Simulating scenarios: comparing the cumulative distance to the distance to the nearest feature

Bernardo Niebuhr, Bram van Moorter

06 January, 2022

Contents

1 Introduction

Our aim here is to assess how the measures of cumulative influence (*CumInf*) and distance to the nearest feature compare when calculated exactly with the same input layer with the locations of infrastructure features. *CumInf* is a measure of cumulative distance, proportional to the density of features (see the main text for details). For this comparison we computed here both measures based on the exponential decay distance, which decreases exponentially from 1 where the infrastructure features are located and approaches asymptotically zero as one walks apart from them. Even though the ZoI is not defined for an exponential decay function, since the function never reaches zero, it has specific properties such as the half-life that might help defining a ZoI. Here we define the ZoI as $4 \cdot$ half life. Since the half life is the distance at which the exponential decay distance decreases to 0.5, in this context the ZoI is defined as the distance at which the function get to 0.5^4 = 0.0625.

To do so, we did the following:

1) simulated a set of points representing the locations of point infrastructure, according to different spatial distributions;
2) calculated the exponential decay distance to each of these point features and put all measures in a RasterStack;
3) calculated the (i) sum of all distance measures (equivalent to the *CumInf*) and the (ii) maximum[1] of all distance measures (equivalent to the distance to the nearest feature);
4) compared the results visually and through correlation analyses.

By comparing these two measures over different scenarios and different ZoIs, we can assess in which cases they represent a similar spatial variation and in which cases they represent different spatial gradients.

2 Simulate landscapes

We started by creating points and separating them into different rasters, so that it is possible to calculate the distance to each of them. For simplicity, here we kept constant the number of points (n = 100). We simulated

[1]Note that, contrary to the Euclidean distance, the exponential decay distance decreases as we step away from the infrastructure features. For this reason, the maximum distance over a set of features corresponds to the distance to the nearest feature.

points with 3 spatial distribution patterns: regular, random, and clumped. For the clumped distribution, we used two scenarios, where the point features are spread in either five clusters or only one single focal cluster, and we varied the width of the clusters (mean distance from the points to the center of the cluster = 5 and 15% of the landscape size). Then, we had six scenarios in a gradient of clumpiness. The clumped scenarios are named based on the number of clusters and the radius of the cluster (e.g. "clumped1_5" means a scenario with one cluster with width equivalent to 5% of the landscape extent).

```r
# Load packages

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

# spatial packages
library(mobsim)
library(terra)
library(landscapetools)
library(sf)
library(spatstat)

# oneimpact package
library(oneimpact)
```

```r
# Set parameters for the simulation
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(500, 750, 1000, 1250, 1500, 2000, 3000, 4000, 5000, 10000)

# 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]])
  })
)

# simplify tibble with scenarios and distances
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(zoi = zoi_vals)
    })
  ) %>%
  tidyr::unnest(parms)
```

After creating the points, we calculated the exponential decay distance for each of the 100 points, separately, in each scenario. We did that considering a range of different ZoI values, from 500 m to 10 km. Based on the stack of rasters with the distance to each of the features, we then computed two variables:

- the cumulative influence, by calculating the sum of the distances of each raster in the series of distance rasters;
- the nearest neighbor distance, by calculating the maximum distance over the series of distance rasters.

```r
# calculate distances to each point using several ZoI/scales
outerlist <- list()
for (i in 1:length(create_scenarios$pt_sep_rast)) {
  innerlist <- sapply(
    vector(mode = "list", length = length(zoi_vals)),
    function(x) list()
  )
  # innerlist <- vector(mode = "list", length = length(zoi_vals))
  # for(k in 1:length(innerlist)) innerlist[[k]] <- list()

  parms <- expand.grid(
    transform_dist = "exp_decay",
    zoi = zoi_vals
  ) %>%
    tibble::as_tibble()
  for (j in 1:length(create_scenarios$pt_sep_rast[[i]])) {
    print(paste(i, j))
    # take the input raster
    r <- create_scenarios$pt_sep_rast[[i]][[j]]
    # calc dist for different parameters
    r_dist <- parms %>% purrr::pmap(calc_dist, x = r)
    # put each ZoI in a different list
    for (element in 1:length(innerlist)) {
      innerlist[[element]][[j]] <- r_dist[[element]]
    }
  }

  outerlist[[i]] <- innerlist
}

# stack each set of 100 dist maps, for each scenario
# stack_list <- purrr::map(outerlist, function(x) purrr::map(x, raster::stack)) # raster
stack_list <- map(outerlist, function(x) purrr::map(x, function(z) do.call(c, z)))
length(stack_list)
length(stack_list[[1]])
```

```r
# 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]])) {
    # print(paste(i, j))
    scenarios_dist$rast_dist[[cont]] <- stack_list[[i]][j]
    scenarios_dist$rast_nndist[[cont]] <- terra::app(stack_list[[i]][j], max, na.rm = T)
    scenarios_dist$rast_cuminf[[cont]] <- terra::app(stack_list[[i]][j], sum, na.rm = T)
    # this is density, just to compare
    scenarios_dist$rast_cuminf_d[[cont]] <- terra::app(stack_list[[i]][j], mean, na.rm = T)
    cont <- cont + 1
  }
}
```

3 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 = 750 m).

```
which_scenarios <- c(0:2, 4)
indices <- which_scenarios * length(zoi_vals) + 2
to_viz <- do.call(c, c(scenarios_dist$rast_nndist[indices], scenarios_dist$rast_cuminf[indices]))
names(to_viz) <- paste(rep(name[which_scenarios + 1], 2),
  rep(c("nearest", "CumInf"), each = 4),
  sep = ": "
)

plot(to_viz, nc = 4)
```

