```
########################################
## BEHAVIORSPACE SENSITIVITY ANALYSIS ##
########################################

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

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

# Browse to the output file created by the mountain and plain simulations separately
db.mountain <- read.csv(file.choose(), sep=",", skip = 6, stringsAsFactors=FALSE)
db.plain <- read.csv(file.choose(), sep=",", skip = 6, stringsAsFactors=FALSE)

# Reduce the dataset to the values important for the sensitivity analyses, rename the columns for easier access,
and calculate speed
db.mountain <- db.mountain[,c(10,12,17,19,20,26:29)]
colnames(db.mountain) <-
c("opt","view","switch","dist","hours","med_pos_slope","med_neg_slope","max_slope","min_slope")
db.mountain$speed <- db.mountain$dist / (db.mountain$hours/3600)

db.plain <- db.plain[,c(10,12,17,19,20,26:29)]
colnames(db.plain) <-
c("opt","view","switch","dist","hours","med_pos_slope","med_neg_slope","max_slope","min_slope")
db.plain$speed <- db.plain$dist / (db.plain$hours/3600)

# Remove the weird characters added to strings by BehaviorSpace
db.mountain$opt <- gsub("^\"|\"$", "", db.mountain$opt)
db.plain$opt <- gsub("^\"|\"$", "", db.plain$opt)

# Create a dataset that will record the mean distance, speed, and slope of the different optimization-environment
combinations
sd_results <- data.frame(Opt=double(0),
                         Envir=double(0),
                         Distance=double(0),
                         Speed=double(0),
                         Mean.slope=double(0),
                         max.slope=double(0),
                         min.slope=double(0))

optimization <- c("Distance","Exploration","Speed")

for (i in optimization)
{
  nrow_results = nrow(sd_results)
  new_row = nrow_results + 1
  sd_results[new_row,] <- as.numeric(0)
  sd_results$Opt[new_row] <- i
  sd_results$Envir[new_row] <- "Mountain"

  sd_results$Distance[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i)) /
mean(subset(db.mountain$dist, db.mountain$opt == i)))
```

```
  sd_results$Speed[new_row] <- (sd(subset(db.mountain$speed, db.mountain$opt == i)) /
mean(subset(db.mountain$speed, db.mountain$opt == i)))
  sd_results$Mean.slope[new_row] <- (sd(subset(db.mountain$med_pos_slope, db.mountain$opt == i)) /
mean(subset(db.mountain$med_pos_slope, db.mountain$opt == i)))
  sd_results$max.slope[new_row] <- (sd(subset(db.mountain$max_slope, db.mountain$opt == i)) /
mean(subset(db.mountain$max_slope, db.mountain$opt == i)))
  sd_results$min.slope[new_row] <- (sd(subset(db.mountain$min_slope, db.mountain$opt == i)) /
mean(subset(db.mountain$min_slope, db.mountain$opt == i)))

  nrow_results = nrow(sd_results)
  new_row = nrow_results + 1
  sd_results[new_row,] <- as.numeric(0)
  sd_results$Opt[new_row] <- i
  sd_results$Envir[new_row] <- "Plain"

  sd_results$Distance[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i)) / mean(subset(db.plain$dist,
db.plain$opt == i)))
  sd_results$Speed[new_row] <- (sd(subset(db.plain$speed, db.plain$opt == i)) / mean(subset(db.plain$speed,
db.plain$opt == i)))
  sd_results$Mean.slope[new_row] <- (sd(subset(db.plain$med_pos_slope, db.plain$opt == i)) /
mean(subset(db.plain$med_pos_slope, db.plain$opt == i)))
  sd_results$max.slope[new_row] <- (sd(subset(db.plain$max_slope, db.plain$opt == i)) /
mean(subset(db.plain$max_slope, db.plain$opt == i)))
  sd_results$min.slope[new_row] <- (sd(subset(db.plain$min_slope, db.plain$opt == i)) /
mean(subset(db.plain$min_slope, db.plain$opt == i)))
}

# Reshape the data so it can be used by ggplot
sd_melt <- melt(sd_results)

# Plot the different values to compare how the environment and optimization impact them
ggplot(sd_melt, aes(x = Opt, y = as.numeric(value), group = Envir)) +
  geom_point() +
  facet_grid(variable~Envir, scales = "free") +
  ggtitle("Standard deviation - Optimization\n") +
  xlab("") + ylab("") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_text(size=10), plot.title = element_text(hjust =
0.5))

############################
## MORE DETAILED GRAPHS ##
############################

# Label the different runs and attach them into a big dataset (db) to compare their results
db.mountain$label <- "Mountain"
db.plain$label <- "Plain"
db <- rbind(db.mountain, db.plain)

# The following graphs compare multiple values between the two settings and the three optimizations, using
boxplots.
