Methods to get more information from sparse vessel monitoring systems data

Hans Gerritsen

Introduction

This vignette accompanies the paper "Methods to get more information from sparse vessel monitoring systems data". It illustrates three of the methods, all of which are based on analysing point data (without attempting to interpolate vessel tracks).

Load libraries

```
library(sf)
library(dplyr)
library(locfit)
library(ggplot2)
library(viridisLite)
```

Example data

First generate some data consisting of VMS locations together with a value quantifying the amount of fishing activity associated with each location (usually this is the time interval between consecutive VMS records of the same vessel). In this case we randomly assign either one or two hours of fishing activity to each record to reflect the variability in polling frequency that occurs in real data. In the simulated data all VMS locations correspond to fishing activity (in other words any steaming or inactive periods have already been removed). The simulated fishing activity is located off the west coast of Ireland. We project the data using UTM zone 29N which converts the latitude and longitude to a grid in meters which is convenient later on.

```
set.seed(123)
n <- 1000 # Number of points to generate
lon <- rnorm(n, -11, 0.1) # longitude in WGS 84 coordinates
lat <- rnorm(n, 53, 0.06) # latitude in WGS 84 coordinates
fishingHrs <- sample(1:2, n, replace = TRUE) #randomly assign either 1 or 2 hours
vms <- data.frame(lon, lat, fishingHrs)

# project to UTM zone 29N (crs 32629) and add X and Y positions (in meters)
vms <- vms %>% st_as_sf(coords=c('lon','lat'), crs=4326, remove=F) %>%
  st_transform(crs=32629) %>%
  mutate(data.frame(st_coordinates(.))) %>%
  st_drop_geometry()
```

Local regression

The *locfit* package is used to estimate densities and apply local regression using the n nearest neighbours. We estimate the density separately from the mean fishing hours in each location. For both fits we use the 20 nearest neighbors. The *locfit* approach allows different degrees of polynomials for each fit. We set this to one

for density fits, which means that a linear regression is applied. For estimating the mean fishing hours we use a polynomial degree of zero, which means an average is applied. This prevents extrapolation of very high or negative values in areas with no little data. *locfit* also allows users to use a global bandwidth to smooth data (h) in our case we set this to zero so the fit is based on the nearest neighbours only.

```
# First set up a prediction grid (for UTM 29 locations)
pred_grid <- expand.grid(X=seq(min(vms$X),max(vms$X),by=250),
                         Y=seq(min(vms$Y),max(vms$Y),by=250))

# The number of nearest neighbours to use
nn <- 20
nn_prop <- nn / nrow(vms)
# estimate the mean fishing hours at each location
fit_meanhrs <- locfit(fishingHrs ~ lp(X, Y, nn=nn_prop, deg=0, h=0), data=vms)
pred_meanhrs <- predict(fit_meanhrs, newdata=pred_grid)
# estimate the density at each location
fit_density <- locfit(~ lp(X, Y, nn=nn_prop, deg=1, h=0), data=vms)
pred_density <- predict(fit_density, newdata=pred_grid) * nrow(vms) * 1e6
# combine the two to estimate the fishing density (in hours per km^2)
fit <- data.frame(pred_grid, fishingHrsPerKm2=pred_meanhrs * pred_density)
```

```
# Plot the result
ggplot(fit, aes(X, Y)) + geom_raster(aes(fill=fishingHrsPerKm2)) +
  scale_fill_gradientn(colours=viridis(256, option="A", direction=-1)) +
  coord_sf(crs=32629, expand=F) +
  labs(title = "Local regression fit of fishing hours per km2",x='',y='') +
  theme_test()
```

Here we use we use a threshold 1 hour per km2 to identify areas that are lightly fished (the areas below this threshold). Note that the number of data points in this example is quite low, resulting in a ragged looking area. Increasing the number of nearest neighbours will result in a smoother estimate.

```
ggplot(fit,aes(X,Y)) + geom_raster(aes(fill=fishingHrsPerKm2<1)) +
  scale_fill_manual(values=c(NA,'#008000'),na.value=NA) +
  coord_sf(crs=32629, expand=F) +
  labs(title = "Nearest neighbour estimate of lightly impacted area",x='',y='') +
  theme_test()
```

