```
#####################################
## TO FILTER SOME RUNS OVER OTHERS ##
#####################################

# This script uses the jackknife approach, and selects routes starting or ending at specific sites,
# and calculates the popularity around all the sites. It repeats this for all individual sites.
# It also focuses on the routes that do NOT go to a specific site and calculates the popularity of all sites,
# on those reduced datasets.

# It uses the csv of individual site-pair routes created in the Parse_LCP_outputs R script. It does NOT go through
the raw data.

library(raster)
library(ggplot2)
library(reshape)
library(dplyr)
library(rgeos)

# Clears the environment to avoid errors from old files
rm(list=ls())

# Choose if you want the mean or the p.value
val <- "p.value"
# val <- "mean"

# Browse to the DEM
DEM <- raster(file.choose(), sep=",")

# Get the dimensions of the raster to collate the route data
dem.x <- dim(DEM)[2]
dem.y <- dim(DEM)[1]

# Browse to the master dataset created by BehaviorSpace
master <- read.csv(file.choose(), sep=",", skip = 6, stringsAsFactors = F)

# Remove weird characters imported from BehaviorSpace
master$optimization <- gsub("^\"|\"$", "", master$optimization)
master$hiker.status <- gsub("^\"|\"$", "", master$hiker.status)

# Filter over the same runs as in the Parse_LCP_outputs script
master <- subset(master, master$hiker.status=="alive") # Removing the runs where the agent died

# Identify the different sites to parse through
sites <- unique(gsub("\\}|\\{", "", master$origin))

# To iterate over different datasets and create raster for each optimization
# Start with Exploration as its map will be used to create a buffer for the other two
optimization <- c("Exploration","Distance","Speed")

# Create an empty dataframe that will record the popularity each site based on the different jackknife choice
results <- data.frame(matrix(vector(), 0, length(sites), dimnames=list(c(), sites)), stringsAsFactors=F)
results$mean <- numeric(nrow(results))
results$legend.1 <- numeric(nrow(results))
```

```
results$legend.2 <- numeric(nrow(results))
results$legend.3 <- numeric(nrow(results))

# Identify the folder in which all the output files are located
my_dir = "[write path to folder]"

# Create a list of all the file names in the identified folder
all_files = list.files(path = my_dir, all.files = TRUE, full.names = TRUE, pattern = "\\.csv$")

# Give the number of files to analyze
list_size = length(all_files)

# Iterate over the different optimization
for (op in optimization){

  for (i in sites){

    # Create a temporary dataframe that will take on coordinates and values
    dat <- data.frame(x=double(),y=double(),value=double())

    # To keep up with progress
    print(i)

    ##############################################
    ## ALL ROUTES THAT START AND END AT SITE i ##
    ##############################################

    # Create a new row for each iteration
    nrow_results = nrow(results)
    new_row = nrow_results + 1
    results[new_row,] <- as.numeric(0)
    results$legend.1[new_row] <- "Only"
    results$legend.2[new_row] <- i
    results$legend.3[new_row] <- op

    # Create the progress bar
    pb <- txtProgressBar(min = 0, max = list_size, style = 3)

    # Iterate over the files in the folder
    for(l in 1:list_size[1]){

      # Reformat the name of the files so that it can be used later
      file.name = strsplit(all_files[l],"/")
      file.name = unlist(file.name)
      name.size = length(file.name)
      new.file = file.name[name.size]

      # Separate out the components of the file name
      filename.split = strsplit(new.file,"_")
      filename.split = unlist(filename.split)
      origin2 = filename.split [4]
      goal2 = filename.split [6]
      opt2 = gsub(".csv", "", filename.split[7])
```

```
    # If that specific run has the correct optimization (op), start or end points, read it and collate its data. If it is not, skip it.