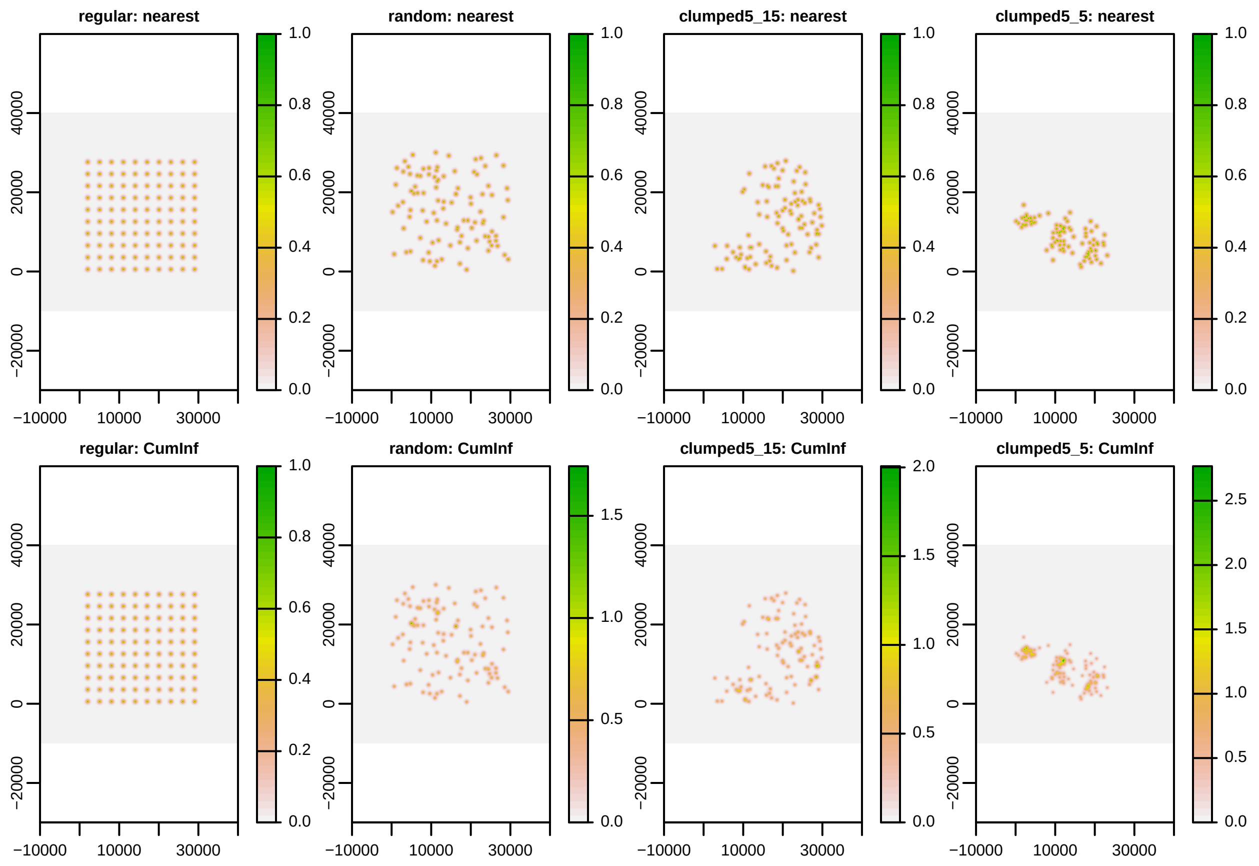

Figure 1: Distance to the nearest feature (nearest, top row) and cumulative influence (CumInf, bottom row) to 100 point infrastructure features spread in space according to four different scenarios (columns, in order): regular, random, and clustered (5 clusters, cluster with = 0.15 and 0.05 of the landscape extent, for columns 3 and 4). The distances are calculated based on an exponential decay distance with ZoI = 750 m.

We see in Fig. 1 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 *CumInf* differing for the fourth scenario (“clumped5_5”).

Now we show the same, but considering a larger ZoI of 3000 m (Fig. 2). We see that, in this case, the distance to the nearest feature is generally different from the *CumInf* measure.

```
which_scenarios <- c(0:2, 5)
indices <- which_scenarios * length(zoi_vals) + 7
to_viz <- do.call(c, c(scenarios_dist$rast_nndist[indices], scenarios_dist$rast_cuminf[indices]))
names(to_viz) <- paste(rep(name[which_scenarios + 1], 2),
  rep(c("nearest", "CumInf"), each = 4),
  sep = ": "
)

plot(to_viz, nc = 4)
```

