Simulating scenarios: comparing log-distance to the nearest feature to the cumulative impact (density) measure

Bernardo Niebuhr

04 December, 2021

Contents

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 cumulative impact measure - based on 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 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 cumulative impact (density) of features, at multiple scales, and assess when and how these variables might represent different sources of spatial variation.

Simulate landscapes

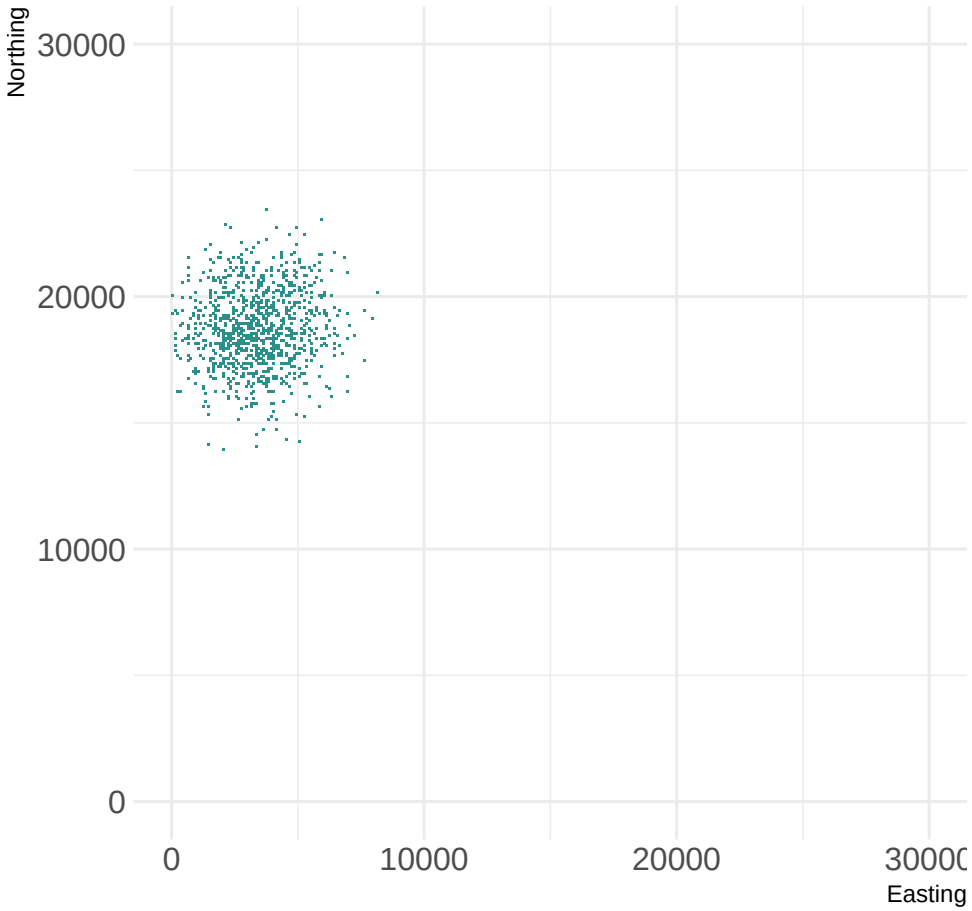
First we simulate some landscape using point-type infrastructure as an example. They could represent the spatial location of houses, cabins, or wind turbines, for example. To do this we'll use a few functions designed within the R package `oneimpact`. We also load other packages from data manipulation and plotting.

```
# Load packages

# data manipulation
library(dplyr)
library(purrr)
library(ggplot2)
library(ggpubr)
library(lsr) # for correlations

# spatial packages
library(mobsim)
library(raster)
library(landscapetools)
```

```r
library(sf)
library(spatstat)

# oneimpact package
library(oneimpact)
```

We set a 30x30 km^2 landscape and can simulate points following different spatial patterns. 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)
)

