Appendix

10 December 2022

Code to reproduce the map of the main text.

# 1 Data

## 1.1 Data wrangling

Data are stored in trees\_2021.zip which contains two GeoJSON files:

* trees\_2021 stores all trees of the city of Paris in 2021.
* trees\_logged contains all trees logged

They must be read. Data are projected into the Lambert 93 datum so that coordinates are in meters.

unzip("data/trees\_2021.zip", exdir = "data")  
library("sf")  
read\_sf("data/trees\_2021.geojson") %>%  
 st\_transform(crs = 2154) -> trees\_all\_raw  
read\_sf("data/trees\_logged.geojson") %>%  
 st\_transform(crs = 2154) -> trees\_logged\_raw

### 1.1.1 All trees

The first dataset contains all trees in Paris in 2021, including those to be cut.

Trees from the Suzanne Lenglen park are selected. Columns of interest are:

* ID: a numeric unique identifier for each tree.
* Species\_name: the scientific name of the tree species, i.e. Genus species.
* Status: Alive.
* Genus.
* Species.
* French\_species\_name: vernacular name.
* Circumference: in cm.

library("dplyr")  
trees\_all\_raw %>%  
 # Filter Suzanne Lenglen park  
filter(adresse == "PARC OMNISPORT SUZANNE LENGLEN / 7 BOULEVARD DES FRERES VOISIN") %>%  
 # Create a field with the species name  
mutate(Species\_name = as.factor(paste(genre, espece))) %>%  
 # Create a field with the status  
mutate(Status = "Alive") %>%  
 # Genus and Species fields  
mutate(Genus = as.factor(genre)) %>%  
 mutate(Species = as.factor(espece)) %>%  
 # Rename and finally select columns  
rename(ID = idbase, French\_species\_name = libellefrancais,  
 Circumference = circonferenceencm) %>%  
 select(ID, Species\_name, Status, Genus, Species,  
 French\_species\_name, Circumference) -> trees\_all  
# Number of trees  
trees\_all %>%  
 nrow()

## [1] 1472

We have 1472 trees in the park.

### 1.1.2 Logged trees

Logged trees are in the second dataset.

Their status is “Logged”. An extra field, Logging\_reason contains the motivation to cut them off (in French). Circumfernce is absent.

# Tree description  
trees\_logged\_raw %>%  
 # Filter Suzanne Lenglen park  
filter(adresse == "PARC OMNISPORT SUZANNE LENGLEN / 7 BOULEVARD DES FRERES VOISIN") %>%  
 # Exclude unidentified trees  
filter(!is.na(especearbreprecedent), !is.na(libellefrancaisarbreprecedent),  
 !is.na(genrearbreprecedent)) %>%  
 filter(libellefrancaisarbreprecedent != "Non spécifié") %>%  
 filter(especearbreprecedent != "n. sp.") %>%  
 # Create a field with the species name  
mutate(Species\_name = as.factor(paste(genrearbreprecedent,  
 especearbreprecedent))) %>%  
 # Create a numeric ID  
mutate(ID = as.integer(idbase)) %>%  
 # Create a field with the status  
mutate(Status = "Logged") %>%  
 # Genus and Species fields  
mutate(Genus = as.factor(genrearbreprecedent)) %>%  
 mutate(Species = as.factor(especearbreprecedent)) %>%  
 # Reason for logging (in French)  
mutate(Logging\_reason = motifabattagearbreprecedent) %>%  
 # Rename and finally select columns  
rename(French\_species\_name = libellefrancaisarbreprecedent) %>%  
 select(ID, Species\_name, Status, Genus, Species,  
 Logging\_reason, French\_species\_name) ->  
 trees\_logged  
# Number of trees  
trees\_logged %>%  
 nrow()

## [1] 48

48 among the 1472 trees of the park were logged.

## 1.2 Merge

The two datasets are merged here.

The logged trees must be removed from the first one. Circumference is removed because it is missing from the logged trees dataset.

# All trees  
trees\_all %>%  
 # Delete the logged trees  
filter(!(ID %in% trees\_logged$ID)) %>%  
 # Delete the circumference that is absent in  
 # trees\_logged  
mutate(Circumference = NULL) %>%  
 # Bind the logged trees  
bind\_rows(trees\_logged) -> trees\_no\_circumference

Circumferences of all trees, including logged ones, are in tree\_all from where they can be recovered.

