Simulating scenarios of feature distribution

Bernardo Niebuhr

19 November, 2021

Contents

Introduction 1
Simulate landscapes 1
Calculate correlations 6
To do 8

Introduction

There are a few big questions that underlie the assessment of the effects of anthropogenic infrastructure on wildlife. When performing environmental impact assessments, one aims not only to to find which factors affect wildlife and how strongly, but also (i) at which spatial scale there are effects and (ii) how these effects sum and interact when (as it is often the case) there are multiple infrastructures and vectors of landscape modification. The first question is also knows as the Zone of Influence (ZoI) problem. The second is generally tackled in the context of cumulative impact assessment.

In the main text of this manuscript, we argue that the density of infrastructure features might better represent the cumulative effect of multiple features in the landscape than considering only the distance to the feature nearest to a given location. This, however, might vary depending on how the number of features in the landscape, how these features are distributed in space, and what is the ZoI of each of those features.

Here we simulate some landscapes with point-type infrastructure spread following different patterns, calculate the distance to the nearest feature and the density of features, at multiple scales, and assess when and how these variables might represent different sources of spatial variation.

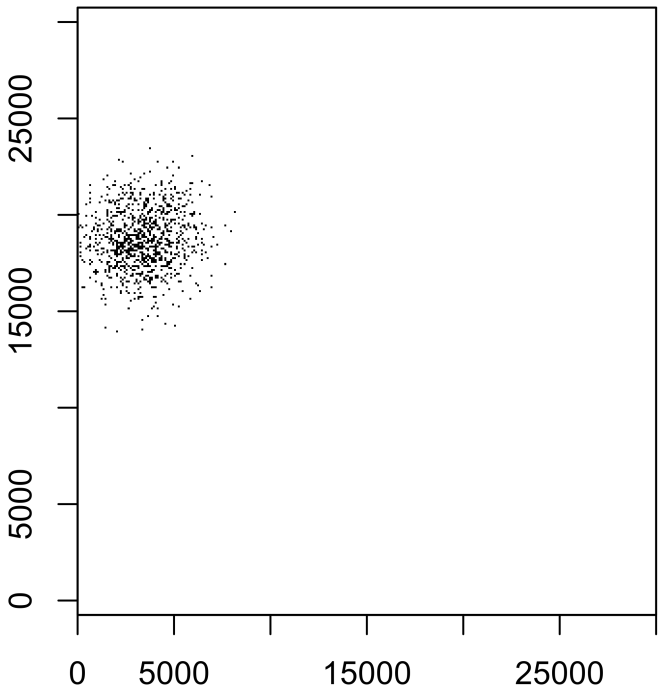
Simulate landscapes

First we simulate some landscape using point-type infrastructure as an example. They could represent the spatial location of houses, cabins, or wind turbines, for example. To do this we'll use a few functions designed within the package `ZOItools`.

```
# Load packages
library(dplyr)
library(ggplot2)
library(mobsim)
library(raster)
library(rasterVis)
library(sf)

library(oneimpact)

library(reshape2)
library(lsr)
```

We set a 30x30 km^2 landscape and can create simulate points following different spatial patterns depending on the number of centers (or patches) around which the points will be located (parameter `centers`) and the width of their distribution around these centers (parameter `width`). Here is an example landscape where the points are spread close to a single center (e.g. houses in a village).

```r
set.seed(1234)

# simulate a single patch
nfeat <- 1000 # number of features
ext <- 30000 # extension of the landscape
nc <- 1 # number of centers or patches
wd <- ext / 20 # width of the patch
pts <- set_points(
  n_features = 1000, centers = nc,
  width = wd, res = 100,
  extent_x = c(0, ext), extent_y = c(0, ext)
)
plot(pts$rast, col = "black", legend = F)
```

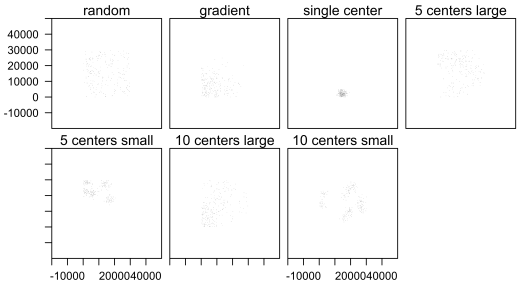


We can now simulate landscapes with different patterns. For now we'll keep the total number of features constant.