landscapetools::show_landscape(pts$rast, legend.position = "none")
```



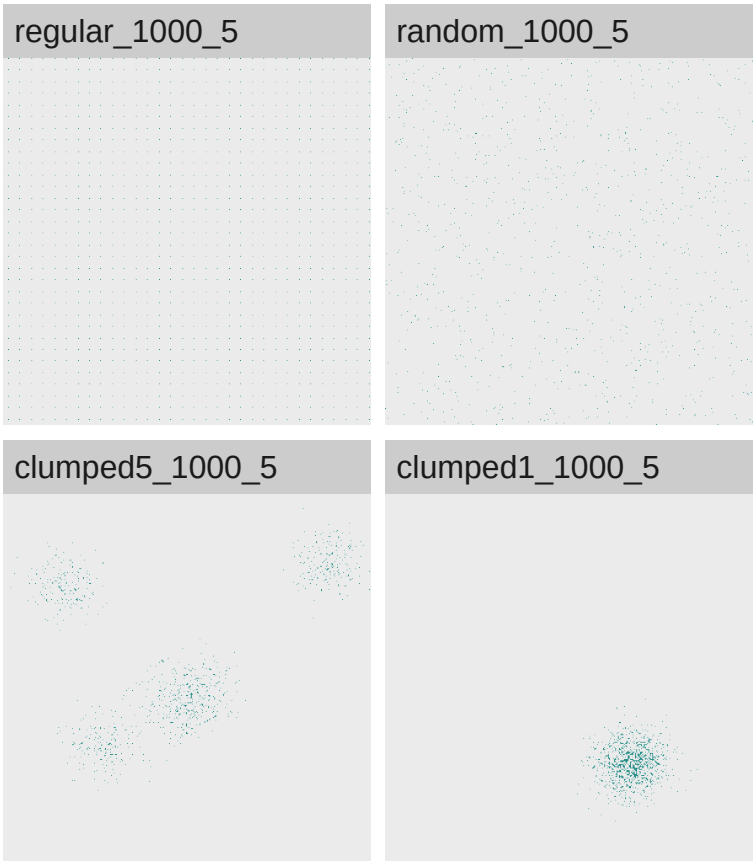
We can now simulate landscapes with different patterns. We start with 3 spatial distribution of points: regular, random, and clumped distribution of points. For the clumped distribution, we use two scenarios, where the point

features are spread in either 5 or only 1 focal patch. Each scenario is simulated with 10, 100, and 1000 points, so that we have 12 scenarios in total.

```
set.seed(1234)

# random, gradient, single center, multiple centers
methods <- c("regular", "random", "mobsim")
name <- c("regular", "random", "clumped5", "clumped1")
nfeat <- c(10, 30, 50, 100, 200, 1000) # number of features
res <- 100 # resolution
ext <- 30000 # extent of the landscape
nc <- c(5, 1) # number of centers for clumped
wd <- c(0.05, 0.15) * ext # width of the "patches"

# parameters
parms_df1 <- expand.grid(
  method = methods, n_features = nfeat,
  centers = nc[1], width = wd
) # first 3 scenarios
parms_df2 <- expand.grid(
  method = methods[3], n_features = nfeat,
  centers = nc[2], width = wd
) # parameters for clumped1
parms_df <- dplyr::bind_rows(parms_df1, parms_df2) %>%
  dplyr::arrange(width, n_features, method)
# names of scenarios
scenarios <- paste0(rep(name, 12), "_", rep(rep(nfeat, each = 4), 2), "_", rep(wd / 300, each = 24))

# simulate points
pts <- parms_df %>% purrr::pmap(set_points,
  res = res,
  extent_x = c(0, ext),
  extent_y = c(0, ext),
  buffer_around = 10000
)

landscapes <- purrr::map(pts, ~ .[[2]])
names(landscapes) <- scenarios
```

Here we visualize the scenarios with 1000 features.

```
# show landscapes with n_features = 1000
# plot(landscapes, col = "black", nc = 4, legend = F)
# rasterVis::levelplot(stack(landscapes), layout = c(4,3), names.attr = scenarios,
#                      par.settings = GrTheme, colorkey = FALSE)


landscapes[21:24] %>%
  purrr::map(raster::crop, y = extent(0, ext, 0, ext)) %>%
  landscapetools::show_landscape()
```



We also store some variables related the distance between points for each of these scenarios. These variables are: (i) the average distance between points, (ii) the average nearest neighbor distance, and (iii) the average isolation. The average is isolation is defined here as the mean nearest neighbor distance between random points created in the landscapes and the simulated point feature locations.

```
# isolation to random points
isolation <- function(x, n_rand = 100, ext = c(0, 1), lonlat = FALSE) {
  # create random points
  rand <- data.frame(x = runif(n_rand, ext[1], ext[2]), y = runif(n_rand, ext[1], ext[2]))
  # calc dist
  dists <- pointDistance(x, rand, lonlat = lonlat)
  # min dist (nearest neighbor)
  apply(dists, 2, min)
}

# mean isolation
mean_isolation <- function(x, n_rand = 100, ext = c(0, 1), lonlat = FALSE) {
  mean(isolation(x, n_rand = n_rand, ext = ext, lonlat = lonlat))
}

# points
pts_coords <- purrr::map(pts, first)
# names(pts_coords) <- scenarios

# calculate distances
dist_scenarios <- data.frame(
  scenario = scenarios,
  spatial_dist = rep(name, 3),
  n_features = rep(nfeat, each = 4),
```

