatio = ifelse(count <= 100, NA, AmpRatio), #if count less than 100 do not report values
           PhaseLag_d = ifelse(count <= 100, NA, PhaseLag_d),
           Ratio_Mean = ifelse(count <= 100, NA, Ratio_Mean),
           )
}
```