Animal movement simulation

Pierre Cottais & An Hoàng

30/11/2021

Contents

1 Generating individual animal step lengths data (utilization distribution) 1

2 Use of `survival` package and its function `clogit()` 6

1 Generating individual animal step lengths data (utilization distribution)

Probability of obtaining a sample at some distance, $l'_{t,i}$ from the previous observed point ($l'_{t,i} = ||x'_{t,i} - x$ is given by the gamma PDF:

$$g(l'_{t,i}|b_1, b_2) = \frac{1}{\Gamma(b_1).b1^{b2}}.l'^{b_1-1}_{t,i}.e^{-\frac{l'_{t,i}}{b_2}}$$

```
# Environment grid ver2 (used later in moves  () function)
lat <- seq(-100,100, 1)
long <- seq(-100,100, 1)
n <- length(lat)
m <- length(long)
envi <- matrix(data = runif(n*m),nrow = n, ncol = m)
colnames(envi) <- lat
rownames(envi) <- long
```

```
moves <- function(steps, rho = 5, mu = 1.7918, omega = 1){

  ####################################################
  # building smoothed gird-landscape depending on rhô
  env_list <- list(0, 0, 0)
    new_env <- envi
    # sliding window
    for (i in 1:(n-rho)){
      for (j in 1:(m-rho)){
        sub <- envi[i:(i+rho),j:(j+rho)]
        new_env[i,j] = mean(sub)
      }
    }
```

```
  env_rho <- new_env[1:(n-rho),1:(m-rho)]
  # apply truncated redistribution calculus kernel to the whole grid-landscape
  env_exp <- exp(omega*env_rho-mu)
  # adding two null columns at the "borders"
  env_exp <- cbind(rep(0, times = nrow(env_exp)),
                   env_exp,
                   rep(0, times = nrow(env_exp)))
  # adding two null rows at the "borders"
  # adding two null columns at the "borders"
  env_exp <- rbind(rep(0, times = ncol(env_exp)),
                   env_exp,
                   rep(0, times = ncol(env_exp)))

  env_list[[1]] <- env_rho
  env_list[[2]] <- env_exp
  env_list[[3]] <- rho

#####################################################
# building movements (from coordinates) of one individual
# vectors to be returned in a list
coordinates <- cbind(rep(0, steps+1), rep(0, steps+1))
prob_avail <- matrix(0, nrow = steps+1, ncol = 9)
num_cell <- rep(0, steps+1)
# matrix of avaible movements
code <- cbind(rep(c(-1, 0, 1), 3), rep(c(-1, 0, 1), each = 3))
# starting point (deterministic)
coordinates[1,] <- which(env_exp == max(env_exp), arr.ind = TRUE)
# coordinates[1,] <- c(2, 197)
latitude <- coordinates[1,1]
longitude <- coordinates[1,2]
prob_avail[1,] <- c(0, 0, 0, 0, 1, 0, 0, 0, 0)
num_cell[1] <- 5

for(i in 1:steps){
  # print(dim(kernel))
  latitude <- coordinates[i,1]
  longitude <- coordinates[i,2]
  kernel <- env_exp[(latitude-1):(latitude+1),
                    (longitude-1):(longitude+1)]
  probs <- kernel/sum(kernel)
  prob_avail[i+1,] <- c(probs)
  num_cell[i+1] <- which(rmultinom(1, 1, prob_avail[i+1,])==1)
  move <- code[num_cell[i+1],]
  coordinates[i+1,] <- c(latitude+move[1], longitude+move[2])
}
coord_dt <- as.data.frame(coordinates, row.names = FALSE)
colnames(coord_dt) <- c("row", "col")
list_avail <- list(prob_avail, num_cell)
#####################################################
res <- list(env_list, coord_dt, list_avail)
names(res) <- c("envi", "coord", "probs")
names(res$envi) <- c("env_rho", "env_exp", "rho")
names(res$probs) <- c("available", "chosen")
```

