Simulating scenarios: comparing log-distance to the nearest feature to the cumulative influence measure

Bernardo Niebuhr, Bram van Moorter

05 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 influence measure (CumInf) - proportional to the density of infrastructure features - might better represent the cumulative effect of multiple features in the landscape than considering only the distance from 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 CumInf of the features, at multiple scales/ZoI, and assess when and how these variables might represent different sources of spatial variation.

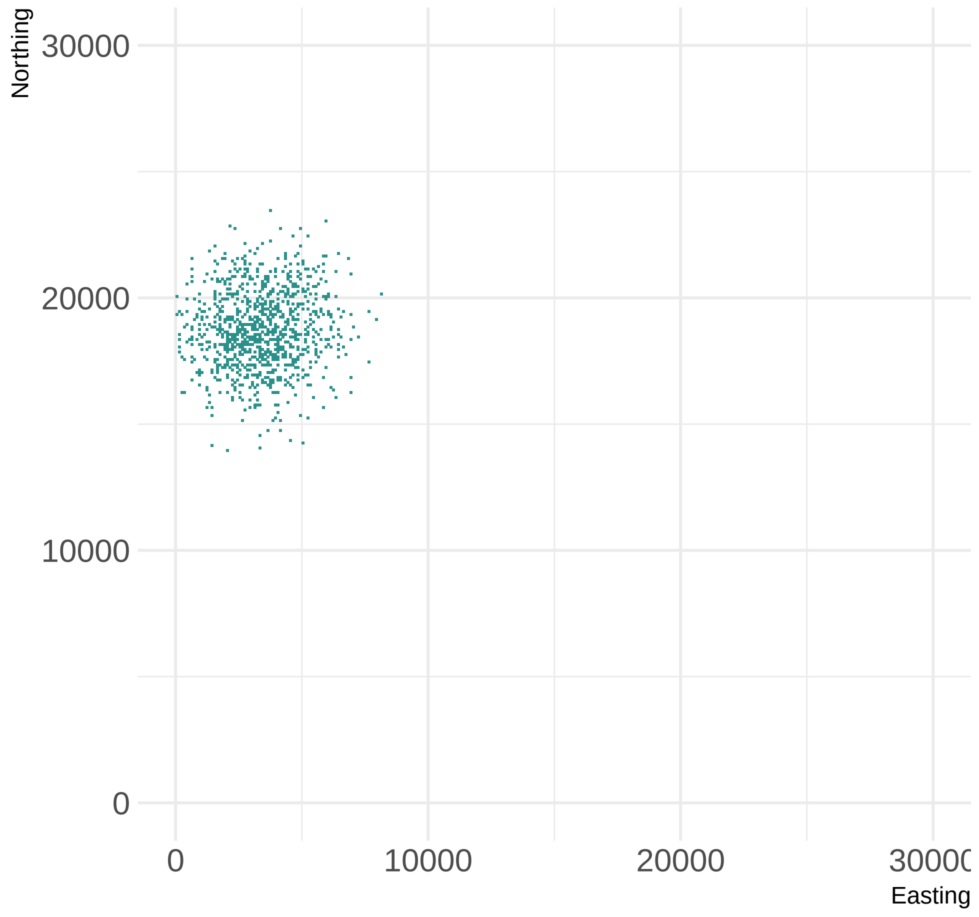
Simulate landscapes

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 spatial packages it depends on and other packages for data manipulation and plotting.

```r
# Load packages

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

# spatial packages
library(mobsim)
library(raster)
library(landscapetools)
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 or cluster (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")
```