    if ((opt2==op && goal2==i) || (opt2==op &&origin2==i)){ # If the site is either a start or a beginning

      # Read the csv outputs and keep only the important columns (2 to 4)
      dat.add <- read.csv(paste(my_dir,new.file,sep=""), stringsAsFactors = F)
      dat.add <- dat.add[,2:4]

      # Transform the times walked on into a 0-1 value (divide by the max times walked)
      dat.add$value <- dat.add$value / max(dat.add$value)

      # Attach the new dataset to the other ones for this specific optimization setting (if applicable)
      dat <- rbind(dat.add,dat)

      # Assign the dataset to the global environment so it can be used outside the loop.
      assign('dat',dat, envir = .GlobalEnv)
     }

     # Update progress bar
     setTxtProgressBar(pb, l)
   }

   # Group the cells by coordinates (here, we may have cells walked on multiple times in different site-pair combinations)
   dat.group <- group_by(dat, x, y)

   # Calculate the mean popularity value of those cells that are walked more then once
   a <- dplyr::summarize(dat.group, value = mean(value))
   dat.final <- as.data.frame(a)

   # To transform into a usable raster by GRASS (get those values from DEM raster)
   dat.final$x <- (dat.final$x * xres(DEM) ) + xmin(DEM) + (xres(DEM) / 2) # xmin extent of the original map (+MID WORKS!!)
   dat.final$y <- (dat.final$y * yres(DEM) ) + ymin(DEM) + (yres(DEM) / 2) # ymin extent of the original map (+MID WORKS!!)

   ## Convert to raster to use the focal function
   ras.dat <- rasterFromXYZ(dat.final)

   if (op == "Exploration"){

     # Create a buffer from the exploration map to show the possible paths between points
     buffer <- ras.dat

     # The buffer has value 0
     buffer[buffer > 0] <- 0
     assign(paste("buffer_",i,"_only",sep=""), buffer, envir = .GlobalEnv)

   }else{

     # Import the appropriate buffer
     buffer <- get(paste("buffer_",i,"_without",sep=""))
```

```
    # Merge map and buffer so that everything in map that is NA and has a value in buffer is set to 0
    map2 <- resample(ras.dat, buffer, "bilinear")
    ras.dat <- merge(map2,buffer) # For all NA in "map2," it takes the values of "buffer"
   }

   # Transform the raster into a matrix of coordinates and values
   whole <- rasterToPoints(ras.dat)

   # Calculate the mean popularity of all path cells and record in the results dataset
   results$mean[new_row] <- format(round(cellStats(ras.dat, stat='mean', na.rm=TRUE), 3), nsmall = 3, na.omit=T)

   # Iterate over each site to calculate their popularity
   for (s in sites){

    # Parse the sites' coordinates and name
    site.x <- as.numeric(unlist(strsplit(s," "))[2])
    site.y <- as.numeric(unlist(strsplit(s," "))[3])
    site.name <- gsub(" ",".", s)
    pos <- which(colnames(results)==site.name)

    # Ensure that the sites are one thr correct coordinate system
    site.coord <- matrix(c(site.x,site.y), nrow=1,ncol=2)
    site.coord[,1] <- (site.coord[,1] * xres(DEM) ) + xmin(DEM) + (xres(DEM) / 2) # xmin extent of the original map
(+MID WORKS!!)
    site.coord[,2] <- (site.coord[,2] * yres(DEM) ) + ymin(DEM) + (yres(DEM) / 2) # ymin extent of the original map
(+MID WORKS!!)

    # Transform the site point into a raster
    site.ras <- rasterize(site.coord, DEM, field = 1)

    # Create a polygon of 1km radius around the site raster
    pol <- rasterToPolygons(site.ras)
    pol.buf <- buffer(pol, width=1000)

    # Use the buffer polygon to clip the Popularity map created
    mask.ras <- mask(ras.dat,pol.buf)
    site.list <- rasterToPoints(mask.ras)

