coex_st

```
# coex_st

# definindo a função ####
coex_st <- function(df, time.series = 1, space = "polygon", plot.longs = F)
{
  # Testes de premissas ####
  packages <- c("plyr","tidyverse","rgeos","rgdal") # define os pacotes
necessários
  suppressMessages(lapply(packages, require, character.only = T)) # carrega
os pacotes e retorna erro se eles não estiverem instalados, mas não retorna
as mensagens de carregamento um por um com o suppressMessages
  cols <- c("taxon_name", "collection_no", "lng", "lat", "max_ma", "min_ma")
# cria o vetor esperado de input do dataframe
  if (class(df)!="data.frame"){
    stop("A classe de 'df' deve ser data.frame")
  } # configura a mensagem de erro caso a estrutura do data.frame esteja
errada
  if (sum(colnames(df) %in% cols) < 6){
    stop("Estrutura de colunas incorreta, as colunas devem estar no seguinte
formato: \"taxon_name\", \"collection_no\", \"lng\", \"lat\", \"max_ma\",
\"min_ma\"")
  } # configura a mensagem de erro caso o nome das colunas do data.frame não
esteja correta
  if(space != "site" & space != "coords" & space != "polygon"){
    stop("'space' só pode ser uma das três possibilidades: 'site', 'coords',
'polygon'")
  }
  # Coexistência temporal ####
  df$taxon_name <- as.character(df$taxon_name) # garante que o nome do taxon
é um caractere
  ts <- tapply(df$max_ma, INDEX = df$taxon_name,FUN = max) # extrai a idade
da base do intervalo da ocorrencia mais antiga, momento de surgimento do
taxon
  te <- tapply(df$min_ma, INDEX = df$taxon_name,FUN = min) # extrai a idade
do topo do intervalo da ocorrencia mais nova, momento de extinção do taxon
  n.occs <- data.frame(table(df$taxon_name)) # tabela com o numero de
ocorrencias por taxon
  colnames(n.occs) <- c("taxon", "n") # nomeia as colunas da tabela
  cfi <- (1-0.5)^(-1/(n.occs$n-1)) - 1 # cria um vetor com os intervalos de
confiança de 50% da duração real das linhagens, baseado no número de
ocorrências de cada espécie, de acordo com o método de Marshall 1990.
Utilizei o intervalo de confiança de 50% pois no intervalo de 95% de
confiança para linhagens com apenas uma ocorrência, o intervalo é esticado
em 19 milhões de anos, o que gera estimativas extremamente irreais de
surgimento e extinção
  cfi[which(cfi == Inf)] <- 0 # linhagens com apenas uma ocorrencia retornam
valores infinitos, e, portanto, não permitem uma estimativa de intervalo de
```

```
confianca. uma abordagem é converter esses valores para 0 e manter a idade
de ultima e primeira ocorrência dessas táxons sem alteração
  longs <- data.frame(ts1 = ts, ts = ts + cfi, te1 = te, te = te - cfi) #
cria o dataframe com as longevidades e adiciona a estimativa do intervalo de
confiança para os momentos de surgimento e extinção. como o tempo é
orientado do passado para o presente, à surgimento se soma, e à extinção se
subtrai. também mantém os momentos de surgimento e extinção originais para
servirem de referência no gráfico
  longs[which(longs[,"te"] < 0), "te"] <- 0 # o momento de extinção não pode
ser menor que 0, pois 0 é o presente. para linhagens em que isso ocorre, o
intervalo de confiaça inclui o presente, e, portanto, não se é capaz de
afirmar o momento de extinção dessas linhagens. dessa forma, é necessário
truncar o momento de extinção em 0.
  longs <- longs[order(longs$ts, decreasing = T),] # ordena o dataframe pelo
momento de surgimento estimado
  taxa <- rownames(longs) # cria vetor com nomes das táxons do mais antigo
para o mais recente
  mat <- matrix(0, ncol = length(taxa), nrow = length(taxa)) # cria a
estrutura básica da matriz que será utilizada em passos posteriores,
preenchendo-a com 0 para posteriormente adicionar 1s na coexistência
  rownames(mat) <- taxa # nomeia as linhas com os táxons
  colnames(mat) <- taxa # nomeia as colunas com os táxons
  if(is.numeric(time.series)!=TRUE & is.integer(time.series)!=TRUE) {
    stop("'time.series' deve ser um vetor numérico")
  } # define que o vetor de série temporal deve ser numérico
  if(length(time.series) == 1){
    points <- rev(seq(from = floor(min(longs$te)), to =
ceiling(max(longs$ts)), by = time.series)) # determina os pontos para a
série temporal, do passado ao presente, caso seja fornecido um valor de
subdvisão do tempo
    message(paste("Subdivindo tempo em intervalos de:", time.series,
"milhão(ões) de anos")) # define a mensagem que será exibida no console
durante a execução da função
  }
  if(length(time.series) > 1){
    points <- time.series # se o vetor de série temporal for fornecido
diretamente, apenas o sobreescreve
    message(paste(c("Utilizando o vetor de série temporal fornecido:",
time.series), collapse = " "))
  } # inclui uma mensagem que indica qual a série temporal utilizada
  alive.in.bin <- list(NA) # cria uma lista em que serão guardados os nomes
dos táxons vivos em cada ponto da série temporal
  for (i in 1:length(points)){
    alive.in.bin[[i]] <- rownames(longs[which(longs$ts >= points[i] &
longs$te <= points[i]),])
  } # retorna os nomes dos táxons que estão vivos em cada momento da série
temporal
  names(alive.in.bin) <- points # dá o nome de cada elemento da lista com o
os pontos da série temporal
