Appendix 1. Modelling lynx density and population size in a spatial explicit capture recapture framework in Southern Carpathians, Romania

25 Sep 2020

Winter session

Description

We here analyze data from field season Winter 2018-2019, lasting between 17.12.2018 - 31.03.2019, hereafter called the “winter session”. The winter session had 59 traps syncronized but for modeling purposes we removed 6 traps from the initial trap array, namely the traps from a game management unit, where we didn’t obtained reliable pictures for lynx identification. Modeling scenario includes all identified, sexed and unsexed, independent individuals (hereafter independent animals). Juveniles still dependent on their mothers (i.e., from family groups) were removed from this analysis. However, when a juvenile from an identified family group was captured, we assigned a capture of his mother in the encounter dataset. Initially we identified 23 independent animals in this session but removed two. Animals L030 and R031 are most probably one and the same individual. None of the trap stations that captured them delivered images from both sides and were removed from further statistical modelling. All predictions are independent of animal’s sex.

Libraries

```
library(secr)
library(rgdal)
library(sp)
library(raster)
library(akima)
library(Hmisc)
library(corrplot)
```

Get and organize data.

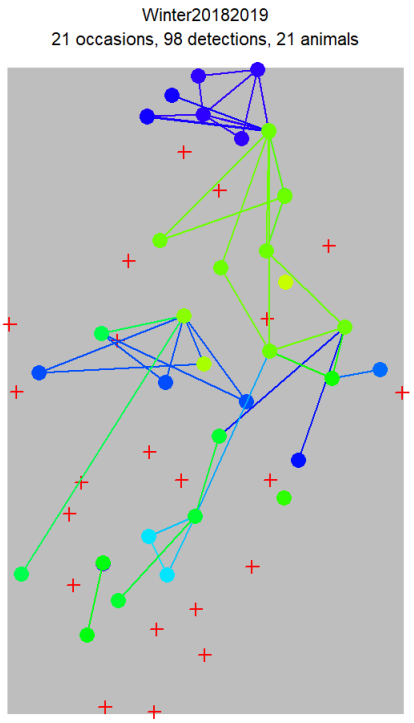
```
setwd('D://secr_s2_Stoenesti_excluded')
#bring data to R and create the capthist object (capt = encounter data file, trap = trap data file)
lynxRO <- read.capthist("lynxcapt.txt", "lynxtrap.txt", detector = "proximity")
```

```
## Session Winter20182019
## More than one detection per detector per occasion at binary detector(s)
```

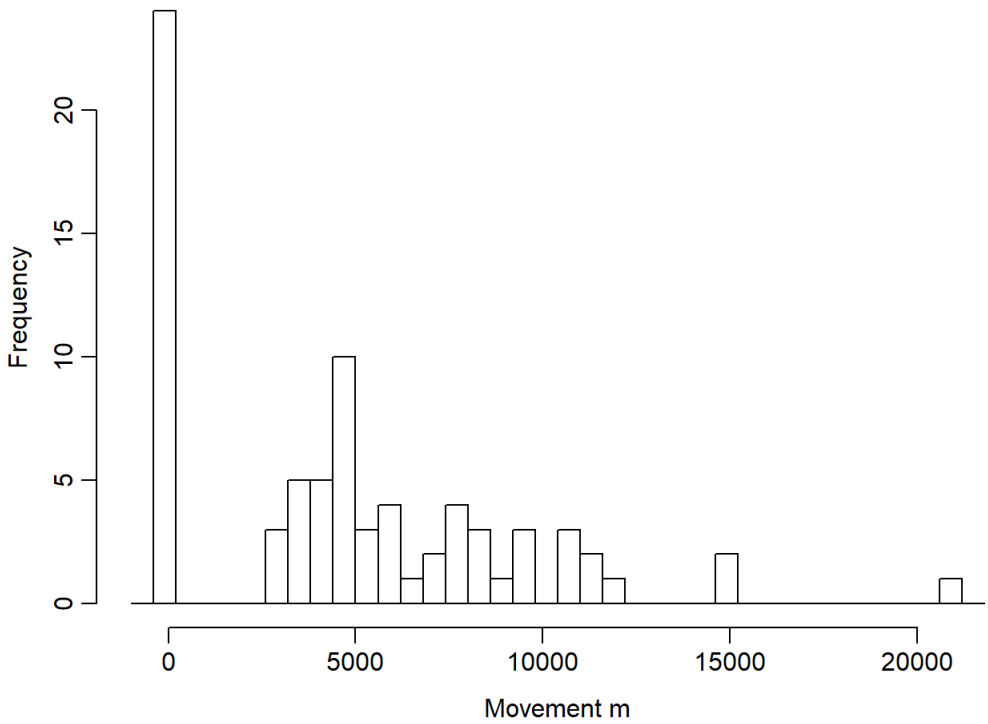
```
#detector = proximity is the correct one for camera trapping. Proximity detectors can be considered to act
#independently of each  other, with a particular individual having equal chances to visit multiple traps.
summary(lynxRO)
```

```
## Object class       capthist
## Detector type      proximity
## Detector number    53
## Average spacing    2738.007 m
## x-range            498199 525245 m
## y-range            425671 468764 m
##
##  Usage range by occasion
##     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
## min 0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## max 1 1 1 1 1 1 1 1 1  1  1  1  1  1  1  1  1  1  1  1  1
##
## Counts by occasion
##                     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## n                   1  2  2  3  3  0  3  2  8  6  5  4  4  4  3  6  5  4  1  3
## u                   1  2  1  3  2  0  2  1  3  0  1  0  1  0  0  1  3  0  0  0
## f                   6  5  2  4  0  0  2  0  1  0  1  0  0  0  0  0  0  0  0  0
## M(t+1)              1  3  4  7  9  9 11 12 15 15 16 16 17 17 17 18 21 21 21 21
## losses              0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## detections          1  2  2  3  4  0  3  3 11  7  5  4  7 10  5 11  6  5  1  5
## detectors visited   1  2  2  3  4  0  3  3  7  3  5  3  6  8  4  8  6  5  1  5
## detectors used     52 52 52 52 53 52 51 51 50 50 52 51 51 51 52 52 50 50 50 50
##                    21 Total
## n                   3    72
## u                   0    21
## f                   0    21
## M(t+1)             21    21
## losses              0     0
## detections          3    98
## detectors visited   3    82
## detectors used     49  1073
##
## Individual covariates
##  V5
##  F: 6
##  M:10
##  U: 5
```

```
#plot the trap array and recaptures of individuals
par(mar = c(1,1,3,1)) # reduce margins of the plot
plot (lynxRO, tracks = TRUE)
```



```
#plot successive trap-revealed movements
m <- unlist(moves(lynxRO))
par(mar = c(3.2,4,1,1), mgp = c(2.1,0.6,0)) # reduce margins
hist(m, breaks = seq(-2000/2, 22000,600), xlab = "Movement m", main = "")
```



We have 5 individuals with unidentified sex, 6F and 10M. In total 21 animals, 98 detections over 21 occasions. Average spacing between traps is 2738.007 m. The trap-revealed movements show two recaptures at 15-20km apart, possible outliers. Also note the high frequency of 0m movements in this session (an animal recaptured at a single trap).

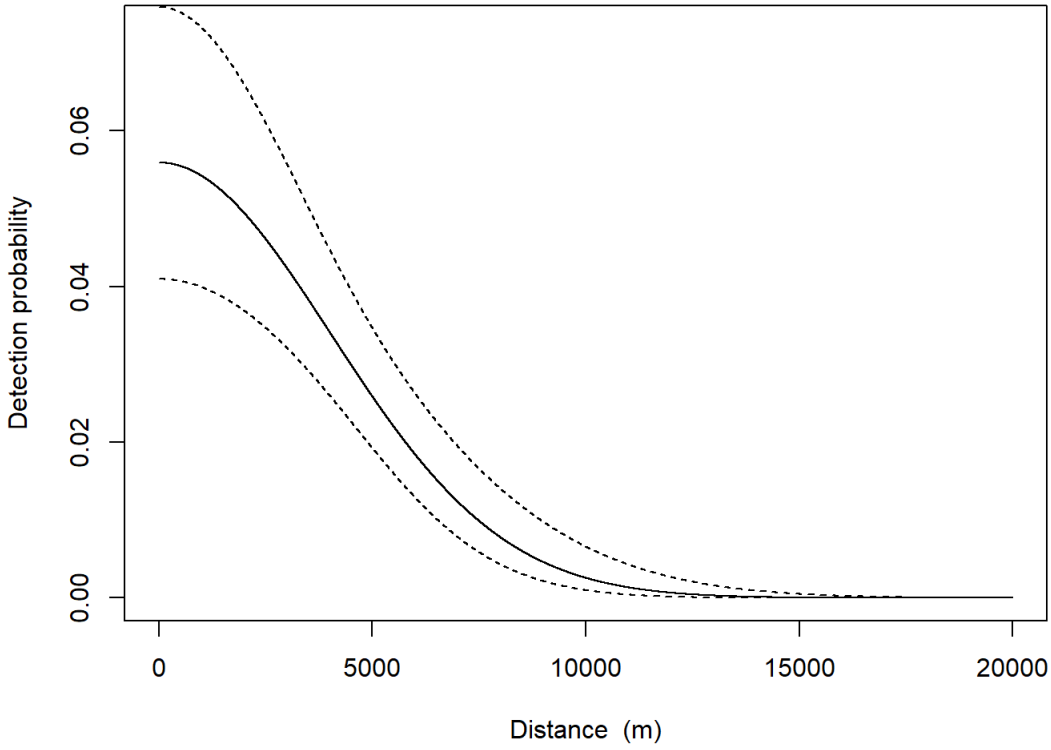
Initializing sigma.

```
#
initialsigma <- RPSV(lynxRO, CC = TRUE)
cat("Quick and biased estimate of sigma =", initialsigma, "m\n")
```

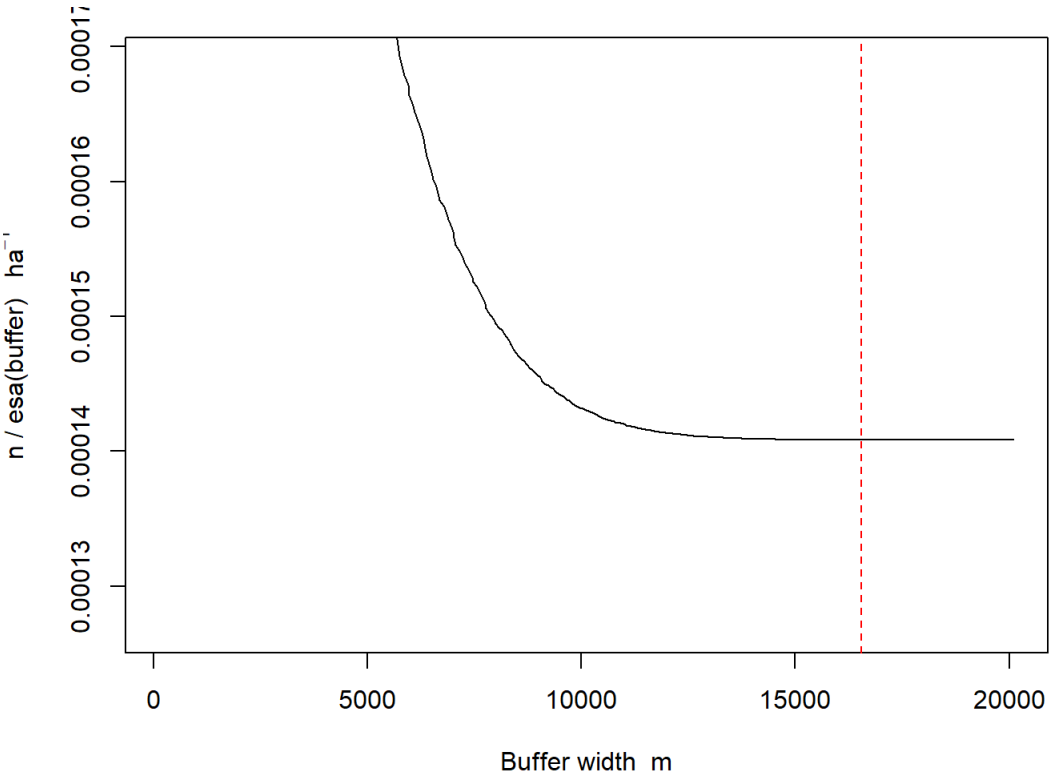
```
## Quick and biased estimate of sigma = 3310.661 m
```

Fit a simple SECR model and choose the buffer width around our trap array.