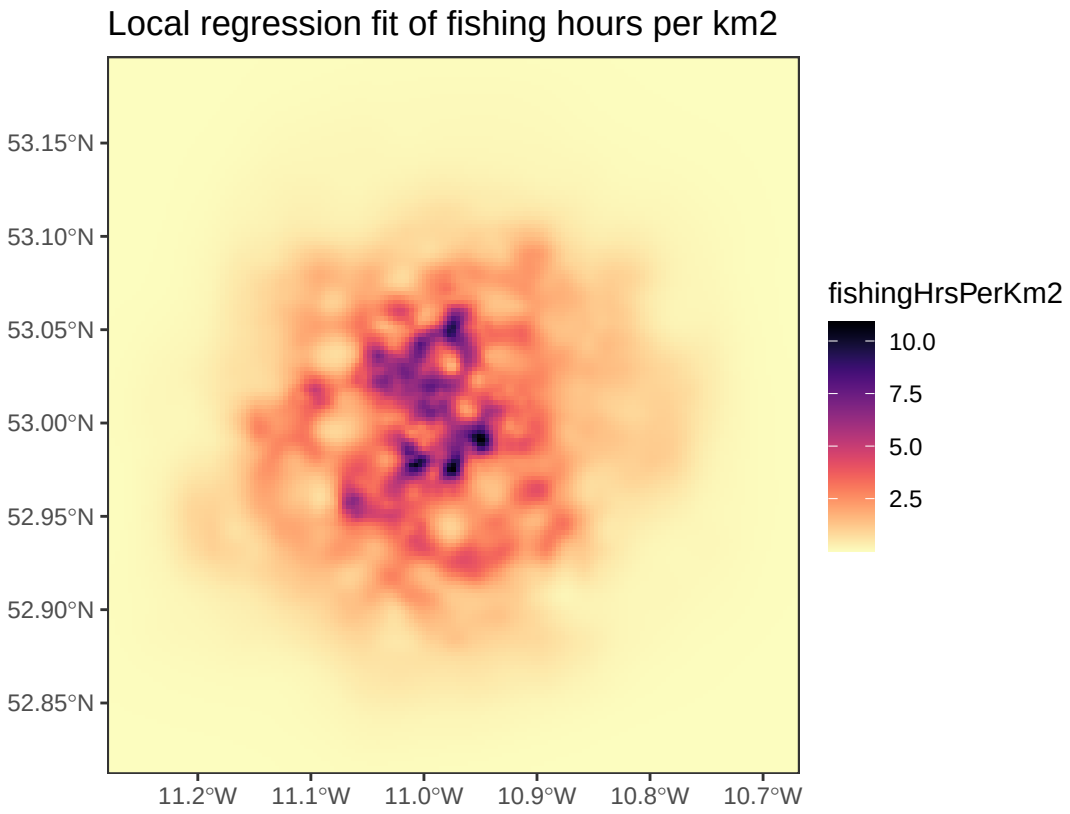

Figure 1: The resulting estimate of fishing hours per km2

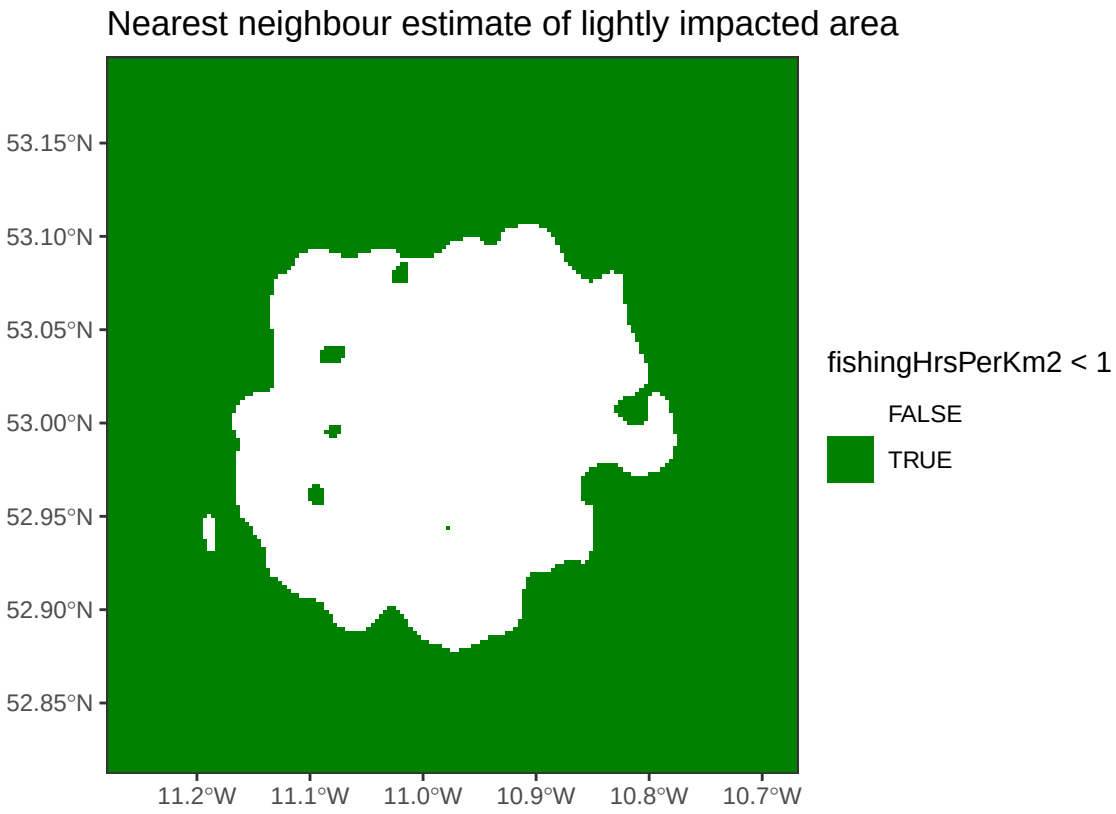

Figure 2: Lightly impacted areas are shown in green

Voronoi tesselation

An alternative approach can be taken by applying Voronoi tessellation to each location to create a polygon around each point We can then calculate the area of this polygon, providing a density estimate for each individual VMS record. In areas with very high densities this could lead to very small polygons (and therefore very high density estimates) so the first step is to aggregate the data to a (fine) spatial grid. This has the added advantage of dealing with any duplicate records (records with the same latitude and longitude). The resolution of the grid is not very important; here we use 500m x 500m. So we round the location data to the nearest 500m and then aggregate the fishing hours.

```
vms_grid <- vms %>%
  mutate(across(c(X, Y), function(x) round(x/500)*500)) %>%
  group_by(X,Y) %>% summarise(fishingHrs=sum(fishingHrs), .groups='drop')

# make it into a simple feature collection
vms_grid_sf <- vms_grid %>% st_as_sf(coords=c("X","Y"), crs = 32629)
