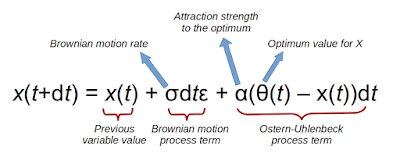
 Hi there!

 Was training some ways to simulate animal (or other organisms) movements having into account habitat suitability. To do this, I used my previous eWalk model as the underlying process to simulate random or directional walks. This model is based on Brownian / Ornstein–Uhlenbeck process.

Code Chunks – eWalk Model

Remember that an OU model describes a stochastic Markov process that consist of two parts: first, a Brownian motion (random walk) model that represents a completely random process of variable evolution, and a second term that represents the attraction of an optimum value of the variable. For this, OU process can be considered as a variation of Brownian motion model. In the last posts we were talking about these kinds of models used to drive trait character evolution, specifically an ecological trait, such as optimum temperature or precipitation.

[](https://2.bp.blogspot.com/-1KJatVQO5tA/WBn8FgdGuFI/AAAAAAAAAJ8/xGvjneoGrcQnPSO3cQjpeSv2RVGT0LEKQCLcB/s1600/Sin%2Bt%25C3%25ADtulo_2.png)

 Mycolleague Javi and I were talking about the applications of this models in two dimensions, representing for example space coordinates (latitude/longitude). They have been used, for example, to simulate predator/prey movements, home range movements, etc. We developed an R function similar to the previous *eMotion* function in order implement this kind of movement models: ***eWalk****(*from *evolutiveWalk...*). *eWalk* is similar to other movement-simulation functions in *adehabitatLT*package (Calenge, 2006). However, you can use *eWalk*to simulate a variety of processes as Brownian motion, OU processes or Lèvy flights (see below).

Here you have the (commented!) code to load the function into R console:

eWalk <- function(sigma,lon,lat, generations,alpha\_x,theta\_x,alpha\_y,theta\_y, color,levy=FALSE, plot=TRUE) {

route=data.frame(1:generations,1:generations) # a data.frame to store the route

names(route)<-c("lon","lat")

generation= generations # generation counter

while (generation > 0){ # stops the simulation when generation counter is 0

if (levy == TRUE){ # only if levy=TRUE: Levy Flight!

rndm=abs(rcauchy(1)) # process to select if I should jump or not

if (rndm > 15){ # Levy jump probability threshold

l=15 # length of the jump

x= lon + (l \* (sigma \* rnorm(1))) + (alpha\_x \* ( theta\_x - lon)) # Brownian or OU process for longitude

y= lat + (l \* (sigma \* rnorm(1))) + (alpha\_y \* ( theta\_y - lat)) # Brownian or OU process for latitude

loc=cbind(x,y)

route[generation,]<- loc # store new location

generation= generation - 1 # advance to next generation

lon<- x # go to the new position!

lat<- y

}

else{ # if Levy jump is not selected, l=1 i.e. proceed with

l=1 #normal Brownian motion or OU process

x= lon + (l \* (sigma \* rnorm(1))) + (alpha\_x \* ( theta\_x - lon))

y= lat + (l \* (sigma \* rnorm(1))) + (alpha\_y \* ( theta\_y - lat))

loc=cbind(x,y)

route[generation,]<- loc

generation= generation - 1

lon<- x

lat<- y

}

}

else{ # if levy=FALSE, l=1, only Brownian motion or OU

l=1

x= lon + (l \* (sigma \* rnorm(1))) + (alpha\_x \* ( theta\_x - lon))

y= lat + (l \* (sigma \* rnorm(1))) + (alpha\_y \* ( theta\_y - lat))

loc=cbind(x,y)

route[generation,]<- loc

generation= generation - 1

lon<- x

lat<- y

}

}

if (plot == TRUE) { # if plot=TRUE, plto the route!

lines(route, col= color, lwd=3)

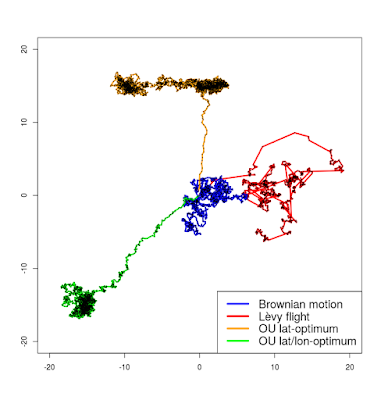
}

return(route) # print the data.frame with each location during the walk!