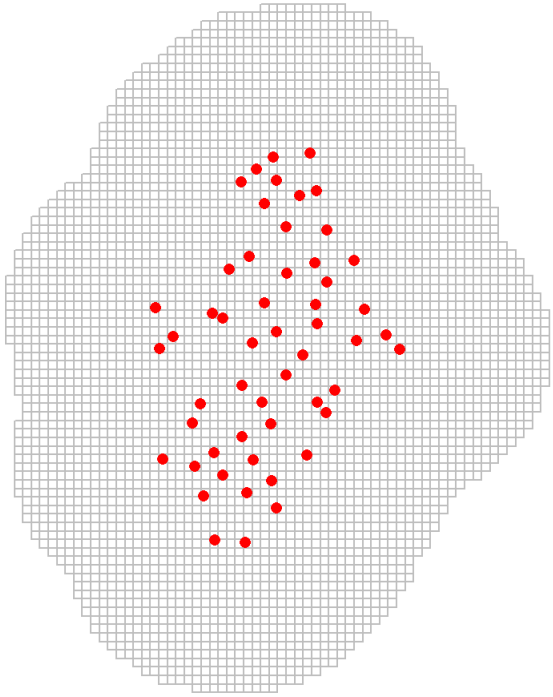
```
fit <- secr.fit (lynxRO, buffer = 5 * initialsigma, trace = FALSE, verify = FALSE)

par(mar = c(4,4,1,1)) # reduce margins
plot(fit, limits = TRUE, xval = 0:20000)
```




```
esa.plot(fit)
abline(v = 5 * initialsigma, lty = 2, col = 'red')
```




```
#let's plot the buffer region to see how it looks for our study area. It makes sense to use a habitat/non-ha
bitat mask instead of the buffer alone. Rationale: the buffer will include cells with human-dominated landsc
ape with no lynx habitat.
par(mar = c(1,1,1,1))
plot(fit$mask, dots = FALSE, mesh = "grey", col = "white")
plot(traps(lynxRO), detpar = list(pch = 16, cex = 1), add = TRUE)
```



We explored different buffer widths from 3 to 6 x sigma but we decided to stick to 5 x sigma for the buffer width. The probability of capturing a lynx coming from outside of this buffer is 0 as the function converged nicely after this distance.

Detection functions

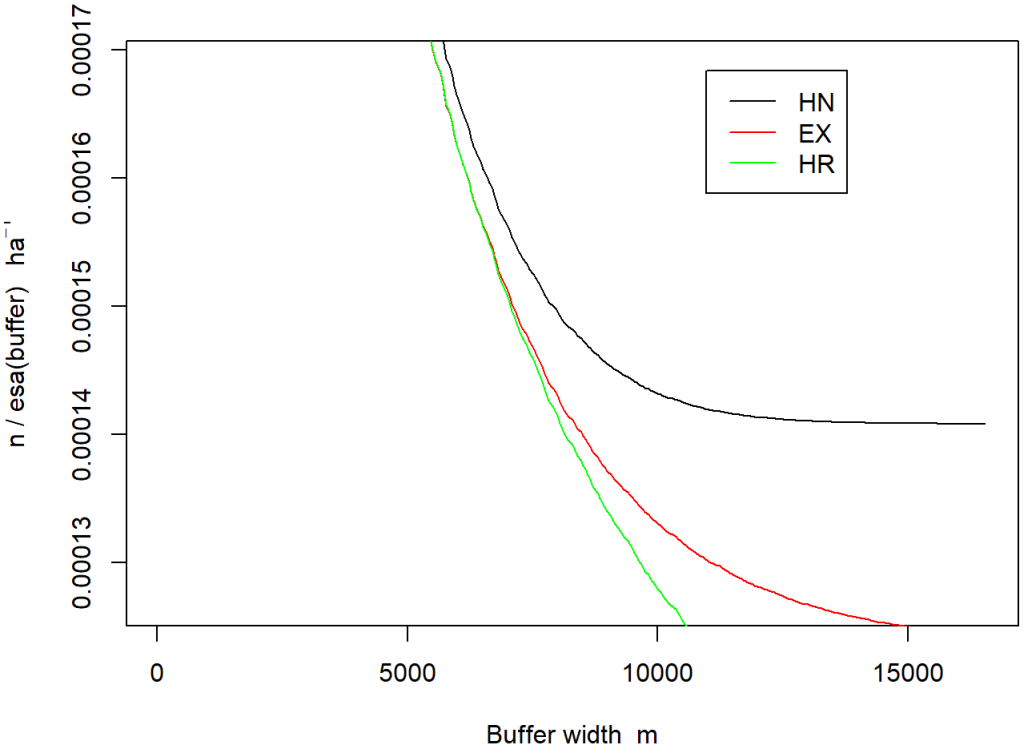
```
#choosing a detection function
fit.HN <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN', trace = FALSE, verify = FALSE)
fit.EX <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'EX', trace = FALSE, verify = FALSE)
fit.HR <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HR', trace = FALSE, verify = FALSE)

#compare the resutled models
fits <- secrlist(HN = fit.HN, EX = fit.EX, HR = fit.HR)
predict(fits)
```

```
## $HN
##         link     estimate  SE.estimate          lcl          ucl
## D        log 1.408280e-04 3.287801e-05 8.965955e-05 2.211983e-04
## g0     logit 5.589790e-02 8.799101e-03 4.095360e-02 7.586413e-02
## sigma    log 4.029075e+03 3.518981e+02 3.396276e+03 4.779779e+03
##
## $EX
##         link     estimate  SE.estimate          lcl          ucl
## D        log 1.242547e-04 2.988991e-05 7.805952e-05 1.977878e-04
## g0     logit 1.705050e-01 3.579567e-02 1.112432e-01 2.523723e-01
## sigma    log 2.644272e+03 2.958952e+02 2.124969e+03 3.290484e+03
##
## $HR
##         link     estimate  SE.estimate          lcl          ucl
## D        log 1.058115e-04 2.585632e-05 6.599940e-05 1.696390e-04
## g0     logit 1.874970e-01 8.949888e-02 6.800096e-02 4.219184e-01
## sigma    log 2.206004e+03 6.719296e+02 1.230458e+03 3.954995e+03
## z        log 2.676644e+00 3.024734e-01 2.146376e+00 3.337916e+00
```

```
AIC(fits)
```

```
##                 model    detectfn npar    logLik     AIC    AICc  dAICc
## HR D~1 g0~1 sigma~1 z~1 hazard rate    4 -224.0537 456.107 458.607  0.000
## EX     D~1 g0~1 sigma~1 exponential    3 -229.5351 465.070 466.482  7.875
## HN     D~1 g0~1 sigma~1  halfnormal    3 -237.7844 481.569 482.981 24.374
##    AICcwt
## HR 0.9809
## EX 0.0191
## HN 0.0000
```

```
par(mar = c(4,4,2,2))
esa.plot(fits, max.buffer = 5 * initialsigma)
```



HalfNormal function

converged first on the plot. Thus, hereafter use the argument [detectfn = 'HN'] in secr.fit.

Detection models.

```
#automatically generated predictors for secr models: model detection parameters
fit.HN <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN', trace = FALSE, verify = FALSE)
fit.HNb <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
                     model = g0 ~ b, trace = FALSE, verify = FALSE)   #permanent global learned response
fit.HNbk <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
                     model = g0 ~ bk, trace = FALSE, verify = FALSE)   #the site specific learned response
#fit.HNt <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
#                       model = g0 ~ t, trace = FALSE, verify = FALSE)  #time consuming  #time factor (one l
evel for each occasion) - no need to run this again as all test we made returned model = g0 ~ bk as performi
ng the best.
#fit.HNT <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
#                      model = g0 ~ T, trace = FALSE, verify = FALSE)  #time consuming  #time trend (integer
covariate 0:(S-1)) - no need to run this again as all test we made returned model = g0 ~ bk as performing th
e best.

#check which detection model works best
fitsb <- secrlist(null = fit$HN, b = fit.HNb, bk = fit.HNbk)
AIC(fitsb)
```

```
##                model    detectfn npar    logLik     AIC    AICc  dAICc AICcwt
## bk D~1 g0~bk sigma~1 halfnormal    4 -434.4800 876.960 879.460  0.000      1
## b    D~1 g0~b sigma~1 halfnormal    4 -450.3595 908.719 911.219 31.759      0
```

```
predict(fitsb)
```

```
## $b
##        link     estimate  SE.estimate          lcl          ucl
## D       log 2.260734e-04 8.298719e-05 1.126187e-04 4.538248e-04
## g0    logit 1.388154e-02 7.095835e-03 5.070686e-03 3.742629e-02
## sigma   log 4.168013e+03 3.547638e+02 3.528654e+03 4.923218e+03
##
## $bk
##        link     estimate  SE.estimate          lcl          ucl
## D       log 1.422779e-04 3.614072e-05 8.715466e-05 2.322654e-04
## g0    logit 2.204220e-02 4.992867e-03 1.411261e-02 3.427238e-02
## sigma   log 5.152122e+03 6.343164e+02 4.051186e+03 6.552244e+03
```

```
coef(fit.HNbk)
```

```
##                beta   SE.beta       lcl       ucl
## D         -8.857728 0.2500547 -9.347826 -8.367630
## g0        -3.792508 0.2316194 -4.246473 -3.338542
## g0.bkTRUE  1.998651 0.2730972  1.463391  2.533912
## sigma      8.547164 0.1226548  8.306765  8.787563
```

```
#model averaging
model.average(fitsb)
```

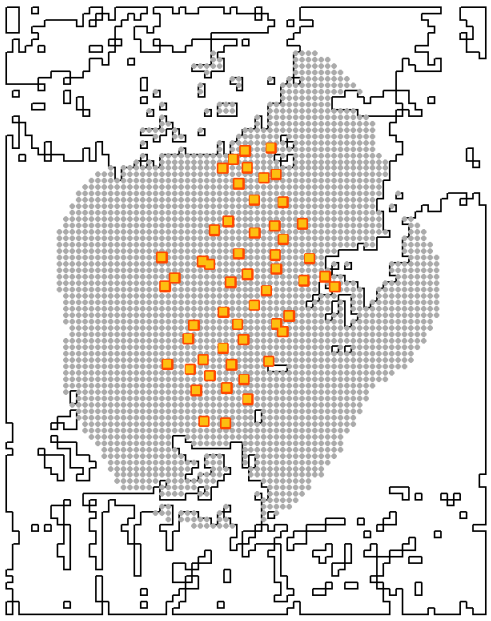
```
##           estimate  SE.estimate          lcl          ucl
## D     1.422779e-04 3.614072e-05 8.715466e-05 2.322654e-04
## g0    2.204220e-02 4.992867e-03 1.411261e-02 3.427238e-02
## sigma 5.152122e+03 6.343164e+02 4.051186e+03 6.552244e+03
```

It seems that [model = g0 ~ bk] works best. This model imply that an individual become trap happy or trap shy in relation to a particular detector. The learned response is positive g0.bTRUE = 1.866389, but we believe the animals didn't became trap happy after camera instalation, but pretty sure this is rather an artefact of the lynx using the same movement paths and become more and more detectable at the same trap. With more than 60% of the traps detecting lynx we can agree we selected the right spots for camera instalation.
Next model is mode = g0 ~ b but the estimate value is unrealistic. Thus use argument [model = g0 ~ bk] hereafter (same as in Rovero & Zimmerman 2016, on the same species and in similar habitat conditions).

Add habitat/non-habitat mask in the story.

We used a mask because the study area, especially after includding 5 x sigma buffer, include non-habitat features as industrial, urban, and agricultural landscape (around Zarnesti and Campulung cities). This is a habitat/non habitat mask of a wider area and it will allow us buffering around traps as mush as we want. We used the following rule for defining non-habitat: we removed cells with human dominated landscape in proportions >0.7 AND forest cover in proportions <0.1. The rest was considered potentially suitable. We considered this rule accounts for the habitat selection requirements of the species during all seasons and during both day and night as documented by Filla et al., (2017).

```
datadir <- system.file("secr/area_extent.shp", package = "secr")
lynxarea_70_10 <- shapefile("D://secr_s2_Stoenesti_excluded/mask_70_10.shp")

#finally, use the 5*sigma buffer width we concluded on earlier
lynxmask_70_10 <- make.mask(traps(lynxRO), spacing = 1000, buffer = 16553.305, type = "trapbuffer", poly = l
ynxarea_70_10)
#We explored mask spacing between 1000 and 5000 but we couldn't find any visible influence on the prediction
s.

#plot the mask
par(mar = c(3,6,6,6), xpd = TRUE)
plot (lynxmask_70_10, col = 'gray67', cex = 0.5, ppoly = TRUE)
#plot(traps(lynxRO), add = T, cex = 4)
plot(traps(lynxRO), detpar = list(pch = 0, cex = 0.8, col="#ff4800ff", lwd = 1.5), add = TRUE)
plot(traps(lynxRO), detpar = list(pch = 15, cex = 0.7, col="#fdbf10ff"), add = TRUE)
```



```
par(mar = c(5,4,4,2) + 0.1, xpd = FALSE)
```

Use argument [mask = lynxmask_70_10] hereafter.

