TMA4250-Project-2

Ole Riddervold, Ole Kristian Skogly

2023-03-06

Loading data

```
cells   <- read.table("data/cells.dat", skip=3, col.names=c('x', 'y'))
redwood <- read.table("data/redwood.dat", skip=3, col.names=c('x', 'y'))
pines   <- read.table("data/pines.dat", skip=3, col.names=c('x', 'y'))
```

Exploring data

```
head(cells)
```

```
##       x     y
## 1 0.350 0.025
## 2 0.062 0.362
## 3 0.938 0.400
## 4 0.462 0.750
## 5 0.462 0.900
## 6 0.737 0.237
```

```
head(redwood)
```

```
##       x     y
## 1 0.364 0.082
## 2 0.898 0.082
## 3 0.864 0.180
## 4 0.966 0.541
## 5 0.864 0.902
## 6 0.686 0.328
```

```
head(pines)
```

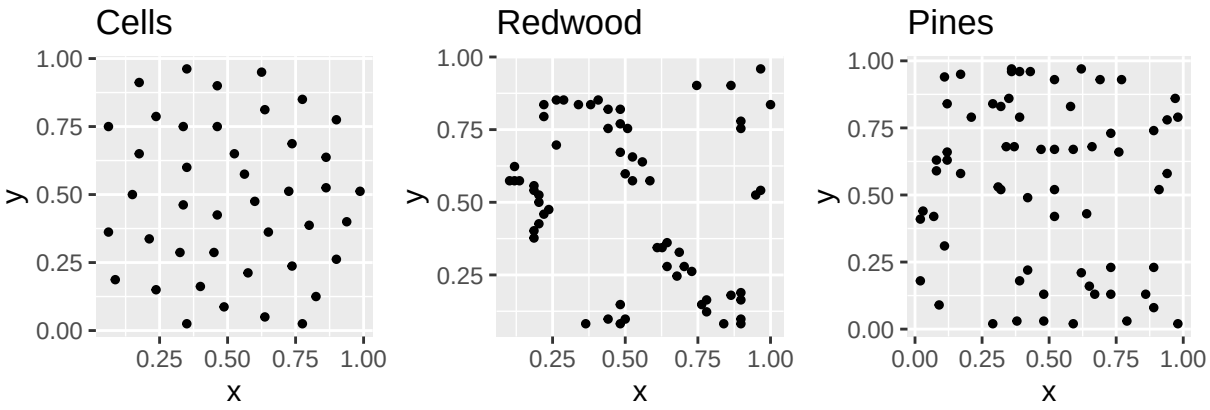
```
##      x    y
## 1 0.09 0.09
## 2 0.59 0.02
## 3 0.86 0.13
## 4 0.42 0.22
## 5 0.02 0.41
## 6 0.08 0.59
```

Problem 1

a)

```
display_point_patterns <- function(dataframes, titles) {
  # Plots point patterns for dataframes with point data
  # Args: Dataframes and titles as lists of the same lengths
  # Returns: None
  n <- length(dataframes)
  do.call(
    wrap_plots,
    lapply(1:n, function(i) {
      ggplot(dataframes[[i]]) + geom_point(aes(x, y), size=1) +
        xlab('x') + ylab('y') + labs(title=titles[i]) +
        coord_fixed()
    })
  )
}
```

```
plot_titles = c(
  "Cells",
  "Redwood",
  "Pines"
)

# png("figures/1_a.png", width=3600, height=1200, units="px", res=300) # Saving

display_point_patterns(list(
  cells,
  redwood,
  pines
), plot_titles)
```



For the Cells dataset, the points seem to be uniformly spread across the whole window. This would make sense since cells exist in very large quantities, and repel each other to a certain degree (https://www.hcplive.com/view/cells_repel). This will result in a uniform distribution of points when we pick a small window.

Comparing the last two dataset, it is evident that redwood trees have a tendency to grow in clusters along streams and other bodies of water (https://www.fs.usda.gov/research/treesearch/41153), which is very evident in the point patern plotted above. Pines, however, tend to follow a more random pattern of growth (https://www.researchgate.net/publication/227015905_The_problem_of_accuracy_in_environmental_analysis), giving the random pattern that is seen in the rightmost plot of Figure (FIGREF).

b)