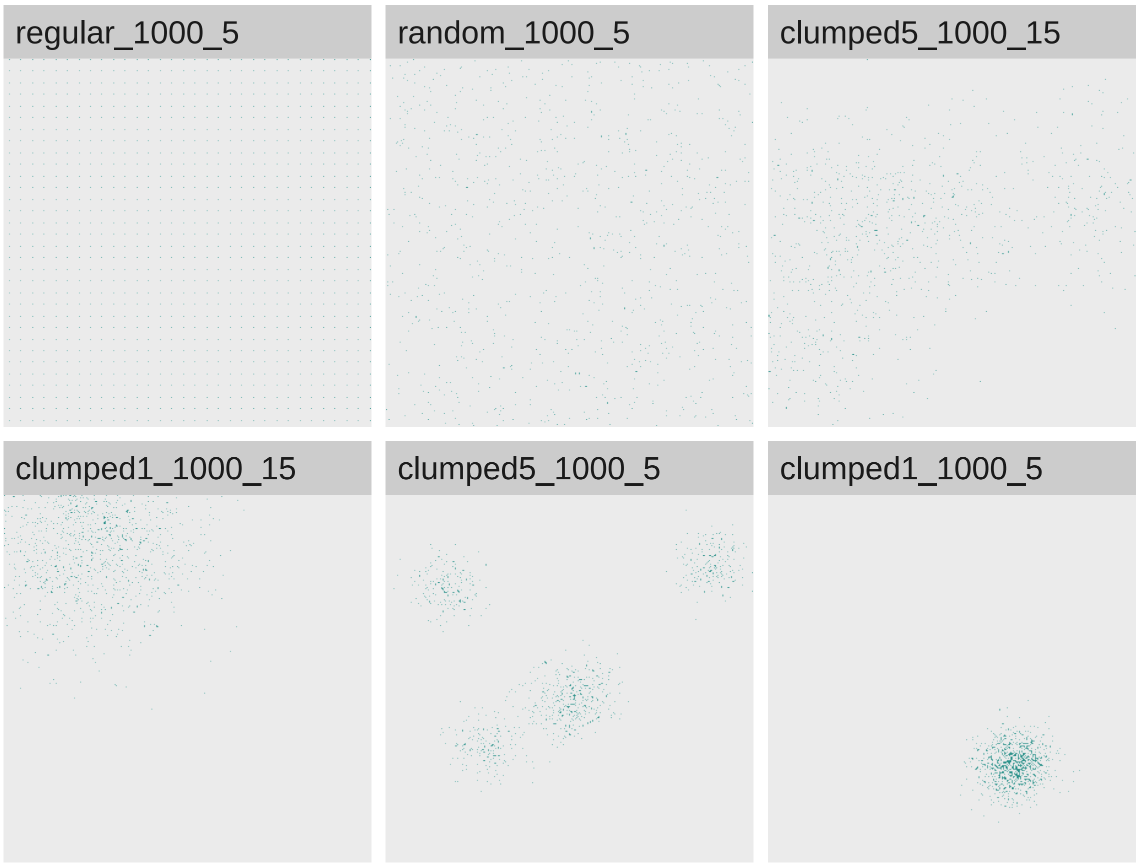


We can now simulate landscapes with different patterns. We start with 3 spatial distribution of points: regular, random, and clumped distribution of points. For the clumped distribution, we use two scenarios, where the point features are spread in either 5 or only 1 focal cluster, and we choose two possible radii of these cluster, corresponding to a more and less clumped point distribution (radius = 5 and 15% of the extent of the landscape). Each scenario is simulated with 10, 30, 50, 100, 200, and 1000 points, so that we have 48 scenarios in total.

```r
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 or clusters for clumped
wd <- c(0.05, 0.15) * ext # width of the "clusters"

# 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
```

As an example, here we visualize the scenarios with 1000 features, for cluster with small radius (5% of the landscape extent).

```r
# 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[c(21:22, 47:48, 23:24)] %>%
  purrr::map(raster::crop, y = extent(0, ext, 0, ext)) %>%
  landscapetools::show_landscape()
```