Adjust for varying effort

```
#adjust for varying effort
#summary(lynxRO)   #see that Usage is binary, and this is wrong. Add data again with 'binary.usage = FALSE'
lynxRO <- read.capthist("lynxcapt.txt", "lynxtrap.txt", detector = "proximity", binary.usage = FALSE)
```

```
## Session Winter20182019
## More than one detection per detector per occasion at binary detector(s)
```

```
summary(traps(lynxRO))   #perfect, now the Usage ranges between 0 and 5 (trap days per occassion)
```

```
## Object class       traps
## Detector type      proximity
## Detector number    53
## Average spacing    2738.007 m
## x-range            498199 525245 m
## y-range            425671 468764 m
##
##  Usage range by occasion
##      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
## min 0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0
## max 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5  5  5  5  5  5  5
```

```
#run the models with adjusted varying effort and within the lynxmask_70_10
fit.usage.nomask <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN', model = g0 ~ bk, trace = F
ALSE, verify = FALSE) #implicitly adjusts for effort, no mask
fit.usage.mask <- secr.fit (lynxRO, mask = lynxmask_70_10, detectfn = 'HN', model = g0 ~ bk, trace = FALSE,
verify = FALSE) #implicitly adjusts for effort with mask
#fit.t.mask <- secr.fit(lynxRO, model = g0 ~ t, mask = lynxmask_70_10, detectfn = 'HN', trace = FALSE, verif
y = FALSE) #ignores effort, but allows for occasion-to-occasion variation by fitting a separate g0 each time
, with mask. No need to run this, it is time consuming and has a poor AIC value.
usage(traps(lynxRO)) <- NULL #we whipe the usage information
fit.nousage.mask <- secr.fit(lynxRO, mask = lynxmask_70_10, detectfn = 'HN', model = g0 ~ bk, trace = FALSE,
verify = FALSE) #no adjustment, with mask

AIC(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)
```

```
##                              model   detectfn npar    logLik     AIC    AICc
## fit.usage.mask   D~1 g0~bk sigma~1 halfnormal    4 -433.3299 874.660 877.160
## fit.usage.nomask D~1 g0~bk sigma~1 halfnormal    4 -433.6227 875.245 877.745
## fit.nousage.mask D~1 g0~bk sigma~1 halfnormal    4 -436.6242 881.248 883.748
##                  dAICc AICcwt
## fit.usage.mask   0.000 0.5607
## fit.usage.nomask 0.585 0.4185
## fit.nousage.mask 6.588 0.0208
```

```
collate(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)[1,,,]
```

```
## , , D
##
##                       estimate  SE.estimate          lcl          ucl
## fit.usage.nomask 0.0001426996 3.622553e-05 8.743848e-05 0.0002328857
## fit.usage.mask   0.0001601425 3.921277e-05 9.979450e-05 0.0002569843
## fit.nousage.mask 0.0001598188 3.911643e-05 9.961278e-05 0.0002564135
##
## , , g0
##
##                      estimate SE.estimate         lcl         ucl
## fit.usage.nomask 0.004510466 0.001028415 0.002883843 0.007048095
## fit.usage.mask   0.004516152 0.001028213 0.002889359 0.007052399
## fit.nousage.mask 0.021317004 0.004843083 0.013630882 0.033191303
##
## , , sigma
##
##                  estimate SE.estimate      lcl      ucl
## fit.usage.nomask 5132.669    633.1502 4034.018 6530.533
## fit.usage.mask   5026.354    589.8135 3996.785 6321.139
## fit.nousage.mask 5037.666    587.8417 4010.873 6327.321
```

```
collate(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)[,,'g0']
```

```
##                      estimate SE.estimate         lcl         ucl
## fit.usage.nomask 0.004510466 0.001028415 0.002883843 0.007048095
## fit.usage.mask   0.004516152 0.001028213 0.002889359 0.007052399
## fit.nousage.mask 0.021317004 0.004843083 0.013630882 0.033191303
```

```
#Density of independent lynx for the studied population
collate(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)[,,'D']
```

```
##                       estimate  SE.estimate          lcl          ucl
## fit.usage.nomask 0.0001426996 3.622553e-05 8.743848e-05 0.0002328857
## fit.usage.mask   0.0001601425 3.921277e-05 9.979450e-05 0.0002569843
## fit.nousage.mask 0.0001598188 3.911643e-05 9.961278e-05 0.0002564135
```

```
#Regional population size for the buffered region.
region.N(fit.usage.nomask)
```

```
##     estimate SE.estimate      lcl      ucl  n
## E.N 47.30971   12.009983 28.98880 77.20943 21
## R.N 47.30980    9.845303 33.94105 74.48914 21
```

```
region.N(fit.usage.mask)
```

```
##     estimate SE.estimate      lcl      ucl  n
## E.N 44.23136   10.830567 27.56324 70.97908 21
## R.N 44.23147    8.548088 32.55480 67.70795 21
```

```
#region.N(fit.t.mask)
region.N(fit.nousage.mask)
```

```
##     estimate SE.estimate      lcl      ucl  n
## E.N 44.14196   10.803957 27.51305 70.82141 21
## R.N 44.14205    8.519596 32.50642 67.54398 21
```

We interpret population size N and density D from model fit.usage.mask (mask included), it has the best AIC and is comparable with the last session - Autumn-early winter. D values has to be multiplied by 10000 to transform density per 1 hectare to density per 100 sqkm.

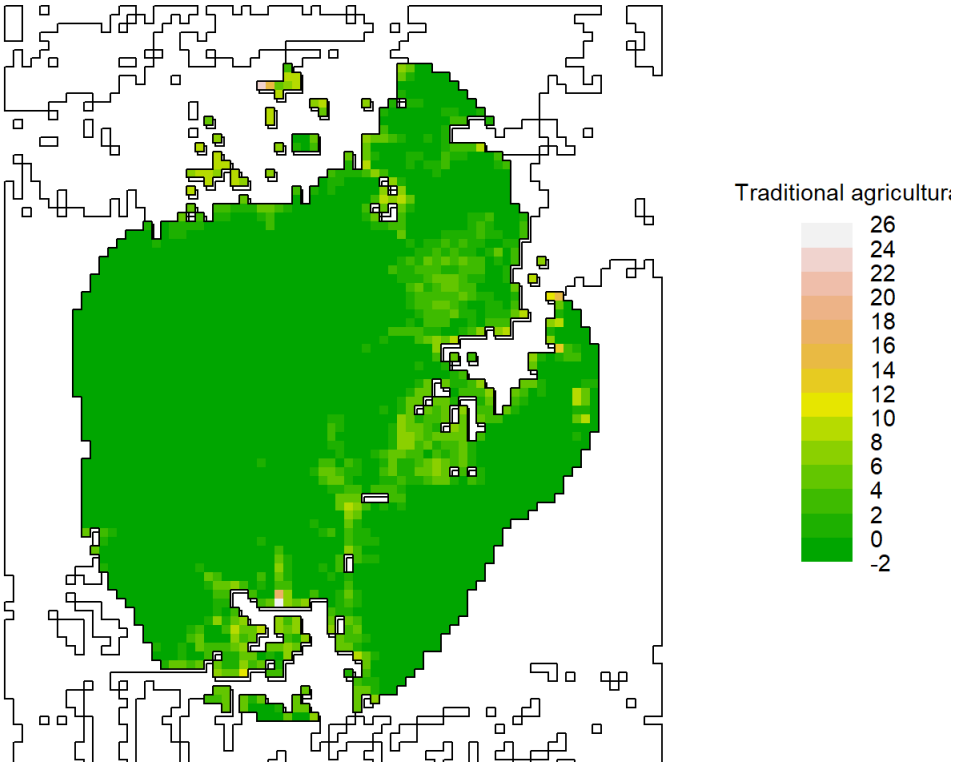
Data prep continued - add covariates to the mask points.

Mask points are generated at a 1km spacing within the habitat mask. We added habitat covariates extracted from open sources like Corine Land Cover 2018 and topography data.

```
#add scaled covariates to the mask points.
lynxarea_70_10 <- shapefile("D://secr_s2_Stoenesti_excluded/mask_70_10.shp")
#return to spacing = 1000
lynxmask_70_10 <- make.mask(traps(lynxRO), spacing = 1000, buffer = 16553.305, type = "trapbuffer", poly = l
ynxarea_70_10)
lynxcov <- rgdal::readOGR(dsn = "grid_covariates_lynx_5.shp", layer = "grid_covariates_lynx_5")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "D:\secr_s2_Stoenesti_excluded\grid_covariates_lynx_5.shp", layer: "grid_covariates_lynx_5"
## with 5542 features
## It has 41 fields
## Integer64 fields read as strings:  CLC Reclass2
```

```
lynxmask <- addCovariates (lynxmask_70_10, lynxcov, columns = c("Alt", "Slo", "TRI5", "TRI9", "PublicRoad",
"Forest", "HumDomin", "OpenHab", "CLC_511", "CLC_222", "CLC_112", "CLC_211", "CLC_242", "CLC_311", "CLC_231"
, "CLC_243", "CLC_321", "CLC_331", "CLC_313", "CLC_324", "CLC_131", "CLC_221", "CLC_312", "CLC_142", "CLC_33
3", "CLC_411", "CLC_322", "CLC_141", "CLC_512", "CLC_332", "CLC_121", "CLC_111", "CLC_132", "CLC", "Reclass"
, "Reclass2", "OpenHaFin", "TradiAgri"),  strict = TRUE, replace = TRUE)
#summary(lynxmask)

#repair missing values
repair <- function (mask, covariate, ...) {
  NAcov <- is.na(covariates(mask)[,covariate])
  OK <- subset(mask, !NAcov)
  require(akima)
  irect <- akima::interp (x = OK$x, y = OK$y, z = covariates(OK)[,covariate],...)
  irectxy <- expand.grid(x = irect$x, y = irect$y)
  i <- nearesttrap(mask[NAcov,], irectxy)
  covariates(mask)[,covariate][NAcov] <- irect[[3]][i]
  mask
}

repaired <- repair(lynxmask, 'TradiAgri')

#interpolate covariates
copynearest <- function (mask, covariate) {
  NAcov <- is.na(covariates(mask)[,covariate])
  OK <- subset(mask, !NAcov)
  i <- nearesttrap(mask, OK)
  covariates(mask)[,covariate][NAcov] <- covariates(OK)[i[NAcov],covariate]
  mask
}

completed <- copynearest(repaired, 'TradiAgri')

#plotting mask with the covariates and explore patterns (just an example here).
par(mar=c(1,1,2,6), xpd=TRUE)
plot(lynxmask, covariate = 'TradiAgri', dots = FALSE, border = 100,
     title = 'Traditional agricultural', polycol = 'blue')
plotMaskEdge(lynxmask, add = TRUE)
```



Model D_surface

```
#We fit simple models with covariates to our data
lynx.0 <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ 1, trace = FALSE, verify = FALSE)
lynx.D_alt <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Alt, method = "Nelder-Mead", trace =
FALSE, verify = FALSE)
lynx.D_tri <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ TRI9, method = "Nelder-Mead", trace =
FALSE, verify = FALSE)
lynx.D_slo <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Slo, method = "Nelder-Mead", trace =
FALSE, verify = FALSE)
lynx.D_landcover <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Reclass, method = "Nelder-Mead"
, trace = FALSE, verify = FALSE)
lynx.D_forest <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Forest, method = "Nelder-Mead", tr
ace = FALSE, verify = FALSE)
lynx.D_open <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ OpenHaFin, method = "Nelder-Mead", t
race = FALSE, verify = FALSE)
lynx.D_agri <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ TradiAgri, method = "Nelder-Mead", t
race = FALSE, verify = FALSE)
lynx.D_roads <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ PublicRoad, method = "Nelder-Mead",
trace = FALSE, verify = FALSE)
lynx.D_decid <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ CLC_311, method = "Nelder-Mead", tr
ace = FALSE, verify = FALSE)
lynx.D_conif <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ CLC_312, method = "Nelder-Mead", tr
ace = FALSE, verify = FALSE)
lynx.D_mixt <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ CLC_313, method = "Nelder-Mead", tra
ce = FALSE, verify = FALSE)

AIC(lynx.0, lynx.D_alt, lynx.D_tri, lynx.D_slo, lynx.D_landcover, lynx.D_forest, lynx.D_open, lynx.D_agri, l
ynx.D_roads, lynx.D_decid, lynx.D_conif, lynx.D_mixt, criterion = "AIC")[,-2]
```

```
##                                    model npar    logLik     AIC    AICc  dAIC
## lynx.D_roads     D~PublicRoad g0~1 sigma~1    4 -237.1413 482.283 484.783 0.000
## lynx.D_mixt         D~CLC_313 g0~1 sigma~1    4 -238.2221 484.444 486.944 2.161
## lynx.0                  D~1 g0~1 sigma~1    3 -239.5805 485.161 486.573 2.878
## lynx.D_agri       D~TradiAgri g0~1 sigma~1    4 -238.7751 485.550 488.050 3.267
## lynx.D_forest        D~Forest g0~1 sigma~1    4 -238.8551 485.710 488.210 3.427
## lynx.D_conif        D~CLC_312 g0~1 sigma~1    4 -239.1395 486.279 488.779 3.996
## lynx.D_open       D~OpenHaFin g0~1 sigma~1    4 -239.1674 486.335 488.835 4.052
## lynx.D_tri             D~TRI9 g0~1 sigma~1    4 -239.2727 486.545 489.045 4.262
## lynx.D_slo              D~Slo g0~1 sigma~1    4 -239.3001 486.600 489.100 4.317
## lynx.D_decid        D~CLC_311 g0~1 sigma~1    4 -239.4611 486.922 489.422 4.639
## lynx.D_alt              D~Alt g0~1 sigma~1    4 -239.5555 487.111 489.611 4.828
## lynx.D_landcover    D~Reclass g0~1 sigma~1    7 -236.9020 487.804 496.419 5.521
##                    AICwt
## lynx.D_roads      0.3697
## lynx.D_mixt       0.1255
## lynx.0            0.0877
## lynx.D_agri       0.0722
## lynx.D_forest     0.0666
## lynx.D_conif      0.0501
## lynx.D_open       0.0488
## lynx.D_tri        0.0439
## lynx.D_slo        0.0427
## lynx.D_decid      0.0364
## lynx.D_alt        0.0331
## lynx.D_landcover 0.0234
```