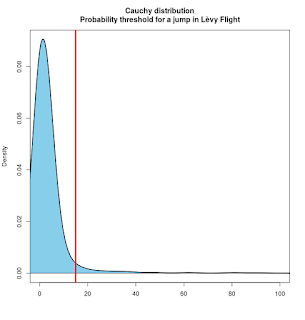
}

>

 The needed parameters to run the function are similar to those for *eMotion*, but taking into account that now you have two variables (latitude/longitude, for example) at the same time.

[](https://2.bp.blogspot.com/-lg3Q0zm_oyY/WBoB2GsSmGI/AAAAAAAAAKQ/TPSIo_hd6loJC2CEZLhAxkNWr6in4zvZgCLcB/s1600/Rplot.png)

eWalk(sigma,initial\_lon,initial\_lat,number of generations, lon\_alpha, lon\_optimum, y\_alpha, y\_optimum, color, levy=FALSE, plot=TRUE)

[](https://3.bp.blogspot.com/-rx_Pv97SFKU/WBoB-_W49EI/AAAAAAAAAKU/Hr_E4jbO5QICihHJmj3SJffC6Xo-n8xwQCLcB/s1600/cauchy.png)

In addition, I have added a complement to simulate Lèvy flights, a non-gaussian process similar to Brownian motion but allowing for jumps during the motion. Lévy flights pattern has been found in searching movements of predators (Bartumeus et al., 2005), or have been used to simulate punctuated equilibrium in traits evolution (Gould and Eldredge, 1977; Landis et al., 2013). To perform Lèvy flight, eWalk selects a random number from Cauchy distribution and if it is higher than a threshold (15 by default), Brownian motion term is multiplied by a constant representing a jump (also 15 by default). Although this is not the formal expression of a Lèvy process, this is a simple way to simulate a negative exponential rate in for the length of each step.

Here you have some examples using eWalk and their code to reproduce them:

par(mfrow=c(2,2))

a=eWalk(0.15,0,0,1000, 0, 0, 0, 0, color="blue", levy=FALSE, plot=FALSE)

plot(a,type="n", main="Brownian motion")

lines(a,lwd=3, col="blue")

points(a, cex=.3, col="black", pch=21)

a=eWalk(0.15,0,0,1000, 0, 0, 0, 0, color="red", levy=TRUE, plot=FALSE)

plot(a,type="n", main="Lêvy flight")

lines(a,lwd=3, col="red")

points(a, cex=.3, col="black", pch=21)

a=eWalk(0.15,0,0,1000, 0.01, 20, 0, 0, color="orange", levy=FALSE, plot=FALSE)

plot(a,type="n", main="OU one optimum")

lines(a,lwd=3, col="orange")

abline(v=20, col="red", lwd="2")

points(a, cex=.3, col="black", pch=21)

a=eWalk(0.15,0,0,1000, 0.01, 20, 0.01, 20, color="green", levy=FALSE, plot=FALSE)

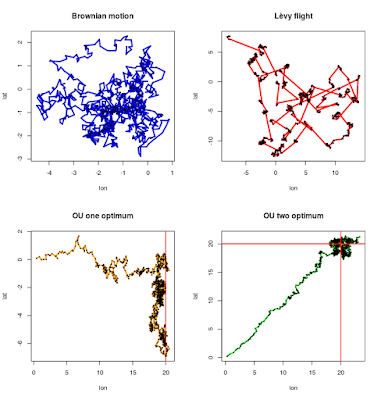
plot(a,type="n", main="OU two optimum")

lines(a,lwd=3, col="green")

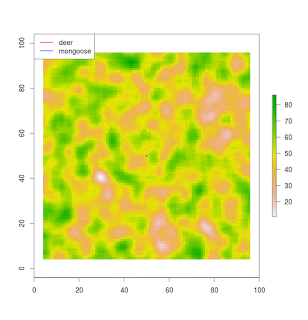
points(a, cex=.3, col="black", pch=21)

abline(v=20, col="red", lwd="2")

abline(h=20, col="red", lwd="2")