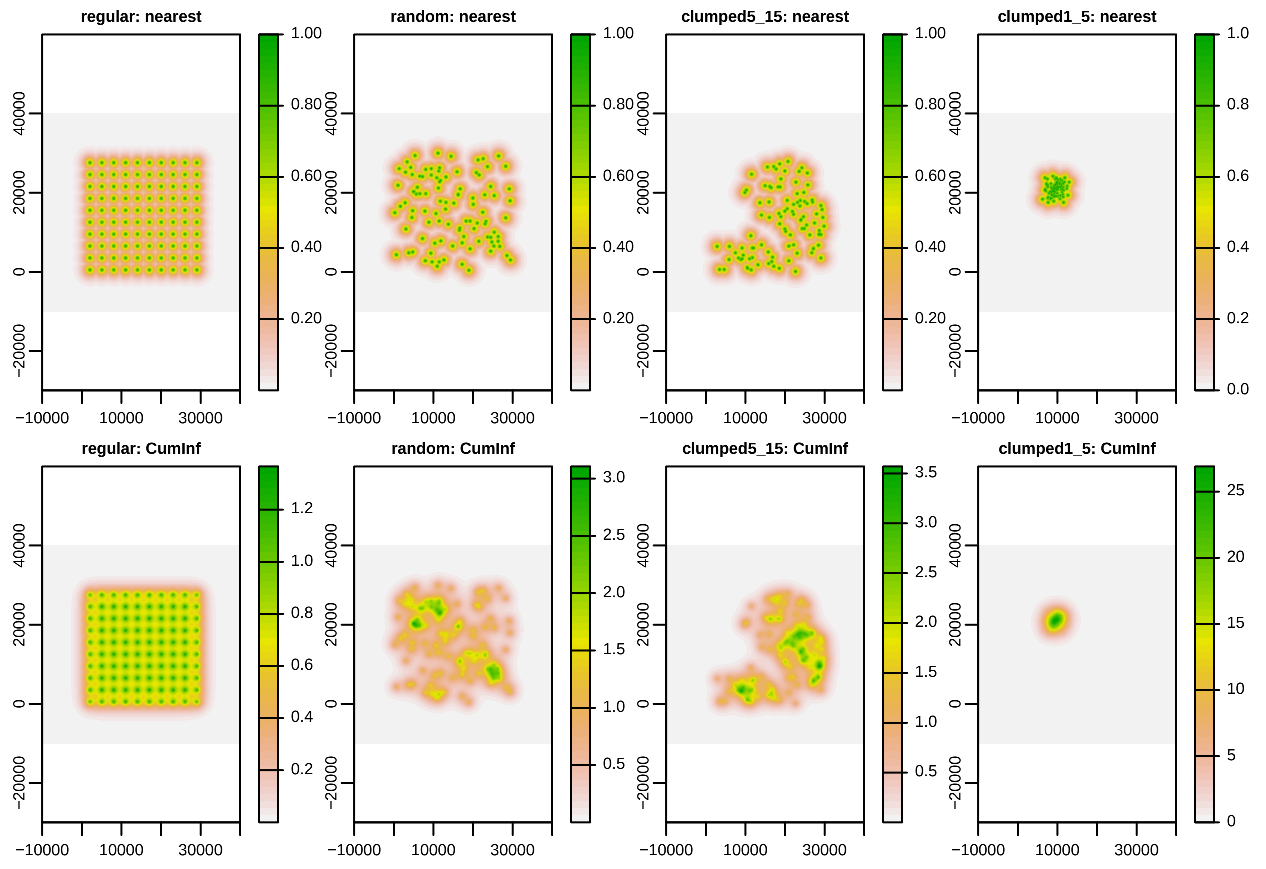

Figure 2: Distance to the nearest feature (nearest, top row) and cumulative influence (CumInf, bottom row) to 100 point infrastructure features spread in space according to four different scenarios (columns, in order): regular, random, and clustered (5 clusters, cluster with = 0.15 and 0.05 of the landscape extent, for columns 3 and 4). The distances are calculated based on an exponential decay distance with ZoI = 3000 m.

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 these correlations and compare the patterns to the nearest neighbor distance between features in each landscape scenario.

```r
# Correlation between scenarios
scenarios_analysis <- scenarios_dist %>%
  dplyr::mutate(corr = purrr::map2(rast_nndist, rast_cuminf, function(x, y) {
    cor(terra::values(x), terra::values(y))
  })) %>%
  tidyr::unnest(corr) %>%
  dplyr::mutate(spatial_dist = factor(spatial_dist, levels = name[c(1, 2, 3, 4, 5, 6)]))

# plot
dat_isol <- scenarios_analysis %>%
  dplyr::group_by(n_features, spatial_dist) %>%
  dplyr::summarise(
    mean_iso = unique(mean_isolation),
    mean_dist = unique(mean_dist),
    mean_nndist = unique(mean_nndist)
  )

fact <- c(1, 1, 0.15, 0.15, 0.05, 0.05)

dat_isol$mean_iso / 30000
```

```
## [1] 0.04294410 0.04989605 0.08795425 0.11309754 0.19090080 0.31913118
```

```r
dat_isol$mean_nndist
```

```
## [1] 3000.0000 1681.9061 1255.7376 1069.7642  756.9967  353.4595
```

```r
(a <- (dat_isol$mean_nndist * 10) * (dat_isol$mean_iso / 30000))
```

```
## [1] 1288.3230  839.2047 1104.4746 1209.8770 1445.1128 1127.9994
```

```r
scenarios_analysis %>%
  ggplot(aes(zoi, abs(corr))) +
  geom_point() +
  geom_line() +
  geom_vline(data = dat_isol, aes(xintercept = a, linetype = spatial_dist)) +
  ylim(0.7, 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))
```

```r
map_dbl(pts_coords, function(x) mean(spatstat.geom::nndist(x)))
```

```
## [1] 3000.0000 1681.9061 1255.7376 1069.7642  756.9967  353.4595
```

```r
a <- map(pts_coords, function(x) spatstat.geom::nndist(x))

a <- (map(pts_coords, function(x) dist(x)))
```