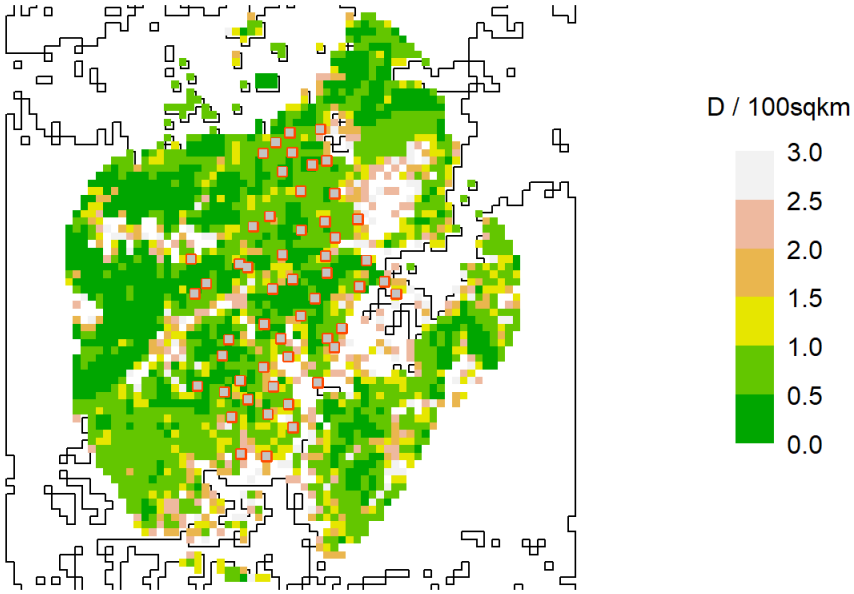
For the winter session, with one exception, none of the variables stands out as really important for D. All models have similar AIC, with a difference lower than 1 unit from a model to another and close to the null model (Table 6). The exception is [PublicRoad], but not necessarily suggesting this predictor explains D, being rather a bias we induced by placing the cameras close to unpaved forest roads (i.e., accessible areas in winter conditions, even though we respected the sampling design with a predefined trap array). Thus, we decided to leave out this predictor from further modelling. For the autumn-early winter session (next chapter), it is shown that some of the predictors explained D surface better, i.e., [TraditionalAgriculture], [Slo]. In the autumn, the [PublicRoad] no longer induced the bias it induced during the winter, suggesting lynx are now detected at traps further from roads too. Thus, the predictors we used for modelling D surface were model = D ~ Forest + TraditionalAgriculture + Slo + I(Slo^2) for both monitoring sessions, and which account for an AICwt of 0.17 for the winter and of 0.30 during the autumn-early winter. We used the syntax I(Slo^2) for accounting for a nonlinear relation with slope.

Map the D_surface prediction and its CIs

```
#final model for S2
lynx.D_s2 <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Forest + TradiAgri + Slo + I(Slo^2), m
ethod = "Nelder-Mead", trace = FALSE, verify = FALSE)
```

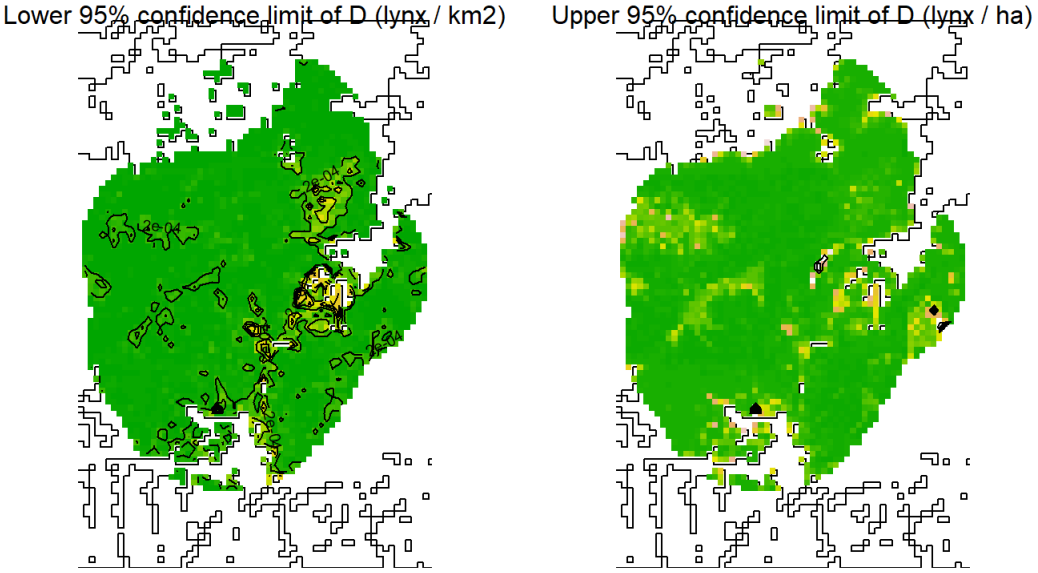
```
## Warning in secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Forest + :
## probable maximization error: optim returned convergence 1. See ?optim
```

```
#plot D surface
par(mar = c(3,6,3,8))
surface.D_s2 <- predictDsurface(lynx.D_s2)
plot(surface.D_s2, plottype = "shaded", poly = FALSE, breaks = seq(0,3,0.5),
     title = "D / 100sqkm", text.cex = 1, scale = 10000)
plot(traps(lynxRO), detpar = list(pch = 0, cex = 0.8, col="#ff4800ff", lwd = 1.5), add = TRUE)
plot(traps(lynxRO), detpar = list(pch = 15, cex = 0.7, col="gray77"), add = TRUE)
```




```
#map CIs
par(mar = c(3,3,3,3), mfrow = c(1,2), xpd = FALSE)
surface.D_s2 <- predictDsurface(lynx.D_s2, cl.D = TRUE)
plot(surface.D_s2, covariate= "lcl", breaks = seq(0,0.0005,0.00001), legend = FALSE)
mtext(side=3,line=-1.5,"Lower 95% confidence limit of D (lynx / km2)")
plot(surface.D_s2, plottype = "contour", breaks = seq(0,0005,0.00001), add = TRUE)
plot(surface.D_s2, covariate= "ucl", breaks = seq(0,0.0060,0.0001), legend = FALSE)
mtext(side=3,line=-1.5,"Upper 95% confidence limit of D (lynx / ha)")
plot(surface.D_s2, covariate= "ucl", plottype = "contour", breaks = seq(0,0.0060,0.0001),
     add = TRUE)
mtext(side=3, line=-1, outer=TRUE, "model = D ~ Forest + TradiAgri + Slo + I(Slo^2)")
```

model = D ~ Forest + TradiAgri + Slo + I(Slo^2)

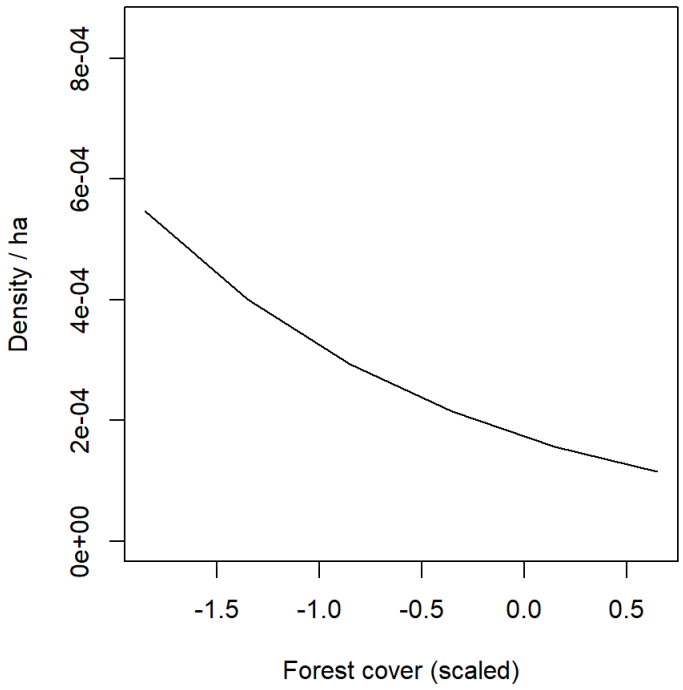


Couldn't find the right breaks

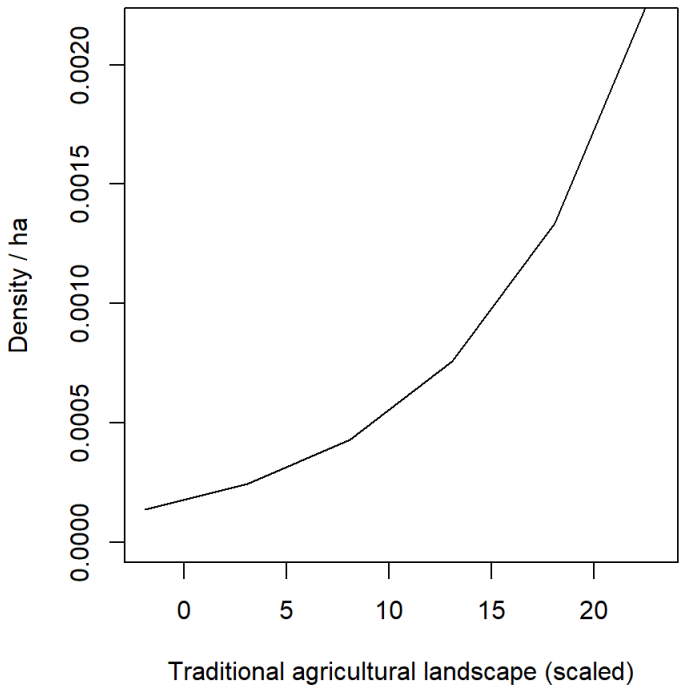
for CIs

Explore coefficients and plot density against covariates

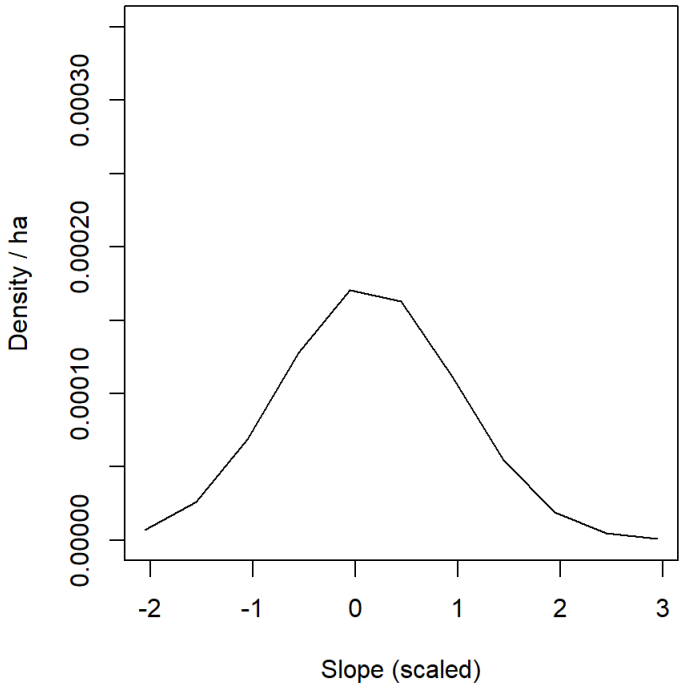
```
#summary(lynxmask)     #copy paste min and max values for each variables in the code below.
#lynx.D_s2
# Beta parameters (coefficients)
#                     beta     SE.beta        lcl        ucl
# D           -8.15249482 0.69444278 -9.5135777 -6.7914120
# D.Forest    -0.54315880 0.45487397 -1.4346954  0.3483778
# D.TradiAgri  0.06582525 0.16968708 -0.2667553  0.3984058
# D.Slo        0.45582034 1.48351813 -2.4518218  3.3634624
# D.I(Slo^2)  -1.72730648 2.33732040 -6.3083703  2.8537573
# g0          -2.83222408 0.16773674 -3.1609821 -2.5034661
# sigma        8.32921186 0.08946104  8.1538714  8.5045523

#plot density against covariates
tmp <- predict(lynx.D_s2, newdata = data.frame(Forest = seq(-1.85145,1.03262,0.5), Slo=0, TradiAgri=0))
par(mar=c(5,8,2,4), pty = "s")
plot(seq(-1.85145,1.03262,0.5), sapply(tmp, "[", "D","estimate"), ylim = c(0,0.00085),
     xlab = "Forest cover (scaled)", ylab = "Density / ha", type = "l")
```




```
tmp <- predict(lynx.D_s2, newdata = data.frame(TradiAgri = seq(-1.9190,24.7610,5), Slo=0, Forest=0))
par(mar=c(5,8,2,4), pty = "s")
plot(seq(-1.9190,24.7610,5), sapply(tmp, "[", "D","estimate"), ylim = c(0,0.00215),
     xlab = "Traditional agricultural landscape (scaled)", ylab = "Density / ha", type = "l")
```




```
tmp <- predict(lynx.D_s2, newdata = data.frame(Slo = seq(-2.0544,3.2576,0.5), TradiAgri=0, Forest=0))
par(mar=c(5,8,2,4), pty = "s")
plot(seq(-2.0544,3.2576,0.5), sapply(tmp, "[", "D","estimate"), ylim = c(0,0.00035),
     xlab = "Slope (scaled)", ylab = "Density / ha", type = "l")
```



Higher densities in places

where we have a mosaic of forest and human dominated landscape feature (at lower altitudes, and consequently at lower slopes where also the roe deer graze during the winter).