```
  patch_width = rep(wd / 300, each = 24),
  mean_dist = map_dbl(pts_coords, function(x) mean(dist(x))),
  mean_nndist = map_dbl(pts_coords, function(x) mean(spatstat.geom::nndist(x))),
  mean_isolation = map_dbl(pts_coords, mean_isolation, n_rand = 150, ext = c(0, ext))
)

dist_scenarios
```

```
##            scenario spatial_dist n_features patch_width mean_dist mean_nndist
## 1      regular_10_5      regular         10           5 15510.736   9486.8330
## 2       random_10_5       random         10           5 13260.840   3619.1979
## 3     clumped5_10_5     clumped5         10           5 10485.280   1465.9146
## 4     clumped1_10_5     clumped1         10           5  2793.825    910.2402
## 5      regular_30_5      regular         30           5 17353.637   5477.2256
## 6       random_30_5       random         30           5 13923.599   3028.2656
## 7     clumped5_30_5     clumped5         30           5  7951.216   1431.1789
## 8     clumped1_30_5     clumped1         30           5  2361.774    618.0082
## 9      regular_50_5      regular         50           5 15632.700   4242.6407
## 10      random_50_5       random         50           5 15850.874   2356.9357
## 11    clumped5_50_5     clumped5         50           5 13768.935    964.3111
## 12    clumped1_50_5     clumped1         50           5  2730.750    517.3460
## 13    regular_100_5      regular        100           5 15717.795   3000.0000
## 14     random_100_5       random        100           5 14812.534   1529.8289
## 15   clumped5_100_5     clumped5        100           5 13753.443    679.2928
## 16   clumped1_100_5     clumped1        100           5  2453.467    339.7859
## 17    regular_200_5      regular        200           5 15524.037   2121.3203
## 18     random_200_5       random        200           5 15531.201   1130.0191
## 19   clumped5_200_5     clumped5        200           5 14294.580    406.2432
## 20   clumped1_200_5     clumped1        200           5  2754.501    249.5978
## 21   