We see that, for regular, random, or less clumped point distributions (upper row in the figure above and "clumped1_15" scenario), the distance to the nearest feature and the cumulative influence are highly correlated at low ZoI values, when the ZoI is smaller than the distance betweeen features, and the correlation decreases as the ZoI increases.

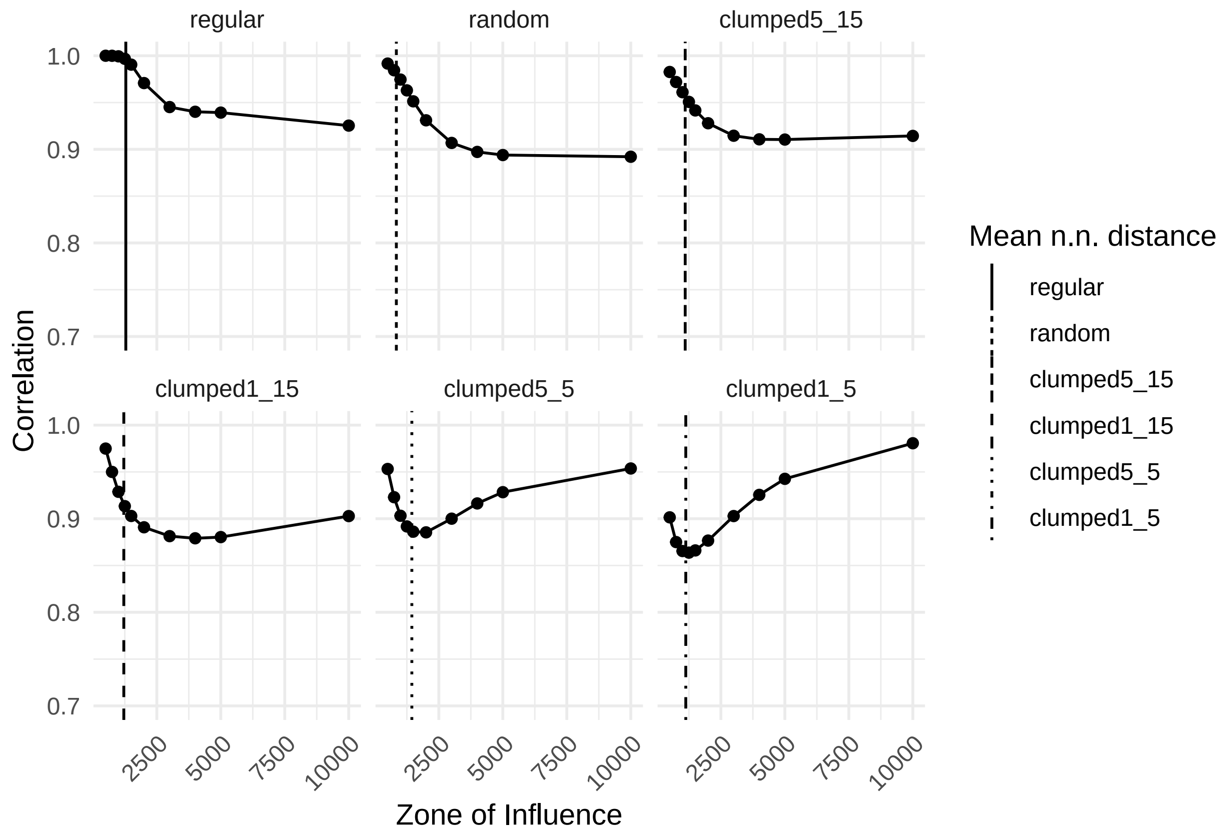

Figure 3: Correlation between the distance to nearest feature and the cumulative influence, for the different scenarios (regular, random, clumped) and ZoI values. In the clumped scenarios the features are spread in either 5 clusters (C and D) or 1 cluster (E and F), whose width is either 0.15 (C, E) or 0.5 (D, F) of the landscape extent.