Should we include a regular pattern as well?

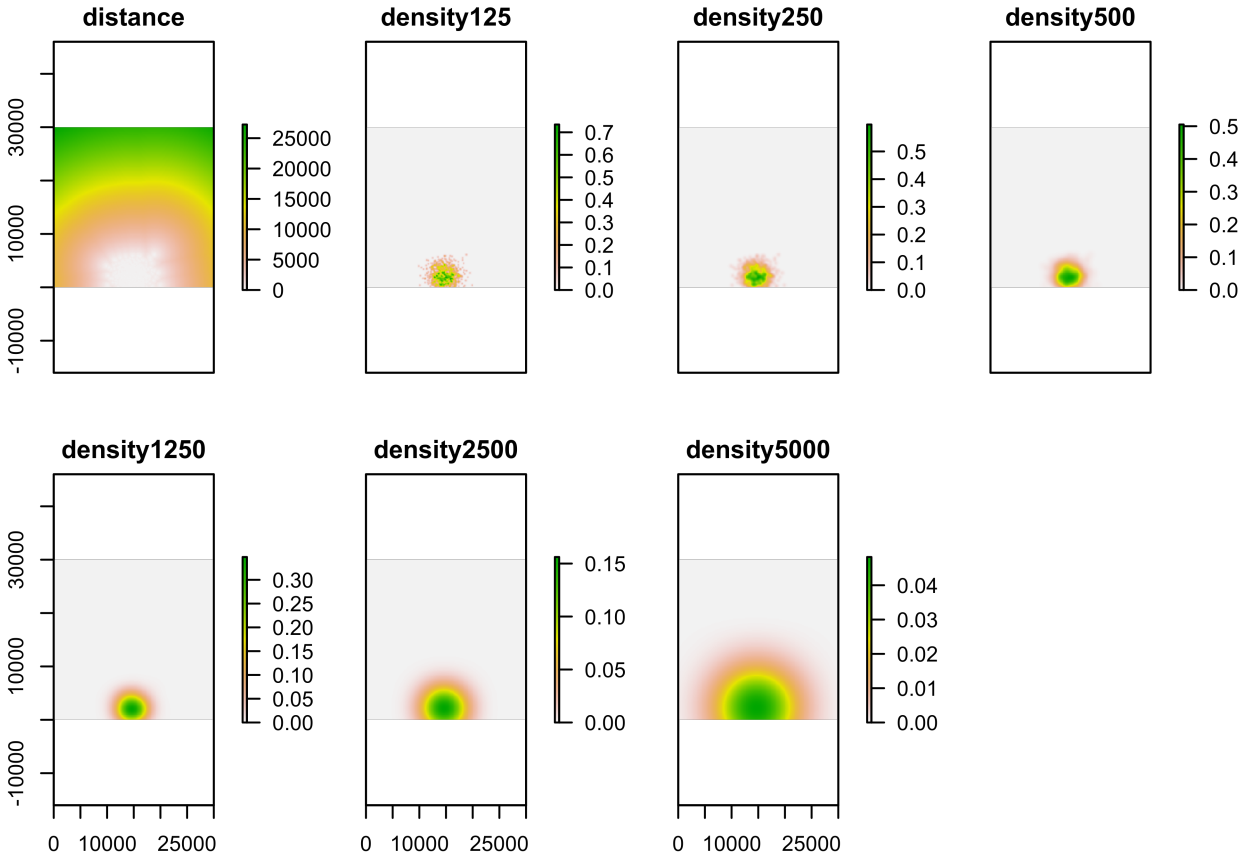
```r
set.seed(123)

# random, gradient, single center, multiple centers
types <- c(
  "random", "gradient", "single center",
  "5 centers large", "5 centers small",
  "10 centers large", "10 centers small"
)
```

```r
nfeat <- 1000 # number of features
ext <- 30000 # extent of the landscape
nc <- c(1, 1, 1, 5, 5, 10, 10) # number of centers
wd <- c(1, 0.3, 0.05, 0.15, 0.05, 0.15, 0.05) * ext # width of the "patches"
buff <- 20000 # add buffer to avoid edge effects when calculating densities

# simulate points
pts <- mapply(set_points,
  centers = nc, width = wd,
  MoreArgs = list(
    n_features = 1000, res = 100,
    extent_x = c(0, ext), extent_y = c(0, ext),
    buffer_around = 20000
  )
)
landscapes <- raster::stack(pts[2, ])
names(landscapes) <- types
# plot(landscapes, col = "black", nc = 4, legend = F)

rasterVis::levelplot(landscapes,
  layout = c(4, 2), names.attr = types,
  par.settings = GrTheme, colorkey = FALSE
)
```