# Prepare a tibble with circumferences  
trees\_all %>%  
 select(ID, Circumference) %>%  
 # inner\_join.sf refuses sf objects  
st\_set\_geometry(NULL) -> Circumferences  
# Add the Circumference of trees  
trees\_no\_circumference %>%  
 inner\_join(Circumferences, by = "ID") -> trees

### 1.2.1 Simpler logging reasons

Logging reasons can be:

* Decaying: the tree’s condition is not healthy enough to keep it safely in a public park.
* Infected: the tree is a maple affected by the (contagious) sooty bark disease, caused by the fungus *Cryptostroma corticale*.

trees$Logging\_reason[is.na(trees$Logging\_reason)] <- ""  
trees$Logging\_reason %>%  
 str\_replace("Arbre.\*", "Decaying") %>%  
 str\_replace("Foyer.\*", "Infected") -> trees$Logging\_reason

### 1.2.2 Factors

Several fields are converted to factors for efficiency.

trees$Logging\_reason <- as.factor(trees$Logging\_reason)  
trees$Status <- as.factor(trees$Status)  
trees$French\_species\_name <- as.factor(trees$French\_species\_name)

## 1.3 Point patterns

dbmms uses weighted, marked, planar point patterns (wmppp). A wmppp named trees\_infected is built. Point marks are their basal area (as weight) and either their logging reason or their genus if they are alive.

library("dbmss")  
trees %>%  
 # Weight is the basal area  
mutate(PointWeight = Circumference^2/4/pi) %>%  
 mutate(PointType = ifelse(Logging\_reason == "",  
 as.character(Genus), as.character(Logging\_reason))) %>%  
 # Add X and Y  
bind\_cols(st\_coordinates(trees)) %>%  
 wmppp(window = as.owin(st\_bbox(trees)), unitname = c("meter",  
 "meters")) -> trees\_infected

We also need a point pattern to describe the park before logging, as a reference.

trees\_all %>%  
 # Weight is the basal area  
mutate(PointWeight = Circumference^2/4/pi) %>%  
 # Genus is the point type  
rename(PointType = Genus) %>%  
 # Add X and Y  
bind\_cols(st\_coordinates(trees\_all)) %>%  
 wmppp(window = as.owin(st\_bbox(trees\_all)), unitname = c("meter",  
 "meters")) -> trees\_2021  
autoplot(trees\_2021)



The map shows the tree genera. Maple (*Acer sp.*) are the most abundant trees in the park.

# 2 Spatial analyses

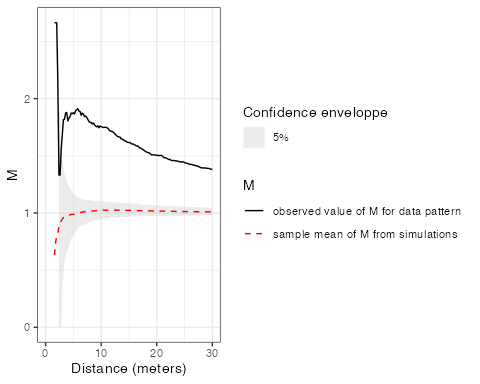
## 2.1 Spatial concentration of maple trees

The M statistic is computed to detect the spatial concentration of maple trees before logging.

Distance <- 15  
NumberOfSimulations <- 1000  
trees\_2021 %>%  
 MEnvelope(r = 0:(10 \* Distance)/5, ReferenceType = "Acer",  
 NeighborType = "Acer", NumberOfSimulations = NumberOfSimulations) ->  
 M\_Acer