  # matriz temporal
```

```
  mat.time <- mat # constrói a matriz temporal baseada na estrutura básica
da matriz
  mat.time.all <- rep(list(mat.time), length(points)) # cria uma lista em
que serão guardadas as matrizes da série temporal
  names(mat.time.all) <- points # dá o nome da cada elemento da lista com os
pontos da série temporal
  for(j in 1:length(mat.time.all)){
    mat.time.all[[j]][alive.in.bin[[j]], alive.in.bin[[j]]] <- 1
  } # preenche cada matriz com 1s quando os táxons naquele momento da série
temporal coexistem no tempo
  # Gráfico de longevidades ####
  # Se tratando de uma série temporal, um gráfico que demonstra a
longevidade dos táxons e os cortes no tempo pode ajudar a visualizar o
efeito da coexistência
  if (plot.longs == T){
  suppressWarnings(plot(1~1, ylim = c(0,nrow(longs)), xlim =
c(max(ceiling(longs$ts)),min(floor(longs$te))),col="white", ylab="Taxa",
xlab="Ma")) # cria base do gráfico a ser preenchida com as longevidades
    segments(x0 = longs$ts, x1 = longs$te, y0 = 1:nrow(longs), y1 =
1:nrow(longs), lwd=2, col = "blue") # adiciona as longevidades estimadas com
o método Marshall
    segments(x0 = longs$ts1, x1 = longs$te1, y0 = 1:nrow(longs), y1 =
1:nrow(longs), lwd=2) # adiciona os segmentos das longevidades originais dos
táxons
  abline(v = points, col = "red", lty = 5) # adiciona as linhas de corte no
tempo
  }
  # Coexistencia espacial ####
  # Coleções como sítio de coleta ####
  if(space == "site"){
    df <- transform(df, Site = as.numeric(factor(collection_no))) # renomeia
sítios de coleta para facilitar a identificação
    sites <- unique(df$Site) # cria o vetor de sítios únicos
    alive.in.site <- list(NA) # cria base da lista de táxons vivos em cada
sítio de coleta
    for (k in 1:length(sites)){
      alive.in.site[[k]] <- df[which(df$Site == sites[k]), "taxon_name"]
    } # preenche a lista com os sítios de coleta
    names(alive.in.site) <- sites # nomeia cada elemento da lista com o
número de sítio de coleta
    mat.site <- mat # cria a matriz de sítios a partir da matriz base
    for(l in 1:length(sites)){
      mat.site[alive.in.site[[l]], alive.in.site[[l]]] <- 1
    } # preenche a matriz com 1s quando pelo menos um fóssil de dois táxons
é encontrado junto
    return(lapply(mat.time.all, function(x) x * mat.site)) # configura o
objeto de retorno da função: multiplica vetorialmente cada matriz da série
temporal de coexistência espacial pela matriz de coexistência em sítios de
coleta
    }
  # Utilizando coordenadas absolutas ####
```

```
  if(space == "coords"){
    coords <- unique(df[,c("lng", "lat")]) # cria o vetor de combinações
únicas de coordenadas
    alive.in.coords <- list(NA) # cria a lista a ser preenchida com os
táxons encontradas nas mesmas coordenadas
    for (m in 1:nrow(coords)){
      suppressMessages(alive.in.coords[[m]] <- df[rownames(match_df(x =
df[,c("lng", "lat")], coords[m,])), "taxon_name"])
    } # preenche a lista com os táxons encontrados nos mesmos pontos. o
ouput dessa função retorna no console a mensagem "matching on: lng, lat" a
cada iteração. a função suppressMessages impede que essa mensagem apareça m
vezes no console.
    names(alive.in.coords) <- 1:length(alive.in.coords) # nomeia cada
elemento da lista com o número do identificador de coordenada
    mat.coords <- mat # cria a matriz de coordenadas a partir da matriz base
    for(n in 1:length(alive.in.coords)){
      mat.coords[alive.in.coords[[n]], alive.in.coords[[n]]] <- 1
    } # preenche a matriz com táxons cujas ocorrências são encontrados na
mesma coordenada geográfica
  return(lapply(mat.time.all, function(x) x * mat.coords)) # configura o
objeto de retorno da função: multiplica vetorialmente cada matriz da série
temporal de coexistência espacial pela matriz de coexistência em coordenadas
geográficas absolutas
}
  # Sobreposição de polígonos ####
  if(space == "polygon"){
    spdf <- SpatialPointsDataFrame(coords = cbind(df$lng, df$lat), data =
df) # cria o objeto do tipo "spatial points data frame", necessário para a
criação dos polígonos e análises de sobreposição
    tx_split <- split(spdf,list(spdf$taxon_name), drop = F) # cria uma lista
com os pontos de cada táxon como dataframes separados
    polygons <- lapply(X = tx_split, FUN = gConvexHull) # aplica a função
gConvexHull que calcula os mínimos polígonos convexos para cada táxon em
cada momento da série temporal
    mat.poly <- sapply(polygons, function(x) sapply(polygons, function(y)
gIntersects(x,y))) # aplica a função gIntersects, que calcula se há
sobreposição entre cada par de polígonos e retorna uma matriz com trues para
polígonos que se sobrepõem. etapa mais demorada da função.
    mat.poly[which(mat.poly == TRUE)] <- 1 # converte TRUES em 1s, que é a
base que estamos usando
    mat.poly <- mat.poly[rownames(mat), colnames(mat)] # reordena os
elementos da matriz para a ordem utilizada nas outras matrizes, para
garantir a multiplicação
  return(lapply(mat.time.all, function(x) x * mat.poly)) # configura o
objeto de retorno da função: multiplica vetorialmente cada matriz da série
temporal de coexistência espacial pela matriz de coexistência de polígonos
  }
}
```