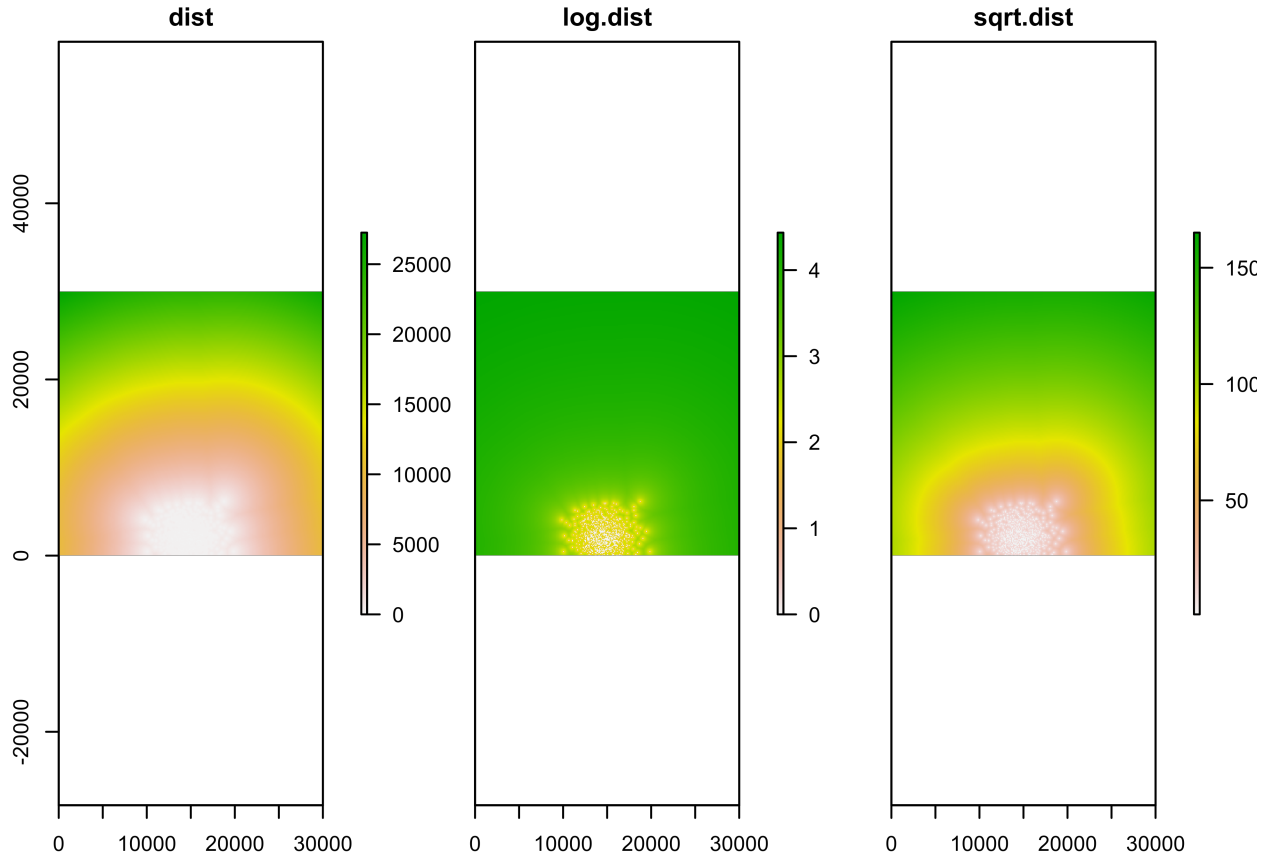
First we illustrate, for one of those scenarios, how the maps change as we calculate either the distance to the nearest feature or the density of features at different scales.

```r
# calculate distance and density at multiple scales for one input
scales <- c(250, 500, 1000, 2500, 5000, 10000) / 2 # scales for Gaussian filter
dist_dens1 <- calc_dist_dens(landscapes[[3]],
  type_density = "Gauss", scale = scales,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
plot(dist_dens1, nc = 4)
```