```

Apply the Voronoi tessellation

```
# Voronoi tessellation and crop to deal with some of the edge effects
vms_voronoi <- vms_grid_sf %>% st_union() %>% st_voronoi() %>%
  st_cast() %>% st_crop(st_bbox(vms_grid_sf))
# Join back to the original data
vms_voronoi <- st_join(st_as_sf(vms_voronoi), vms_grid_sf)
# Calculate fishing hours per km2 in each Voronoi polygon
vms_voronoi$area <- st_area(vms_voronoi)
vms_voronoi$fishingHrsPerKm2 <- vms_voronoi$fishingHrs / as.numeric(vms_voronoi$area) * 1e6
```

```
ggplot() +
  geom_sf(aes(fill = fishingHrsPerKm2),vms_voronoi) +
  scale_fill_gradientn(colours = viridis(256, option = "A", direction=-1)) +
  geom_sf(data=vms_grid_sf,size=0.2,pch=16) +
  coord_sf(crs=32629, expand=F) +
  labs(title = "Voronoi polygons with fishing hours per km2") +
  theme_test()
```

We use we use the same threshold as before to identify areas that are lightly fished: 1 hour per km2.

```
ggplot(vms_voronoi) + geom_sf(aes(fill = fishingHrsPerKm2<1),col=NA) +
  scale_fill_manual(values=c(NA,'#008000'),na.value=NA) +
  coord_sf(crs=32629, expand=F) +
  labs(title = "Voronoi estimate of lightly impacted area") +
  theme_test()
```