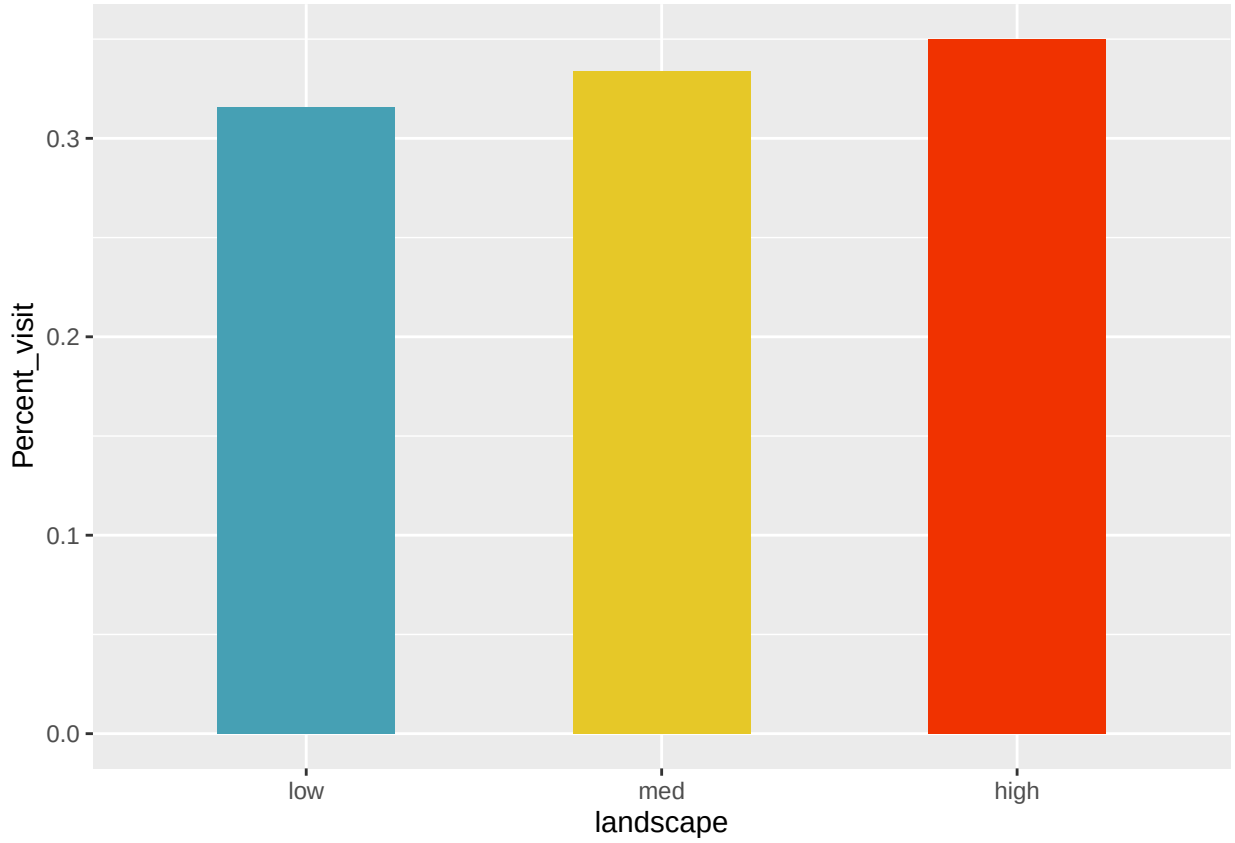
```
  return(res)
}
```

```
nb_steps <- 1000000
res <- moves(nb_steps, rho = 10)
env_rho <- res$envi$env_rho  # grid-landscape with rhô transformation
env_exp <- res$envi$env_exp  # grid-landscape with exp transformation
rho <- res$envi$rho  # rhô value used in smoothing grid
coord <- res$coord  # coordinates of used cells
probs <- res$probs  # list of :
#                     - vector of available probabilities
#                     - rank of the chosen probability in the vector

dim <- ncol(env_exp)-2
env_dt <- cbind(rep(2:(dim+1), times = dim),
                rep(2:(dim+1), times = rep(dim, dim)),
                c(env_rho)) %>% as.data.frame()
colnames(env_dt) <- c("lat", "long", "resource")
env_dt <- env_dt %>%
  mutate(landscape = cut(
    resource,
    breaks = quantile(resource, probs = c(0:3/3), na.rm = TRUE),
    # labels = c("Montant faible", "Montant moyen", "Montant fort"),
    labels = c("low", "med", "high"),
    include.lowest = TRUE
  ))
```