regular_1000_5      regular       1000           5 15836.660    948.6833
## 22    random_1000_5       random       1000           5 15544.613    470.9937
## 23  clumped5_1000_5     clumped5       1000           5 11580.682    212.1259
## 24  clumped1_1000_5     clumped1       1000           5  2600.941    113.5977
## 25    regular_10_15      regular         10          15 15510.736   9486.8330
## 26     random_10_15       random         10          15 14631.971   5901.3242
## 27   clumped5_10_15     clumped5         10          15 15199.039   4910.7269
## 28   clumped1_10_15     clumped1         10          15  6575.873   1737.7268
## 29    regular_30_15      regular         30          15 15981.307   5477.2256
## 30     random_30_15       random         30          15 16045.489   2944.7290
## 31   clumped5_30_15     clumped5         30          15 11925.356   1841.9030
## 32   clumped1_30_15     clumped1         30          15  6066.064   1372.6964
## 33    regular_50_15      regular         50          15 15632.700   4242.6407
## 34     random_50_15       random         50          15 16415.665   2284.6495
## 35   clumped5_50_15     clumped5         50          15 13761.444   1908.3041
## 36   clumped1_50_15     clumped1         50          15  7683.572   1381.2363
## 37   regular_100_15      regular        100          15 15717.795   3000.0000
## 38    random_100_15       random        100          15 15467.926   1766.2872
## 39  clumped5_100_15     clumped5        100          15 11736.599   1142.5785
## 40  clumped1_100_15     clumped1        100          15  6169.952    766.5546
## 41   regular_200_15      regular        200          15 15524.037   2121.3203
## 42    random_200_15       random        200          15 15576.791   1085.5294
## 43  clumped5_200_15     clumped5        200          15 13755.216   1003.9548
## 44  clumped1_200_15     clumped1        200          15  6724.922    587.0384
## 45  regular_1000_15      regular       1000          15 15342.259    948.6833
## 46   random_1000_15       random       1000          15 15862.488    478.6560
## 47 clumped5_1000_15     clumped5       1000          15 12428.291    406.8023
```

