```
library(raster)
library(ggplot2)
library(reshape)
library(dplyr)

# This parses the model outputs and creates collated raster(s).

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

# 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 and read the main output file created by BehaviorSpace
master <- read.csv(file.choose(), sep=",", skip = 6, stringsAsFactors = F)

# Remove the weird characters added to strings
master$optimization <- gsub("^\"|\"$", "", master$optimization)
master$hiker.status <- gsub("^\"|\"$", "", master$hiker.status)

# Filter some runs over others (dead agent or too steep slopes)
master <- subset(master, master$hiker.status=="alive") # Removing the runs where the agent died
master <- subset(master, master$min.list.slope > -12) # Removing the runs where too steep slopes were used
master <- subset(master, master$max.list.slope < 12) # Removing the runs where too steep slopes were used
master <- subset(master, master$median.filter....i.....i....0...list.slope < 5) # Removing the runs where too steep
slopes were used
master <- subset(master, master$median.filter....i.....i....0...list.slope.1 > -5) # Removing the runs where too steep
slopes were used

# Reduce the stamp decimals to 8 so that they can be compared to the file name
master$stamp1 <- formatC(master$stamp1, digits = 8, format = "f")

# Create a dataset recording the unique site-pairs combinations
master.comb <- unique(master[,c('origin','goal')])

# To iterate over different datasets and create raster for each optimization
optimization <- c("Distance","Exploration","Speed")

counter <- 0  # Help keep track of progress

# Assign the folder in which all the individual simulation output files are located
my_dir = "[write path to folder here]"

# 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 files to collate the routes for each optimization
for (op in optimization){

  # Create a temporary dataframe that will take on the collated coordinate and popularity values
  dat.final <- data.frame(x=double(0),
                          y=double(0),
                          value=double(0))

  # Create a temporary dataframe that will take on the collated coordinate and popularity values (different from
dat.final to avoid overwiting data)
  dat.prev <- data.frame(x=double(0),
                         y=double(0),
                         value=double(0))

  print(op)

  # Iterate over the different site-pair combinations to identify the popular routes between each
  for (m in 1:nrow(master.comb)){

    ori <- gsub("\\}|\\{", "", master.comb$origin[m])  # goal (start point) of row m with the {} removed
    print(ori) # To help visualize the run progress
    g <- gsub("\\}|\\{", "", master.comb$goal[m])  # goal (end point) of row m with the {} removed
    print(g) # To help visualize the run progress
    counter <- counter + 1
    print(counter) # To help visualize the run progress

    # Create a temporary dataframe that will help create raster
    routes <- data.frame(x=double(0),
                         y=double(0),
                         value=double(0))

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

    # Iterate over all the files located in the given 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's name
      filename.split = strsplit(new.file,"_")
      filename.split = unlist(filename.split)
      opt = filename.split [3] # Optimization of the run
      origin = gsub("\\)|\\(", "", filename.split[6]) # Origin of the run
      goal = gsub("\\)|\\(", "", filename.split[7]) # Goal of the run
      stamp = format(round(as.numeric(gsub(".csv", "", filename.split[8])), 8), nsmall = 8) # Numerical stamp of the
run (reduced decimals to 8 to be comparable with file data)
```

```
    # If that specific run has the correct optimization (op), start and end points, read it and collate its data. If it is not, skip it.