Export final prediction

```
#export as Raster Layer, format = tif
D_s2 <- raster(surface.D_s2, covariate = "D.0", crs = "+proj=sterea +lat_0=46 +lon_0=25 +k=0.99975 +x_0=500000 +y_0=500000 +ellps=krass +towgs84=28,-121,-77,0,0,0,0 +units=m +no_defs")
writeRaster(D_s2, 'D_s2_20200925.tif', format='GTiff',  overwrite=TRUE)
surface.D_s2 <- predictDsurface(lynx.D_s2, cl.D = TRUE)
D_s2_lcl <- raster(surface.D_s2, covariate= "lcl.0", crs = "+proj=sterea +lat_0=46 +lon_0=25 +k=0.99975 +x_0=500000 +y_0=500000 +ellps=krass +towgs84=28,-121,-77,0,0,0,0 +units=m +no_defs")
writeRaster(D_s2_lcl, 'D_s2_lcl_20200925.tif', format='GTiff',  overwrite=TRUE)
D_s2_ucl <- raster(surface.D_s2, covariate= "ucl.0", crs = "+proj=sterea +lat_0=46 +lon_0=25 +k=0.99975 +x_0=500000 +y_0=500000 +ellps=krass +towgs84=28,-121,-77,0,0,0,0 +units=m +no_defs")
writeRaster(D_s2_ucl, 'D_s2_ucl_20200925.tif', format='GTiff',  overwrite=TRUE)
```

Autumn-early winter session

Description

We here analyze data from the field season lasting between 09 October 2019 - 16 January 2020 and hereafter called “the autumn-early winter session”. The autumn-early winter session had 76 traps syncronized but for modeling purposes we removed 15 traps from the initial trap array, from the same game management unit where we didn’t obtained reliable pictures for lynx identification. Modeling scenario includes all identified, sexed and unsexed, independent individuals. Juveniles still dependent on their mothers (i.e., from family groups) were removed from this analysis. All predictions are thus independent of animal’s sex.

Libraries

Get and organize data.

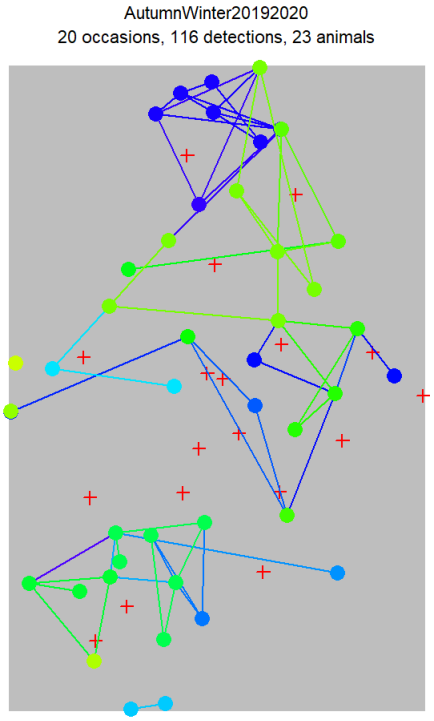
```
setwd('D://secr_s3_Stoenesti_excluded')
#bring data to R and create the capthist object (capt = encounter data file, trap = trap data file)
lynxRO <- read.capthist("lynxcapt.txt", "lynxtrap.txt", detector = "proximity")
```

```
## Session AutumnWinter20192020
## More than one detection per detector per occasion at binary detector(s)
## Detections at 'unused' detectors
##          cbind(ID, detector, occasion)[conflcts, ]
## ID                                            B21
## detector                                        3
## occasion                                       19
```

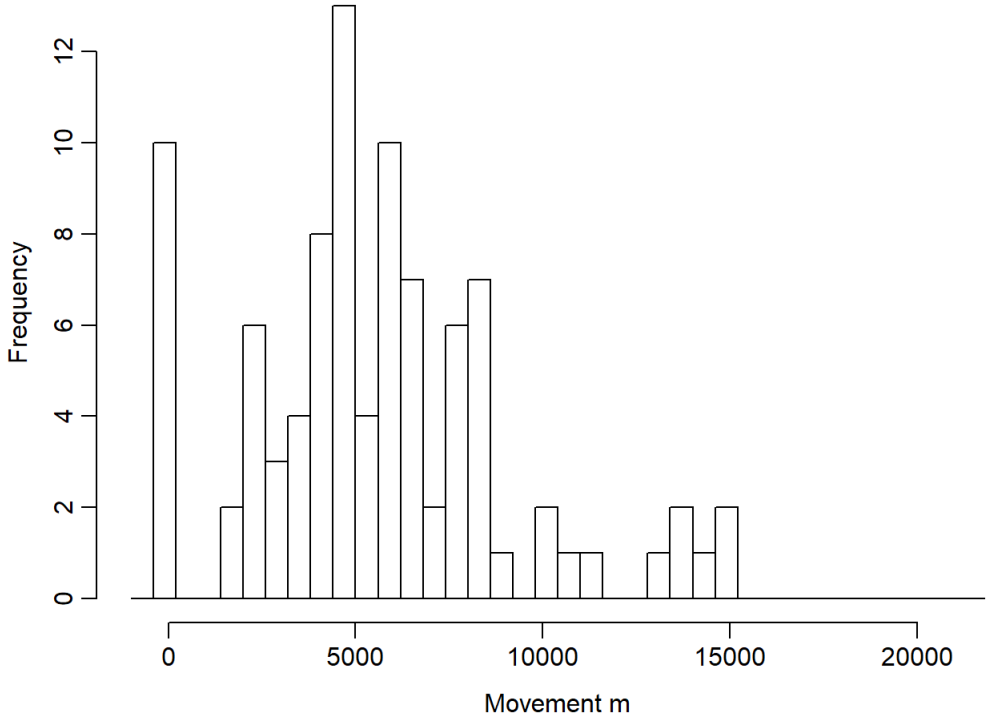
```
#detector = proximity is the correct one for camera trapping. Proximity detectors act independently of each other, with a particular individual having equal chances to visit multiple traps.
summary(lynxRO)
```

```
## Object class       capthist
## Detector type      proximity
## Detector number    60
## Average spacing    2470.473 m
## x-range            497925 525709 m
## y-range            426538 468669 m
##
##  Usage range by occasion
##     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## min 0 0 0 0 0 0 0 0 0  0  0  0  1  1  1  1  1  0  0  0
## max 1 1 1 1 1 1 1 1 1  1  1  1  1  1  1  1  1  1  1  1
##
## Counts by occasion
##                     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## n                   4  0  7  4  3  5  2  6  8  6  3  6  5  5  9  2  3  5  6  6
## u                   4  0  6  2  2  2  0  1  1  0  1  1  1  0  1  0  0  1  0  0
## f                   4  5  3  1  1  5  1  1  2  0  0  0  0  0  0  0  0  0  0  0
## M(t+1)              4  4 10 12 14 16 16 17 18 18 19 20 21 21 22 22 22 23 23 23
## losses              0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## detections          6  0  9  4  3  7  2  8 11  7  3  6  6  6 10  2  4  8  6  8
## detectors visited   5  0  8  4  3  6  2  7 11  7  3  6  6  6  9  2  4  8  6  7
## detectors used     54 56 56 56 55 55 55 58 58 58 58 59 60 60 60 60 60 59 57 56
##                    Total
## n                     95
## u                     23
## f                     23
## M(t+1)                23
## losses                 0
## detections           116
## detectors visited    110
## detectors used      1150
##
## Individual covariates
##  V5
##  F:7
##  M:8
##  U:8
```

```
#plot the trap array and recaptures of individuals
par(mar = c(1,1,3,1)) # reduce margins of the plot
plot (lynxRO, tracks = TRUE)
```




```
#plot successive trap-revealed movements
m <- unlist(moves(lynxRO))
par(mar = c(3.2,4,1,1), mgp = c(2.1,0.6,0)) # reduce margins
hist(m, breaks = seq(-2000/2, 22000,600), xlab = "Movement m", main = "")
```



We have 8 individuals with Sex unidentified, 7F and 8M = 23 unique animals. Average spacing between traps 2477.378 m The trap-revealed movements show a nice distribution symetrical aroung 5000m, not that many single detector recaptures (0m), and just a few long movements of 12-15000m. In general the plot looks much better than during the winter.

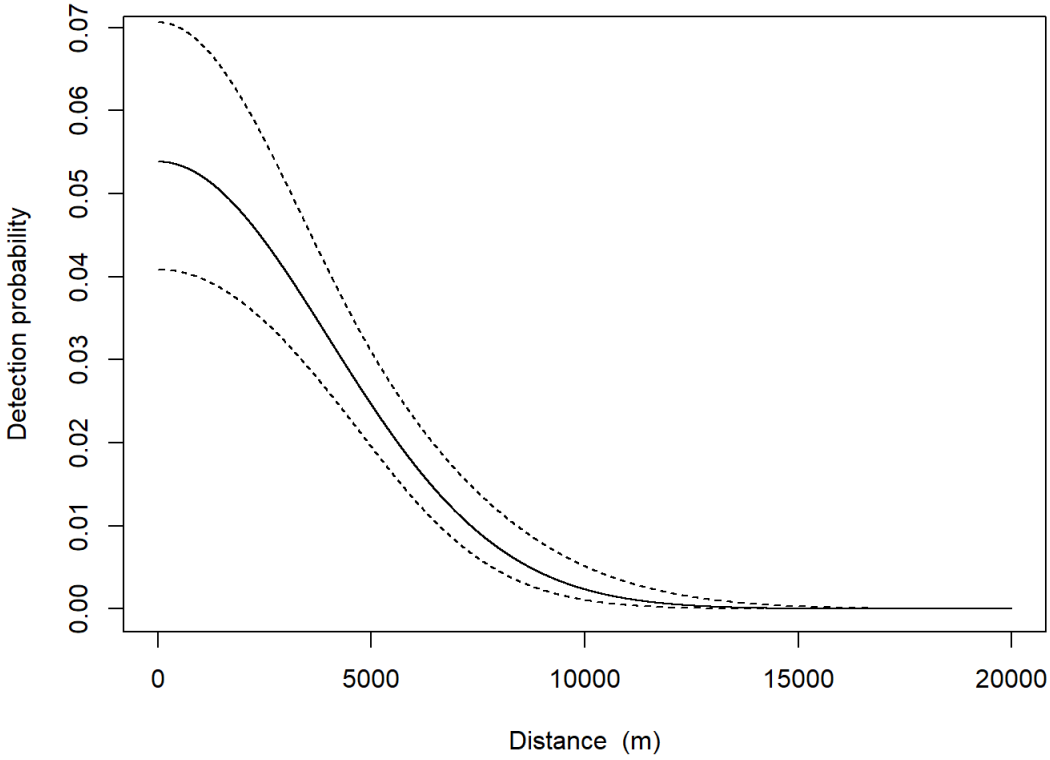
Initializing sigma.

```
#
initialsigma <- RPSV(lynxRO, CC = TRUE)
cat("Quick and biased estimate of sigma =", initialsigma, "m\n")
```

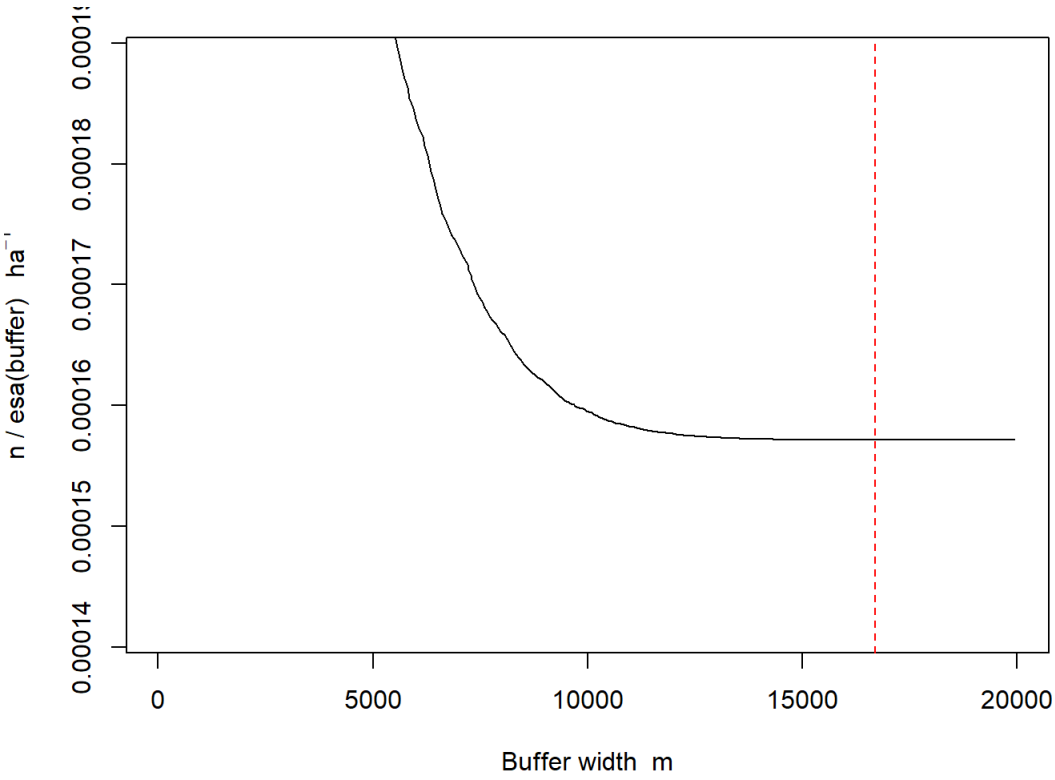
```
## Quick and biased estimate of sigma = 3342.148 m
```

Fit a simple SECR model and choose the buffer width around our trap array.