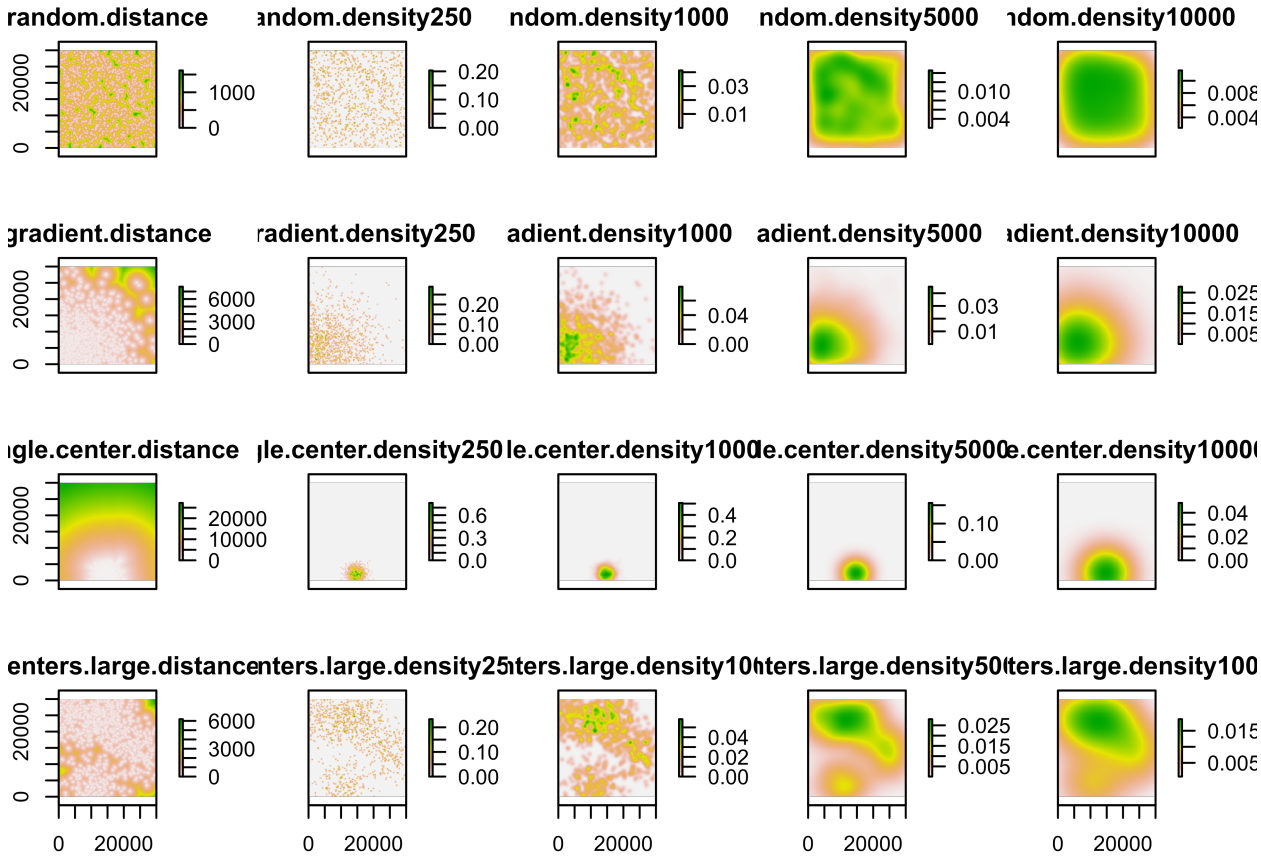
We can see that the results are different if one transforms the distance variable, as it is generally done in ecological models. Below we show, as an example, how it looks like when the distances are log- and sqrt-transformed. It probably affects how distances and densities correlate, as we'll see.

```
log_dist <- calc_dist(landscapes[[3]],
  transform_dist = "log", log_base = 10,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
sqrt_dist <- calc_dist(landscapes[[3]],
  transform_dist = "sqrt",
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
dists <- raster::stack(dist_dens1[[1]], log_dist, sqrt_dist)
names(dists) <- c("dist", "log-dist", "sqrt-dist")
plot(dists, nc = 3)
```