    # If the site's buffer falls onto at least 2 cells walked on by an agent,
    # this calculates the mean popularity of the cells within the buffer.
    # One can uncomment the line to instead get the p-value of a two-sided t-test to see if the mean is
    # significantly higher than the average of all cells.

    if (nrow(site.list) > 1){

     # Make sure you defined p.value or mean above.
     if (val == "p.value"){
      temp <- t.test(site.list[,3],whole[,3], alternative = "greater")$p.value
     }else{
      temp <- mean(site.list[,3])
      }
```

```
    # Record the value into the results dataset
    results[new_row,pos] <- format(round(temp, 3), nsmall = 3, na.omit=T)

    # Assign to the Global Environment to be used outside the loop
    assign('results',results, envir = .GlobalEnv)
   }else{
    # If the site buffer is outside of the paths, nothing happens.
    results[new_row,pos] <- "NA"

    # Assign to the Global Environment to be used outside the loop
    assign('results',results, envir = .GlobalEnv)
   }
  }

  ######################################################
  ## ALL ROUTES THAT DO NOT START AND END AT SITE i ##
  ######################################################

  # Create a new temporary dataframe that will take on coordinates and values
  dat <- data.frame(x=double(),y=double(),value=double())

  # To keep up with progress
  print(i)

  # Create a new row for each iteration
  nrow_results = nrow(results)
  new_row = nrow_results + 1
  results[new_row,] <- as.numeric(0)
  results$legend.1[new_row] <- "Without"
  results$legend.2[new_row] <- i
  results$legend.3[new_row] <- op

  # Create a new progress bar
  pb <- txtProgressBar(min = 0, max = list_size, style = 3)

  # Iterate over the files in the folder
  for(l in 1:list_size[1]){

   # Reformat the name of the files so that it can be used later
   file.name = strsplit(all_files[l],"/")
   file.name = unlist(file.name)
   name.size = length(file.name)
   new.file = file.name[name.size]

   # Separate out the components of the file name
   filename.split = strsplit(new.file,"_")
   filename.split = unlist(filename.split)
   origin2 = filename.split [4]
   goal2 = filename.split [6]
   opt2 = gsub(".csv", "", filename.split[7])

   # If that specific run has the correct optimization (op), start or end points, read it and collate its data. If it is not,
skip it.
```

```
    if (opt2==op && goal2!=i && origin2!=i){    # If site i is NOT a start nor a beginning

      # Read the csv outputs and keep only the important columns (2 to 4)
      dat.add <- read.csv(paste(my_dir,new.file,sep=""), stringsAsFactors = F)
      dat.add <- dat.add[,2:4]

      # Transform the times walked on into a 0-1 value (divide by the max times walked)
      dat.add$value <- dat.add$value / max(dat.add$value)

      # Attach the new dataset to the other ones for this specific optimization setting (if applicable)
      dat <- rbind(dat.add,dat)

      # Assign the dataset to the global environment so it can be used outside the loop.
      assign('dat',dat, envir = .GlobalEnv)
    }

    # Update progress bar
    setTxtProgressBar(pb, l)
  }

  # Group the cells by coordinates (here, we may have cells walked on multiple times in different site-pair
combinations)
  dat.group <- group_by(dat, x, y)

  # Calculate the mean popularity value of those cells that are walked more then once
  a <- dplyr::summarize(dat.group, value = mean(value))
  dat.final <- as.data.frame(a)

  # To transform into a usable raster by GRASS (get those values from DEM raster)
  dat.final$x <- (dat.final$x * xres(DEM) ) + xmin(DEM) + (xres(DEM) / 2) # xmin extent of the original map (+MID
WORKS!!)
  dat.final$y <- (dat.final$y * yres(DEM) ) + ymin(DEM) + (yres(DEM) / 2) # ymin extent of the original map (+MID
WORKS!!)