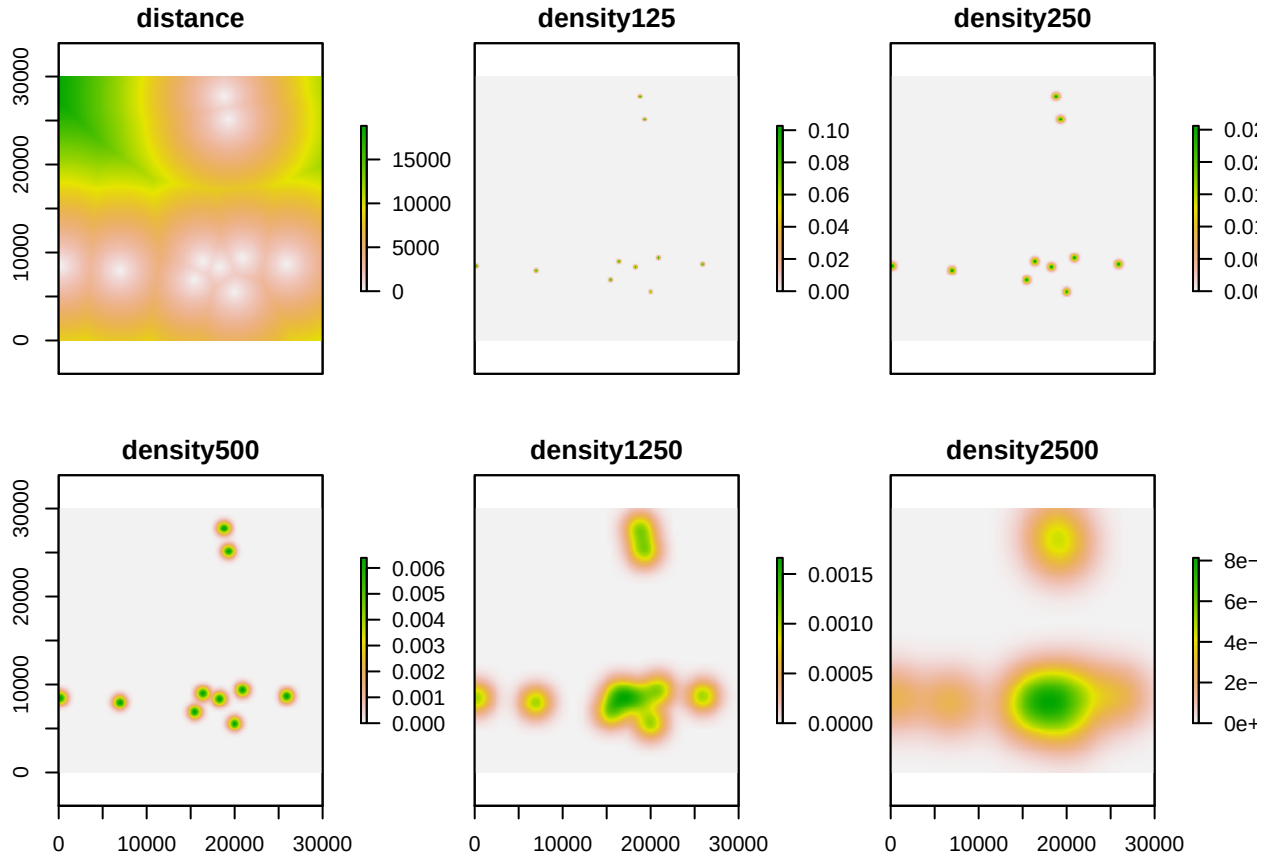
```
## 48 clumped1_1000_15     clumped1       1000          15  6612.116    248.4088
##    mean_isolation
## 1       4822.9495
## 2       5527.5554
## 3       7980.0693
## 4      11457.2618
## 5       2018.9347
## 6       3013.2951
## 7       6214.5410
## 8      12022.1163
## 9       1711.4763
## 10      2340.6671
## 11      4296.5861
## 12      9345.9451
## 13      1165.8395
## 14      1575.3059
## 15      4596.8840
## 16     11743.8054
## 17       786.7392
## 18      1043.8903
## 19      6247.4284
## 20     11834.8546
## 21       352.3430
## 22       462.9789
## 23      3646.2634
## 24     10024.8183
## 25      4034.8733
## 26      5807.2655
## 27      6486.6755
## 28      9689.1184
## 29      2232.4175
## 30      3071.8898
## 31      7358.2780
## 32      7444.7470
## 33      1717.7837
## 34      2097.9504
## 35      3003.4754
## 36      5876.3027
## 37      1161.8352
## 38      1461.5977
## 39      4822.5806
## 40      8475.1594
## 41       848.4060
## 42      1074.9182
## 43      1560.5772
## 44      6627.2802
## 45       367.8233
## 46       485.7060
## 47      1077.1697
## 48      5014.4699
```

Calculate distance and cumulative effects

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. For illustration purposes, we use here a Gaussian filter, where

the scale corresponds to the standard deviation σ of the Gaussian distribution, and the moving window has a size of $3 \cdot \sigma$.

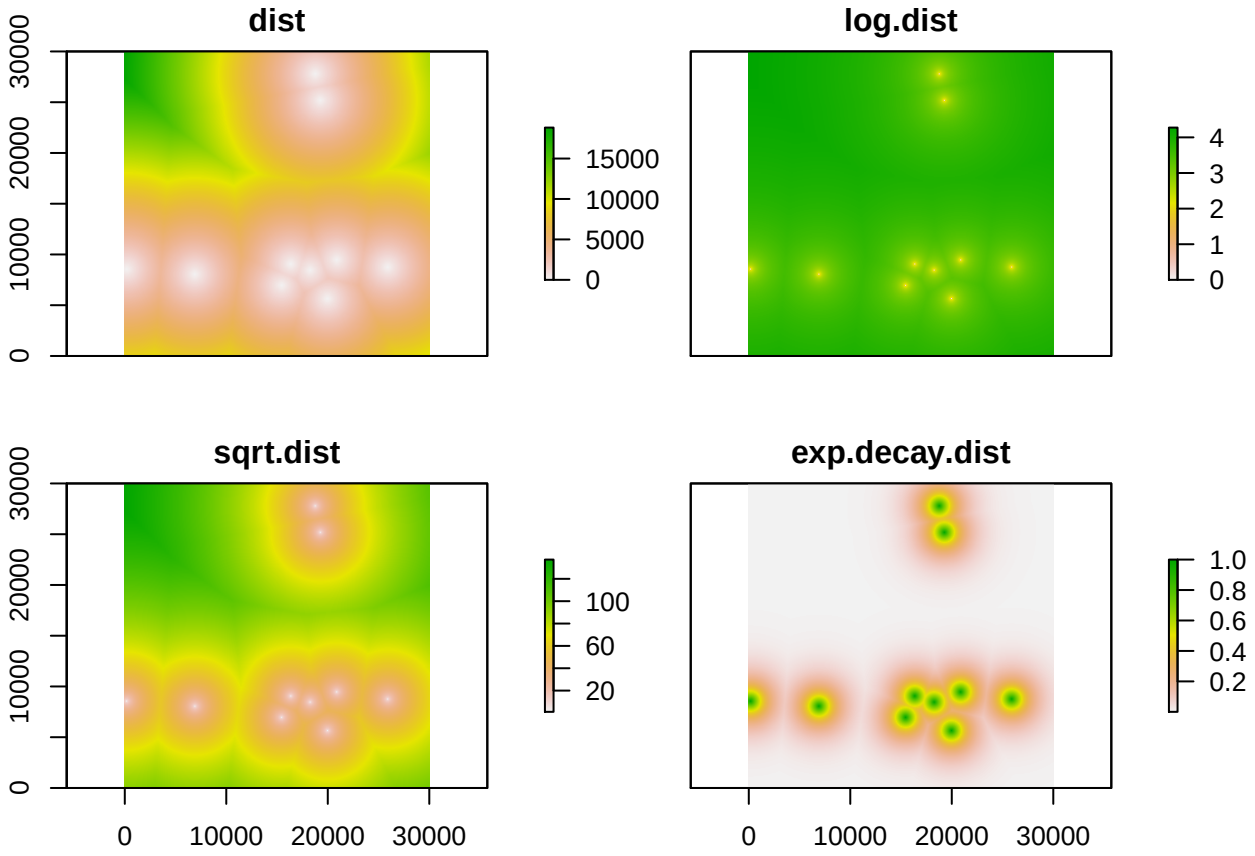
```
# calculate distance and density at multiple scales for one input
scales <- c(250, 500, 1000, 2500, 5000) / 2 # scales for Gaussian filter
dist_dens1 <- calc_dist_dens(landscapes[[2]],
  type_density = "Gauss", scale = scales,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
plot(dist_dens1,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