Now we compare these variables visually for the different spatial point patterns, using the non-transformed distance, for now.

```
# calculate distance and density at multiple scales for each input
dist_dens <- sapply(as.list(landscapes), calc_dist_dens,
  type_density = "Gauss", scale = scales,
  # transform_dist = "log", log_base = 10,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)

dist_dens <- raster::stack(dist_dens)
# change names according to the scenario and the variable
vars <- c("distance", paste0("density", scales * 2))
names(dist_dens) <- paste(rep(types, each = length(vars)), vars, sep = ".")

# slice a few of them to plot
slices <- as.vector(outer(c(1, 2, 4, 6, 7), (0:3) * 7, FUN = "+"))
plot(raster::subset(dist_dens, slices), nc = 5, maxnl = 30) # , main = "")
```



Calculate correlations

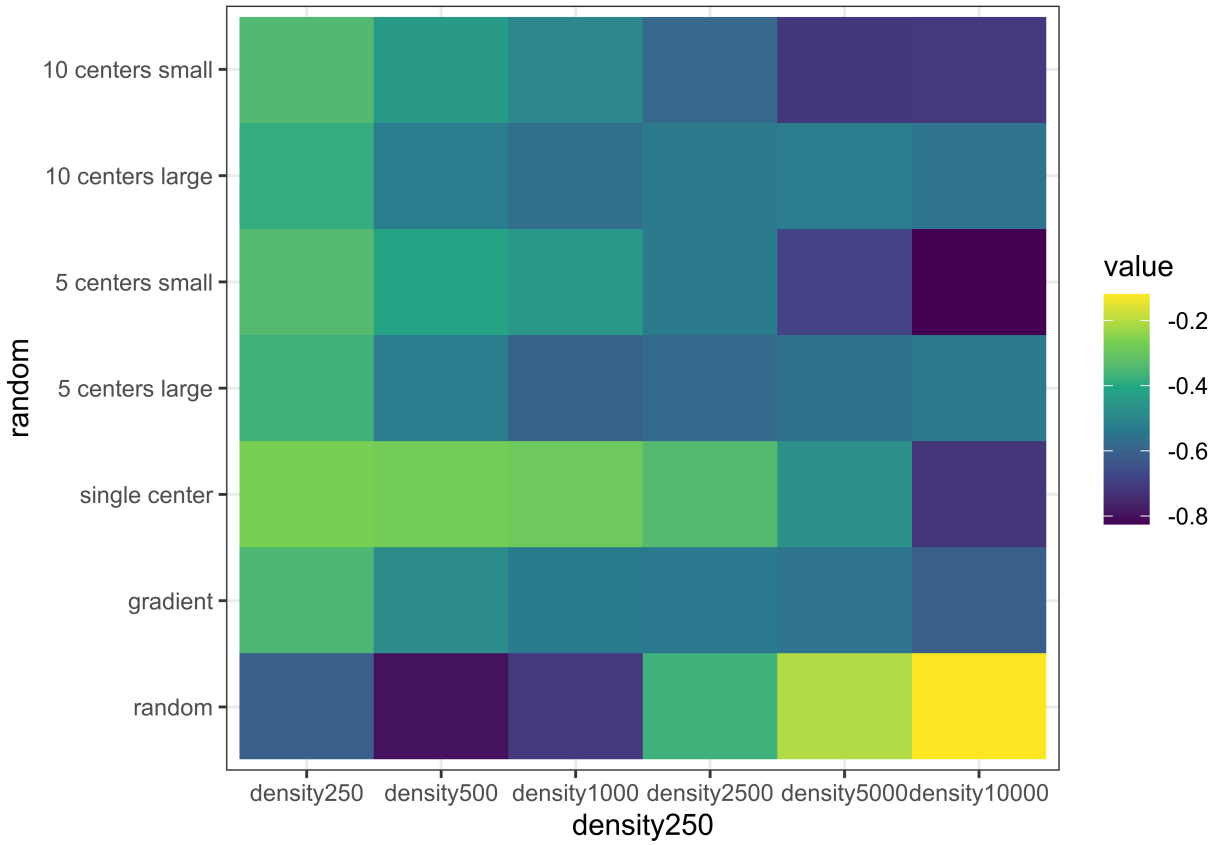
Now we are going to check how correlated these variables are in space, to evaluate how much they can have different ecological interpretations.

```
# Calculate pairwise Pearson correlation coefficient
# rast_cor <- layerStats(dist_dens, stat = "pearson")[[1]]
#
# # Extract the ones which are interesting for us
# dist_dens_cor <- rbind(rast_cor[1, 2:7], rast_cor[8, 9:14],
#                        rast_cor[15, 16:21], rast_cor[22, 23:28])
# colnames(dist_dens_cor) <- vars[2:7]
# rownames(dist_dens_cor) <- types
#
# # print
# dist_dens_cor

# Different approach
dist_dens_df <- as.data.frame(dist_dens)

# Pairwise correlation
corr <- correlate(dist_dens_df)
# Select distance vs density for each scenario
dist_dens_corr <- rbind(
  corr$correlation[1, 2:7], corr$correlation[8, 9:14],
  corr$correlation[15, 16:21], corr$correlation[22, 23:28],
