Simulating scenarios: Comparing distance to the nearest feature to the sum of distances, using on exponential decay

Bernardo Niebuhr, Bram van Moorter

09 December, 2021

Contents

Introduction

Our aim here is to compare how distance and density/cumulative impact measures (CI) compare when calculated exactly with the same basis. We use here exponential decay distance as an example since one can more directly extract thresholds (given its half-life and decayment nature) or measures that in some way represent the ZoI or scale of effects of a given infrastructure.

To do so, we do the following: 1) create a set of points representing the locations of point infrastructure, according to different spatial distributions; 2) calculate the exponential decay distance to each of these point features and put all measures in a RasterStack; 3) calculate the (i) minimum of all distance measures (equivalent to the distance to the nearest feature) and the (ii) sum of all distance measures (equivalent to the CI); 4) compare the results.

Should we also include the comparison with the filter using focal here? Yes

Simulate landscapes

We start by creating points and separating them in different rasters, so that it is possible to calculate distance to each of them. For simplicity, here we keep constant the density of points (n = 100) and the radius of the feature clusters (radius = 15% of the landscape size).

We simulate points with 3 spatial distribution patterns: regular, random, and clumped. For the clumped distribution, we use two scenarios, where the point features are spread in either 5 or only 1 focal cluster. So we have four scenarios in a gradient of clumpiness.

After creating the points, we calculate the exponential decay distance for each of the 100 points, separately, and then compute two variables: - the nearest neighbor distance, by calculating the maximum distance over the series of distance rasters; - the cumulative influence, by calculating the sum of the distances of each raster in the series of distance rasters.
For the nearest neighbor distance, we use here the maximum instead of the minimum because it corresponds to decay distance (it decreases with distance).

```
# Load packages

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

```r
# spatial packages
library(mobsim)
library(raster)
library(landscapetools)
library(sf)
library(spatstat)

# oneimpact package
library(oneimpact)
```

```r
set.seed(12)

# random, gradient, single center, multiple centers
methods <- c("regular", "random", "mobsim")
name <- c("regular", "random", "clumped5_15", "clumped1_15", "clumped5_5", "clumped1_5")
nfeat <- c(100) # 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(0.15, 0.05) * ext # width of the "clusters"

# ZoI values
zoi_vals <- c(250, 500, 750, 1000, 1250, 1500, 2000, 3000, 4000, 5000)

# 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(desc(width), n_features, method) %>%
  dplyr::slice(1:4, 7:8)
```

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

# coordinates
pts_coords <- pts %>%
  purrr::map(first)

# raster with points
pts_all_rast <- pts %>%
  purrr::map(~ .[[2]])

# crate scenarios, separating each point in a different
# raster, to calculate distances
create_scenarios <- tibble::tibble(
  spatial_dist = name,
```