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, or when an exponential decay distance is used instead. It also affects how distances and densities correlate, as we'll see.

```
# transformed distance
log_dist <- calc_dist(landscapes[[2]],
  transform_dist = "log", log_base = 10,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
sqrt_dist <- calc_dist(landscapes[[2]],
  transform_dist = "sqrt",
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
exp_decay_dist <- calc_dist(landscapes[[2]],
  transform_dist = "exp_decay",
  exp_hl = 1000,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)
```

```r
# combine
dists <- raster::stack(dist_dens1[[1]], log_dist, sqrt_dist, exp_decay_dist)

# plot
names(dists) <- c("dist", "log-dist", "sqrt-dist", "exp-decay-dist")
plot(dists, nc = 2)
```

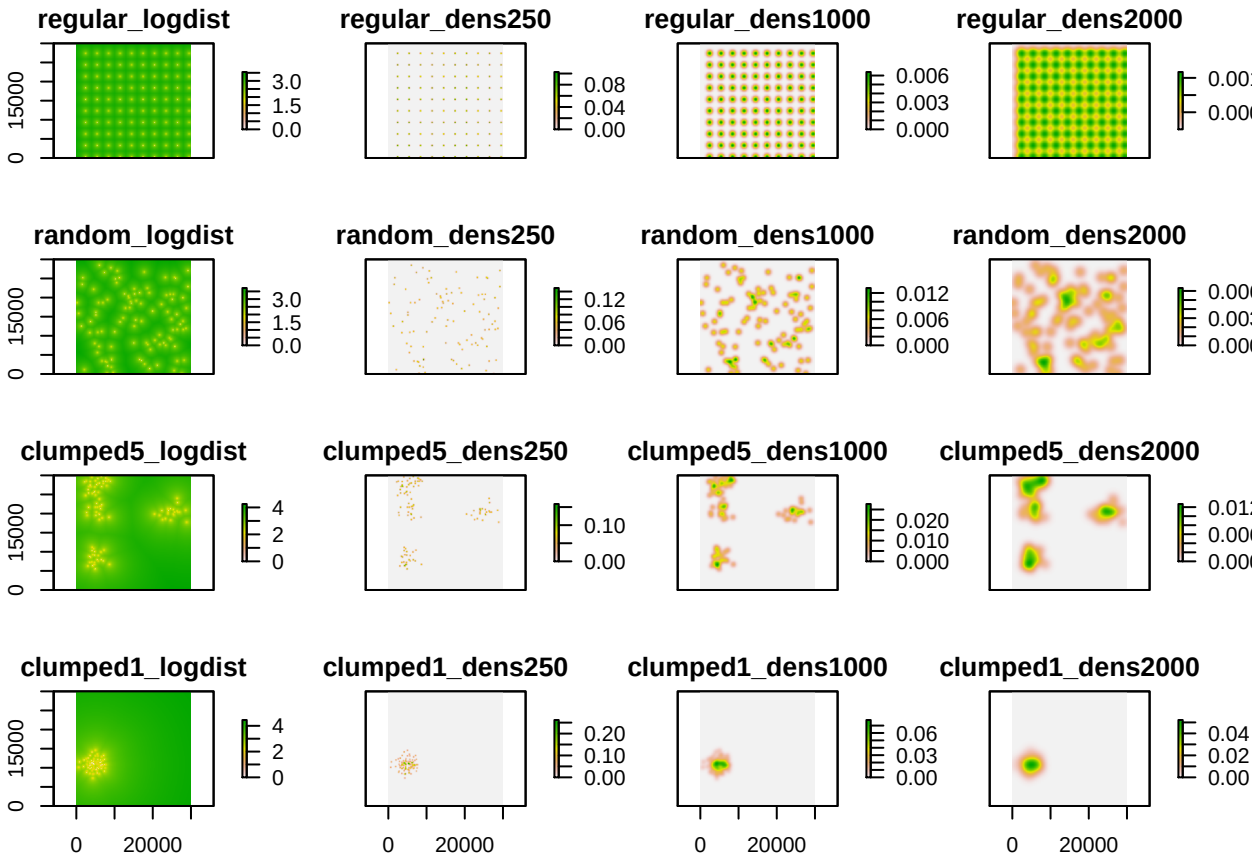


Visual comparison

Now we compare these variables visually for the different spatial point patterns, using the log-transformed distance, for now. We select the log-distance since it is a measure commonly used in ecological studies to include distance to the nearest features into the statistical models.

```r
# redefine scales
scales <- c(250, seq(500, 5000, by = 500)) / 2 # scales for Gaussian filter

# calculate distance and density at multiple scales for each input
dist_dens <- purrr::map(landscapes, calc_dist_dens,
  type_density = "Gauss", scale = scales,
  # type_density = "circle", scale = scales,
  transform_dist = "log", log_base = 10,
  extent_x_cut = c(0, ext), extent_y_cut = c(0, ext)
)

# change names according to the scenario and the variable
vars <- c("logdist", paste0("dens", scales * 2))

for (i in 1:length(dist_dens)) {
```

```
  names(dist_dens[[i]]) <- paste(strsplit(names(dist_dens)[i], "_")[[1]][1], vars, sep = "_")
}

# slice a few of them to plot - for
slices <- c(1, 2, 4, 6) # as.vector(outer(c(1,2,4,6), (0:3)*6, FUN = "+"))
maps100 <- dist_dens[13:16]
stk <- stack(maps100[[1]][[slices]], maps100[[2]][[slices]], maps100[[3]][[slices]], maps100[[4]][[slices]]
plot(stk, nc = 4)
```