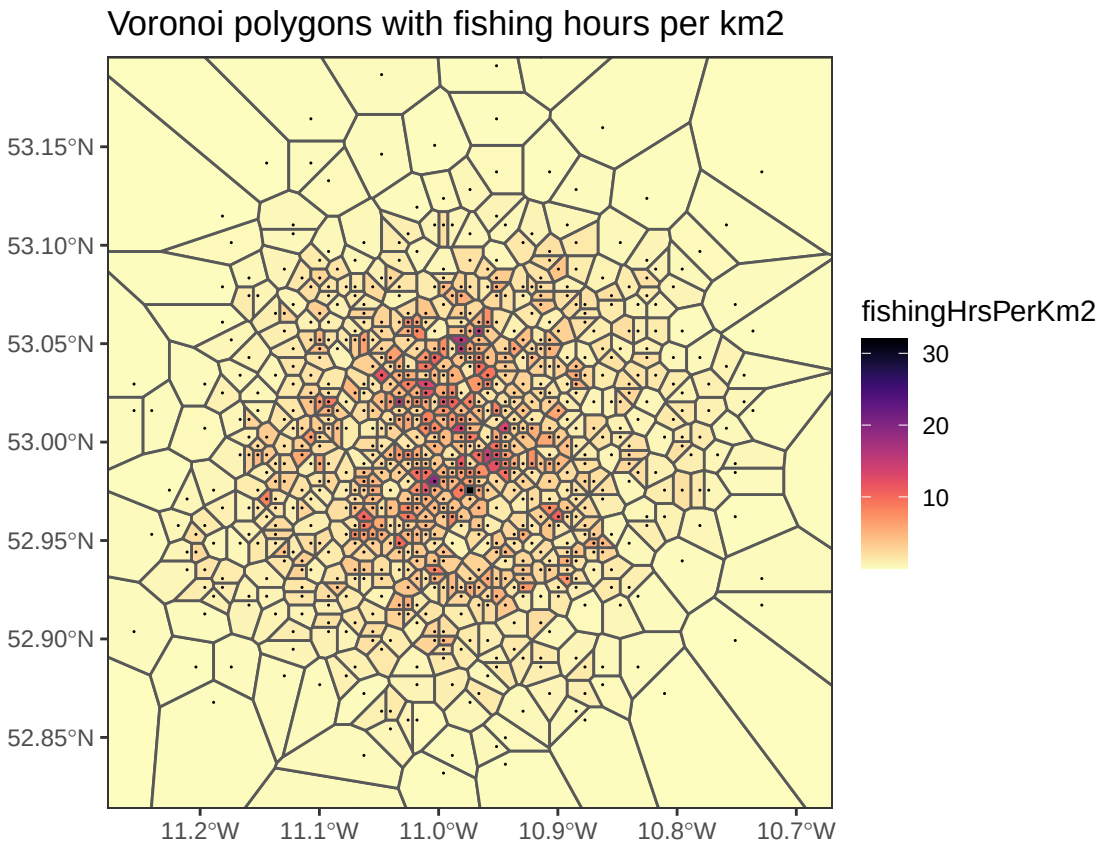

Figure 3: The resulting estimate of fishing hours per km2

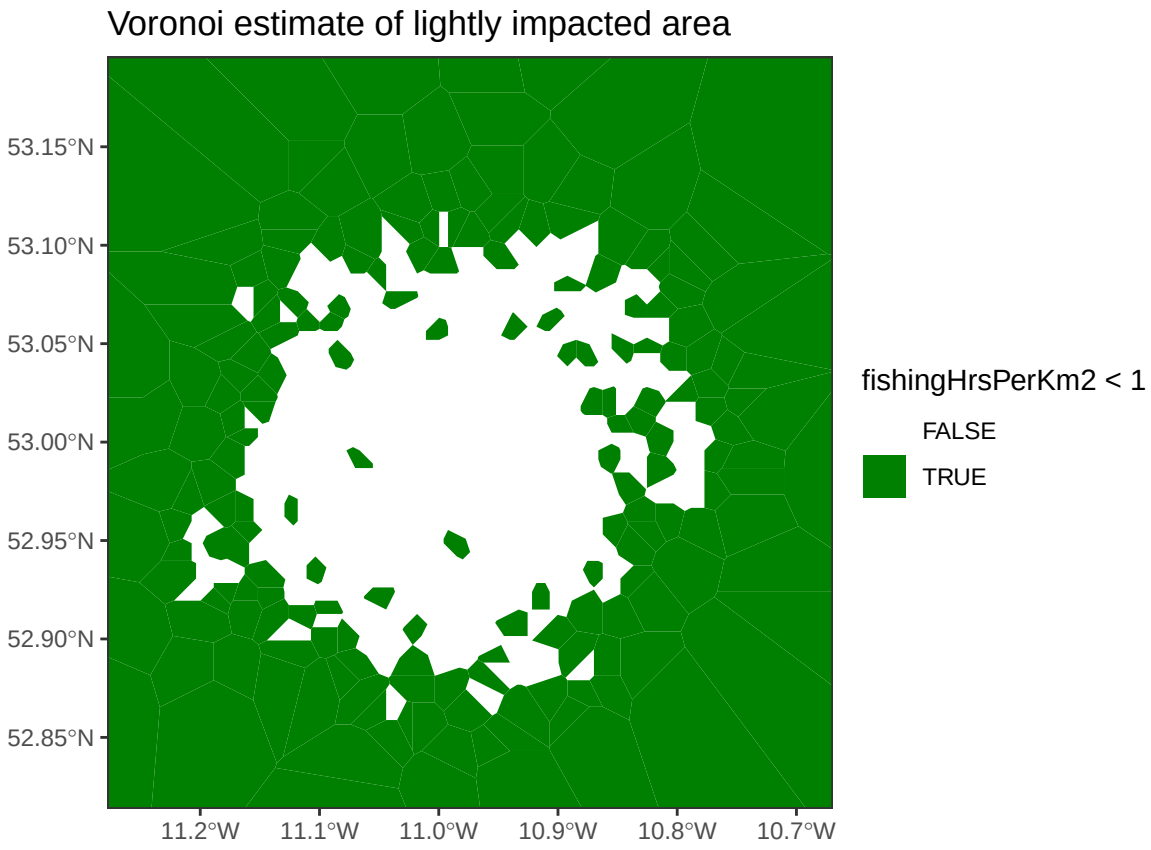

Figure 4: Lightly impacted areas are shown in green

Nested grid approach

This approach creates a grid with variable cell sizes, based on the number of observations in each grid cell. The idea is to start with a large rectangle, covering the whole area and to iteratively divide this in increasingly small rectangles.The rectangles are split so they will contain equal number of data points by identifying the median x and y position of the points inside the rectangle. Rectangles that are higher than they are wide are split along the x direction rectangles that are wider than high are split along the y direction. We used a threshold of n = 5, meaning that rectangles will not be further split if that would result in fewer than 5 data points in the resulting rectangle.