## Generating 1000 simulations by evaluating   
## expression ...  
## 1, 2, 3, ......10.........20.........30.........40..  
## .......50.........60.........70.........80....  
## .....90.........100.........110.........120......  
## ...130.........140.........150.........160........  
## .170.........180.........190.........200.........210  
## .........220.........230.........240.........250..  
## .......260.........270.........280.........290....  
## .....300.........310.........320.........330......  
## ...340.........350.........360.........370........  
## .380.........390.........400.........410.........420  
## .........430.........440.........450.........460..  
## .......470.........480.........490.........500....  
## .....510.........520.........530.........540......  
## ...550.........560.........570.........580........  
## .590.........600.........610.........620.........630  
## .........640.........650.........660.........670..  
## .......680.........690.........700.........710....  
## .....720.........730.........740.........750......  
## ...760.........770.........780.........790........  
## .800.........810.........820.........830.........840  
## .........850.........860.........870.........880..  
## .......890.........900.........910.........920....  
## .....930.........940.........950.........960......  
## ...970.........980.........990......... 1000.  
##   
## Done.

autoplot(M\_Acer)



To map it, individual values must be calculated at the chosen distance, that is 15 meters.

trees\_2021 %>%  
 Mhat(r = c(0, Distance), ReferenceType = "Acer",  
 NeighborType = "Acer", Individual = TRUE) ->  
 M\_ind\_Acer

The map requires krigeing the individual values on a grid of points. To build the grid, the size ratio of the spatial window is calculated. The number of rows and columns of the grid will respect this ratio so that its points are equally spaced.

# Window ratio  
ratio <- with(trees\_infected$window, {  
 (yrange[2] - yrange[1])/(xrange[2] - xrange[1])  
})  
# Map resolution: number of columns of the grid.  
resolution <- 512

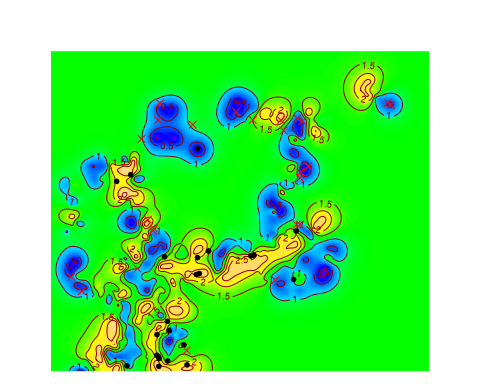
A kriged weighted, marked, planar point patterns (kwmppp) object is produced and plotted. Logged trees are added to the map:

* Infected trees are black points,
* Decaying trees are red crosses.

trees\_2021 %>%  
 kwmppp(fvind = M\_ind\_Acer, distance = Distance,  
 ReferenceType = "Acer", Nbx = resolution, Nby = resolution \*  
 ratio) -> map\_acer

## [using ordinary kriging]

plot(map\_acer)  
# Add infected trees  
points(trees\_infected[trees\_infected$marks$PointType ==  
 "Infected"], pch = 20)  
# And decaying trees  
points(trees\_infected[trees\_infected$marks$PointType ==  
 "Decaying"], pch = 4, col = "red")



Infected trees are present in the areas where maples are concentrated.

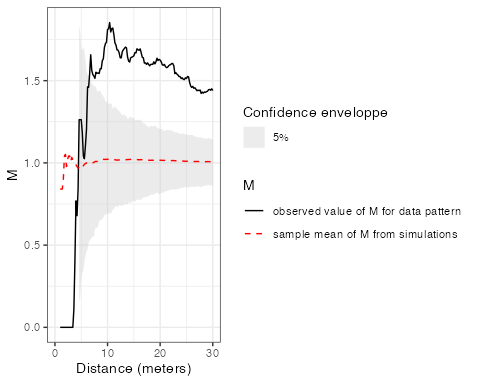
## 2.2 Concentration of maples around infected trees

To test the intertype concentration between sane and infected maple trees, the intertype M statistic is computed.

trees\_infected %>%  
 MEnvelope(r = 0:(10 \* Distance)/5, ReferenceType = "Infected",  
 NeighborType = "Acer", NumberOfSimulations = NumberOfSimulations) ->  
 M\_Infected\_Acer