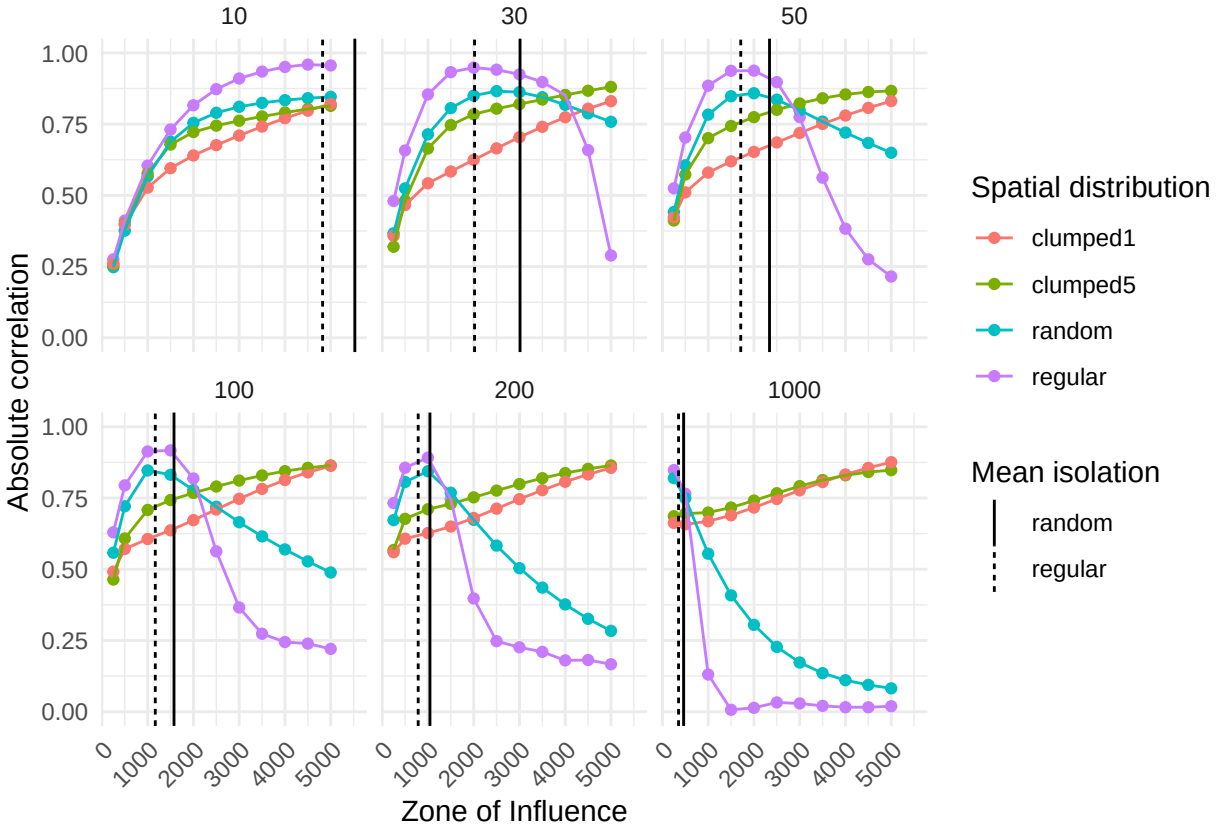


Calculate correlations between distance and cumulative impact (density)

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

# put distances with rasters
scenarios_analysis_full <- dist_scenarios %>%
```

```
  tibble::as_tibble() %>%
  dplyr::mutate(
    rasts = dist_dens,
    rasts_df = purrr::map(rasts, as.data.frame),
    corr_raw = purrr::map(rasts_df, lsr::correlate),
    corr_list = purrr::map(corr_raw, function(x) x$correlation[1, 2:12]),
    corr_df = purrr::map(corr_list, tibble::enframe, name = "dens", value = "corr")
  ) %>%
  tidyr::unnest(corr_df) %>%
  dplyr::mutate(
    variable = gsub(".*_", "log-dist_", dens),
    scale = as.numeric(gsub("\\D+", "", variable))
  )

scenarios_analysis <- scenarios_analysis_full %>%
  dplyr::select(-c(rasts:corr_list)) %>%
  dplyr::mutate(n_features = factor(n_features))

# plot
dat_isol <- scenarios_analysis %>%
  dplyr::filter(patch_width == 5, spatial_dist %in% c("random", "regular")) %>%
  dplyr::group_by(n_features, spatial_dist) %>%
  dplyr::summarise(mean_iso = unique(mean_isolation))

