;Issues

; step-length - linearly modelled - maybe not the best way to do it?

; Notes

;

; Running ticks until 25568 is approx equal to running from the first release date to the 16th Dec 2050

extensions [

gis ; Loading extra packages

time

csv

profiler

table

]

globals [

dt ; The date-time of the model

lcover ; Landcover values

slo ; Slope values

outline ; Outline of corsica

distance-rd ; Distance to road

sites ; Release sites shp

test\_file ; Just a global variable for exporting the n-visit.asc file

season ; tracks whether it's summer or winter -

focal-deer ; the deer currently being modelled

step-distances ; list of step distances for evaluation

target-patches ; The target patches used for deciding which patches to move deer to

hpc-stop ;should the hpc stop?

run-id ;identifying shared runs

dead-deer-who ; list of dead deer HR sizes

dead-deer-HR ; list of dead deer HR sizes

parameter-table

]

breed [

deer a-deer ; Assigning deer as a breed (singlular and plural)

]

breed [

release-sites release-site ; Release sites

]

patches-own [

type-of-landcover ; The type of landcover

slope-value ; The slope

distance-to-road ; The distance of every patch to the nearest significant road

land ; are the patches on land?

movement-prob ; for each deer and each tick, the probability said deer will move to the patch

n-visits ; Number of deer visiting each patch

]

deer-own [

age ; age of deer in 12 hour ticks

sex-of-deer ; Each deer is assigned a sex

prior-patch ; For calculating step distances

home-range-patches ; List of patches representing an individuals home range

release-site-x ; x coords of release site

release-site-y ; y coords of release site

offspring ; Variable for if the mother has offspring

mother ; Who ID for mother deer, to follow

independence-countdown ; A countdown for independence for offspring

mature ; Whether a deer is mature or not - and thus whether or not they can breed

]

release-sites-own [

sex\_deer ; The sex of the deer released from this site (different beacuse it's a different variable assoicated with the patch)

time ; The time (in ticks) the deer is to be released - again calculated with lubridates assistance so should be accurate!

]

to setup

clear-all ; Clears everything

setup-parameter-table

set dt time:create-with-format "16-12-2015 10:00:00" "dd-MM-yyyy HH:mm:ss"

set season "winter"

; Patches

;

; Note that setting the coordinate system here is optional, as

; long as all of your datasets use the same coordinate system.

; Read in the spatial data.

set lcover gis:load-dataset "landcover.asc" ; Loading the rasters and assigning them to global variables

set slo gis:load-dataset "slope.asc"

set distance-rd gis:load-dataset "distance.asc"

; make each raster cell = patch in NetLogo

let width floor (gis:width-of lcover / 2)

let height floor (gis:height-of lcover / 2)

resize-world (-1 \* width ) width (-1 \* height ) height

; define your patch size in pixels (makes your world size bigger/smaller in the Interface):

set-patch-size 1

; Set the world envelope to the union of all of our dataset's envelopes

gis:set-world-envelope (gis:envelope-union-of (gis:envelope-of distance-rd) (gis:envelope-of lcover) (gis:envelope-of slo))

; Applying global raster values to patch values (gis:apply-raster global patch)

gis:apply-raster slo slope-value

gis:apply-raster lcover type-of-landcover

gis:apply-raster distance-rd distance-to-road

; Modifying patches (see below comments)

ask patches [

ifelse type-of-landcover >= 0 [set land "yes"] [set land "no"] ; Assigning a patch variable to identify if the patch is land or water (used later)

ifelse (slope-value <= 0) or (slope-value >= 0) [ ; A known 'bug' with the GIS extension assigns patches without a value as 'NaN' - this replaces those with '0'

; do nothing

] [

set slope-value 0 ; If above code is not met (i.e., it's NaN) then replaces with 0

]

]

; Assigning landcover values

ask patches [

(ifelse type-of-landcover = 1 [set type-of-landcover "artificial"]

type-of-landcover = 2 [set type-of-landcover "agricultural"]

type-of-landcover = 3 [set type-of-landcover "forest"]

type-of-landcover = 4 [set type-of-landcover "scrub"]

type-of-landcover = 5 [set type-of-landcover "bare"]

type-of-landcover = 6 [set type-of-landcover "wetlands"]

[set land "no"]

)

]

set step-distances []

set dead-deer-who []

set dead-deer-HR []

; Create release points as turtles

set sites gis:load-dataset "new\_release\_sites\_for\_NetLogo.shp"

gis:create-turtles-from-points sites release-sites [set shape "circle"]

set run-id (round ((random-float 1) \* 1000000))

reset-ticks ; Sets the tick counter to 0

;export-visit-map

end

to setup-parameter-table

set parameter-table table:make

let landcovers ["artificial" "agricultural" "forest" "scrub" "bare" "wetlands"] ; Add all relevant landcover types

let seasons ["summer" "winter"] ; Add all relevant seasons

let sexes ["male" "female"] ; If sex affects parameters

foreach landcovers [ lc ->

; Store general landcover parameters

table:put parameter-table (word "beta-" lc) (runresult (word "beta-" lc))

table:put parameter-table (word "se-" lc) (runresult (word "se-" lc))