Avec ggplot2...

```
dim <- ncol(env_exp)-2
env_dt <- cbind(rep(2:(dim+1), times = dim),
                 rep(2:(dim+1), times = rep(dim, dim)),
                 c(env_rho)) %>% as.data.frame()

colnames(env_dt) <- c("lat", "long", "resource")

env_dt <- env_dt %>%
  mutate(landscape = cut(
    resource,
    breaks = quantile(resource, probs = c(0:3/3), na.rm = TRUE),
    # labels = c("Montant faible", "Montant moyen", "Montant fort"),
    labels = c("low", "med", "high"),
    include.lowest = TRUE
  ))

burn_coord <- coord[1:nb_steps*0.2,]
coord <- coord[(nb_steps*0.2+1):nrow(coord),]

index <- as.numeric(row.names(coord))
samp_ind <- index%%100==0
samp <- coord[samp_ind,]

perc_visit <- samp %>%
```

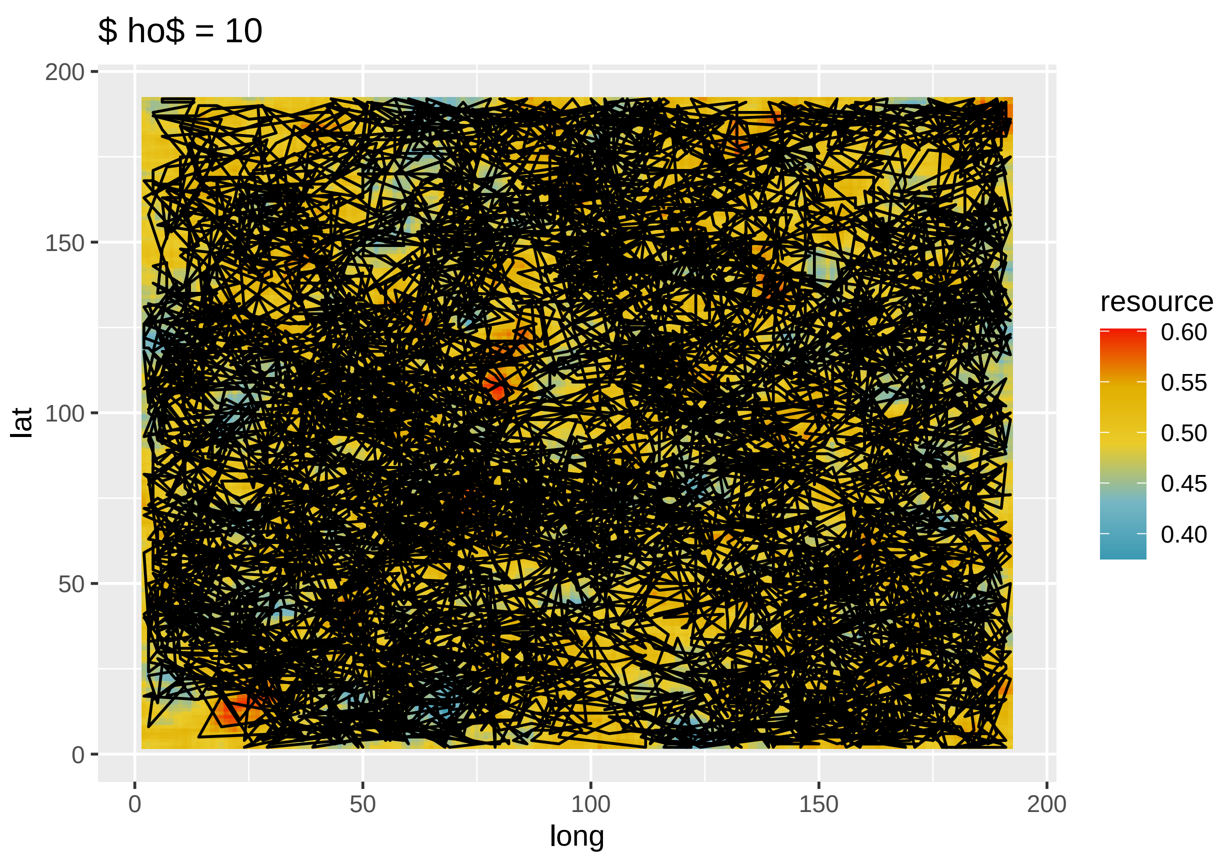
```
  inner_join(env_dt, by = c("row"="lat", "col"="long")) %>%
  group_by(landscape) %>%
  summarise(Percent_visit = n()/nrow(samp))