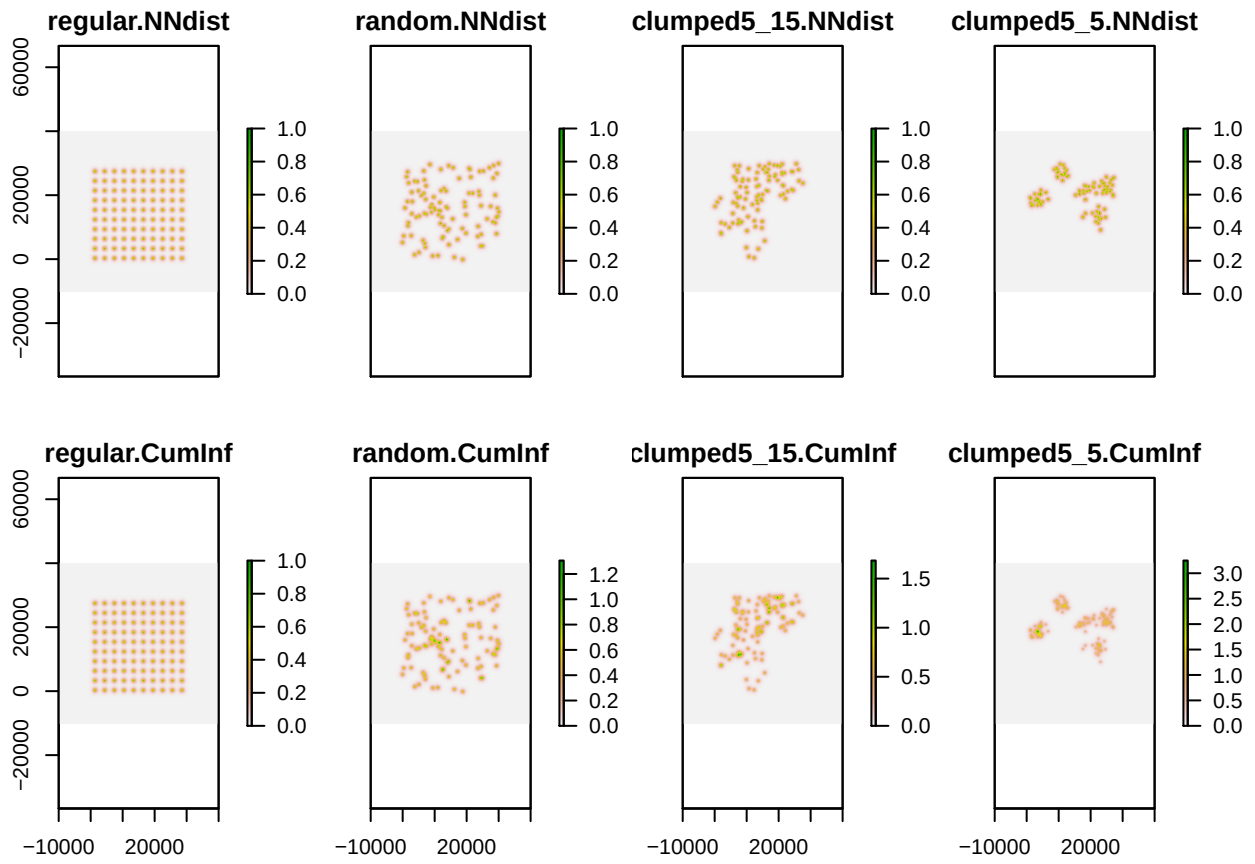
```
  n_features = rep(nfeat, length(name)),
  # mean isolation
  mean_isolation = map_dbl(pts_coords, mean_isolation, n_rand = 150, ext = c(0, ext)),
  # mean distance between points
  mean_dist = map_dbl(pts_coords, function(x) mean(dist(x))),
  # mean nearest neighbor distance
  mean_nndist = map_dbl(pts_coords, function(x) mean(spatstat.geom::nndist(x)))
  # points coordinates, all together
  points_all = pts_coords,
  # separate points
  points_sep = purrr::map(points_all, function(x)
    purrr::map(1:nfeat, function(a, b) slice(b, a), b = x)),
  # rasterize points
  pt_sep_rasterized = purrr::map(points_sep, function(x)
    purrr::pmap(list(point_coordinates = x), set_points, res = res,
                extent_x = c(0, ext), extent_y = c(0, ext),
                buffer_around = 10000)),
  # pick only the raster
  pt_sep_rast = purrr::map(pt_sep_rasterized, function(x)
    purrr::map(x, function(z) z[[2]]))
)

# calculate distances to each point using several ZoI/scales
outerlist <- list()
for(i in 1:length(create_scenarios$pt_sep_rast)) {
  innerlist <- list(list(), list(), list(), list(), list(),
                    list(), list(), list(), list(), list())
  for(j in 1:length(create_scenarios$pt_sep_rast[[i]])) {
    print(paste(i, j))
    # calc dist for different parameters of exp_decay
    r <- create_scenarios$pt_sep_rast[[i]][[j]]
    parms <- expand.grid(transform_dist = "exp_decay",
                         exp_hl = zoi_vals) %>%
      tibble::as_tibble()
    r_dist <- parms %>% pmap(calc_dist, points = r)
    # put each ZoI in a different list
    innerlist[[1]][[j]] <- r_dist[[1]]