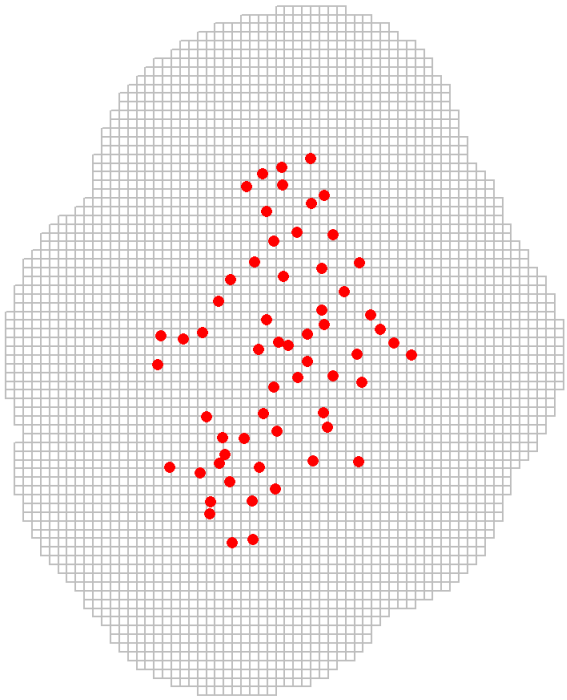
```
fit <- secr.fit (lynxRO, buffer = 5 * initialsigma, trace = FALSE, verify = FALSE)

par(mar = c(4,4,1,1)) # reduce margins
plot(fit, limits = TRUE, xval = 0:20000)
```




```
esa.plot(fit)
abline(v = 5 * initialsigma, lty = 2, col = 'red')
```




```
#let's plot the buffer region
par(mar = c(1,1,1,1))
plot(fit$mask, dots = FALSE, mesh = "grey", col = "white")
plot(traps(lynxRO), detpar = list(pch = 16, cex = 1), add = TRUE)
```



5 x sigma conveged nicely as during the winter session

Detection functions

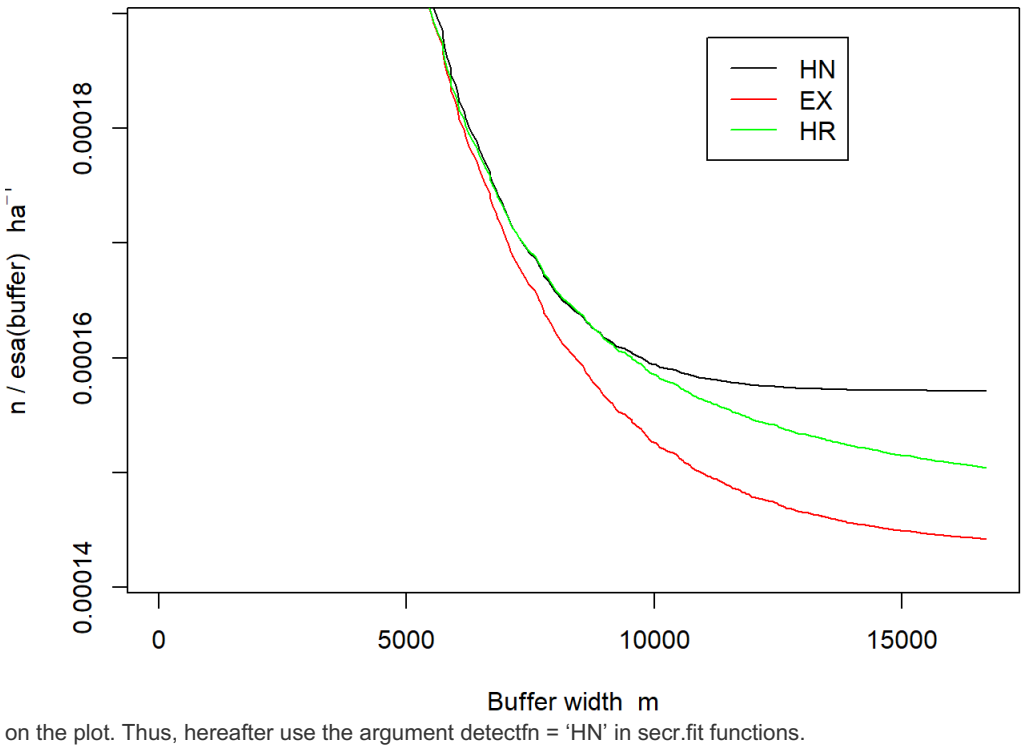
```
#choosing a detection function
fit.HN <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN', trace = FALSE, verify = FALSE)
fit.EX <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'EX', trace = FALSE, verify = FALSE)
fit.HR <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HR', trace = FALSE, verify = FALSE)

#compare the resutled models
fits <- secrlist(HN = fit.HN, EX = fit.EX, HR = fit.HR)
predict(fits)
```

```
## $HN
##         link     estimate  SE.estimate          lcl          ucl
## D        log 1.571398e-04 3.445344e-05 1.027646e-04 2.402861e-04
## g0     logit 5.385503e-02 7.520848e-03 4.087949e-02 7.064578e-02
## sigma    log 3.995769e+03 2.972486e+02 3.454348e+03 4.622050e+03
##
## $EX
##         link     estimate  SE.estimate          lcl          ucl
## D        log 1.441998e-04 3.215835e-05 9.363428e-05 2.220723e-04
## g0     logit 1.423783e-01 2.543910e-02 9.939146e-02 1.998317e-01
## sigma    log 2.610229e+03 2.468990e+02 2.169420e+03 3.140608e+03
##
## $HR
##         link     estimate  SE.estimate          lcl          ucl
## D        log 1.504174e-04 3.365540e-05 9.753611e-05 2.319693e-04
## g0     logit 5.004335e-02 9.305974e-03 3.464982e-02 7.176720e-02
## sigma    log 4.807474e+03 5.774808e+02 3.802204e+03 6.078530e+03
## z        log 4.471981e+00 6.611810e-01 3.352199e+00 5.965819e+00
```

```
AIC(fits)
```

```
##                   model    detectfn npar    logLik     AIC    AICc dAICc AICcwt
## HR D~1 g0~1 sigma~1 z~1 hazard rate    4 -261.2018 530.404 532.626 0.000 0.7647
## EX     D~1 g0~1 sigma~1 exponential    3 -263.9902 533.980 535.244 2.618 0.2066
## HN     D~1 g0~1 sigma~1  halfnormal    3 -265.9642 537.928 539.191 6.565 0.0287
```

```
par(mar = c(4,4,2,2))
esa.plot(fits, max.buffer = 5 * initialsigma)
```



HN is reaching the asymptote

on the plot. Thus, hereafter use the argument detectfn = 'HN' in secr.fit functions.

Detection models.

```
#automatically generated predictors for secr models: model detection parameters
fit.HN <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN', trace = FALSE, verify = FALSE)
fit.HNb <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
                     model = g0 ~ b, trace = FALSE, verify = FALSE)   #permanent global learned response
fit.HNbk <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
                     model = g0 ~ bk, trace = FALSE, verify = FALSE)   #the site specific learned response
# fit.HNt <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
#                       model = g0 ~ t, trace = FALSE, verify = FALSE)  #time consuming  #time factor (one
level for each occasion) - no need to run this again as all tests we made returned model = g0 ~ bk as perfor
ming the best.
# fit.HNT <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN',
#                      model = g0 ~ T, trace = FALSE, verify = FALSE)  #time consuming  #time trend (intege
r covariate 0:(S-1)) - no need to run this again as all tests we made returned model = g0 ~ bk as performing
the best.

#check which detection model works best
fitsb <- secrlist(null = fit$HN, b = fit.HNb, bk = fit.HNbk)
AIC(fitsb)
```

```
##                model   detectfn npar    logLik      AIC     AICc  dAICc AICcwt
## bk D~1 g0~bk sigma~1 halfnormal    4 -549.5216 1107.043 1109.265  0.000      1
## b   D~1 g0~b sigma~1 halfnormal    4 -558.4300 1124.860 1127.082 17.817      0
```

```
predict(fitsb)
```

```
## $b
##        link     estimate  SE.estimate          lcl          ucl
## D       log 1.802832e-04 4.314483e-05 1.135214e-04 2.863074e-04
## g0    logit 3.129284e-02 9.964039e-03 1.667858e-02 5.795777e-02
## sigma   log 4.035719e+03 2.992770e+02 3.490478e+03 4.666132e+03
##
## $bk
##        link     estimate  SE.estimate          lcl          ucl
## D       log 1.586708e-04 3.566573e-05 1.026877e-04 2.451748e-04
## g0    logit 3.262192e-02 6.038925e-03 2.265042e-02 4.677313e-02
## sigma   log 4.502215e+03 4.051883e+02 3.775501e+03 5.368807e+03
```

```
coef(fit.HNbk)
```

```
##                beta    SE.beta        lcl       ucl
## D         -8.748679 0.22201407 -9.1838184 -8.313539
## g0        -3.389605 0.19136122 -3.7646662 -3.014544
## g0.bkTRUE  1.160863 0.24085643  0.6887933  1.632933
## sigma      8.412325 0.08981612  8.2362884  8.588361
```

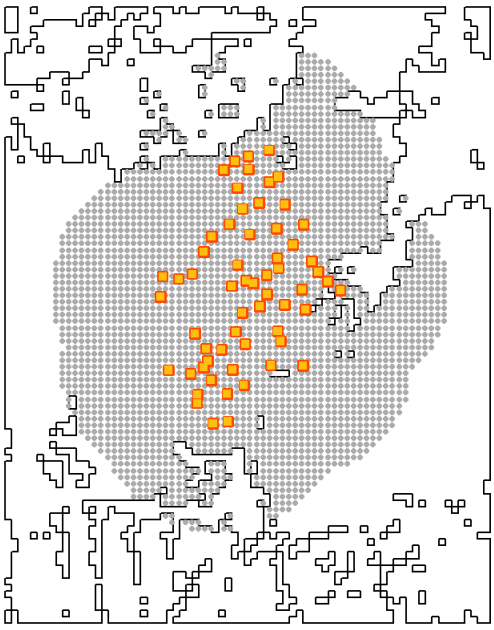
```
#model averaging
model.average(fitsb)
```

```
##           estimate  SE.estimate          lcl          ucl
## D     1.586708e-04 3.566573e-05 1.026877e-04 2.451748e-04
## g0    3.262192e-02 6.038925e-03 2.265042e-02 4.677313e-02
## sigma 4.502215e+03 4.051883e+02 3.775501e+03 5.368807e+03
```

It seems that [model = g0 ~ bk] works best as during the winter session. The learned response is positive g0.bTRUE = 1.866389, but as described already, we believe the animals didn't became trap happy after camera instalation, but this is rather an artefact of the lynx using the same movement paths and become more and more detectable at the same trap.

Add habitat/non-habitat mask in the story.

```
datadir <- system.file("secr/area_extent.shp", package = "secr")
lynxarea_70_10 <- shapefile("D://secr_s3_Stoenesti_excluded/mask_70_10.shp")

#finally, use the 5*sigma buffer width we concluded on earlier
lynxmask_70_10 <- make.mask(traps(lynxRO), spacing = 1000, buffer = 16710.74, type = "trapbuffer", poly = ly
nxarea_70_10)

#plot the mask
par(mar = c(3,6,6,6), xpd = TRUE)
plot (lynxmask_70_10, col = 'gray67', cex = 0.5, ppoly = TRUE)
#plot(traps(lynxRO), add = T, cex = 4)
plot(traps(lynxRO), detpar = list(pch = 0, cex = 0.8, col="#ff4800ff", lwd = 1.5), add = TRUE)
plot(traps(lynxRO), detpar = list(pch = 15, cex = 0.7, col="#fdbf10ff"), add = TRUE)
```



```
par(mar = c(5,4,4,2) + 0.1, xpd = FALSE)
```

Use argument [mask = lynxmask_70_10] hereafter.