The L-function is defined by

$$L(r) = \sqrt[d]{\frac{K(r)}{b_d}} \tag{1}$$

where

$$K(r) = \frac{1}{\lambda}\mathbb{E}_{\mathbf{0}}\left[N\left(B_r(\mathbf{0}\backslash\{\mathbf{0}\})\right)\right] \tag{2}$$

$$b_d = \nu(B_1(\mathbf{0})) \tag{3}$$

, $B_r(\mathbf{x})$ denotes a ball of radius r centered in $\mathbf{x}$ in the relevant space (here $\mathbb{R}^2$), and ν denotes the volume-function. The K-function $K(r)$ can be interpreted as the ratio between the expected number of points in a ball centered at $\mathbf{0}$ excluding the origin itself, *given* that there is a point at the origin, and the rate of the point process itself. Therefore one can say that the L-function is a variant of the K-function that takes the

curse of dimensionality, i.e. the fact that points get further apart the larger the dimension, into account. In $\mathbb{R}^2$, we have

$$L(r) = \sqrt{(\frac{K(r)}{\pi})} \tag{4}$$

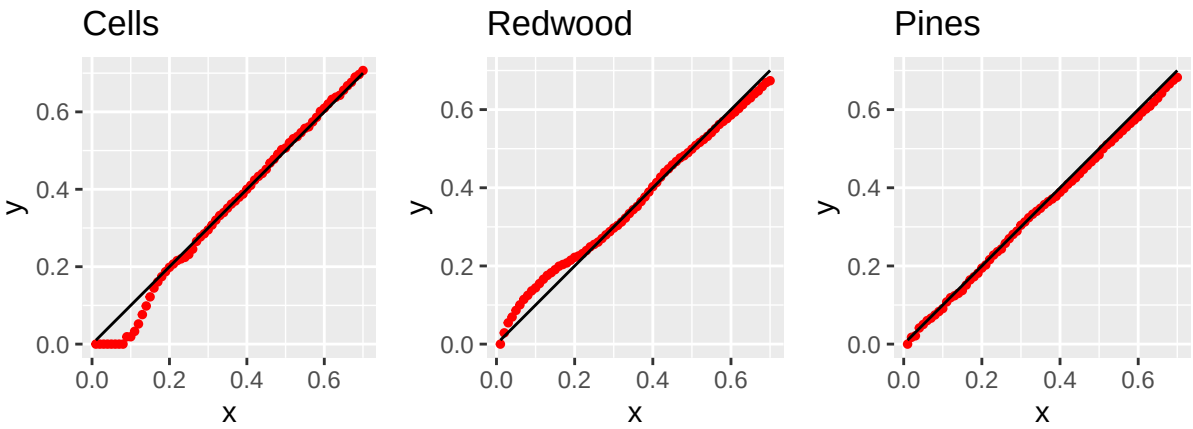
Estimation of the L-function:

```
pp.cells   <- ppinit("data/cells.dat")
pp.redwood <- ppinit("data/redwood.dat")
pp.pines   <- ppinit("data/pines.dat")

L.cells   <- data.frame(Kfn(pp.cells, fs=1.0)[1:2])
L.redwood <- data.frame(Kfn(pp.redwood, fs=1.0)[1:2])
L.pines   <- data.frame(Kfn(pp.pines, fs=1.0)[1:2])
```

L-function plotting:

```
display_L_functions <- function(dataframes, titles) {
  # Plots K-functions and affine lines with growth 1 for comparison
  # Args: Dataframes and titles as same length lists
  # Returns: None
  n <- length(dataframes)
  do.call(
    wrap_plots,
    lapply(1:n, function(i) {
      ni <- nrow(dataframes[[i]])
      df <- dataframes[[i]] %>% mutate(xline=1:ni/100, yline=1:ni/100)
      ggplot(df) + geom_point(aes(x, y), size=1, col="red") +
        xlab('x') + ylab('y') + labs(title=titles[i]) +
        geom_line(aes(xline, yline)) +
        coord_fixed()
    })
  )
}
```

```
# png("figures/1_b.png", width=3600, height=1200, units="px", res=300) # Saving

display_L_functions(list(
  L.cells,
  L.redwood,
  L.pines
), titles=plot_titles)
```



If, for small distances, the L-function diverges from the theoretical straight function (a straight line), then that is a implication that there is either clustering or repulsion in the dataset. For each of the datasets provided: - Cells: The L-function goes below the theoretical L-function, meaning that there are fewer points than expected in the ball with small r. This implies that there is repulsion in the point process, and a Gibbs process might be more suitable. - Redwood: The L-function is slightly higher than the theoretical function for small r, which is an implication of clustering, as discussed in 1a), and a Neymann-Scott process may therefore be more suitable. - Pines: The estimated L-function is mostly a straight line corresponding to the theoretical function, implying that a Poisson point process is a suitable model.

c)

For a homogeneous Poisson point process on a window $W \subseteq \mathbb{R}^m$, we have the convenient result:

$$[\mathbf{x}_1. \cdots, \mathbf{x}_n | N(W) = n] \overset{\text{i.i.d.}}{\sim} U(W) \tag{5}$$