pal_disc <- c("#46A0B4",  "#E6C828", "#F03200")

perc_visit %>% ggplot() + aes(y = Percent_visit, x = landscape, fill = landscape) +
  geom_col(width = 0.5, fill = pal_disc)
```




```
pal <- wes_palette(40401, name = "Zissou1", type = "continuous")

p_move <- env_dt %>%
  ggplot() + geom_tile(aes(x = long, y = lat, fill = resource)) +
  scale_fill_gradientn(colours = pal) +
  geom_path(data = samp, aes(x = row, y = col)) +
  ggtitle(label = paste("$\rho$ =", rho))
  # geom_path(data = samp, aes(x = row, y = col))+
  # geom_point(data = samp, aes(x = row, y = col))
p_move
```

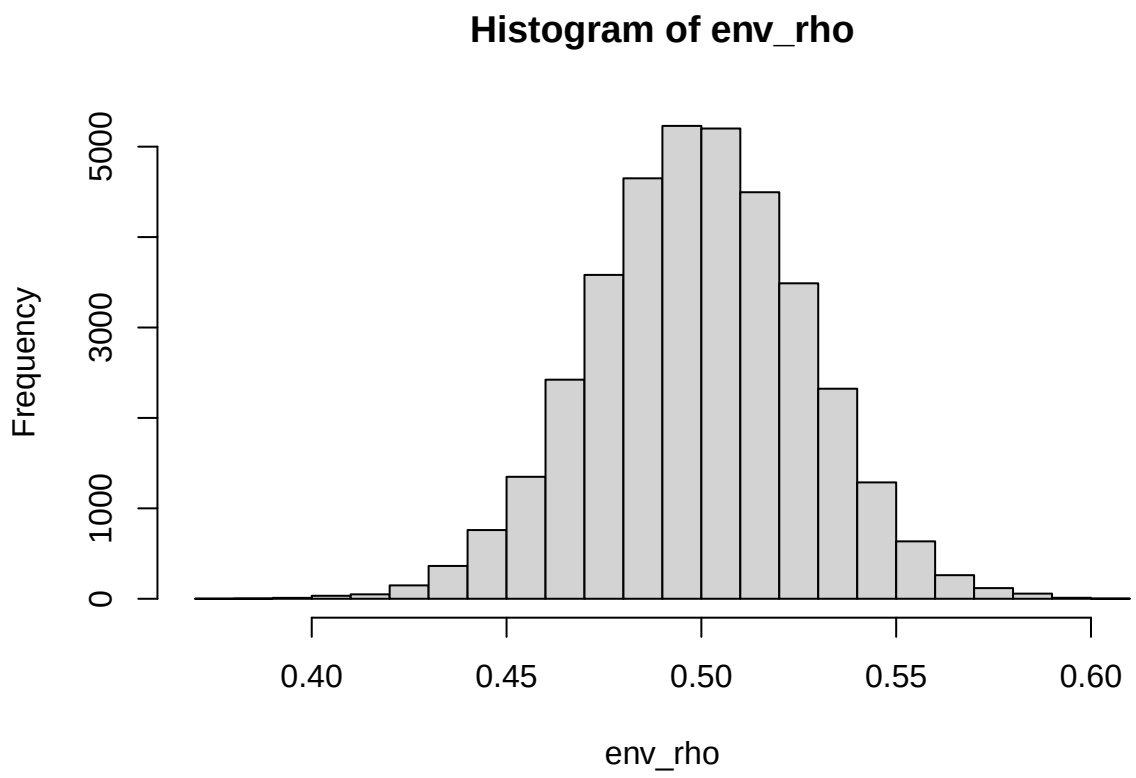
```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## largeur de police inconnue pour le caractère 0xd
```




```
# png("figures/moves.png",width=9,height=6.5,units="in",res=300)
# par(omi=c(0.65,0.25,0.75,0.75),mai=c(0.3,2,0.35,0),mgp=c(3,3,0), las=1)
# p_move
# dev.off()
```

```
hist(env_rho)
```