```

```
ggplot(db, aes(x = 1, y = as.numeric(dist), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(label~opt) +
  ggtitle("Traveled km - Optimization\n") +
  xlab("") + ylab("Distance traveled (km)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db, aes(x = 1, y = as.numeric(speed), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(label~opt) +
  ggtitle("Speed - Optimization\n") +
  xlab("") + ylab("Speed traveled (km/h)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db, aes(x = 1, y = as.numeric(mean_slope), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(label~opt) +
  ggtitle("Mean slope - Optimization\n") +
  xlab("") + ylab("Mean slope (degree)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db, aes(x = 1, y = as.numeric(max_slope), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(label~opt) +
  ggtitle("Max slope - Optimization\n") +
  xlab("") + ylab("Max slope (degree)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db, aes(x = 1, y = as.numeric(min_slope), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(label~opt) +
  ggtitle("Min slope - Optimization\n") +
  xlab("") + ylab("Min slope (degree)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

########################################
## IMPACT OF VIEWSHED AND SWITCHBACK ##
########################################

# Calculate the standard deviation (variability) created by each setting
sd_results <- data.frame(Opt=double(0),
                         Envir=double(0),
                         v0=double(0),
```

```
                    v001=double(0),
                    v01=double(0),
                    v02=double(0),
                    v05=double(0),
                    v1=double(0),
                    switch.t=double(0),
                    switch.f=double(0))

items <- c("Distance","Exploration","Speed")
for (i in items)
{
  nrow_results = nrow(sd_results)
  new_row = nrow_results + 1
  sd_results[new_row,] <- as.numeric(0)
  sd_results$Opt[new_row] <- i
  sd_results$Envir[new_row] <- "Mountain"

  sd_results$v0[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0)) /
mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0)))
  sd_results$v001[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view ==
0.001)) / mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.001)))
  sd_results$v01[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.01)) /
mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.01)))
  sd_results$v02[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.02)) /
mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.02)))
  sd_results$v05[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.05)) /
mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.05)))
  sd_results$v1[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.1)) /
mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == 0.1)))
  sd_results$switch.t[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$switch ==
"true")) / mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$switch == "true")))
  sd_results$switch.f[new_row] <- (sd(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$switch ==
"false")) / mean(subset(db.mountain$dist, db.mountain$opt == i & db.mountain$switch == "false")))

  nrow_results = nrow(sd_results)
  new_row = nrow_results + 1
  sd_results[new_row,] <- as.numeric(0)
  sd_results$Opt[new_row] <- i
  sd_results$Envir[new_row] <- "Plain"

  sd_results$v0[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0)) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0)))
  sd_results$v001[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.001)) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.001)))
  sd_results$v01[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.01)) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.01)))
  sd_results$v02[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.02)) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.02)))
  sd_results$v05[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.05)) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.05)))
  sd_results$v1[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.1)) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$view == 0.1)))
```

```
  sd_results$switch.t[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$switch == "true")) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$switch == "true")))
  sd_results$switch.f[new_row] <- (sd(subset(db.plain$dist, db.plain$opt == i & db.plain$switch == "false")) /
mean(subset(db.plain$dist, db.plain$opt == i & db.plain$switch == "false")))
}

# These graphs show the impact of changing the different viewshed values and switchback (ON/OFF) on important
metrics.