  corr$correlation[29, 30:35], corr$correlation[36, 37:42],
  corr$correlation[43, 44:49]
```

```
)
rownames(dist_dens_corr) <- types
colnames(dist_dens_corr) <- vars[-1]

# print
dist_dens_corr
```

```
##                  density250 density500 density1000 density2500 density5000
## random           -0.6120884 -0.7982375  -0.7093161  -0.3718260  -0.2053362
## gradient         -0.3510334 -0.4830840  -0.5357922  -0.5400772  -0.5539752
## single center    -0.2647836 -0.2775889  -0.2885488  -0.3406239  -0.4698438
## 5 centers large  -0.3704843 -0.5280753  -0.6046977  -0.5774101  -0.5582975
## 5 centers small  -0.3406804 -0.4134138  -0.4490224  -0.5334645  -0.6898730
## 10 centers large -0.3839009 -0.5302476  -0.5683766  -0.5334847  -0.5310896
## 10 centers small -0.3429757 -0.4458933  -0.4970344  -0.5833531  -0.7141403
##                  density10000
## random             -0.1197783
## gradient           -0.6113346
## single center      -0.7212928
## 5 centers large    -0.5344893
## 5 centers small    -0.8250903
## 10 centers large   -0.5548095
## 10 centers small   -0.7115492
```

```
# plot
dist_dens_corr %>%
  reshape2::melt() %>%
  ggplot(aes(x = Var2, y = Var1)) +
  geom_raster(aes(fill = value)) +
  scale_fill_viridis_c() +
  labs(x = vars[-1], y = types) +
  theme_bw()
```



Some text here.

To do

1. Repeat the example for log(dist) and sqrt(dist)
2. Repeat the test including multiple (n = 100?) simulations for each scenario and taking the average correlation
3. Use GRASS GIS functions for neighborhood analyses
4. Relationship between mean free path and the correlation?

Maybe: 5. Plot number of significant correlation tests as well.