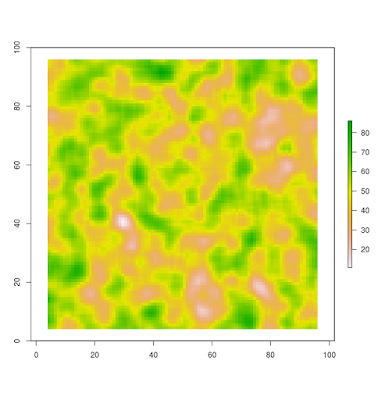
[](https://2.bp.blogspot.com/-LmvQdcrYgdE/WBoCNZhNsjI/AAAAAAAAAKY/Mu_IRYrAmVoY6vWo7op-U9wEkejeNiO7ACLcB/s1600/4plots2.png)

 Today, I will add one more element to this movement simulations. In this case, we will have into account the habitat or environmental preferences of the simulated species, to perform a simulation like this:

[](https://i0.wp.com/4.bp.blogspot.com/-l-bCOkGMWMw/Wui-894bmwI/AAAAAAAAAP4/nD5BRuZWLPAVt7t3Zh57dRr-t4CwmWMxACLcBGAs/s1600/paths.gif?ssl=1)

 First, we will create a raster layer as a random environmental variable, for example tree cover.

library (raster)   
 library (dismo)   
   
 tc <- raster(nrows=100, ncols=100, xmn=0, xmx=100, ymn=0,ymx=100)   
 tc[] <- runif(10000, -80, 180)   
 tc <- focal(tc, w=matrix(1, 5, 5), mean)   
 tc <- focal(tc, w=matrix(1, 5, 5), mean)   
 plot(tc)

[](https://i2.wp.com/1.bp.blogspot.com/-Bb2edCESEEs/Wuivgcq5P6I/AAAAAAAAAO4/EjV9L9_ymSInc4gsxrXHQFBy3FVlqgiSwCLcBGAs/s1600/tc.png?ssl=1)

 Second, we will define the species class. The species will be defined by their position (coordinates), and their optimum for the environmental variable.

species <- setClass("species", slots=c(x="numeric", y="numeric", opt="numeric"))

 Here we will define the Red deer species as a specimen in the coordinates (50,50) and an optimum of 80 for the environmental variable (tree cover). In the same way, we will define the Egyptian mongoose as a specimen in the coordinates (50,50) and an optimum of 30 for the tree cover variable.

Red\_deer <- species(x= 50, y =50, opt= 90)   
 Egyptian\_mongoose <- species(x= 50, y =50, opt= 30)

 Now, we will load the “*go*” function (I do not have a name for it yet). It require a species (*sp*), a raster layer with any environmental variable (*env*), number of iterations (*n*), a Brownian motion parameter (that is, how random is the movement of your species), a geographical optimum (the wanted destination of your species *theta\_x* and *theta\_y*), and the attraction strength or “interest” of the species to get this position (*alpha\_x* and *alpha\_y*). The syntaxis should be something like this:

path <- go (sp, env, n, sigma, theta\_x, alpha\_x, theta\_y, alpha\_y)

 Here is the function to load (I will comment the function in a future post):

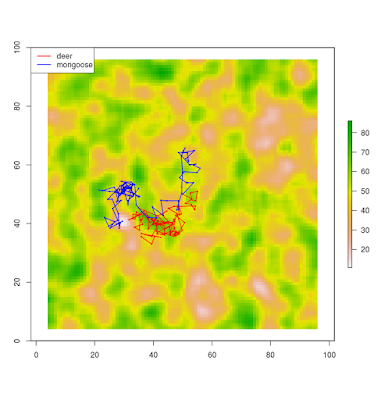
go <- function (sp, env, n, sigma, theta\_x, alpha\_x, theta\_y, alpha\_y) {   
 track <- data.frame()   
 track[1,1] <- sp@x   
 track[1,2] <- sp@y   
 for (step in 2:n) {   
 neig <- adjacent(env,   
 cellFromXY(env, matrix(c(track[step-1,1],   
 track[step-1,2]), 1,2)),   
 directions=8, pairs=FALSE )   
 options <- data.frame()   
 for (i in 1:length(neig)){   
 options[i,1]<-neig[i]   
 options[i,2]<- sp@opt - env[neig[i]]   
 }   
 option <- c(options[abs(na.omit(options$V2)) == min(abs(na.omit(options$V2))), 1 ],   
 options[abs(na.omit(options$V2)) == min(abs(na.omit(options$V2))), 1 ])   
 new\_cell <- sample(option,1)   
 new\_coords <- xyFromCell(env,new\_cell)   
 lon\_candidate<--9999   
 lat\_candidate<--9999   
   