    if (stamp %in% master$stamp1 && opt==op && goal==g && origin==ori){

      # Import the data without headers
      ds <- read.table(paste(my_dir,new.file,sep=""), fill = TRUE, skip = 19, stringsAsFactors = FALSE, sep = ",")

      # Keep only the coordinates of the paths
      ds <- ds[,c(2,6)]

      # Change the names and reduce the floats coordinates to integers
      colnames(ds) <- c("x","y")
      ds$x <- as.integer(ds$x)
      ds$y <- as.integer(ds$y)

      # Remove duplicates cells created because of knight movements that often stops on cell edges before reaching its destination
      ds <- ds [!duplicated(ds[c(1,2)]),]

      # Extract the coordinates of the path's start and end points
      start.x.raw <- unlist(strsplit(gsub("[[:punct:]]", "", ori), " "))
      start.x <- start.x.raw[2]
      start.y <- start.x.raw[3]
      end.x.raw <- unlist(strsplit(gsub("[[:punct:]]", "", g), " "))
      end.x <- end.x.raw[2]
      end.y <- end.x.raw[3]

      # Remove the origin and goal's patches (if present) to avoid skewing the data
      ds<-ds[!(ds$x==start.x & ds$y==start.y),]
      ds<-ds[!(ds$x==end.x & ds$y==end.y),]

      # For each path, each cell is walked on only once
      ds$value <- 1

      # If this is not an empty dataset
      if(nrow(ds) > 1){

        # Add this new path to the big dat dataset
        routes <- rbind(routes,ds)

        # Then group by coordinates and sum up the number of times each cell is walked on
        route.group <- group_by(routes, x, y)
        b <- dplyr::summarize(route.group, value = sum(value))
        routes <- as.data.frame(b)

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

  # Update progress bar
  setTxtProgressBar(pb, l)
  }
```

```
  ##########################
  ## CREATING THE RASTER ##
  ##########################

  print("creating route") # Show progress
  dat <- routes

  # Change the name of the dat file because we will add new columns
  colnames(dat) <- c("long","lat","value")

  # Ensure that the x and y columns are numeric
  dat$x <- as.numeric(as.character(dat[,1]))
  dat$y <- as.numeric(as.character(dat[,2]))

  # Change the order of the dat file to have x,y,value.
  dat <- dat[,c(4,5,3)]

  # Output the compiled routes for each combination origin-goal-optimization
  write.csv(dat, paste("[write path to folder here]/Iterated_route_from_", ori, "_to_", g, "_", op, ".csv", sep = ""))

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

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

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

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

 # Transform into a raster with the same coordinates as the imported DEM
 dat.final$x <- (dat.final$x * xres(DEM) ) + xmin(DEM) + (xres(DEM) / 2) # xmin extent of the original map
 dat.final$y <- (dat.final$y * yres(DEM) ) + ymin(DEM) + (yres(DEM) / 2) # ymin extent of the original map
 dat.final.cut <- dat.final

 # Create the raster
 r.sub <- rasterFromXYZ(dat.final.cut)

 # Output it
 writeRaster(r.sub, paste("[write path to folder here]/Iterated_sites_",op,".asc", sep = ""), overwrite = T)
}

#################################################################
```

```
## TO READ THE CSV OUTPUTS SEPARATELY, AND CONVERT TO ASCII ##
#####################################################################

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

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

# 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]

# To iterate over different datasets and create raster for each optimization
optimization <- c("Distance","Exploration","Speed")

counter <- 0  # Help keep track of progress

# Assign the folder in which csv path outputs (created by the code above) are located
my_dir = "[write path to folder here]"

# 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)

for (op in optimization){

  # Create a temporary dataframe that will take on the collated coordinate and popularity values
  dat.final <- data.frame(x=double(0),
                          y=double(0),
                          value=double(0))

  # Create a temporary dataframe that will help create raster
  dat <- data.frame(x=double(0),
                    y=double(0),
                    value=double(0))

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

  # Keep count of progress
  counter = counter + 1
  print(counter)

  # Iterate over the list of files and collage their data
```

```
  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)
    opt = gsub(".csv", "", filename.split [7])

    # If that specific run is of the correct optimization, read it and collate its data. If it is not, skip it.
    if (opt==op){

      # Take on the important file name components
      origin = gsub("\\)|\\(", "", filename.split[4])
      goal = gsub("\\)|\\(", "", filename.split[6])

      # 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
  dat.final$y <- (dat.final$y * yres(DEM) ) + ymin(DEM) + (yres(DEM) / 2) # ymin extent of the original map
dat.final.cut <- dat.final

  r.sub <- rasterFromXYZ(dat.final.cut)
```

```
  writeRaster(r.sub, paste("[write path to folder here]/Iterated_sites_",op,".asc", sep = ""), overwrite = T)
}
```