; Store step-length, turning angle, and road interaction parameters

table:put parameter-table (word "beta-" lc ":step-length") (runresult (word "beta-" lc ":step-length"))

table:put parameter-table (word "se-" lc ":step-length") (runresult (word "se-" lc ":step-length"))

table:put parameter-table (word "beta-" lc ":turning-angle") (runresult (word "beta-" lc ":turning-angle"))

table:put parameter-table (word "se-" lc ":turning-angle") (runresult (word "se-" lc ":turning-angle"))

table:put parameter-table (word "beta-" lc ":distance-to-road") (runresult (word "beta-" lc ":distance-to-road"))

table:put parameter-table (word "se-" lc ":distance-to-road") (runresult (word "se-" lc ":distance-to-road"))

; Store step-length interaction terms per season and sex

foreach seasons [ s ->

foreach sexes [ sex ->

table:put parameter-table (word "beta-" lc ":step-length:" s ":" sex)

(runresult (word "beta-" lc ":step-length:" s ":" sex))

table:put parameter-table (word "se-" lc ":step-length:" s ":" sex)

(runresult (word "se-" lc ":step-length:" s ":" sex))

]

]

]

; Store slope and road coefficients by season

foreach seasons [ s ->

table:put parameter-table (word "beta-slope:" s) (runresult (word "beta-slope:" s))

table:put parameter-table (word "se-slope:" s) (runresult (word "se-slope:" s))

table:put parameter-table (word "beta-distance-to-road:" s) (runresult (word "beta-distance-to-road:" s))

table:put parameter-table (word "se-distance-to-road:" s) (runresult (word "se-distance-to-road:" s))

]

end

to go

set hpc-stop "no"

;Deer release submodel below

ask release-sites [

if time = ticks [ hatch-deer 1 [

set sex-of-deer [sex\_deer] of myself

set age 731

set home-range-patches []

set home-range-patches fput patch-here home-range-patches

set release-site-x xcor

set release-site-y ycor

set mother "none" ; No mother

set offspring "no" ; No offspring at start

set mature "yes" ; Can breed

]]

]

ask deer [ ifelse pen-down? [pen-down][pen-up] ] ; Record pen marks or not

birth ; Deer birth procedure

move-mature ; deer move procedure

move-immature ; offspring move

death ; Deer death procedure

;Exporting a visit map every 5 years from release until 2050

if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2020 and time:get "hour" dt = 10 [export-visit-map]

if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2025 and time:get "hour" dt = 10 [export-visit-map]

if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2030 and time:get "hour" dt = 10 [export-visit-map]

if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2035 and time:get "hour" dt = 10 [export-visit-map]

if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2040 and time:get "hour" dt = 10 [export-visit-map export-home-range-sizes set hpc-stop "yes"]

;if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2045 and time:get "hour" dt = 10 [export-visit-map]

;if time:get "day" dt = 16 and time:get "month" dt = 12 and time:get "year" dt = 2050 and time:get "hour" dt = 10 [export-visit-map export-home-range-sizes set hpc-stop "yes"]

if hpc-stop = "yes" [stop]

set dt time:plus dt 12 "hours"

if time:get "day" dt = 15 and time:get "month" dt = 4 [set season "summer"]

if time:get "day" dt = 15 and time:get "month" dt = 10 [set season "winter"]

tick-advance 1 ; Add 1 onto the tick counter

update-plots ; Updates any plots on the interface

end

; Sub-models

;

;

to birth

ask deer with [sex-of-deer = "female" and mature = "yes"] [ ; For female mature deer

let rand-number random-float 1

if rand-number > (1 - annual-birth-prob) ^ (1 / 730.5) and offspring = "no" [ ; Need to do calculate from the prob of not giving birth because the only way to not give birth is to not give birth in every 12 hour step, so the maths is easy.

set offspring "yes"

hatch 1 [

ifelse random 2 = 0 [set sex-of-deer "male"][set sex-of-deer "female"]

set mature "no" ; Means that the offspring won't reproduce - they're not mature

set prior-patch patch-here

set mother [who] of myself

set offspring "no" ; Has no offspring

set release-site-x xcor

set release-site-y ycor

ifelse random 2 = 0 [ set independence-countdown 730 ][ set independence-countdown 731 ] ; Average of about a years worth of 12 hour gaps

] ]