In this case, $m = 2$ and $W = [0, 1] \times [0, 1]$. The points are therefore very easy to simulate, as they follow a uniform distribution on W.

```
simulate_R2_unif <- function(n, m) {
  # Simulates a homogeneous Poisson point process on R^2 in m realizations on [0, 1]x[0, 1].
  # Args: n (amount of points to simulate in each realization) and m (amount of realizations)
  # Returns: mxnx2 array.
  dims <- c(m, n, 2)
  realizations <- array(runif(m*n*2), dims)
  return(realizations)
}
```

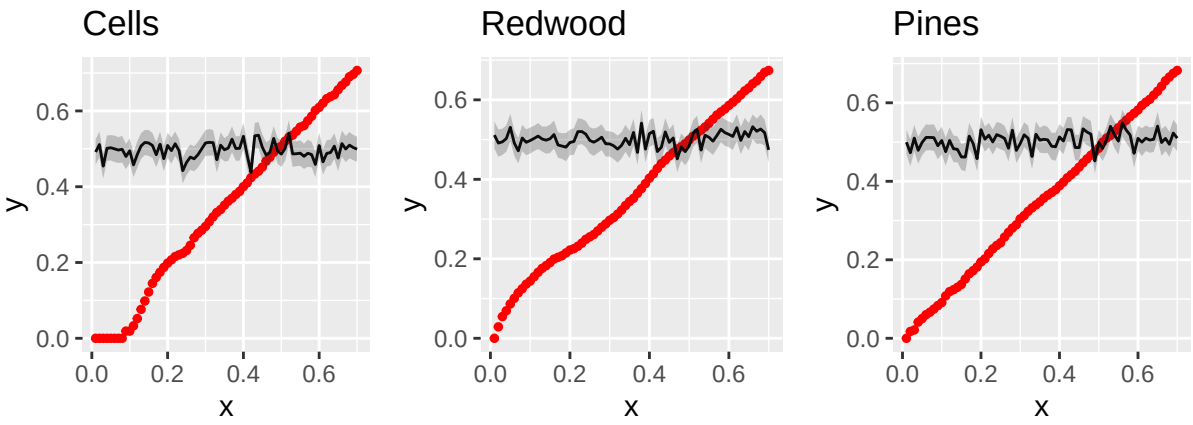
```r
calculate_L_functions <- function(realizations) {
  # Given realizations of some point process in the form of an array, calculates
  # the L-function for each realiztion and returns as an array.
  # Args: realizations (mxnxd array)
  #         - m: Amount of realzations
  #         - n: Amount of points per realization
  #         - d: Amount of dimensions
  # Returns: mxnx(d-1)-array
  return(realizations)
}
```

```r
L_function_stats <- function(L_functions) {
  # Given an array of m L-functions at n points, calculates mean and 90%
  # confidence intervals.
  # Args: L_functions (mxn)
  # Returns: Dataframe of L-functions
  stats <- L_functions %>% melt() %>%
    group_by(Var2) %>%
    summarize(mean=mean(value), CI90=1.64*sd(value/sqrt(length(value))))
  return(stats)
}
```

```r
simulate_R2_ppp_Lfn <- function(n, m) {
  # Simulates a Poisson point process conditional on N = n, and returning
  # a dataframe containing empirically estimated L-function and 90% CI.
  return(
    simulate_R2_unif(n, m) %>%
      calculate_L_functions() %>%
      L_function_stats()
  )
}
```

```r
display_L_functions_with_ppp_sim <- function(dataframes, titles) {
  # Plots K-functions and affine lines with growth 1 for comparison
  # Args: Dataframes and titles as same length lists
  # Returns: None
  n <- length(dataframes)
  do.call(
    wrap_plots,
    lapply(1:n, function(i) {
      ni <- nrow(dataframes[[i]])
      simi <- simulate_R2_ppp_Lfn(n=ni, m=100)
      df <- dataframes[[i]] %>% mutate(Lmean=simi$mean, LCI=simi$CI90)
      ggplot(df) + geom_point(aes(x, y), size=1, col="red") +
        geom_line(aes(x, Lmean)) +
        geom_ribbon(aes(x, ymin=Lmean-LCI, ymax=Lmean+LCI), alpha=0.25) +
        xlab('x') + ylab('y') + labs(title=titles[i]) +
        coord_fixed()
    })
  )
}
```

```
# png("figures/1_b.png", width=3600, height=1200, units="px", res=300) # Saving

display_L_functions_with_ppp_sim(list(
  L.cells,
  L.redwood,
  L.pines
), titles=plot_titles)
```



Problem 2

a)