  ## Convert to raster to use the focal function
  ras.dat <- rasterFromXYZ(dat.final)

  if (op == "Exploration"){

    # Create a buffer from the exploration map to show the possible paths between points
    buffer <- ras.dat

    # The buffer has value 0
    buffer[buffer > 0] <- 0
    assign(paste("buffer_",i,"_without",sep=""), buffer, envir = .GlobalEnv)

  }else{

    # Import the appropriate buffer
    buffer <- get(paste("buffer_",i,"_without",sep=""))

    # Merge map and buffer so that everything in map that is NA and has a value in buffer is set to 0
    map2 <- resample(ras.dat, buffer, "bilinear")
```

```
  ras.dat <- merge(map2,buffer) # For all NA in "map2," it takes the values of "buffer"
 }

 # Transform the raster into a matrix of coordinates and values
 whole <- rasterToPoints(ras.dat)

 # Calculate the mean popularity of all path cells and record in the results dataset
 results$mean[new_row] <- format(round(cellStats(ras.dat, stat='mean', na.rm=TRUE), 3), nsmall = 3, na.omit=T)

 # Iterate over each site to calculate their popularity
 for (s in sites){

  # Parse the sites' coordinates and name
  site.x <- as.numeric(unlist(strsplit(s," "))[2])
  site.y <- as.numeric(unlist(strsplit(s," "))[3])
  site.name <- gsub(" ",".", s)
  pos <- which(colnames(results)==site.name)

  # Ensure that the sites are one thr correct coordinate system
  site.coord <- matrix(c(site.x,site.y), nrow=1,ncol=2)
  site.coord[,1] <- (site.coord[,1] * xres(DEM) ) + xmin(DEM) + (xres(DEM) / 2) # xmin extent of the original map
(+MID WORKS!!)
  site.coord[,2] <- (site.coord[,2] * yres(DEM) ) + ymin(DEM) + (yres(DEM) / 2) # ymin extent of the original map
(+MID WORKS!!)

  # Transform the site point into a raster
  site.ras <- rasterize(site.coord, DEM, field = 1)

  # Create a polygon of 1km radius around the site raster
  pol <- rasterToPolygons(site.ras)
  pol.buf <- buffer(pol, width=1000)

  # Use the buffer polygon to clip the Popularity map created
  mask.ras <- mask(ras.dat,pol.buf)
  site.list <- rasterToPoints(mask.ras)

  # If the site's buffer falls onto at least 2 cells walked on by an agent,
  # this calculates the mean popularity of the cells within the buffer.
  # One can uncomment the line to instead get the p-value of a two-sided t-test to see if the mean is
  # significantly higher than the average of all cells.

  if (nrow(site.list) > 1){

   # Make sure you defined p.value or mean above.
   if (val == "p.value"){
    temp <- t.test(site.list[,3],whole[,3], alternative = "greater")$p.value
   }else{
    temp <- mean(site.list[,3])
   }

   # Record the value into the results dataset
   results[new_row,pos] <- format(round(temp, 3), nsmall = 3, na.omit=T)
```

```
      # Assign to the Global Environment to be used outside the loop
      assign('results',results, envir = .GlobalEnv)
     }else{
      # If the site buffer is outside of the paths, nothing happens.
      results[new_row,pos] <- "NA"

      # Assign to the Global Environment to be used outside the loop
      assign('results',results, envir = .GlobalEnv)
     }
    }
  }
}

# Clean up the row and column names for export
colnames(results) <-
c("Rascano","Castillo","Juyo","Miron","Pendo","Cierro","Altamira","mean","type","site","optimization")
results$site[results$site == "patch 241 78"] <- "Rascano"
results$site[results$site == "patch 221 85"] <- "Castillo"
results$site[results$site == "patch 232 97"] <- "Juyo"
results$site[results$site == "patch 259 66"] <- "Miron"
results$site[results$site == "patch 231 93"] <- "Pendo"
results$site[results$site == "patch 140 133"] <- "Cierro"
results$site[results$site == "patch 212 98"] <- "Altamira"

# Export as a csv
if (val == "p.value"){
  write.csv(results, "[write path to folder]/Pop_jk_pvalue_test.csv")
}else{
  write.csv(results, "[write path to folder]/Pop_jk_mean_test.csv")
}
```