]

end

; Deer movement procedure

to move-mature

ask deer with [mother = "none" and mature = "yes"] [ ; Asking the deer (this ask deer individually in a random order) who is mature and independent of a mother

set focal-deer self ; Sets itself as a focal deer (I need to do this for some code below)

ask patch-here [ ; The deer has to ask the patch it is situated on so that the code below works (i.e., the patch the deer is on asks the other patches in a set radius).

set target-patches other patches in-radius max-step-distance with [land = "yes"] ; creates an agentset of land patches within the max-step distance

ask target-patches [ ; Asks these 'target patches'

set movement-prob calculate-movement-probability ; ...to calculate the probability for the focal deer to move towards it

]

let total-movement-prob (sum [movement-prob] of target-patches) ; Finding the sum of the probability for all considered patches for that deer

ask target-patches [ ; Ask these patches...

set movement-prob movement-prob / total-movement-prob ; ... to scale their probabilities to add to one

]

]

move-deer-based-on-probability ; Select a patch to move to based on these probabilities

;let distance-moved (distance prior-patch)

;set step-distances fput distance-moved step-distances

;set prior-patch patch-here

]

end

to move-immature

ask deer with [independence-countdown > 0 and offspring = "no"] [

face a-deer ([mother] of self)

move-to a-deer ([mother] of self)

set independence-countdown independence-countdown - 1

; For the next few lines - saying 'if the distance between my current location and my mums release site/centre of home range is > distance between my release site/home range centre and my mums...update my home range centre to the new one, because it's further away from mums centre of home range without being outside it

if distancexy [release-site-x] of a-deer ([mother] of self) [release-site-y] of a-deer ([mother] of self) > (([release-site-x] of a-deer ([mother] of self) - [release-site-x] of self) ^ 2 + ([release-site-y] of a-deer ([mother] of self) - [release-site-y] of self) ^ 2 ) ^ 0.5 [

set release-site-x xcor

set release-site-y ycor

]

if independence-countdown <= 0 [

ask a-deer ([mother] of self) [set offspring "no"] ; Mother can breed again

set offspring "no" ; Ready to breed

set mother "none" ; No mother anymore

set mature "yes" ; Is now a mature deer and can breed

]

]

end

to death

; Survival rates from 95-99%

ask deer [if age > max-lifespan-years \* 730.5 [die]] ; Deer dies due to intrinsic senescnenc

foreach sort deer [t -> ask t [

let rand-number random-float 1

if mature = "yes" and rand-number > (annual-survival-prob) ^ (1 / 730.5) [

set dead-deer-who fput who dead-deer-who

set dead-deer-HR fput (length remove-duplicates home-range-patches) dead-deer-HR

die

] ; For adult deer, if random number is greater than survival rate for a 12 hour period - deer dies

if mature = "no" and a-deer ([mother] of self) = nobody [die] ; For offspring, if mum dies then they die too

]]

end

; Sub-prodecures and reporters

;

;

;

;

;

;

;

;

;

;

;

;

;

to-report calculate-movement-probability

let step-length ln (distance myself) ; Compute log step-length once

let turning-angle cos (subtract-headings (towards myself - 180) ([heading] of focal-deer)) ; Compute turning angle once

let dist-from-release distancexy [release-site-x] of focal-deer [release-site-y] of focal-deer ; Compute distance from release once

let deer-sex [sex-of-deer] of focal-deer

; Store landcover-based beta and SE values to avoid redundant calls

let beta-landcover table:get parameter-table (word "beta-" type-of-landcover)

let se-landcover table:get parameter-table (word "se-" type-of-landcover)

let beta-landcover-step table:get parameter-table (word "beta-" type-of-landcover ":step-length")

let se-landcover-step table:get parameter-table (word "se-" type-of-landcover ":step-length")

let beta-landcover-turn table:get parameter-table (word "beta-" type-of-landcover ":turning-angle")

let se-landcover-turn table:get parameter-table (word "se-" type-of-landcover ":turning-angle")

let beta-landcover-road table:get parameter-table (word "beta-" type-of-landcover ":distance-to-road")

let se-landcover-road table:get parameter-table (word "se-" type-of-landcover ":distance-to-road")

let beta-slope-season table:get parameter-table (word "beta-slope:" season)

let se-slope-season table:get parameter-table (word "se-slope:" season)

let beta-road-season table:get parameter-table (word "beta-distance-to-road:" season)

let se-road-season table:get parameter-table (word "se-distance-to-road:" season)

let beta-complex table:get parameter-table (word "beta-" type-of-landcover ":step-length:" season ":" deer-sex)

let se-complex table:get parameter-table (word "se-" type-of-landcover ":step-length:" season ":" deer-sex)