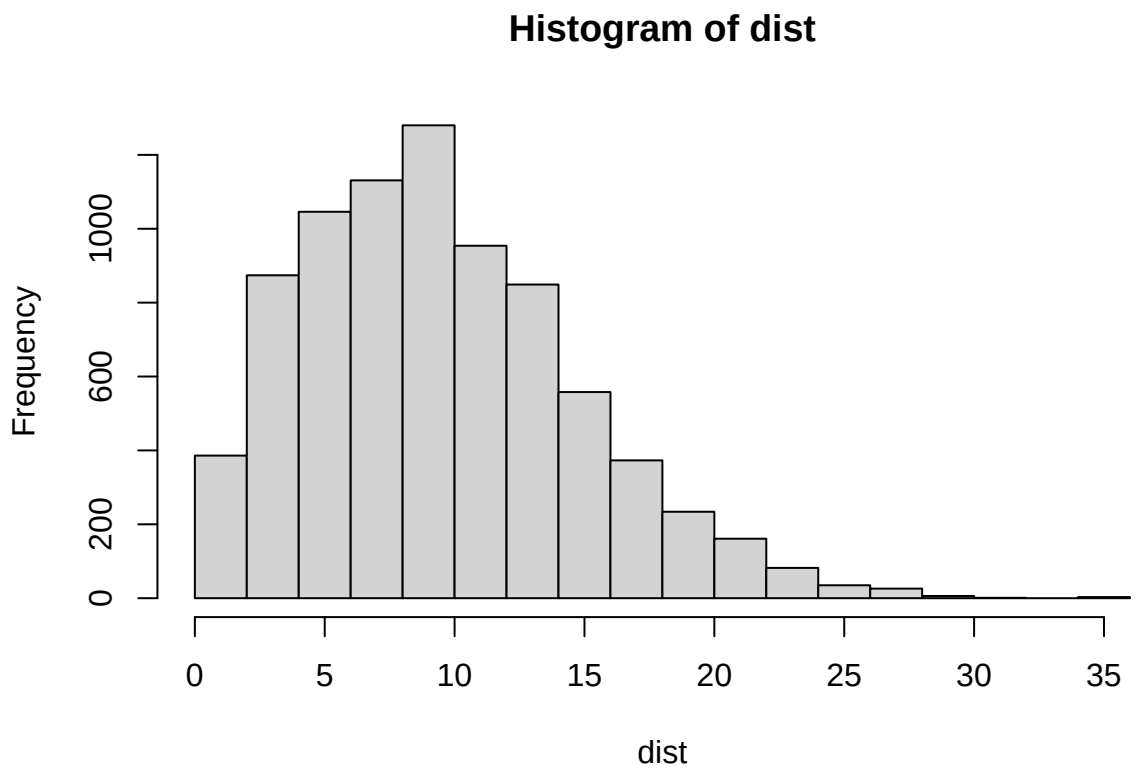
2 Use of `survival` package and its function `clogit()`

```
library(survival)
str(logan)
```