# They are separated by environments (Plain and Mountain) to keep things readable.

## PLAIN ##

ggplot(db.plain, aes(x = 1, y = as.numeric(dist), group = opt)) +
  geom_boxplot(notch = T, fill="grey") +
  facet_grid(opt~view, scales="free_y") +
  ggtitle("Walked distance on plain\n") +
  xlab("\nViewshed threshold") + ylab("Walked distance (km)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5),
axis.ticks = element_blank())

ggplot(db.plain, aes(x = 1, y = as.numeric(max_slope), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(opt~view, scales="free_y") +
  ggtitle("Plain - Max slope - Viewshed\n") +
  xlab("") + ylab("Max slope (degrees)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5),
axis.ticks = element_blank())

ggplot(db.plain, aes(x = 1, y = as.numeric(dist), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(opt~switch, scales="free_y") +
  ggtitle("Plain - Distance - Switchbacks\n") +
  xlab("") + ylab("Distance traveled (km)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5),
axis.ticks = element_blank())

ggplot(db.plain, aes(x = 1, y = as.numeric(speed), group = opt)) +
  geom_boxplot(notch = T) +
  facet_grid(opt~switch, scales="free_y") +
  ggtitle("Plain - Speed - Switchbacks\n") +
  xlab("") + ylab("Speed traveled (km/h)\n") +
  theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
  guides(fill=FALSE) +
  theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5),
axis.ticks = element_blank())

# Use t-tests to see if the differences noted in the graphs are statistically significant.
```

```
# The results are exported as a csv file.

# Create the dataset that will record the results.
res.x <- data.frame(optimization=double(0),
                    val.a=double(0),
                    val.b=double(0),
                    p.value=double(0))

# The viewshed values used in the simulations
view.pos <- c(0,0.001,0.01,0.02,0.05,0.1)

# The optimization used in the simulations
opt.pos <- c("Distance","Exploration","Speed")

# Iterate over the optimization possibilities
for (i in opt.pos){

  # Iterate over the viewshed values
  for (j in view.pos){

    # This second iteration over the viewshed values ensure the t-test is not done on two identical samples
    for (k in view.pos){

      # The t-test is done only on samples from simulations using different viewshed values
      if (j != k){

        # Create a new row for each combination
        nrow_res = nrow(res.x)
        new_row = nrow_res + 1
        res.x[new_row,] <- as.numeric(0)

        # Assign values to each row.
        res.x[new_row,]$optimization <- i
        res.x[new_row,]$val.a <- j
        res.x[new_row,]$val.b <- k

        # Create subsets of the db.plain dataset where each sample has the correct optimization (i) and viewshed
value (j and k, respectively)
        sub.a <- subset(db.plain$dist, db.plain$opt == i & db.plain$view == j)
        sub.b <- subset(db.plain$dist, db.plain$opt == i & db.plain$view == k)

        # The test is only done if both samples have more than one unique value (avoid errors)
        if (length(unique(sub.b))>1 & length(unique(sub.a))>1){

          # Records the p-value of the two sided t-test.
          p <- t.test(sub.a,sub.b)$p.value
        }else{

          # If the samples cannot be used for t-test, no value is recorded
          p <- NA
        }