; Compute probability

let probs

(random-normal beta-landcover se-landcover) +

(random-normal beta-slope se-slope) \* slope-value +

(random-normal beta-distance-to-road se-distance-to-road) \* distance-to-road +

(random-normal beta-step-length se-step-length) \* step-length +

(random-normal beta-turning-angle se-turning-angle) \* turning-angle +

(random-normal beta-distance-from-release se-distance-from-release) \* dist-from-release +

(random-normal beta-landcover-step se-landcover-step) \* step-length +

(random-normal beta-distance-to-road:turning-angle se-distance-to-road:turning-angle) \* distance-to-road \* turning-angle +

(random-normal beta-landcover-turn se-landcover-turn) \* turning-angle +

(random-normal beta-slope-season se-slope-season) \* slope-value +

(random-normal beta-road-season se-road-season) \* distance-to-road +

(random-normal beta-slope:step-length se-slope:step-length) \* slope-value \* step-length +

(random-normal beta-landcover-road se-landcover-road) \* distance-to-road +

(random-normal beta-slope:turning-angle se-slope:turning-angle) \* slope-value \* turning-angle +

(random-normal beta-complex se-complex) \* step-length

report exp probs

end

to move-deer-based-on-probability; Moves deer to a new patch based on the calculated probabilities

let r random-float 1 ; Random threshold at which the patch will be selected

let cumulative-probability 0 ; Sets the cumulative probability

let selected-patch nobody ; Sets the selected patch as nobody

ask target-patches [ ; Only asking patches the deer could move to

set cumulative-probability cumulative-probability + movement-prob ; Adds on the movement probability - a higher probability means more likely to pass the r threshold

if (selected-patch = nobody and r < cumulative-probability) [ ; If it does pass the threshold AND a patch hasn't already been selected (I.e., it's the first time the threshold is passed)...

set selected-patch self ; ...selects this patch to move towards

stop

]

]

ifelse selected-patch != nobody [ ; If a patch is selected

while [patch-here != selected-patch] [ ; While the patch the focal deer is on is not the selected patch...

let current-patch patch-here ; (Used for avoiding double-counting)

face selected-patch ; ...the focal deer faces the selected patch...

forward 1 ; ...and moves one patch towards it (i.e., 100m here)

if patch-here != current-patch [ask patch-here [patch-count]] ; Each time, the patch the deer is on adds 1 onto it's n-visit variable BUT doesn't doublecount if it doesn't move off the patch

if patch-here != current-patch and offspring = "yes" [ask patch-here [patch-count]] ; Each time, the patch the deer is on adds 1 onto it's n-visit variable for an offspring BUT doesn't doublecount if it doesn't move off the patch

set home-range-patches fput patch-here home-range-patches ; Add the patch to the deers home range

set home-range-patches remove-duplicates home-range-patches ; Removes duplicates - don't need duplicate patches and I'm hoping this may reduce list size and speed the procedure up

]

] [

error (word "Deer has not moved from " patch-here " and sum of patches surrounding the focal deer is " cumulative-probability); If a deer doesn't move there's an error message - currently this is not showing :)

]

end

to patch-count

set n-visits n-visits + 1 ; Adds on one to the patch count

end

to export-visit-map

ask one-of patches [

set test\_file gis:patch-dataset n-visits ; Collects the n-visit variables from the patches

]

gis:store-dataset test\_file (word "output\_maps/n-visits\_" run-id "\_ticks\_" ticks) ; Exports the n-visit raster as an .asc raster

end

to export-home-range-sizes

ask deer [ set home-range-patches remove-duplicates home-range-patches ] ; Double checks that duplicate patches are removed

csv:to-file (word "Results/home-range-sizes\_still\_living\_deer\_" run-id ".csv") [ (list who (length home-range-patches)) ] of deer

csv:to-file (word "Results/home-range-sizes\_dead\_deer\_" run-id ".csv") (list dead-deer-who dead-deer-HR)

end

to colour-by-slope

gis:paint slo 0 ; To colour the map by slope value

end

to colour-by-landcover

clear-drawing

ask patches [ ; To colour the patches by landcover

if type-of-landcover = "artificial" [set pcolor 5]

if type-of-landcover = "agricultural" [set pcolor 25]

if type-of-landcover = "forest" [set pcolor 65]

if type-of-landcover = "scrub" [set pcolor 45]

if type-of-landcover = "bare" [set pcolor 35]

if type-of-landcover = "wetlands" [set pcolor 85]

]

end

to colour-by-distance-to-road

gis:paint distance-rd 0 ; To colour the patches by distance to road.

end

to colour-by-n-visits

gis:paint n-visits 0

end

; Profiling

;

;

;

;

;

;

;

;

to profile

profiler:reset ;; clear the data

setup ;; set up the model

profiler:start ;; start profiling

repeat profile-tick-number [ go ] ;; run something you want to measure

profiler:stop ;; stop profiling

print profiler:report ;; view the results

profiler:reset ;; clear the data

end