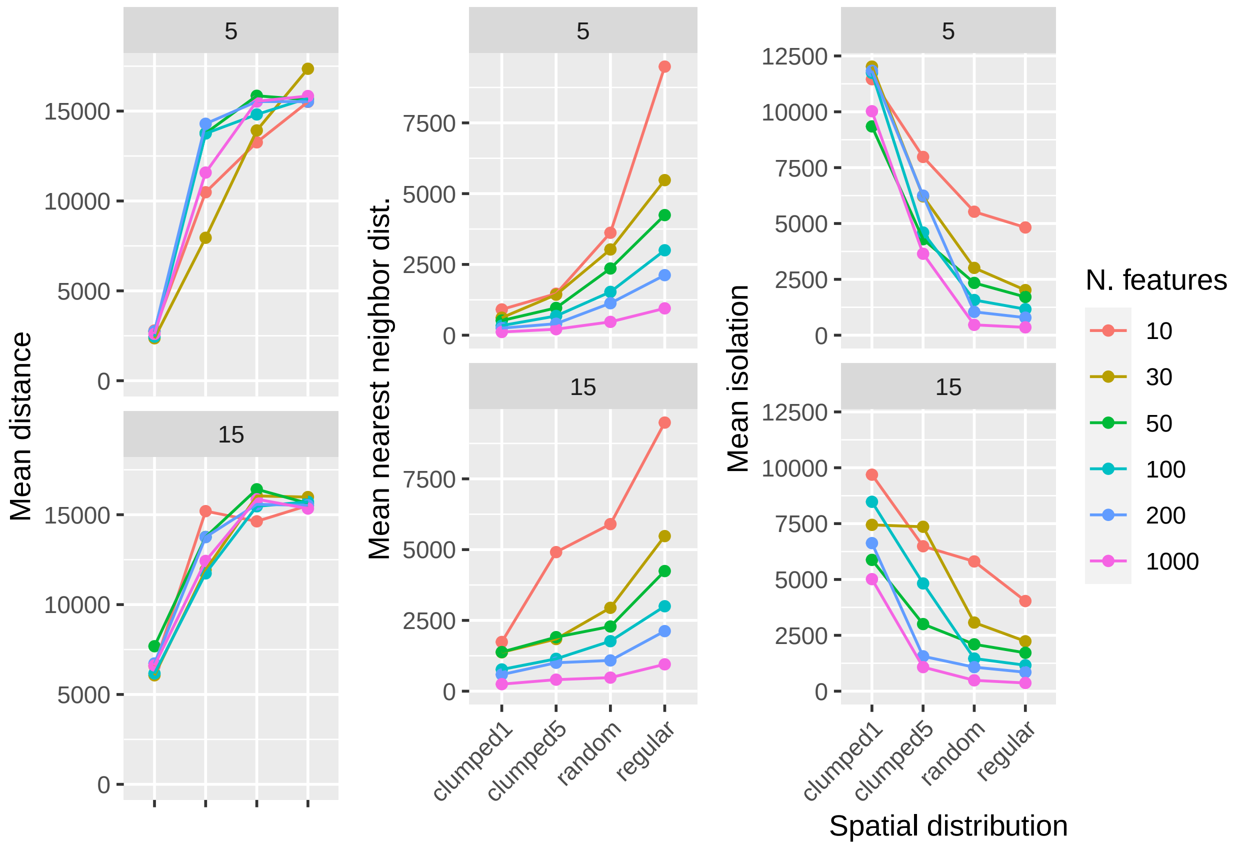
We also store some variables related the distance between points for each of these scenarios. These variables are: (i) the average distance between points, (ii) the average nearest neighbor distance, and (iii) the average isolation. The average isolation is defined here as the mean nearest neighbor distance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

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

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

head(dist_scenarios)
```

```
##         scenario spatial_dist n_features cluster_radius 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
##   mean_isolation
## 1       4822.949
## 2       5527.555
## 3       7980.069
## 4      11457.262
## 5       2018.935
## 6       3013.295
```

Just to understand how these distance and isolation measures change among scenarios, we plot them below.

```r
# relationship between scenarios
ds1 <- dist_scenarios %>%
  ggplot(aes(x = spatial_dist, y = mean_dist, color = factor(n_features))) +
  geom_point() +
  geom_line(aes(group = n_features)) +
  facet_wrap(~cluster_radius, ncol = 1) +
  ylim(0, NA) +
  labs(
    x = "",
    y = "Mean distance",
    color = "N. features"
  ) +
  theme(axis.text.x = element_blank())
ds2 <- dist_scenarios %>%
  ggplot(aes(x = spatial_dist, y = mean_nndist, color = factor(n_features))) +
  geom_point() +
  geom_line(aes(group = n_features)) +
  facet_wrap(~cluster_radius, ncol = 1) +
  ylim(0, NA) +
  labs(
    x = "",
    y = "Mean nearest neighbor dist.",
    color = "N. features"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
ds3 <- dist_scenarios %>%
  ggplot(aes(x = spatial_dist, y = mean_isolation, color = factor(n_features))) +
  geom_point() +
  geom_line(aes(group = n_features)) +
  facet_wrap(~cluster_radius, ncol = 1) +
  ylim(0, NA) +
  labs(
    x = "Spatial distribution",
    y = "Mean isolation",
    color = "N. features"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))