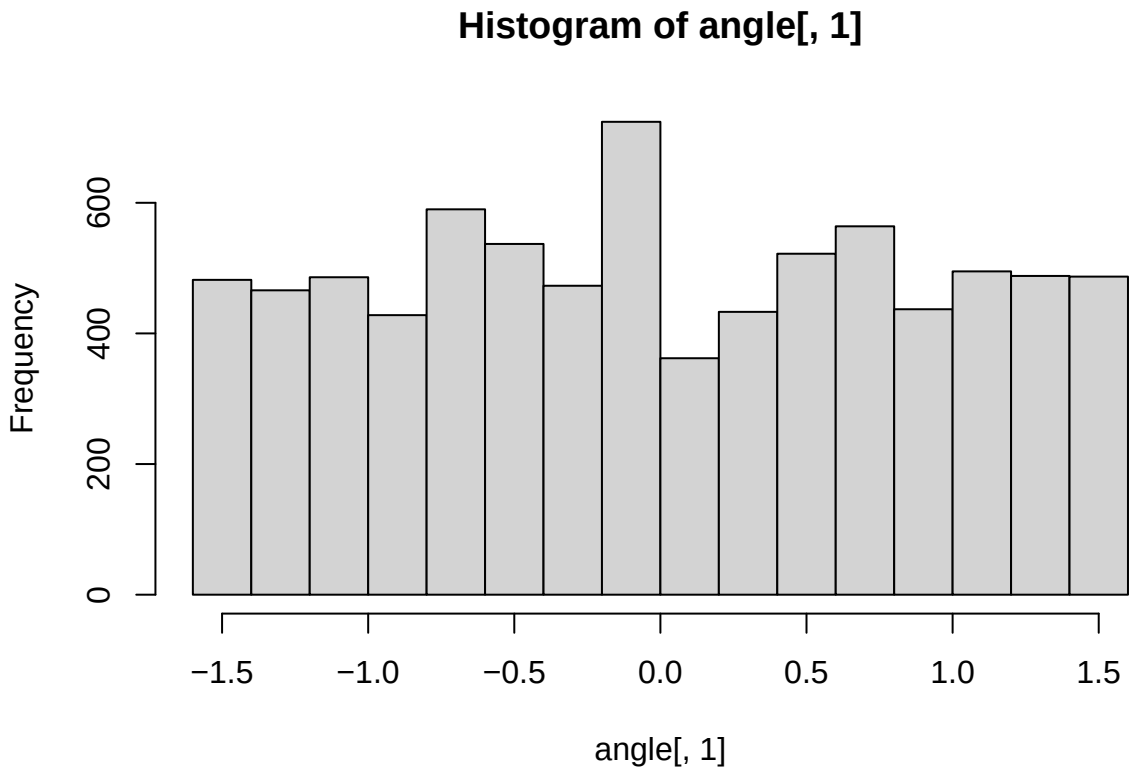
```
data("infert")
clogit(case ~ spontaneous + induced + strata(stratum), data=infert)
# }
# NOT RUN {
# A multinomial response recoded to use clogit
#  The revised data set has one copy per possible outcome level, with new
#  variable tocc = target occupation for this copy, and case = whether
#  that is the actual outcome for each subject.
# See the reference below for the data.
resp <- levels(logan$occupation)
n <- nrow(logan)
indx <- rep(1:n, length(resp))

logan2 <- data.frame(logan[indx,],
                     id = indx,
```

```
                    tocc = factor(rep(resp,
                                      each=n)))
logan2$case <- (logan2$occupation == logan2$tocc)
# logan2 %>%
#   mutate(tocc_num = case_when(case==TRUE~1,
#                               case==FALSE~0)) %>%
#   group_by(id) %>%
#   summarise(sum(tocc_num, na.rm = TRUE)) %>%
#   head()
clogit(case ~ tocc + tocc:education + strata(id), logan2)
# }
```

```
#distribution of step length

dist <- rep(0,length(samp$row)-1)

for (i in 1:(length(samp$row)-1)){
  dist[i] = sqrt(sum((samp[i+1,]-samp[i,])^2))
}

dist <- round(dist)

hist(dist)
```



```r
#distribution of angle
angle <- rep(0,length(samp$row)-1)

for (i in 1:(length(samp$row)-1)){
  a = samp[i+1,] - samp[i,]
  angle[i] = atan(a$row/a$col)
}

angle <- as.data.frame(angle)
hist(angle[,1])
```




```r
n <- length(samp$row)-1 # number of steps
s <- 10                 # number of spatial points estimate at each step

# match coordinate with env_rho
summary(samp)
samp <- samp-1

# for loop
max <- max(env_rho)
min <- min(env_rho)
delta <- max - min

# subset environment habitat
h <- rep(0,n+1)
```

```