 while ( [is.na](http://is.na)(extract(env, matrix(c(lon\_candidate,lat\_candidate),1,2)))) {   
 lon\_candidate <- new\_coords[1]+ (sigma \* rnorm(1)) + (alpha\_x \* ( theta\_x - new\_coords[1]))   
 lat\_candidate <- new\_coords[2]+ (sigma \* rnorm(1)) + (alpha\_y \* ( theta\_y - new\_coords[2]))   
 }   
 track[step,1] <- lon\_candidate   
 track[step,2] <- lat\_candidate   
 }   
 return(track)   
 }

 Well, now we can perform a simple experiment with our two specimens. We will simulate random movement of these two species having into account their environmental optimums. The “*go*” function will return us the track or the path followed by each specimen (coordinates by each step).

deer\_simul <- go (Red\_deer, tc, 100, 2, 90, 0, 90, 0)   
 mongoose\_simul <- go (Egyptian\_mongoose, tc, 100, 2, 90, 0, 90, 0)

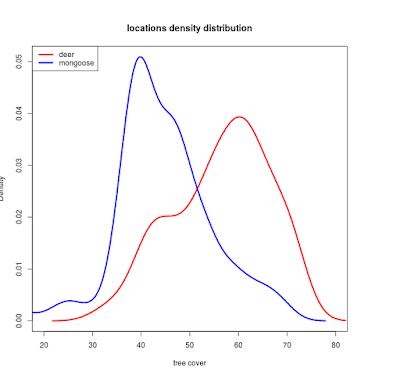
 We can plot the paths…

plot(tc)   
 lines(deer\_simul, lwd=1.5, col="red")   
 points(deer\_simul, cex=0.3, col="red")   
 lines(mongoose\_simul, lwd=1.5, col="blue")   
 points(mongoose\_simul, cex=0.3, col="blue")   
 legend("topleft", legend=c("deer","mongoose"), col=c("red","blue"),   
 lty=c(1,1), lwd=c(2,2))

[](https://i0.wp.com/1.bp.blogspot.com/-e1lO1DOhrrg/Wui2udjjlDI/AAAAAAAAAPY/BJhOCFDPT4YvPDhk66RUCWmwcB2R2PT3wCLcBGAs/s1600/paths.png?ssl=1)

 To test if each species is actually “searching” their environmental optimum, we can extract the environmental values by step for each species and plot their density distributions.

plot(density(extract(tc, deer\_simul)),lwd=3, col="red", xlim=c(20,80),   
 ylim=c(0,max(c(density(extract(tc, deer\_simul))$y,   
 density(extract(tc, mongoose\_simul))$y))),   
 main="locations density distribution", xlab="tree cover")   
 lines(density(extract(tc, mongoose\_simul)),lwd=3, col="blue")   
 legend("topleft", legend=c("deer","mongoose"), col=c("red","blue"),   
 lty=c(1,1), lwd=c(3,3))

[](https://i0.wp.com/2.bp.blogspot.com/-uDp2lf-lA_w/Wui3gvsGFMI/AAAAAAAAAPg/J9Fy7Z9TUpYUk0SMSuACEmafnJJui_T6QCLcBGAs/s1600/Rplot.png?ssl=1)

 So, we can see that the deer is actually using patches with a higher value of tree cover than the mongoose… our simulation worked!

Code Chunks – GIF to explain Brownian Motion

Here I will show you how I built a GIF to explain Brownian Motion in my talk using R and ImageMagick.