This method is not recommended for identifying lightly impacted areas, particularly when data are sparse as in this example. The code below is included for completeness only.

```
# the starting point is to make a rectangle encompassing all datapoints
vms_nest <- vms_grid %>% mutate(x1 = min(X),
                                x2 = max(X),
                                y1 = min(Y),
                                y2 = max(Y),
                                n = length(X))

# set the minimum number of records in each rectangle
# do not split a rectangle if that results in fewer
# than 5 records in one of the two new rectangles
minn <- 5
while(max(vms_nest$n)>=minn*2) {
  vms_nest <- vms_nest %>% group_by(x1,x2,y1,y2) %>%
      mutate(cutx=median(X)  # where to cut along the x axis
             ,cuty=median(Y) # where to cut along the y axis
             ,w=abs(x1-x2)   # width of the rectangle
             ,h=abs(y1-y2)   # height of the rectangle
             # if it is wider than high,cut along x
             ,x1=ifelse(w>h & X>cutx & length(x1)>minn,cutx,x1)
             ,x2=ifelse(w>h & X<=cutx & length(x2)>minn,cutx,x2)
             # if it is higher than wide,cut along y
             ,y1=ifelse(w<=h & Y>cuty & length(y1)>minn,cuty,y1)
             ,y2=ifelse(w<=h & Y<=cuty & length(y2)>minn,cuty,y2)
             ,n=length(X)) %>%
    ungroup()
}

vms_nest <- vms_nest %>%
  mutate(poly = paste('POLYGON ((',x1,y1,',',x2,y1,', ',x2,y2,',',x1,y2,',',x1,y1,'))'))

# aggregate by rectangle and calculate the fishing hours per km2
vms_nest <- vms_nest %>% group_by(poly) %>%
  summarise(fishingHrs=sum(fishingHrs)) %>% subset(!is.na(poly))
vms_nest$geometry <- st_as_sfc(vms_nest$poly,crs=32629)
vms_nest$area <- st_area(vms_nest$geometry)
vms_nest$fishingHrsPerKm2 <- vms_nest$fishingHrs / as.numeric(vms_nest$area) * 1e6
```

```
ggplot() +
  geom_sf(aes(geometry=geometry,fill = fishingHrsPerKm2),vms_nest) +
  scale_fill_gradientn(colours = viridis(256, option = "A", direction=-1)) +
  geom_sf(data=vms %>% st_as_sf(coords=c('lon','lat'), crs = 4326) ,size=0.2,pch=16) +
  coord_sf(crs=32629, expand=F) +
  labs(title = "Nested grid with fishing hours per km2") +
```

```
  theme_test()
```