        # Assign the p-value to the results dataset, rounding the decimals.
```

```
      res.x[new_row,]$p.value <- format(round(p, 4), nsmall = 4, na.omit=T)
      assign('res.x',res.x, envir = .GlobalEnv)

    }
   }
  }
}

res.x$var <- paste(res.x$optimization,"_",res.x$val.a, sep = "")
res.x$optimization <- NULL
res.x$val.a <- NULL
res.y <- reshape(res.x, idvar="var", timevar="val.b", direction="wide")

write.csv(res.y, "[write path to file here]")

## MOUNTAIN ##

ggplot(db.mountain, aes(x = 1, y = as.numeric(dist), group = opt)) +
 geom_boxplot(notch = T) +
 facet_grid(opt~view, scales="free_y") +
 ggtitle("Mountain - Distance - Viewshed\n") +
 xlab("") + ylab("Distance traveled (km)\n") +
 theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
 guides(fill=FALSE) +
 theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db.mountain, aes(x = 1, y = as.numeric(speed), group = opt)) +
 geom_boxplot(notch = T) +
 facet_grid(opt~view, scales="free_y") +
 ggtitle("Mountain - Speed - Viewshed\n") +
 xlab("") + ylab("Speed traveled (km/h)\n") +
 theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
 guides(fill=FALSE) +
 theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db.mountain, aes(x = 1, y = as.numeric(dist), group = opt)) +
 geom_boxplot(notch = T) +
 facet_grid(opt~switch, scales="free_y") +
 ggtitle("Mountain - Distance - Switchbacks\n") +
 xlab("") + ylab("Distance traveled (km)\n") +
 theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
 guides(fill=FALSE) +
 theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))

ggplot(db.mountain, aes(x = 1, y = as.numeric(speed), group = opt)) +
 geom_boxplot(notch = T) +
 facet_grid(opt~switch, scales="free_y") +
 ggtitle("Mountain - Speed - Switchbacks\n") +
 xlab("") + ylab("Speed traveled (km/h)\n") +
 theme_bw(base_size = 16) %+replace% theme(strip.background  = element_blank()) +
 guides(fill=FALSE) +
 theme(axis.text.y=element_text(size=10), axis.text.x = element_blank(), plot.title = element_text(hjust = 0.5))
```

```
# Use t-tests to see if the differences noted in the graphs are statistically significant.
# The results are exported as a csv file.

# Create the dataset that will record the results.
res.x <- data.frame(optimization=double(0),
                    val.a=double(0),
                    val.b=double(0),
                    p.value=double(0))

# The viewshed values used in the simulations
view.pos <- c(0,0.001,0.01,0.02,0.05,0.1)

# The optimization used in the simulations
opt.pos <- c("Distance","Exploration","Speed")

# Iterate over the optimization possibilities
for (i in opt.pos){

  # Iterate over the viewshed values
  for (j in view.pos){

    # This second iteration over the viewshed values ensure the t-test is not done on two identical samples
    for (k in view.pos){

      # The t-test is done only on samples from simulations using different viewshed values
      if (j != k){

        # Create a new row for each combination
        nrow_res = nrow(res.x)
        new_row = nrow_res + 1
        res.x[new_row,] <- as.numeric(0)

        # Assign values to each row.
        res.x[new_row,]$optimization <- i
        res.x[new_row,]$val.a <- j
        res.x[new_row,]$val.b <- k

        # Create subsets of the db.mountain dataset where each sample has the correct optimization (i) and viewshed
value (j and k, respectively)
        sub.a <- subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == j)
        sub.b <- subset(db.mountain$dist, db.mountain$opt == i & db.mountain$view == k)

        # The test is only done if both samples have more than one unique value (avoid errors)
        if (length(unique(sub.b))>1 & length(unique(sub.a))>1){

          # Records the p-value of the two sided t-test.
          p <- t.test(sub.a,sub.b)$p.value
        }else{

          # If the samples cannot be used for t-test, no value is recorded
          p <- NA
        }
```