    innerlist[[2]][[j]] <- r_dist[[2]]
    innerlist[[3]][[j]] <- r_dist[[3]]
    innerlist[[4]][[j]] <- r_dist[[4]]
    innerlist[[5]][[j]] <- r_dist[[5]]
    innerlist[[6]][[j]] <- r_dist[[6]]
    innerlist[[7]][[j]] <- r_dist[[7]]
    innerlist[[8]][[j]] <- r_dist[[8]]
    innerlist[[9]][[j]] <- r_dist[[9]]
    innerlist[[10]][[j]] <- r_dist[[10]]
  }
  outerlist[[i]] <- innerlist
}

# stack each set of 100 dist maps, for each scenario
stack_list <- map(outerlist, function(x) map(x, raster::stack))
length(stack_list)
length(stack_list[[1]])

# simplify tibble with scenarios and distances
```

```r
scenarios_dist <- create_scenarios %>%
  dplyr::select(-c(points_sep, pt_sep_rasterized, pt_sep_rast)) %>%
  dplyr::mutate(
    parms = purrr::map(spatial_dist, function(x) {
      tibble::tibble(exp_hl = zoi_vals) })
  ) %>%
  tidyr::unnest(parms)

# calculate nearest neighbor distance (max over the raster)
# and cumulative influence (sum over the raster)
scenarios_dist$rast_dist <- list(NA)
scenarios_dist$rast_nndist <- list(NA)
scenarios_dist$rast_cuminf <- list(NA)
scenarios_dist$rast_cuminf_d <- list(NA)
cont <- 1
for(i in 1:length(stack_list)) {
  for(j in 1:length(stack_list[[i]])) {
    scenarios_dist$rast_dist[[cont]] <- stack_list[[i]][[j]]
    scenarios_dist$rast_nndist[[cont]] <- raster::calc(stack_list[[i]][[j]], max, na.rm = T)
    scenarios_dist$rast_cuminf[[cont]] <- raster::calc(stack_list[[i]][[j]], sum, na.rm = T)
    # this is density, just to compare
    scenarios_dist$rast_cuminf_d[[cont]] <- raster::calc(stack_list[[i]][[j]], mean, na.rm = T)

    cont <- cont + 1
  }
}

# save
save(scenarios_dist, file = "../data/sm02_scenarios_dist_min_sum.rda")
```