Adjust for varying effort

```
#adjust for varying effort
#summary(lynxRO)   #see that Usage is binary, and it is wrong. Add data again with 'binary.usage = FALSE'
lynxRO <- read.capthist("lynxcapt.txt", "lynxtrap.txt", detector = "proximity", binary.usage = FALSE)
```

```
## Session AutumnWinter20192020
## More than one detection per detector per occasion at binary detector(s)
## Detections at 'unused' detectors
##          cbind(ID, detector, occasion)[conflcts, ]
## ID                                            B21
## detector                                        3
## occasion                                       19
```

```
summary(traps(lynxRO))   #perfect, now the Usage ranges between 0 and 5 (trap days per occassion)
```

```
## Object class       traps
## Detector type      proximity
## Detector number    60
## Average spacing    2470.473 m
## x-range            497925 525709 m
## y-range            426538 468669 m
##
##  Usage range by occasion
##     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## min 0 0 0 0 0 0 0 0 0  0  0  0  3  5  5  5  1  0  0  0
## max 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5  5  5  5  5  5
```

```
#run the models with adjusted varying effort and within the lynxmask_70_10
fit.usage.nomask <- secr.fit (lynxRO, buffer = 5 * initialsigma, detectfn = 'HN', model = g0 ~ bk, trace = F
ALSE, verify = FALSE) #implicitly adjusts for effort, no mask
fit.usage.mask <- secr.fit (lynxRO, mask = lynxmask_70_10, detectfn = 'HN', model = g0 ~ bk, trace = FALSE,
verify = FALSE) #implicitly adjusts for effort
#fit.t.mask <- secr.fit(lynxRO, model = g0 ~ t, mask = lynxmask_70_10, detectfn = 'HN', trace = FALSE, verif
y = FALSE) #ignores effort, but allows for occasion-to-occasion variation by fitting a separate g0 each time
.
usage(traps(lynxRO)) <- NULL #we whipe the usage information
fit.nousage.mask <- secr.fit(lynxRO, mask = lynxmask_70_10, detectfn = 'HN', model = g0 ~ bk, trace = FALSE,
verify = FALSE) #no adjustment

AIC(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)
```

```
##                              model   detectfn npar    logLik      AIC     AICc
## fit.usage.mask   D~1 g0~bk sigma~1 halfnormal    4 -547.8781 1103.756 1105.978
## fit.usage.nomask D~1 g0~bk sigma~1 halfnormal    4 -549.1619 1106.324 1108.546
## fit.nousage.mask D~1 g0~bk sigma~1 halfnormal    4 -556.3996 1120.799 1123.021
##                    dAICc AICcwt
## fit.usage.mask     0.000 0.7831
## fit.usage.nomask   2.568 0.2169
## fit.nousage.mask 17.043 0.0000
```

```
#Density estimates
collate(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)[1,,,]
```

```
## , , D
##
##                       estimate  SE.estimate          lcl          ucl
## fit.usage.nomask 0.0001588030 3.568607e-05 0.0001027847 0.0002453517
## fit.usage.mask   0.0001731291 3.847808e-05 0.0001125811 0.0002662408
## fit.nousage.mask 0.0001731426 3.836274e-05 0.0001127354 0.0002659177
##
## , , g0
##
##                      estimate SE.estimate         lcl         ucl
## fit.usage.nomask 0.006661959 0.001244791 0.004617223 0.009603469
## fit.usage.mask   0.006646646 0.001243830 0.004604028 0.009586762
## fit.nousage.mask 0.031643094 0.005797507 0.022055069 0.045206651
##
## , , sigma
##
##                  estimate SE.estimate      lcl      ucl
## fit.usage.nomask 4486.672    403.9463 3762.211 5350.638
## fit.usage.mask   4504.168    408.7455 3771.620 5378.998
## fit.nousage.mask 4470.786    398.8370 3754.905 5323.151
```

```
collate(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)[,,,'g0']
```

```
##                      estimate SE.estimate         lcl         ucl
## fit.usage.nomask 0.006661959 0.001244791 0.004617223 0.009603469
## fit.usage.mask   0.006646646 0.001243830 0.004604028 0.009586762
## fit.nousage.mask 0.031643094 0.005797507 0.022055069 0.045206651
```

```
collate(fit.usage.nomask, fit.usage.mask, fit.nousage.mask)[,,,'D']
```

```
##                      estimate  SE.estimate          lcl          ucl
## fit.usage.nomask 0.0001588030 3.568607e-05 0.0001027847 0.0002453517
## fit.usage.mask   0.0001731291 3.847808e-05 0.0001125811 0.0002662408
## fit.nousage.mask 0.0001731426 3.836274e-05 0.0001127354 0.0002659177
```

```
#Regional population size for the buffered region.
region.N(fit.usage.nomask)
```

```
##     estimate SE.estimate      lcl      ucl  n
## E.N 53.24416   11.964979 34.46209 82.26256 23
## R.N 53.24427    9.482435 39.59580 78.11732 23
```

```
region.N(fit.usage.mask)
```

```
##     estimate SE.estimate      lcl      ucl  n
## E.N 48.18183   10.708449 31.33132 74.09482 23
## R.N 48.18184    8.154082 36.56135 69.75971 23
```

```
#region.N(fit.t.mask)
region.N(fit.nousage.mask)
```

```
##     estimate SE.estimate      lcl      ucl  n
## E.N 48.18558   10.676350 31.37427 74.00490 23
## R.N 48.18570    8.111651 36.60631 69.61951 23
```

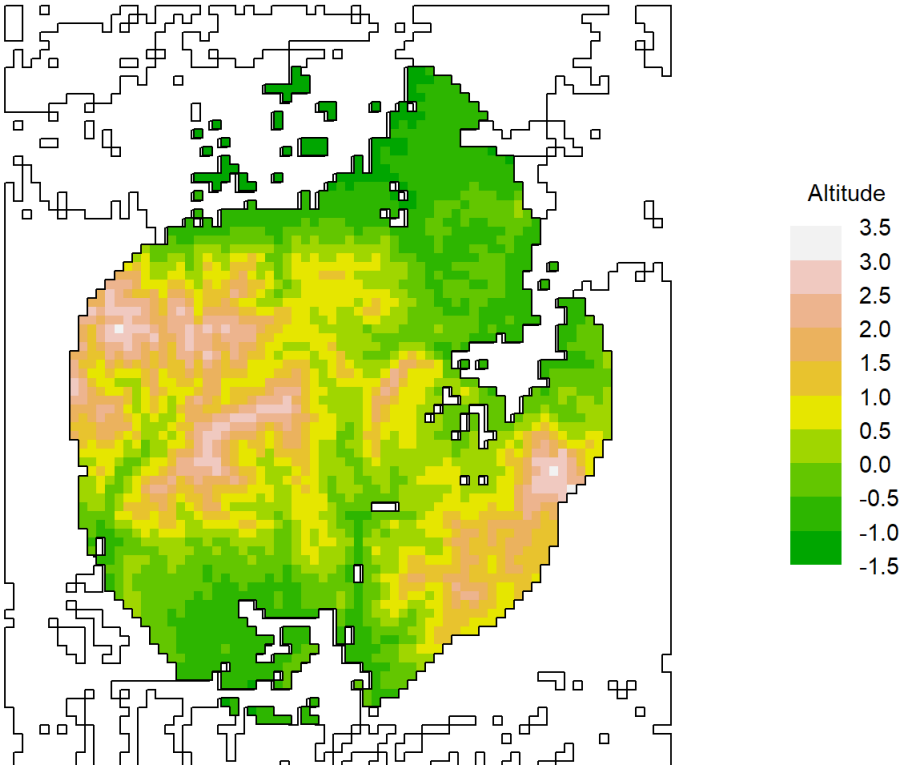
We interpret density values from model fit.usage.mask (mask included), it has the best AIC. Values has to be multiplied by 10000 to transform density per 1 hectare to density per 100 sqkm.

Data prep continued - add covariates to the mask points.

```
#add scaled covariates to the mask points.
lynxarea_70_10 <- shapefile("D://secr_s3_Stoenesti_excluded/mask_70_10.shp")
#return to spacing = 1000
lynxmask_70_10 <- make.mask(traps(lynxRO), spacing = 1000, buffer = 16710.74, type = "trapbuffer", poly = ly
nxarea_70_10)
lynxcov <- rgdal::readOGR(dsn = "grid_covariates_lynx_5.shp", layer = "grid_covariates_lynx_5")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "D:\secr_s3_Stoenesti_excluded\grid_covariates_lynx_5.shp", layer: "grid_covariates_lynx_5"
## with 5542 features
## It has 41 fields
## Integer64 fields read as strings:  CLC Reclass2
```

```
lynxmask <- addCovariates (lynxmask_70_10, lynxcov, columns = c("Alt", "Slo", "TRI5", "TRI9", "PublicRoad",
"Forest", "HumDomin", "OpenHab", "CLC_511", "CLC_222", "CLC_112", "CLC_211", "CLC_242", "CLC_311", "CLC_231"
, "CLC_243", "CLC_321", "CLC_331", "CLC_313", "CLC_324", "CLC_131", "CLC_221", "CLC_312", "CLC_142", "CLC_33
3", "CLC_411", "CLC_322", "CLC_141", "CLC_512", "CLC_332", "CLC_121", "CLC_111", "CLC_132", "CLC", "Reclass"
, "Reclass2", "OpenHaFin", "TradiAgri"),  strict = TRUE, replace = TRUE)
#summary(lynxmask)

#repair missing values
repair <- function (mask, covariate, ...) {
  NAcov <- is.na(covariates(mask)[,covariate])
  OK <- subset(mask, !NAcov)
  require(akima)
  irect <- akima::interp (x = OK$x, y = OK$y, z = covariates(OK)[,covariate],...)
  irectxy <- expand.grid(x = irect$x, y = irect$y)
  i <- nearesttrap(mask[NAcov,], irectxy)
  covariates(mask)[,covariate][NAcov] <- irect[[3]][i]
  mask
}

repaired <- repair(lynxmask, 'TradiAgri')

#interpolate covariates
copynearest <- function (mask, covariate) {
  NAcov <- is.na(covariates(mask)[,covariate])
  OK <- subset(mask, !NAcov)
  i <- nearesttrap(mask, OK)
  covariates(mask)[,covariate][NAcov] <- covariates(OK)[i[NAcov],covariate]
  mask
}

completed <- copynearest(repaired, 'TradiAgri')

#plotting mask with the covariates and explore patterns (just an example).
par(mar=c(1,1,2,6), xpd=TRUE)
plot(lynxmask, covariate = 'Alt', dots = FALSE, border = 100,
     title = 'Altitude', polycol = 'blue')
plotMaskEdge(lynxmask, add = TRUE)
```



Predict D_surface

```
#We fit simple models with covariates to our data
lynx.0 <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ 1, trace = FALSE, verify = FALSE)
lynx.D_alt <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Alt, method = "Nelder-Mead", trace =
FALSE, verify = FALSE)
lynx.D_tri <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ TRI9, method = "Nelder-Mead", trace =
FALSE, verify = FALSE)
lynx.D_slo <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Slo, method = "Nelder-Mead", trace =
FALSE, verify = FALSE)
lynx.D_landcover <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Reclass, method = "Nelder-Mead"
, trace = FALSE, verify = FALSE)
lynx.D_forest <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Forest, method = "Nelder-Mead", tr
ace = FALSE, verify = FALSE)
lynx.D_open <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ OpenHaFin, method = "Nelder-Mead", t
race = FALSE, verify = FALSE)
lynx.D_agri <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ TradiAgri, method = "Nelder-Mead", t
race = FALSE, verify = FALSE)
lynx.D_roads <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ PublicRoad, method = "Nelder-Mead",
trace = FALSE, verify = FALSE)
lynx.D_decid <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ CLC_311, method = "Nelder-Mead", tr
ace = FALSE, verify = FALSE)
lynx.D_conif <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ CLC_312, method = "Nelder-Mead", tr
ace = FALSE, verify = FALSE)
lynx.D_mixt <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ CLC_313, method = "Nelder-Mead", tra
ce = FALSE, verify = FALSE)

AIC(lynx.0, lynx.D_alt, lynx.D_tri, lynx.D_slo, lynx.D_landcover, lynx.D_forest, lynx.D_open, lynx.D_agri, l
ynx.D_roads, lynx.D_decid, lynx.D_conif, lynx.D_mixt, criterion = "AIC")[,-2]
```

```
##                                    model npar    logLik     AIC    AICc  dAIC
## lynx.D_slo              D~Slo g0~1 sigma~1    4 -263.8259 535.652 537.874 0.000
## lynx.D_tri             D~TRI9 g0~1 sigma~1    4 -263.8980 535.796 538.018 0.144
## lynx.0                   D~1 g0~1 sigma~1    3 -265.0308 536.062 537.325 0.410
## lynx.D_agri       D~TradiAgri g0~1 sigma~1    4 -264.1705 536.341 538.563 0.689
## lynx.D_alt             D~Alt g0~1 sigma~1    4 -264.3604 536.721 538.943 1.069
## lynx.D_conif        D~CLC_312 g0~1 sigma~1    4 -264.5717 537.143 539.366 1.491
## lynx.D_roads     D~PublicRoad g0~1 sigma~1    4 -264.6068 537.214 539.436 1.562
## lynx.D_open       D~OpenHaFin g0~1 sigma~1    4 -264.7790 537.558 539.780 1.906
## lynx.D_forest        D~Forest g0~1 sigma~1    4 -264.8469 537.694 539.916 2.042
## lynx.D_decid        D~CLC_311 g0~1 sigma~1    4 -265.0055 538.011 540.233 2.359
## lynx.D_mixt         D~CLC_313 g0~1 sigma~1    4 -265.0212 538.042 540.265 2.390
## lynx.D_landcover    D~Reclass g0~1 sigma~1    7 -263.3727 540.745 548.212 5.093
##                    AICwt
## lynx.D_slo        0.1561
## lynx.D_tri        0.1452
## lynx.0            0.1272
## lynx.D_agri       0.1106
## lynx.D_alt        0.0915
## lynx.D_conif      0.0741
## lynx.D_roads      0.0715
## lynx.D_open       0.0602
## lynx.D_forest     0.0562
## lynx.D_decid      0.0480
## lynx.D_mixt       0.0472
## lynx.D_landcover 0.0122
```

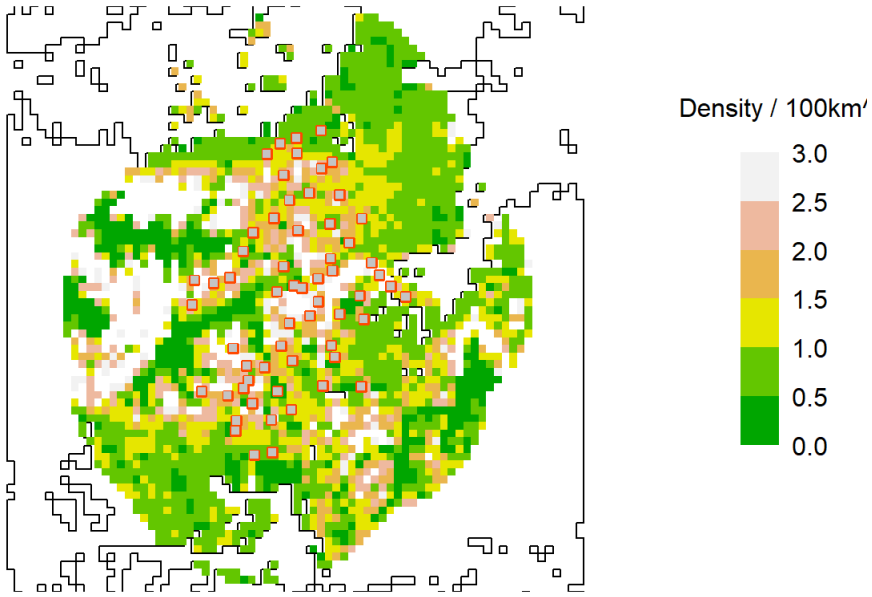
Note that during this session some of the variables really stands out as important for D (Slo, TRI, TradiAgri accout for AICwt of .45, and if we add Alt and Forest it goes up to .6. Look now at the PublicRoad that induced bias during the winter. No longer important during the autumn, suggesting lynx are now detected at traps further from roads too. Thus, the model covariates used for predicting D_surface were model = D ~ Forest + TradiAgri + Slo + I(Slo^2).