For the more clumped distribution of points ("clumped5_5" and "clumped1_5"), as the point patterns become more clumped in less and smaller clusters, there starts to appear an inflection in the correlation so that the correlation starts to increase with the ZoI. **Check the following:** 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.

5 Using neighborhood analysis to calculate cumulative influence

Now we show that the cumulative influence might be calculated using neighborhood analysis - by defining the spatial filter according to the aimed distance measure - here the exponential decay distance - for the whole set of point features at once, instead of calculating the distance for each of the features and then summarizing them. This is because *CumInf* is proportional to the density of features in space.

Here we first define a exponential decay window for each of the corresponding ZoI values used above and we calculate *CumInf* using neighborhood analysis. Then we compare the *CumInf* measure based on both approaches - a summary of the distance to each of the features, and a neighborhood analysis for all features at once.

```r
# create filter matrices
parms_filter <- expand.grid(method = "exp_decay", zoi = zoi_vals, scenario = name) %>%
  dplyr::select(-scenario)
filter_matrices <- parms_filter %>% pmap(create_filter, res = res)
```

```r
# create density maps based on the rasters with all features
in_rast <- rep(pts_all_rast, each = length(zoi_vals))
maps_density <- purrr::map2(in_rast, filter_matrices, calc_dens, type = "mfilter") # ,
# extent_x_cut = c(0, ext), extent_y_cut = c(0, ext))

scenarios_analysis <- scenarios_analysis %>%
  dplyr::mutate(rast_density = maps_density)

# correlation
scenarios_analysis <- scenarios_analysis %>%
  dplyr::mutate(corr_dens = purrr::map2(rast_density, rast_cuminf, function(x, y) {
    cor(terra::values(x), terra::values(y), use = "complete.obs")
  })) %>%
  tidyr::unnest(corr_dens)

# check
all(scenarios_analysis$corr_dens > 0.99)
```

```
## [1] TRUE
```

We see that the correlation between the two measures is higher than 0.99 is all cases. The small differences that arise are a result of using a limited-sized window instead of calculating the distance for each pixel based on the whole landscape, but for practical purposes they are negligible. Below we see the plot is basically the same for different scenarios.

```r
# plot
which_scenarios <- c(0:2, 5)
indices <- which_scenarios * length(zoi_vals) + 7
to_viz <- do.call(c, c(scenarios_analysis$rast_density[indices], scenarios_analysis$rast_cuminf[indices]))
names(to_viz) <- paste(rep(name[which_scenarios + 1], 2),
  rep(c("CumInf distance", "CumInf neighborhood"), each = 4),
  sep = ": "
)

plot(to_viz, nc = 4)
```

6 Discussion

In this document we have shown that:

- In regular, random, and less clustered spatial distributions of features, the distance to the nearest feature and the cumulative influence represent different sources of spatial variation as the zone of influence of the infrastructure increase. This happens because with large ZoIs the effects of the multiple infrastructure start to interact with each other and accumulate.
- In clumped distributions of features, on the contrary, when the ZoI is sufficiently high the distance to the nearest feature and the cumulative influence are similar. This happens for more isolated infrastructure, suck as wind parks and small villages or urban centers, for instance. In these cases, as one walks far apart from the infrastructure, the accumulated effect of the infrastructure features (e.g. wind turbines or houses) is small and both measures are equivalent.
- Cumulative Influence (or, equivalently, feature density) might be calculated using common neighborhood (spatial filtering) analyses, what turns this procedure computationally fast and feasible.