# combine
ggpubr::ggarrange(ds1, ds2, ds3,
  nrow = 1, common.legend = T,
  legend = "right"
)
```



As expected, the mean distance between points and the mean distance to the nearest neighbor point increase when we depart from a more clumped scenario to a random or regular distribution. They represent the distance relationship between points only, regardless of the size of the landscape

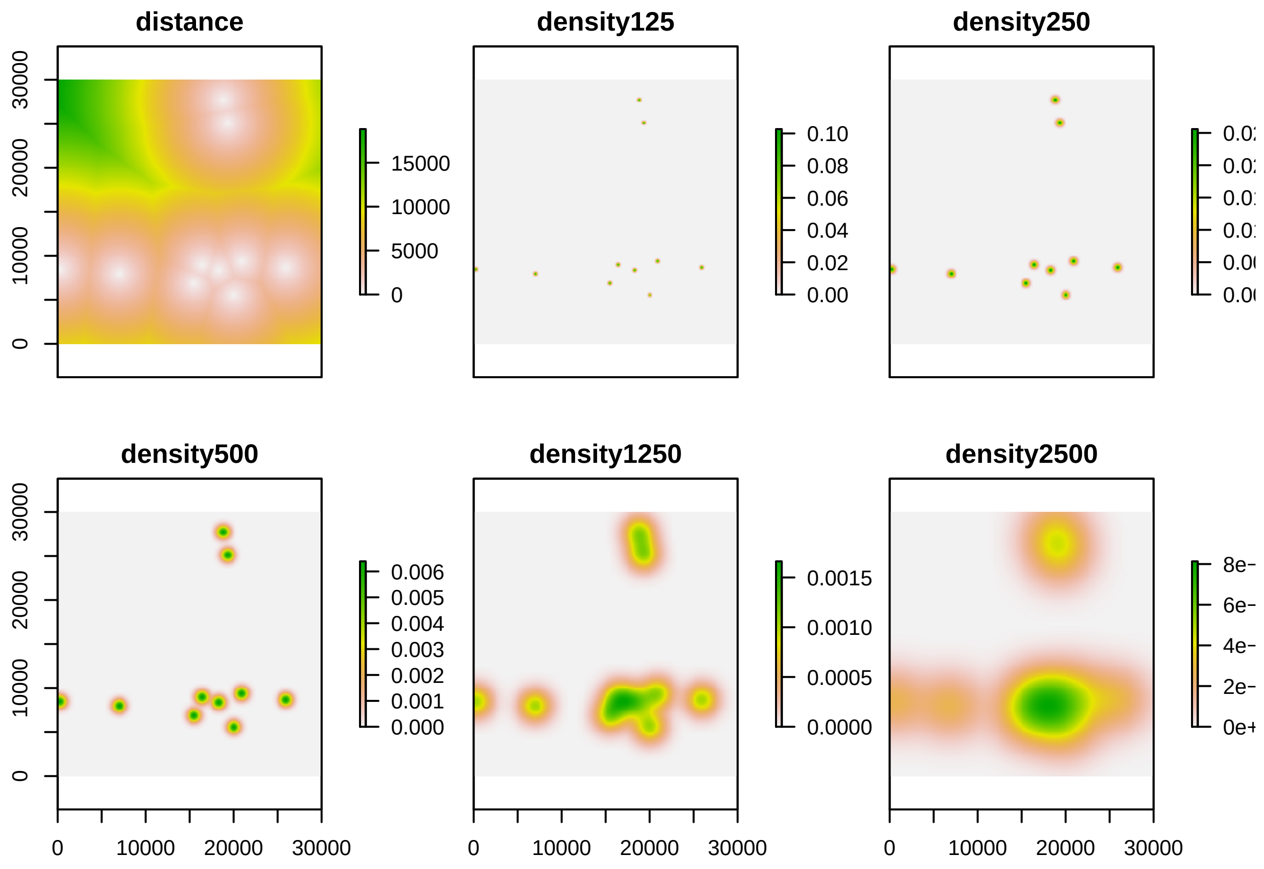
The mean isolation follows an opposite pattern: it decreases when we depart from clumped to random or regular distribution of points. In all cases, even though the general pattern is qualitatively the same, quantitatively these quantities vary with the number of features - obviously the distances or isolation are larger when the density of points is smaller.