Map the D_surface prediction and its CIs

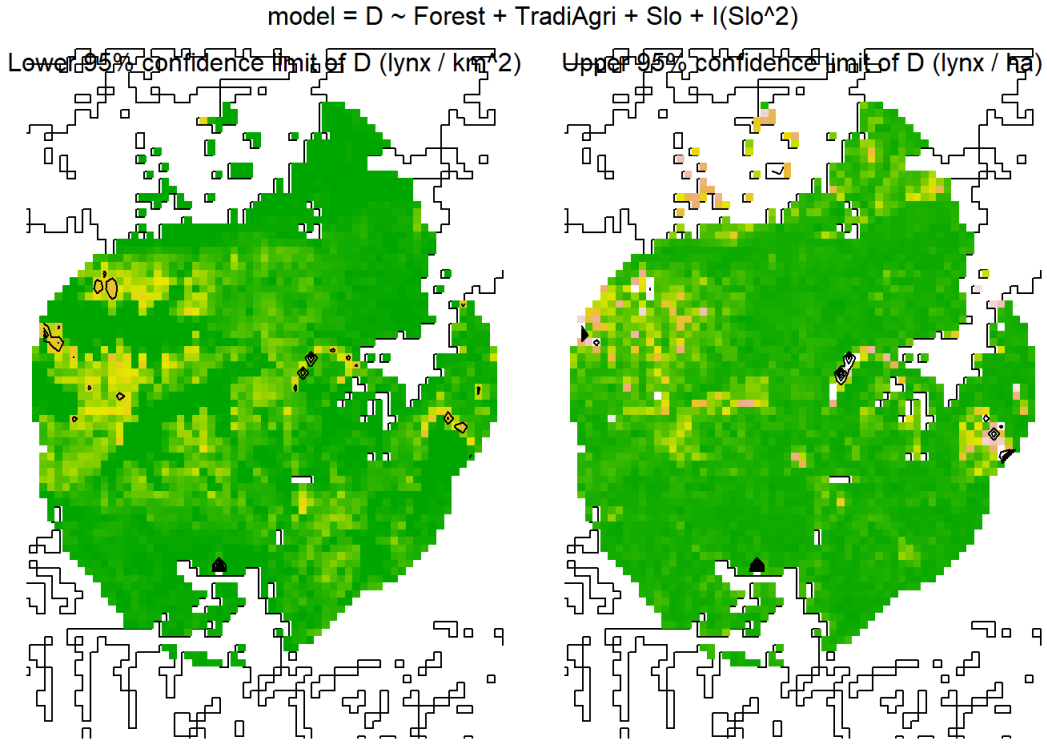
```
#final model for S3
lynx.D_s3 <- secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Forest + TradiAgri + Slo + I(Slo^2), m
ethod = "Nelder-Mead", trace = FALSE, verify = FALSE)
```

```
## Warning in secr.fit(capthist = lynxRO, mask = lynxmask, model = D ~ Forest + :
## probable maximization error: optim returned convergence 1. See ?optim
```

```
#plot D surface
par(mar = c(3,6,3,8))
surface.D_s3 <- predictDsurface(lynx.D_s3)
plot(surface.D_s3, plottype = "shaded", poly = FALSE, breaks = seq(0,3,0.5),
     title = "Density / 100km^2", text.cex = 1, scale = 10000)
plot(traps(lynxRO), detpar = list(pch = 0, cex = 0.8, col="#ff4800ff", lwd = 1.5), add = TRUE)
plot(traps(lynxRO), detpar = list(pch = 15, cex = 0.7, col="gray77"), add = TRUE)
```



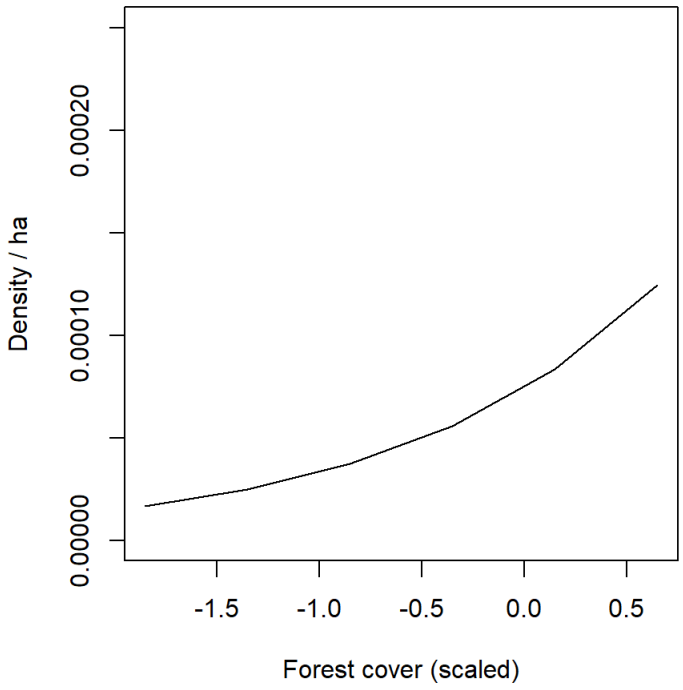

```
#map CIs
par(mar = c(1,1,1,1), mfrow = c(1,2), xpd = FALSE)
surface.D_s3 <- predictDsurface(lynx.D_s3, cl.D = TRUE)
plot(surface.D_s3, covariate= "lcl", breaks = seq(0,0.0005,0.00001), legend = FALSE)
mtext(side=3,line=-1.5,"Lower 95% confidence limit of D (lynx / km^2)")
plot(surface.D_s3, plottype = "contour", breaks = seq(0,0005,0.00001), add = TRUE)
plot(surface.D_s3, covariate= "ucl", breaks = seq(0,0.0060,0.0001), legend = FALSE)
mtext(side=3,line=-1.5,"Upper 95% confidence limit of D (lynx / ha)")
plot(surface.D_s3, covariate= "ucl", plottype = "contour", breaks = seq(0,0.0060,0.0001),
     add = TRUE)
mtext(side=3, line=-1, outer=TRUE, "model = D ~ Forest + TradiAgri + Slo + I(Slo^2)")
```



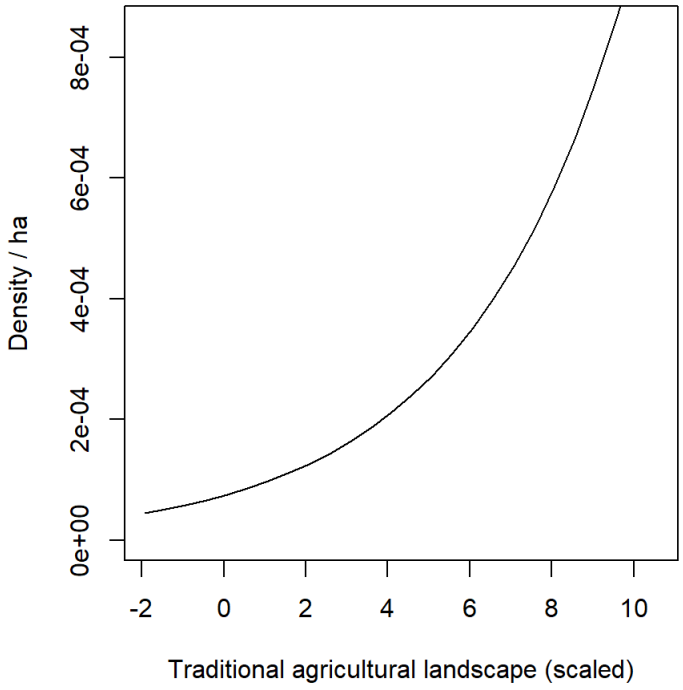
Note that for the alpine area the upper CIs is high. The model is probably overestimating D at high elevations but at mid and low elevations the D surface patterns and CIs look fine. Note a swith of density hotspots from the traditional agricultural landcsape in winter to the compact forest in autumn-early winter.

Explore coefficients and plot density against covariates

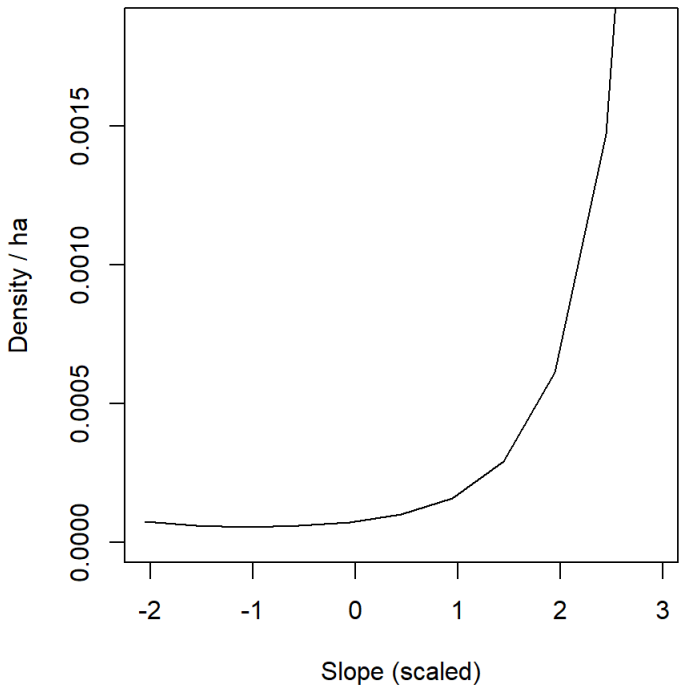
```
#summary(lynxmask)
#lynx.D_s3
# Beta parameters (coefficients)
#                   beta    SE.beta          lcl        ucl
# D           -9.3558263 0.66185985 -10.65304781 -8.0586049
# D.Forest     0.8518444 0.92558309  -0.96226510  2.6659539
# D.TradiAgri  0.3051157 0.10990719   0.08970161  0.5205299
# D.Slo        1.0446217 1.05862281  -1.03024088  3.1194843
# D.I(Slo^2)  -0.1310709 0.71445840  -1.53138361  1.2692419
# g0          -2.8724992 0.14539475  -3.15746767 -2.5875307
# sigma        8.2921674 0.07095075   8.15310653  8.4312284


#plot density against covariates
tmp <- predict(lynx.D_s3, newdata = data.frame(Forest = seq(-1.85145,1.03262,0.5), TradiAgri=0, Slo=0))
par(mar=c(5,8,2,4), pty = "s")
plot(seq(-1.85145,1.03262,0.5), sapply(tmp, "[", "D","estimate"), ylim = c(0,0.00025),
     xlab = "Forest cover (scaled)", ylab = "Density / ha", type = "l")
```




```
tmp <- predict(lynx.D_s3, newdata = data.frame(TradiAgri = seq(-1.9190,10.7610,0.5), Forest=0, Slo=0))
par(mar=c(5,8,2,4), pty = "s")
plot(seq(-1.9190,10.7610,0.5), sapply(tmp, "[", "D","estimate"), ylim = c(0,0.00085),
     xlab = "Traditional agricultural landscape (scaled)", ylab = "Density / ha", type = "l")
```




```
tmp <- predict(lynx.D_s3, newdata = data.frame(Slo = seq(-2.0544,3.2576,0.5), TradiAgri=0, Forest=0))
par(mar=c(5,8,2,4), pty = "s")
plot(seq(-2.0544,3.2576,0.5), sapply(tmp, "[", "D","estimate"), ylim = c(0,0.00185),
     xlab = "Slope (scaled)", ylab = "Density / ha", type = "l")
```



Export final prediction

```
#export as Raster Layer, format = tif
D_s3 <- raster(surface.D_s3, covariate = "D.0", crs = "+proj=sterea +lat_0=46 +lon_0=25 +k=0.99975 +x_0=500000 +y_0=500000 +ellps=krass +towgs84=28,-121,-77,0,0,0,0 +units=m +no_defs")
writeRaster(D_s3, 'D_s3_20200925.tif', format='GTiff',  overwrite=TRUE)
D_s3_lcl <- raster(surface.D_s3, covariate = "lcl.0", crs = "+proj=sterea +lat_0=46 +lon_0=25 +k=0.99975 +x_0=500000 +y_0=500000 +ellps=krass +towgs84=28,-121,-77,0,0,0,0 +units=m +no_defs")
writeRaster(D_s3_lcl, 'D_s3_lcl_20200925.tif', format='GTiff',  overwrite=TRUE)
D_s3_ucl<- raster(surface.D_s3, covariate = "ucl.0", crs = "+proj=sterea +lat_0=46 +lon_0=25 +k=0.99975 +x_0=500000 +y_0=500000 +ellps=krass +towgs84=28,-121,-77,0,0,0,0 +units=m +no_defs")
writeRaster(D_s3_ucl, 'D_s3_ucl_20200925.tif', format='GTiff',  overwrite=TRUE)
```