# First, we simulate continuous trait evolution by adding in each iteration

# a random number from a normal distribution with mean equal to 0 and standard

# deviation equal to 1. We simulate a total of 4 processes, to obtain at first

# two species and a specieation event at the middle of the simulation, obtaining

# a total of 3 species at the end.

df1<- data.frame(0,0)

names(df1)<- c("Y","X")

y<-0

for (g in 1:750){

df1[g,2] <- g

df1[g,1] <- y

y <- y + rnorm(1,0,1)

}

#plot(df1$X,df1$Y, ylim=c(-100,100), xlim=c(0,1500), cex=0)

#lines(df1$X,df1$Y, col="red")

df2<- data.frame(0,0)

names(df2)<- c("Y","X")

y<-0

for (g in 1:1500){

df2[g,2] <- g

df2[g,1] <- y

y <- y + rnorm(1,0,1)

}

#lines(df2$X,df2$Y, col="blue")

df3<- data.frame(750,df1[750,1])

names(df3)<- c("Y","X")

y<-df1[750,1]

for (g in 750:1500){

df3[g-749,2] <- g

df3[g-749,1] <- y

y <- y + rnorm(1,0,1)

}

#lines(df3$X,df3$Y, col="green")

df4<- data.frame(750,df1[750,1])

names(df4)<- c("Y","X")

y<-df1[750,1]

for (g in 750:1500){

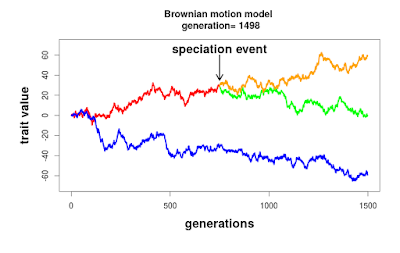
df4[g-749,2] <- g

df4[g-749,1] <- y

y <- y + rnorm(1,0,1)

}

#lines(df4$X,df4$Y, col="orange")

[](https://3.bp.blogspot.com/-cnb_UJUWwCc/WiwEekDcxnI/AAAAAAAAANI/4e9BO5YespYtoetLI1wQ_315ayQd4Kp1wCEwYBhgL/s1600/bm1498.png)

# Now, we have to plot each simmulation lapse and store them in our computer.

# I added some code to make lighter the gif (plotting just odd generations) and

# to add a label at the speciation time. Note that, since Brownan Model is a

# stocasthic process, my simulation will be different from yours.

# You should adjust labels or repeat the simulation process if you don't

# like the shape of your plot.

parp<-rep(0:1, times=7, each= 15)

parp<- c(parp, rep(0, 600))

for (q in 1:750){

if ( q %% 2 == 1) {

id <- sprintf("%04d", q+749)

png(paste("bm",id,".png", sep=""), width=900, height=570, units="px",

pointsize=18)

par(omd = c(.05, 1, .05, 1))

plot(df1$X,df1$Y, ylim=c(-70,70), xlim=c(0,1500), cex=0,

main=paste("Brownian motion model \n generation=", 749 + q) ,

xlab="generations", ylab="trait value", font.lab=2, cex.lab=1.5 )

lines(df1$X,df1$Y, col="red", lwd=4)

lines(df2$X[1:(q+749)],df2$Y[1:(q+749)], col="blue", lwd=4)

lines(df3$X[1:q],df3$Y[1:q], col="green", lwd=4)

lines(df4$X[1:q],df4$Y[1:q], col="orange", lwd=4)

if (parp[q]==0)

text(750, 65,labels="speciation event", cex= 1.5, col="black", font=2)

if (parp[q]==0)

arrows(750, 60, 750, 35, length = 0.20, angle = 30, lwd = 3)

dev.off()

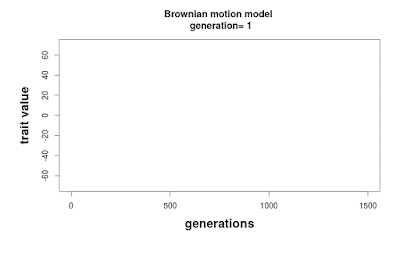
}

}

 Now, you just have to use ImageMagick to put all the PNG files together in a GIF using a command like this in a terminal:

convert -delay 10 \*.png bm.gif

Et voilà!

[](https://1.bp.blogspot.com/-y_ys5AJz5SQ/WiwF_nq8JKI/AAAAAAAAANY/hS7IIviDbPwdSlTmOJEbEtfxdFB7MJ06ACEwYBhgL/s1600/bm_fast2.gif)