We take the last measure - mean isolation - to represent the relationship between the spatial distribution of points and the correlation between log-distances and cumulative influence (density) measures.

Calculate distance and cumulative influence

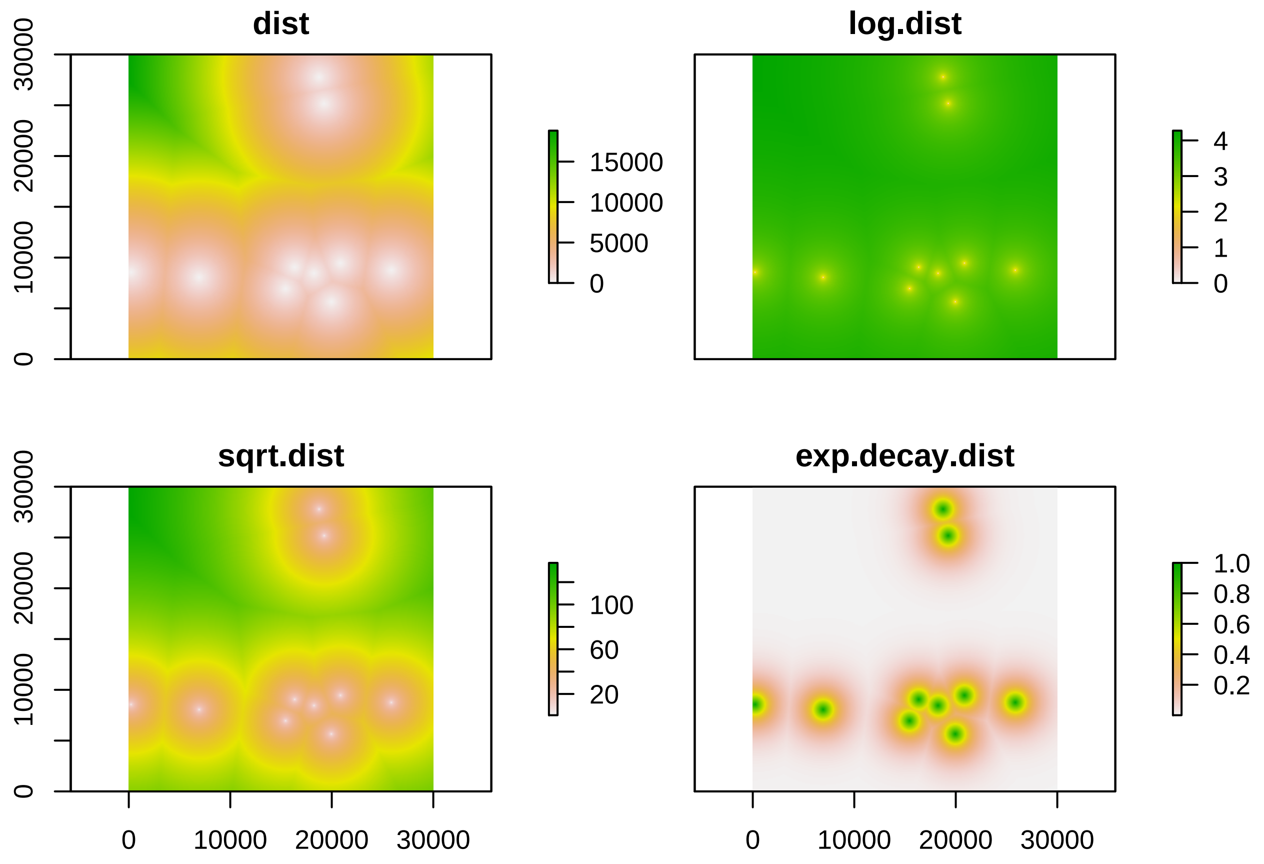
First we illustrate, for one of those scenarios, how the maps change as we calculate either the distance to the nearest feature or the CumInf (or density) of features at different scales. For illustration purposes, we use here a Gaussian filter to calculate CumInf, where the scale corresponds to the standard deviation σ of the Gaussian distribution, and the moving window has a size of $3 \cdot \sigma$.