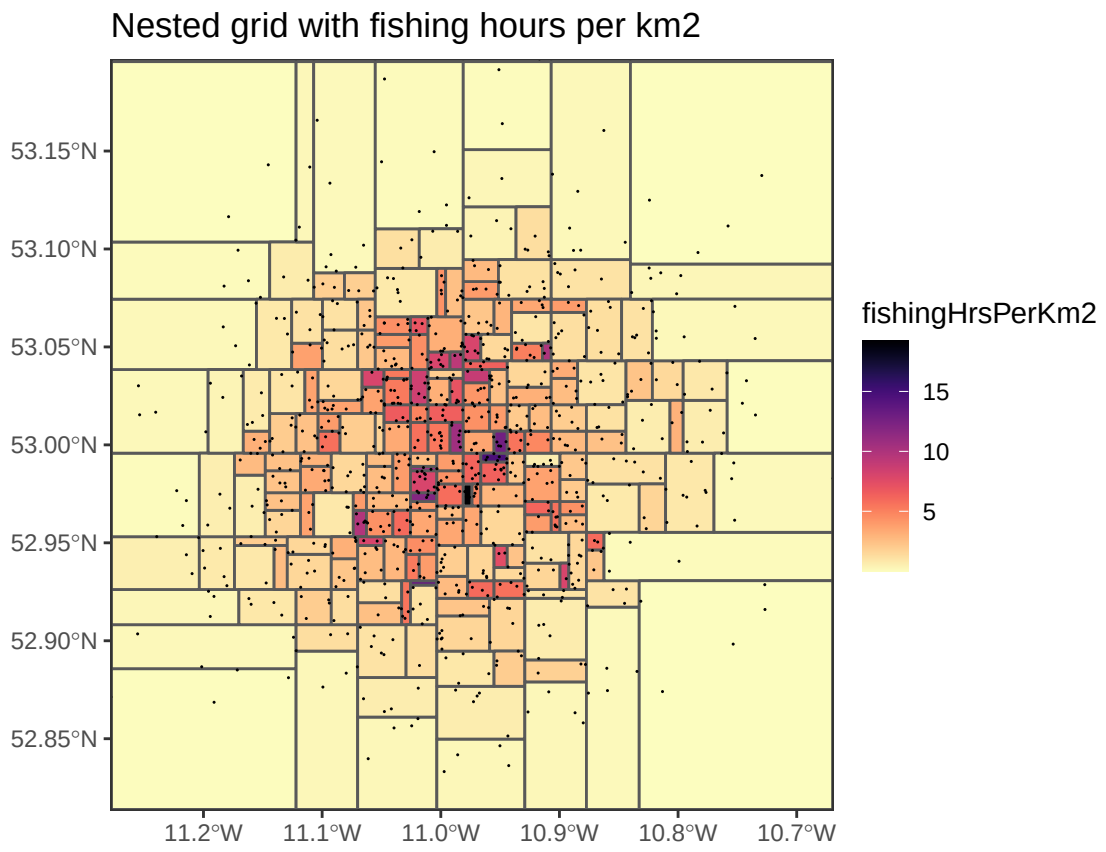

Figure 5: The resulting estimate of fishing hours per km2

We use we use the same threshold as before to identify areas that are lightly fished: 1 hour per km2.

```
ggplot(vms_nest) + geom_sf(aes(geometry=geometry,fill = fishingHrsPerKm2<1),col=NA) +
  scale_fill_manual(values=c(NA,'#008000'),na.value=NA) +
  coord_sf(crs=32629,expand=F) +
  labs(title = "Nested grid estimate of lightly impacted area") +
  theme_test()
```

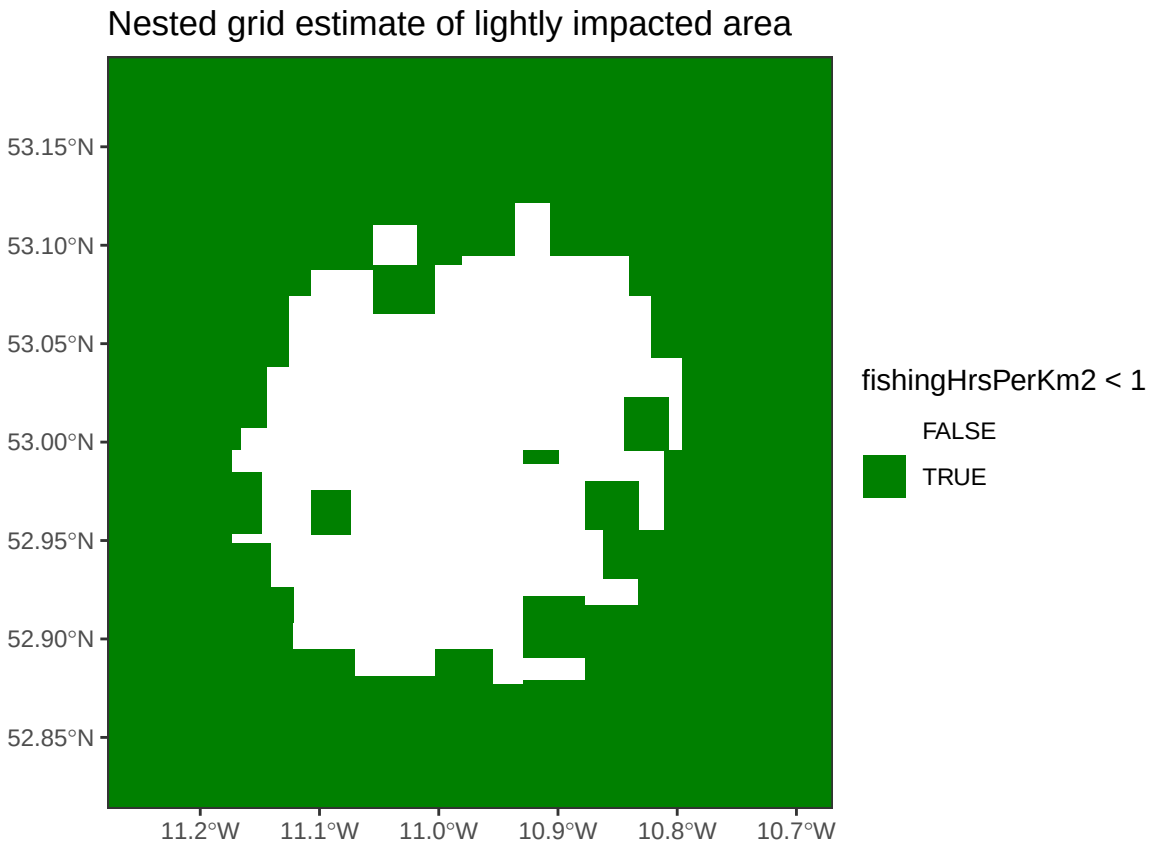

Figure 6: Lightly impacted areas are shown in green