7 References

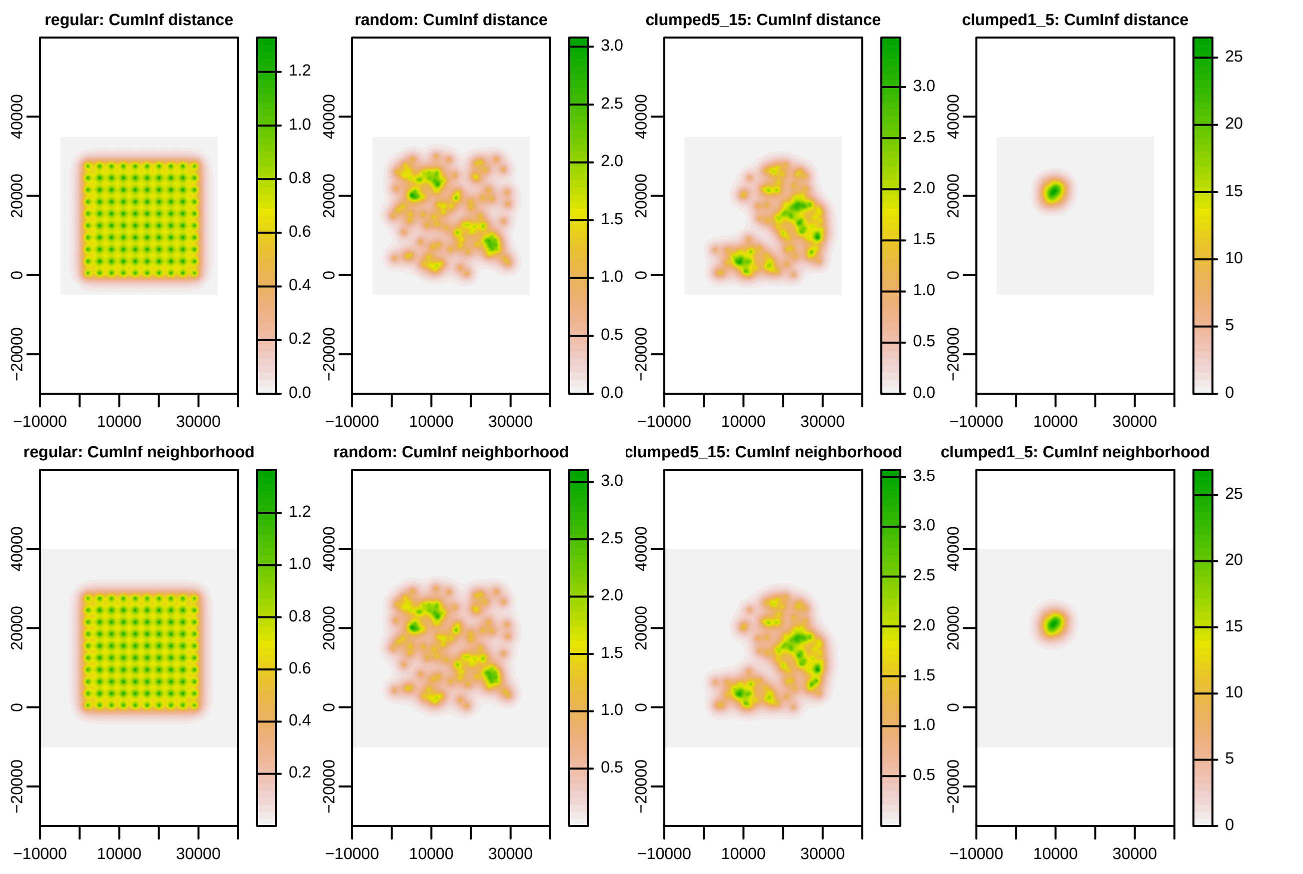
regular: CumInf distance
random: CumInf distance
clumped5_15: CumInf distance
clumped1_5: CumInf distance
regular: CumInf neighborhood
random: CumInf neighborhood
clumped5_15: CumInf neighborhood
clumped1_5: CumInf neighborhood
−10000
10000
30000
−20000
0
20000
40000

Figure 4: Cumulative influence calculated based on the distance to each feature and then summed (CumInf distance, top row) and based on neighborhood analyses (CumInf neighborhood, bottom row), for ZoI = 3000 m.