```r
# 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.

```
# 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)
)

# 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)
```

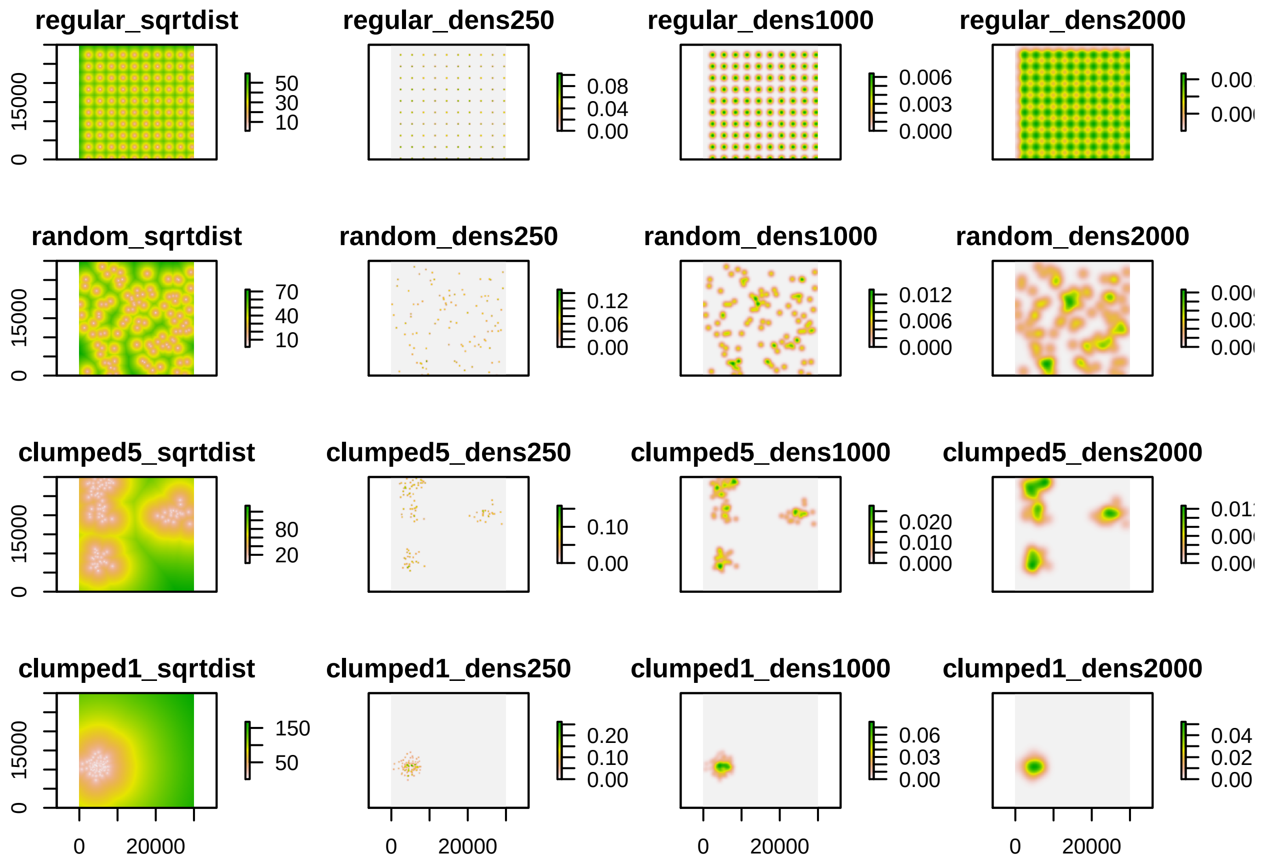


Visual comparison

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 = "sqrt",
  # 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("sqrtdist", 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]
```

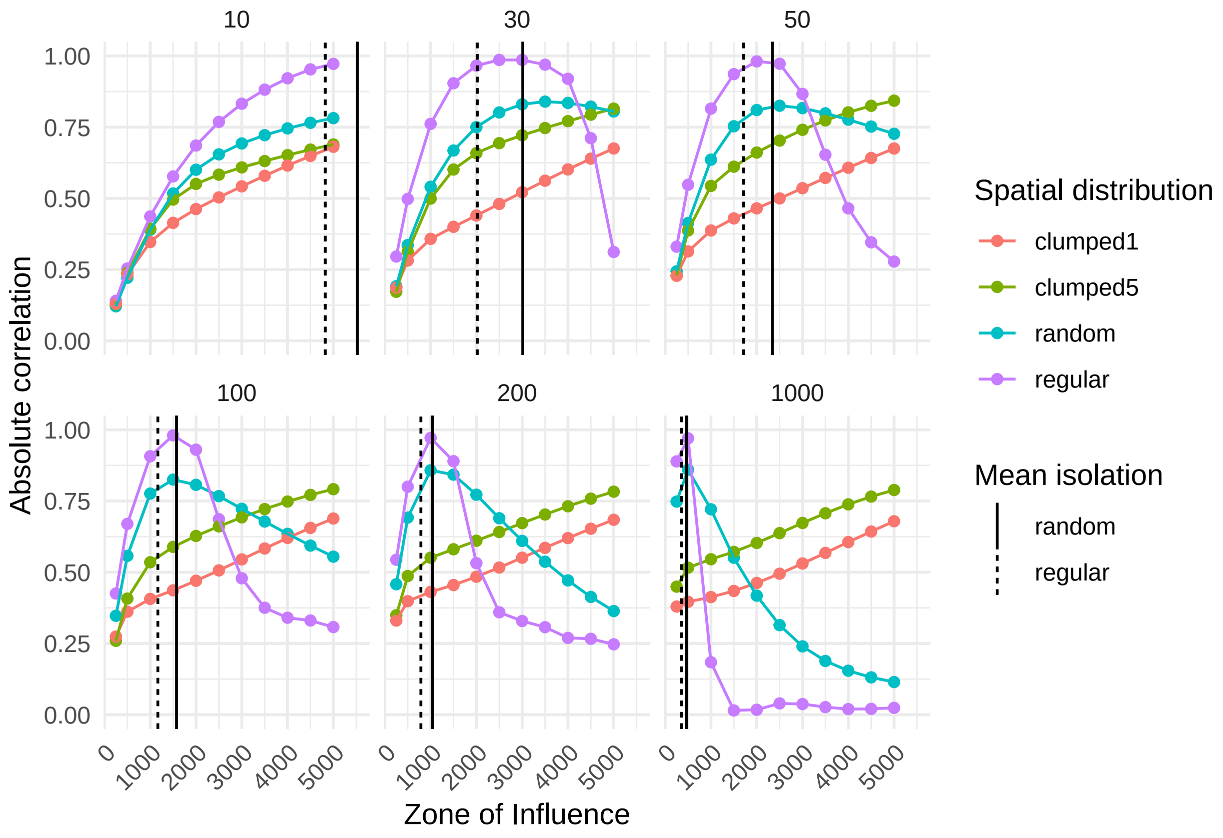
```
stk <- raster::stack(
  maps100[[1]][[slices]], maps100[[2]][[slices]],
  maps100[[3]][[slices]], maps100[[4]][[slices]]
)
plot(stk, nc = 4)
```



Calculate correlations between distance and cumulative influence

Now we are going to check how correlated these variables are in space, to evaluate how their ecological interpretations might differ.

```
# put distances with rasters
scenarios_analysis_full <- dist_scenarios %>%
  tibble::as_tibble() %>%
  dplyr::mutate(
    rasts = dist_dens,
    # rasters as data.frame
    rasts_df = purrr::map(rasts, as.data.frame),
    # correlation between pairs of rasters
    corr_raw = purrr::map(rasts_df, lsr::correlate),
    # select only correlation between log-dist and CumInf at different ZoI
    corr_list = purrr::map(corr_raw, function(x) x$correlation[1, 2:12]),
    # transform into tibble
    corr_df = purrr::map(corr_list, tibble::enframe, name = "dens", value = "corr")
  ) %>%
  tidyr::unnest(corr_df) %>%
  # organize names
  dplyr::mutate(
    variable = gsub(".*_", "log-dist_", dens),
    scale = as.numeric(gsub("\\D+", "", variable))
  )

# select only relevant variables for plotting
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(cluster_radius == 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(cluster_radius == 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 plot here the the absolute value of the Pearson correlation between values of the log-distance and the CumInf at mutltiple scales/ZoI, and highlight in each plot the average isolation between points in the random and regular scenarios (vertical lines). We use the absolute correlation because we are interested in the degree of correlation but not in the signal of the correlation. The plot above is for small clusters of point features (radius = 5% of the study area extent).

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 cluster of features are small (as in the figure above), the point features are in general quite apart from each other, so as the ZoI increases the cumulative influence (density) measure gets more correlated to the log-distance, regardless of the density of features in space.