Visualize the differences

Now we visualize the differences between the distance to the nearest feature and the cumulative influence. We start by looking at the scenarios considering a low ZoI or scale of influence (ZoI - here the exponential decay half-life = 250 m).

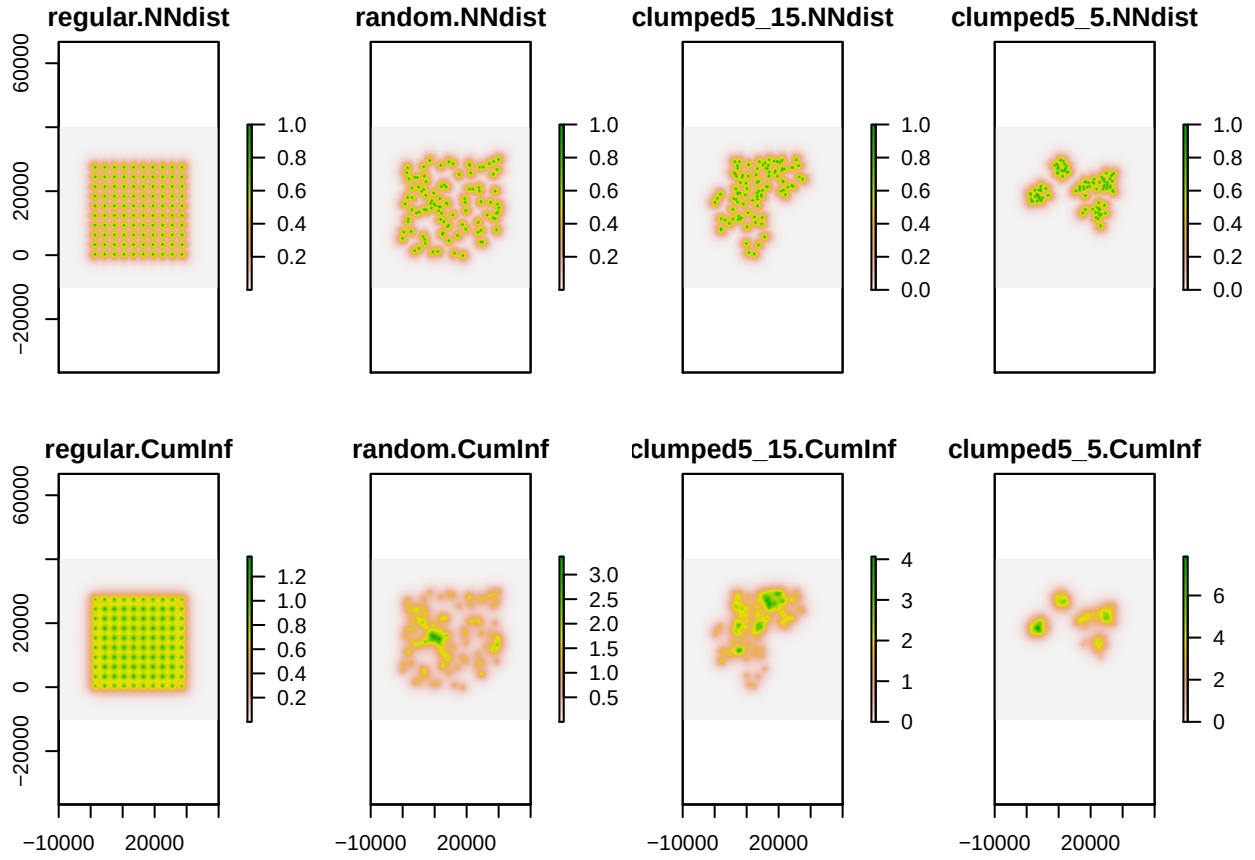
```r
which_scenarios <- c(0:2, 4)
indices <- which_scenarios * length(zoi_vals) + 1
to_viz <- c(scenarios_dist$rast_nndist[indices], scenarios_dist$rast_cuminf[indices]) %>%
  raster::stack()
names(to_viz) <- paste(rep(name[which_scenarios + 1], 2), rep(c("NNdist", "CumInf"), each = 4))

plot(to_viz, nc = 4)
```



We see that, since the ZoI is smaller than the average distance between points, the distance to the nearest feature and the cumulative influence are qualitatively similar for most cases - with some small area under cumulative influence differing for the clumped1 scenario.

Now we show the same, but considering a larger ZoI of 1000 m. We see that, in this case, the distance to the nearest feature is generally different from the cumulative influence measure.

```
which_scenarios <- c(0:2, 4)
indices <- which_scenarios * length(zoi_vals) + 3
to_viz <- c(scenarios_dist$rast_nndist[indices], scenarios_dist$rast_cuminf[indices]) %>%
  raster::stack()
names(to_viz) <- paste(rep(name[which_scenarios + 1], 2), rep(c("NNdist", "CumInf"), each = 4))

plot(to_viz, nc = 4)
```

