#'@name streaminterp

#'@title Interpolation of streaming data

#'@description Interpolation of streaming data

#'@param dt input data frame

#'@param paramlist list of parameters (dt column names) to interpolate

#'@param yearmon a file path used to extract basename

#'@param trim\_rstack logical trim the raster stack by the convex hull of training points?

#'@param trim\_negative logical set negative values to zero?

#'@param tname file.path location to save training dataset

#'@param vname file.path location to save validation dataset

#'@param missprop numeric proportion of missing data allowed. Variables with a greater proportion of missing data will be dropped.

#'@param fdir character file path to local data directory

#'@param costrasname character file.path to cost raster

#'@export

#'@importFrom raster raster writeRaster mask

#'@importFrom gdata resample

#'@importFrom sp SpatialPointsDataFrame coordinates CRS spTransform proj4string

#'@importFrom ipdw ipdwInterp pathdistGen

#'@import rgeos

#'@examples \dontrun{

#'dt <- streamget(yearmon = 201513, qa = TRUE)

#'streaminterp(dt, paramlist = c("sal"), yearmon = 201513)

#'}

streaminterp <- function(dt, paramlist, yearmon, trim\_rstack = TRUE, trim\_negative = TRUE, costrasname = "barrier60large2e2.tif", tname = NA, vname = NA, missprop = 0.16, fdir = getOption("fdir")){

#CHANGE "costrasname="CostRas\_Full\_Clip.tif" OR IF TOO BIG, USE "CostRas\_Optimize\_Clip.tif"

#define projections

projstr <- "+proj=utm +zone=17 +datum=NAD83 +units=m +no\_defs +ellps=GRS80 +towgs84=0,0,0"

latlonproj <- "+proj=longlat +datum=WGS84 +no\_defs +ellps=WGS84 +towgs84=0,0,0"

costras <- raster::raster(file.path(fdir, "DF\_Basefile", costrasname))

#clean paramlist and dt names####

#might need to add translation key for corner case names

paramlist <- tolower(paramlist)

names(dt) <- tolower(names(dt))

if(!all(paramlist %in% names(dt))){

stop("check to make sure paramlist entries matches column names")

}

#remove entries from paramlist that have many NA

naparam <- lapply(paramlist, function(x)length(which(is.na(data.frame(dt[,x])))))

if(any(which(naparam > (nrow(dt) / (1 / missprop))))){

paramlist <- paramlist[-which(naparam > (nrow(dt) \* missprop))]

warning(paste(-which(naparam > (nrow(dt) \* missprop)), "has many missing values."))

}

if(length(paramlist) == 0){

stop("too many missing values")

}

if(is.na(tname) & is.na(vname)){

tname <- file.path(fdir, "/DF\_Subsets/", yearmon, "s.csv", fsep = "")

vname <- file.path(fdir, "/DF\_Validation/", yearmon, "s.csv", fsep = "")

}

if(!file.exists(tname) & !file.exists(vname)){

fulldataset.over <- dt

gridlev <- unique(fulldataset.over$gridcode)

for(i in 1:length(gridlev)){

activesub <- subset(fulldataset.over, fulldataset.over$gridcode == gridlev[i])

selectnum <- gdata::resample(1:nrow(activesub), 1)

if(i == 1){

training <- activesub[selectnum,]

}

else{

training <- rbind(training, activesub[selectnum,])

}

}

validate <- fulldataset.over[!(row.names(fulldataset.over) %in% row.names(training)),]

xy <- cbind(validate$lon\_dd, validate$lat\_dd)

validate <- sp::SpatialPointsDataFrame(xy, validate)

training <- training[!is.na(training$lat\_dd),]

xy <- cbind(training$lon\_dd, training$lat\_dd)

training <- sp::SpatialPointsDataFrame(xy, training)

write.csv(training, tname)

write.csv(validate, vname)

}else{

warning("using previously defined subset")

training <- read.csv(tname, sep = ",")

sp::coordinates(training)<-c("lon\_dd", "lat\_dd")

}

sp::proj4string(training) <- sp::CRS(latlonproj)

training <- sp::spTransform(training, sp::CRS(projstr))

##start interpolation==============================================#

rstack <- ipdw::pathdistGen(training, costras, 3750, yearmon = yearmon)

if(trim\_rstack == TRUE){

rstack <- raster::mask(rstack, rgeos::gConvexHull(coordinatize(streamget(yearmon), latname = "lat\_dd", lonname = "lon\_dd")), inverse = FALSE)

}

dir.create(file.path(fdir, "/DF\_Surfaces/", yearmon))

for(j in 1:length(paramlist)){

finalras <- ipdw::ipdwInterp(training, rstack, paramlist[j], overlapped = TRUE, yearmon = yearmon)

if(trim\_negative){

finalras[finalras < 0] <- 0

}

rf <- raster::writeRaster(finalras, filename = file.path(fdir, "DF\_Surfaces", yearmon, paste(paramlist[j], ".grd", sep = "")), overwrite = T)

rf <- raster::writeRaster(finalras, filename = file.path(fdir, "DF\_Surfaces", yearmon, paste(paramlist[j], ".tif", sep = "")), overwrite = T, format = "GTiff")

}

# for(i in 1:length(paramlist)){

# test <- raster::raster(file.path(fdir, "DF\_Surfaces", yearmon, paste(paramlist[j], ".tif", sep = "")))

# sp::plot(test)

# }

}

#subset a particular basin

#bayproper<-subset(fulldataset,FBFS\_Zones==2)

#longsound<-subset(fulldataset,Fathom\_ID==7)

#longsound<-subset(fulldataset,NAME=="Long Sound")

##subset dt#######

#dt<-read.csv(dtname.new,header=T,na.strings="NA")

#fulldataset.over<-dt

#grid selection#

#set.seed(2)

#test parallel pathdistGen

# a<-Sys.time()

# rstack2<-pathdistGen2(spdf=training,costras=costras,range=3750,yearmon=yearmon,paralleltf=TRUE)

# #rstack<-rstack[[1]]

# a-Sys.time()