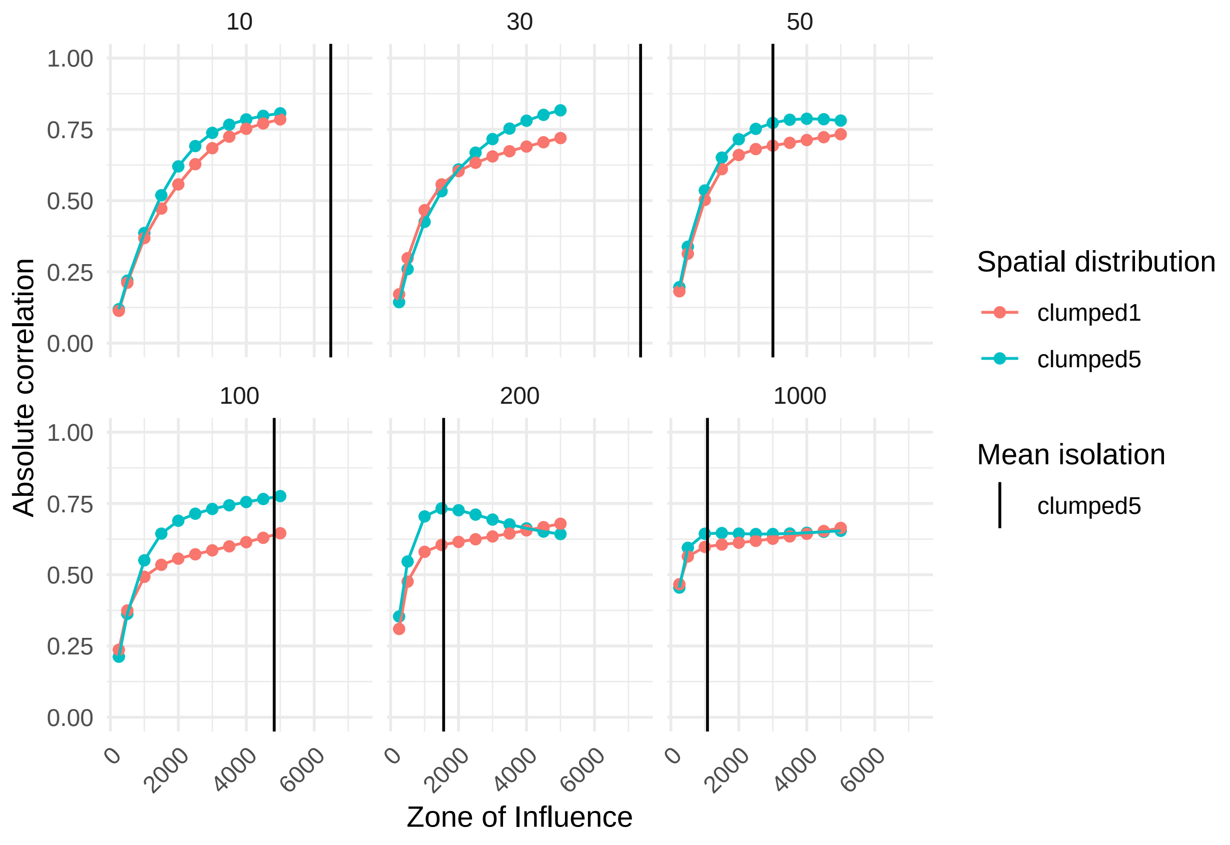
That is also true for regular and random distribution of features when the density of features is small (n = 10) and, consequently, their mean isolation is also high. As the density of features increase and the mean isolation between points decreases, the CumInf (density) measure starts to present a 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 values, the correlation drops very quickly, especially for the regular distribution scenario.

This means that, when comparing log-distances to CumInf 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 also true for clumped distribution when the radius of the clusters of features is larger, such as in the figure below (radius = 15% of the landscape extent).

When the features are distributed in more than one cluster, the correlation between log-distance and the cumulative influence measure also starts to show a peak related to the average distance between features. As this radius increases, the average isolation between features decreases and the results of the clumped distributions gets closer to the one for random distribution.

The only case when this changes is when there is a single small clump of features.

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

scenarios_analysis %>%
  dplyr::filter(cluster_radius == 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 cumulative influences (or densities) from point infrastructure. To do so, we simulated a different number of point features in space spread according to different and spatial distributions, calculated the log-distances and CumInf at multiple scales (ZoI) and compared them through correlation.

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

When all point infrastructure features are limited to a single, small cluster, (e.g. one single village or wind park in the landscape) so that their mean isolation is very high, log-distance and CumInf are little correlated, and their correlation increases as the scale (ZoI) increases. 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 relatively small (scenario with 5 small clumps shown above), a case in which the mean isolation is still 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 CumInf (density) when the scale (ZoI) is at the same order of magnitude of the mean isolation to 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 CumInf or the density of infrastructure. In all other cases, the CumInf might represent a wider range of spatial variation metrics, that in turn should better represent the cumulative effect of the multiple infrastructure in space.