```
      # Assign the p-value to the results dataset, rounding the decimals.
      res.x[new_row,]$p.value <- format(round(p, 4), nsmall = 4, na.omit=T)
      assign('res.x',res.x, envir = .GlobalEnv)
     }
    }
   }
}

res.x$var <- paste(res.x$optimization,"_",res.x$val.a, sep = "")
res.x$optimization <- NULL
res.x$val.a <- NULL
res.y <- reshape(res.x, idvar="var", timevar="val.b", direction="wide")

write.csv(res.y, "[write path to file here]")

################################################
## IDENTIFYING THE NUMBER OF NECESSARY RUNS ##
################################################

# Use CV to evaluate after how many run the variability stabilizes

# Create a function that analyses and produce the wanted graph automatically.
ttstplot.cv <- function(data, variable){

 # Select the values of the variable identified from the data (reduce the size of the data handled)
 b = match(variable, colnames(data))
 full_samp <- data[,b]

 # Calculate the Coefficient of Variation of the full sample as a reference
 full_cv <- (sd(full_samp, na.rm=TRUE))/(mean(full_samp, na.rm=TRUE))

 # Create a dataset to record the CV values
 res_cv <- nrow(600)

 # Iterate over a list range 5-3000 at intervals of 10.
 for (i in seq(5,3000,5)){

  # Select the first i values of the full_samp dataset
  samp <- full_samp[0:i]

  # Calculate the CV of that smaller sample and assign it to the res_cv dataset
  res_cv[i] <- (sd(samp, na.rm=TRUE))/(mean(samp, na.rm=TRUE))

  # Assign the res_cv to the Global Environment to use outside the loop
  assign('res_cv',res_cv, envir = .GlobalEnv)
 }

 # Create a graph that shows the changes in CV as the sample size increases
 ggplot() +
  geom_point(aes(x=c(1:length(res_cv)), y=res_cv)) +
  ggtitle("Distance traveled") +
  geom_segment(aes(x = 0, y = full_cv, xend = 3000, yend = full_cv, colour = "red")) +
  xlab("Sample size") + ylab("CV\n") +
```

```
  theme_bw(base_size = 16) +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_continuous(minor_breaks = seq(0 , 3000, 100), breaks = seq(0, 3000, 1000)) +
  guides(colour = "none")
}

# Browse to the output file from BehaviorSpace
results <- read.csv(file.choose(), skip = 6)

# Call on the function, here looking at distance traveled (can be changed)
ttstplot.cv(results, "dist.traveled")

################################################################
## EFFECT OF VIEWSHED AND SWITCHBACK ON ONE SPECIFIC ROUTE ##
################################################################

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

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

# Browse to the main 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)

# This creates a dataset recording the unique sites' combinations
master.comb <- unique(master[,c('switchbacks','viewshed.threshold')])

# 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

# Assign the folder in which all the individual simulation output files are located
my_dir = "[write folder path 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 (m in 1:nrow(master.comb)){
```

```
  # 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))

  # Take the switchbacks and viewshed threshold values to select the files to analyze
  sw <- master.comb$switchbacks[m]
  vt <- master.comb$viewshed.threshold[m]

  # To see progress
  print(sw)
  print(vt)

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

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

  # 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 map name
    filename.split = strsplit(new.file,"_")
    filename.split = unlist(filename.split)
    time.stamp = filename.split [6]
    switch = filename.split[4]
    viewshed = filename.split[5]

    # If that specific run has the correct viewshed and switchback values, read it and collate its data. If it is not, skip
it.
    if (switch==sw && viewshed==vt){

      ori = gsub("\\)|\\(", "", filename.split[6]) # Origin of the run
      g = gsub("\\)|\\(", "", filename.split[7]) # Goal of the run

      # 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)
    }
  }
}
Sys.sleep(0.1)
# update progress bar
setTxtProgressBar(pb, l)

################################################
## USING ROUTES TO IDENTIFY MOST POPULAR PATH ##
################################################
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)]

  # 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 <- dat

  # 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]/Sensitivity_",sw,"_",vt,".asc", sep = ""), overwrite = T)
}

r.sub[is.na(r.sub)] <- 0
```