## Generating 1000 simulations by evaluating   
## expression ...  
## 1, 2, 3, ......10.........20.........30.........40..  
## .......50.........60.........70.........80....  
## .....90.........100.........110.........120......  
## ...130.........140.........150.........160........  
## .170.........180.........190.........200.........210  
## .........220.........230.........240.........250..  
## .......260.........270.........280.........290....  
## .....300.........310.........320.........330......  
## ...340.........350.........360.........370........  
## .380.........390.........400.........410.........420  
## .........430.........440.........450.........460..  
## .......470.........480.........490.........500....  
## .....510.........520.........530.........540......  
## ...550.........560.........570.........580........  
## .590.........600.........610.........620.........630  
## .........640.........650.........660.........670..  
## .......680.........690.........700.........710....  
## .....720.........730.........740.........750......  
## ...760.........770.........780.........790........  
## .800.........810.........820.........830.........840  
## .........850.........860.........870.........880..  
## .......890.........900.........910.........920....  
## .....930.........940.........950.........960......  
## ...970.........980.........990......... 1000.  
##   
## Done.

autoplot(M\_Infected\_Acer)

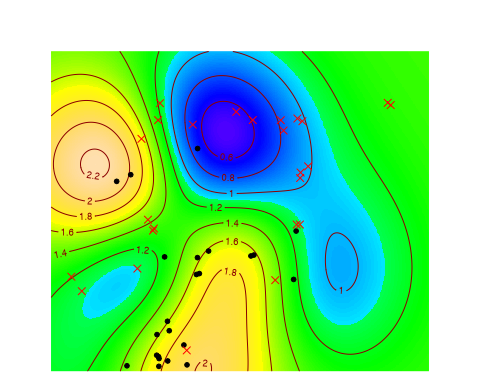


The map is produced.

trees\_infected %>%  
 Mhat(ReferenceType = "Infected", NeighborType = "Acer",  
 Individual = TRUE) -> M\_ind\_Infected\_Acer  
trees\_infected %>%  
 kwmppp(fvind = M\_ind\_Infected\_Acer, distance = Distance,  
 ReferenceType = "Infected", Nbx = resolution,  
 Nby = resolution \* ratio) -> map\_infected\_acer

## [using ordinary kriging]

plot(map\_infected\_acer)  
# Add infected trees  
points(trees\_infected[trees\_infected$marks$PointType ==  
 "Infected"], pch = 20)  
# And decaying trees  
points(trees\_infected[trees\_infected$marks$PointType ==  
 "Decaying"], pch = 4, col = "red")



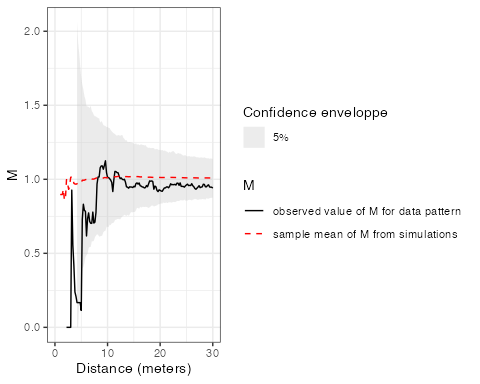
Infected and sane maple trees are significantly concentrated.

In contrast, decaying trees and mapples rather repulse each others.

trees\_infected %>%  
 MEnvelope(r = 0:(10 \* Distance)/5, ReferenceType = "Decaying",  
 NeighborType = "Acer", NumberOfSimulations = NumberOfSimulations) ->  
 M\_Decaying\_Acer

## Generating 1000 simulations by evaluating   
## expression ...  
## 1, 2, 3, ......10.........20.........30.........40..  
## .......50.........60.........70.........80....  
## .....90.........100.........110.........120......  
## ...130.........140.........150.........160........  
## .170.........180.........190.........200.........210  
## .........220.........230.........240.........250..  
## .......260.........270.........280.........290....  
## .....300.........310.........320.........330......  
## ...340.........350.........360.........370........  
## .380.........390.........400.........410.........420  
## .........430.........440.........450.........460..  
## .......470.........480.........490.........500....  
## .....510.........520.........530.........540......  
## ...550.........560.........570.........580........  
## .590.........600.........610.........620.........630  
## .........640.........650.........660.........670..  
## .......680.........690.........700.........710....  
## .....720.........730.........740.........750......  
## ...760.........770.........780.........790........  
## .800.........810.........820.........830.........840  
## .........850.........860.........870.........880..  
## .......890.........900.........910.........920....  
## .....930.........940.........950.........960......  
## ...970.........980.........990......... 1000.  
##   
## Done.

autoplot(M\_Decaying\_Acer)



# References