scenarios_analysis %>%
  dplyr::filter(patch_width == 5) %>%
  ggplot(aes(scale, abs(corr), color = spatial_dist)) +
  geom_point() +
  geom_line() +
  geom_vline(data = dat_isol, aes(xintercept = mean_iso, linetype = spatial_dist)) +
  ylim(0, 1) +
  facet_wrap(~n_features) +
  labs(
    x = "Zone of Influence",
    y = "Absolute correlation",
    color = "Spatial distribution",
    linetype = "Mean isolation"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
```



We can see that the pattern changes greatly with both the total number of features and their spatial distribution.

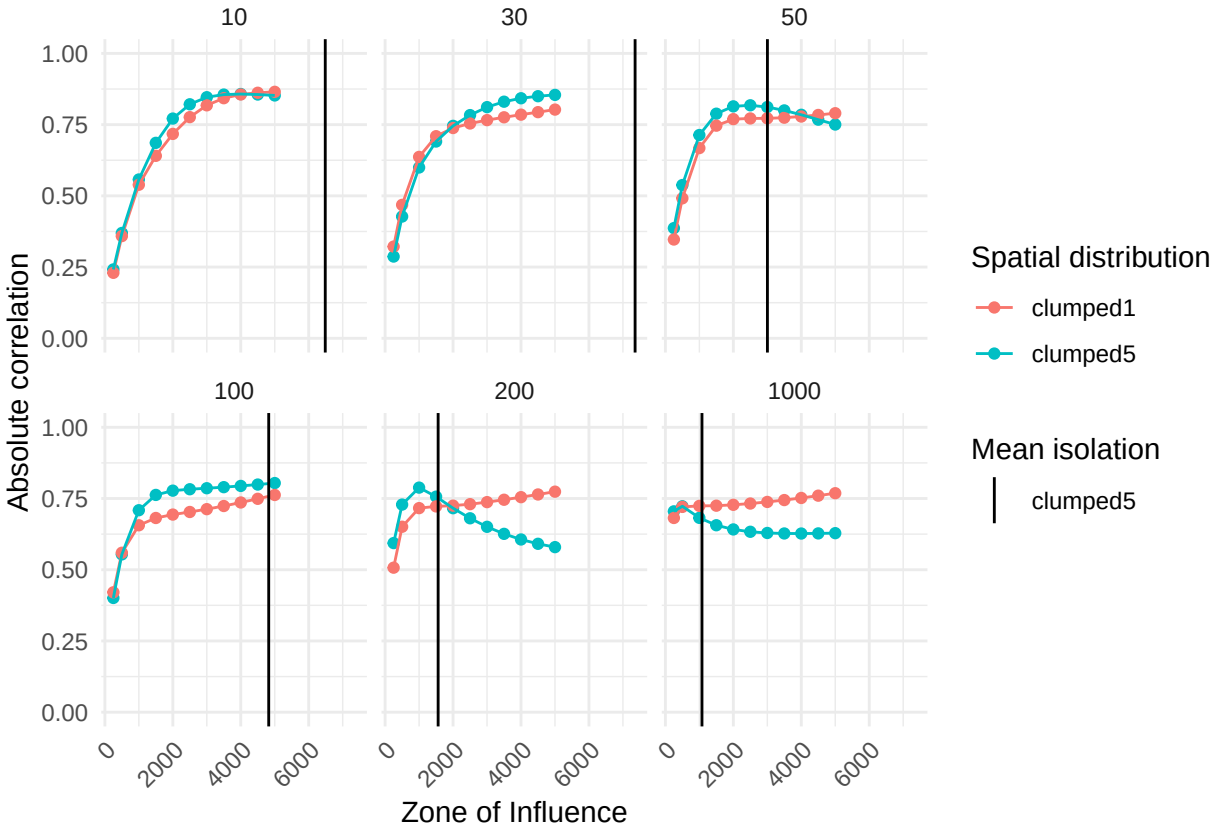
For clumped distribution of features and when the patches of features are small (radius = 5% of the study area extent, in the figure above), the point features are quite apart from each other in general, so as the ZoI increases the cumulative impact (density) measure gets more correlated to the log-distance

That is also true for regular and random distribution of features when the density of features is small (n = 10) and, consequently, their mean isolation is high. As the density density of features increase, and the mean isolation between points decreases, the cumulative impact (density) measure starts to present a peak when ZoI is close to the mean isolation between features in the landscape - compare the peaks in the plots above with the mean isolation for the random and regular scenarios. For larger ZoI valuws, the correlation drops very quickly, especially for the regular distribution scenario.

This means that, when comparing log-distances to the cumulative impact measure for random and regular distribution of features, they only represent similar spatial variation when the the mean isolation of the features is around the same size of their ZoI. This is still true for clumped distribution when the radius of the "patches" of features is larger, such as in the figure below (radius = 15% of the landscape extent). For more than one cluster, the correlation between log-distance and the cumulative impact measure also starts to show a peak related to the average distance between features. The only case when this changes is when there is a single small clump of features.

```
# plot
dat_isol <- scenarios_analysis %>%
  dplyr::filter(patch_width == 15, spatial_dist %in% c("clumped5")) %>%
  dplyr::group_by(n_features, spatial_dist) %>%
  dplyr::summarise(mean_iso = unique(mean_isolation))

scenarios_analysis %>%
  dplyr::filter(patch_width == 15, spatial_dist %in% c("clumped5", "clumped1")) %>%
  ggplot(aes(scale, abs(corr), color = spatial_dist)) +
  geom_point() +
  geom_line() +
```

```
  geom_vline(data = dat_isol, aes(xintercept = mean_iso, linetype = spatial_dist)) +
  ylim(0, 1) +
  facet_wrap(~n_features) +
  labs(
    x = "Zone of Influence",
    y = "Absolute correlation",
    color = "Spatial distribution",
    linetype = "Mean isolation"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
```



Discussion

Here we have compared how different is the spatial variation when one computes log-distances and CI (density) from point infrastructure. To do so, we simulated point features in space in different densities and spatial distributions, calculated the log-distances and CI (density) at multiple scales (ZoI) and compared them through correlation.

We found the correlation between log-distance and CI is intrinsically correlated with the mean isolation of the point features in space.

When all point infrastructure features are limited to a single, small cluster, so that their mean isolation is very high, log-distance and CI are little correlated, and their correlation increases as the scale (ZoI) increses. In this case, they start to represent a more similar variation when the scale/ZoI is high. This also holds true when there more then one cluster of points but their radius is very small (scenario with 5 small clumps shown above), a case in which the mean isolation is still too high.

In all other cases - either a regular or random distribution of points, or a clumped distribution with more clusters or larger clusters - the log-distance is only highly correlated with the CI (density) when the scale (ZoI) is at the same order of magnitude of the mean isolation of the point features.

Given that the determination of the scale/ZoI is an empirical question and varies according to the ecological system and context of the study area, we argue here that in very limited cases using the distance to the nearest feature might be equivalent to using CI or the density of infrastructure. In all other cases, the CI might represent a wider range of spatial variation metrics, that in turn should better represent the cumulative effect of the multiple infrastructure in space.