col <- samp$col
row <- samp$row
for (i in 1:(n+1)){
  h[i] <- env_rho[row[i],col[i]]
}


#data frame for clogit function
esti.l <- as.data.frame(dist) #distance
colnames(esti.l) <- "dist"
esti.l$ID <-  paste(samp$row,samp$col, sep = "")[-1]#point ID
esti.l$h_end <- h[-1]
esti.l$case_control <- rep(1,n)
esti.l$h_start <- h[-length(h)]

# length match for each step
k <- 2*(1+(max - esti.l$h_start)/delta)
q <- 5/(1+(esti.l$h_start - min)/delta)
avail_matrx <- matrix(round(rgamma(s, shape = k, scale = q), 2),
                      nrow = n, ncol = s)
use_length <- dist

# for (i in 1:n){
#   #estimate gamma distribution
#   k <- 2*(1+(max - esti.l$h_start[i])/delta)
#   q <- 5/(1+(esti.l$h_start[i] - min)/delta)
#
#   use_length <- dist[i]
#
#   # if(sum(length == use_length) > 0){
#   #   esti.l$match[i] <- 1
#   # } else {
#   #   esti.l$match[i] <- 0
#   # }
# }
```

```
# generating data frame of available steps
df_avail <- as.data.frame(avail_matrx)
df_avail$ID <- esti.l$ID
df_avail <- df_avail %>% pivot_longer(cols = 1:10, values_to = "dist") %>%
  select(-name) %>%
  left_join(esti.l[,-which(names(esti.l)=="dist")], by = "ID") %>%
  mutate(case_control = 0)
df_step <- bind_rows(df_avail, esti.l) %>%
  arrange(ID, desc(case_control))
```

Run `clogit` regression

```
#add some variables
df_step$ln_dist <- log(df_step$dist) #ln(dist)
df_step$lh <- df_step$dist*df_step$h_start  # dist * h at previous step
df_step$ln_lh <- df_step$ln_dist*df_step$h_start #ln(dist) * h at previous step
```

```r
#remove step where dist = 0
df_reg <- subset(df_step, dist > 0)

mod1 <- clogit(case_control ~ h_end + dist + ln_dist + lh + ln_lh + strata(ID), data=df_reg)

stargazer(mod1, type = "latex")
```