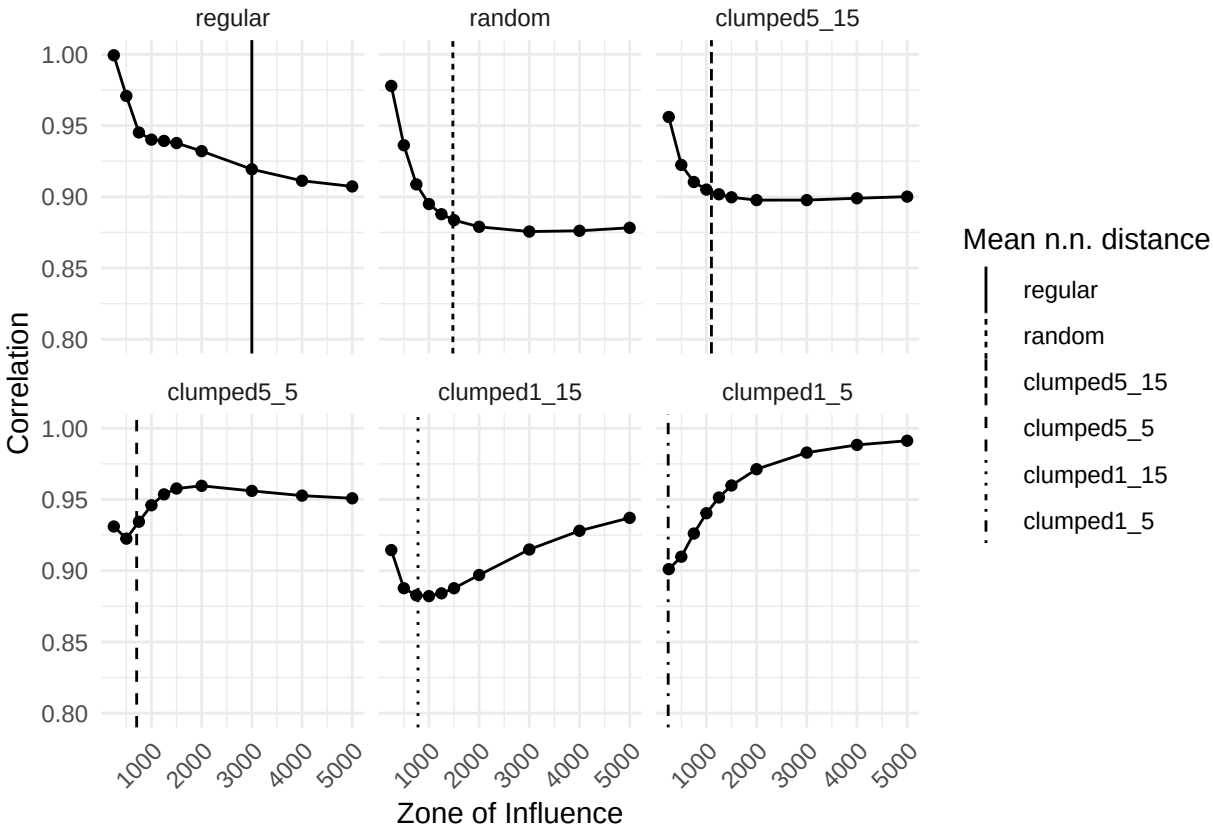


Correlation

Now we calculate the correlation between the distance to the nearest feature and the cumulative influence for each scenario and ZoI. We plot the correlation between the distance to the nearest feature and the cumulative influence for each ZoI or scale of influence, and compare the pattern of correlation to the nearest neighbor distance between points in each landscape scenario.

```
scenarios_analysis <- scenarios_dist %>%
  dplyr::mutate(corr = purrr::map2(rast_nndist, rast_cuminf, function(x, y) {
    cor(raster::values(x), raster::values(y))
  })) %>%
  tidyr::unnest(corr) %>%
  dplyr::mutate(
    spatial_dist = factor(spatial_dist, levels = name[c(1, 2, 3, 5, 4, 6)]),
    mean_dist = map_dbl(points_all, function(x) mean(dist(x))),
    mean_nndist = map_dbl(points_all, function(x) mean(spatstat.geom::nndist(x)))
  ) %>%
  dplyr::rename(scale = exp_hl)

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

```r
scenarios_analysis %>%
  ggplot(aes(scale, abs(corr))) + # , color = spatial_dist)) +
  geom_point() +
  geom_line() +
  geom_vline(data = dat_isol, aes(xintercept = mean_nndist, linetype = spatial_dist)) +
  ylim(0.8, 1) +
  facet_wrap(~spatial_dist) +
  labs(
    x = "Zone of Influence",
    y = "Correlation",
    color = "Spatial distribution",
    linetype = "Mean n.n. distance"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
```



We see that, for regular, random, or less clumped point distributions (upper row in the figure above), the distance to the nearest feature and the cumulative influence are highly correlated at low ZoI values, and the correlation decreases as the ZoI increases.

For the more clumped distribution of points (lower row in the figure above), as the point patterns become more clumped in in less and smaller clusters, there starts to appear an inflection in the correlation so that the correlation starts to increase with the ZoI. The point of inflection seems to be related to the average nearest neighbor distance between features in each scenario. When the ZoI is greater than this value, the cumulative influence gets different from the distance to the nearest feature, and what happens at ZoI values smaller or larger than this average depends on the spatial distribution of the features.

Using neighborhood analysis to calculate cumulative influence

Add this part here. First I need to implement the exponential filter function in the `oneimpact` package.

Discussion

Add some discussion here.