STAT 302: Statistical Computing **Autumn 2023**

Final Project Description

Yikun Zhang *November 10, 2023*

Disclaimer: Parts of this project are modified from the midterm project of STAT 133 in Fall 2016 at UC Berkeley by Professor Deborah Nolan.

1 Introduction

Technological advancement has been changing our daily life. Nowadays, people rely heavily on their cell phones to receive messages and communicate with others around the world. All these cell phone communications are made possible by a concept called *ad hoc wireless network*[1], which has no centralized nodes or fixed structures. Instead, such a network structure is dynamically determined by the current locations and usages of all the communication devices in the nearby region. In other words, all the cell phones and other communication devices can create and join the ad-hoc wireless networks "on the fly".

A very basic aspect of ad hoc networks that people need to understand is how the communication and complete connectivity changes with respect to the broadcasting power. Increased power levels allow one to send a message over a larger distance.

2 Project Descriptions

This project consists of three main tasks.

1. Start by writing a function to generate nodes in a given (two-dimensional) ad hoc network. The function should meet the following conditions:
   (a) The name of the functions is `genNodes()`.
   (b) It has one input argument: `n`, which is the number of nodes to generate.
   (c) The return value of `genNodes()` is an $n \times 2$ matrix with the first column representing the x coordinates and the second column as the y coordinates of the nodes.

   Use the *acceptance-rejection sampling* in Lecture 6 to randomly generate the nodes of the network according to a pre-specified node density function `nodeDensity()`. This node density function has been written in R and can be downloaded on our Canvas page. It takes two input arguments x and y, which should be two numeric vectors of the same length. The function returns a numeric vector of values that are proportional to the node density value at the (x, y) pairs. To read this function into the current R environment, run

   ```
   source("nodeDensity.R")
   ```

   in an R code chunk of your .Rmd file. The two-dimensipnal contour plot and three-dimensional perspective plot of the node density function are shown in Figure 1 and Figure 2 below.

2. For a given configuration of nodes, we intend to find the smallest radius $R_c > 0$ such that the network is completely connected. Please write an R function to R_c for a given collection of nodes. This function has the following properties:

[1]See https://en.wikipedia.org/wiki/Wireless_ad_hoc_network.

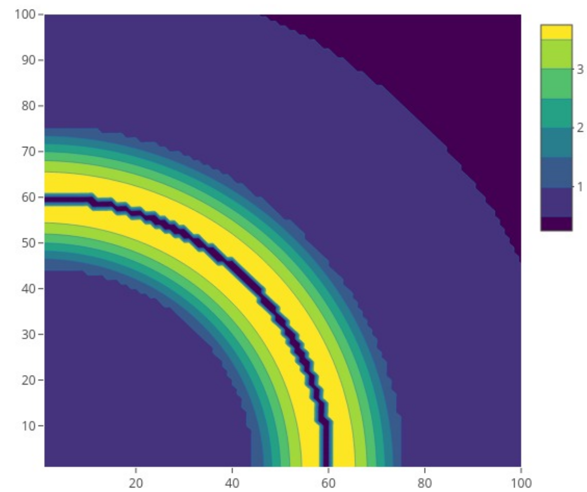

Figure 1: A two-dimensional contour plot of `nodeDensity()` in a region of interest. The contours are proportional to the density of nodes.

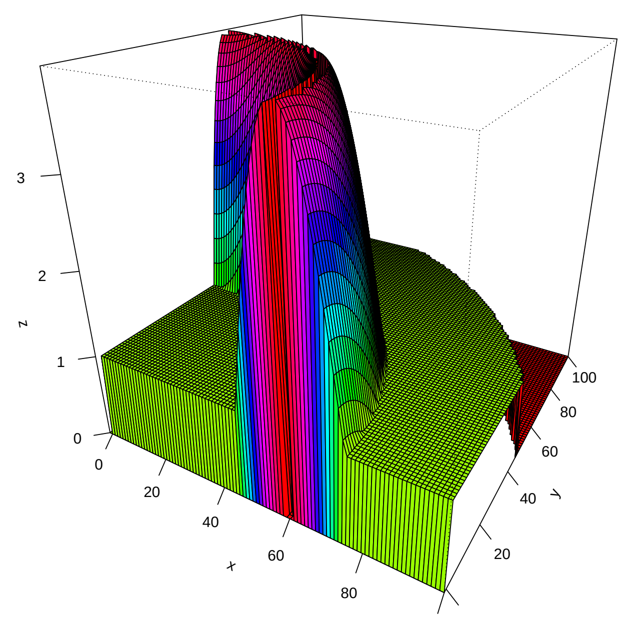

Figure 2: A three-dimensional perspective plot of `nodeDensity()` in a region of interest. Note that there are river curves passing through the center of the region so that no nodes are located near the river.

(a) The name of the function is called findRc().

(b) The first input argument is `nodes`, which is a two-column matrix of the x and y coordinates/locations of the nodes.

(c) The second input argument is `tol` with a default value of 0.05, which is the tolerance level for how close we need to get to the true value of R_c for the provided node configuration.

(d) The return value if a numeric variable that holds the value of R_c (or a value that is close to it under the tolerance level).

We are interested in finding the smallest power level that leads to a connected network. Here, we assume that the power level is proportional to the radius R of a circle centered on the node, and two nodes in the network are in direct communication if the distance between them is less than R.

The task of finding R_c relies on some probability arguments and a searching algorithm, which are described in Section 3. There, you will also be asked to write a couple of helper functions for findRc().

3. Select a range of sample sizes for n. Then, for each n, generate 1000 networks and find the value of R_c for each of these 1000 networks. Examine the distribution of these R_c values. There are some questions that you need to answer in words with supplementary plots:

   (a) How does R_c, the smallest radius such that the network is connected, change with different node configurations?

   (b) Explore the distribution of R_c (for each n). Is it symmetric, skewed, heavy-tailed, and multimodal?

   (c) For $n = 50$, plot the network of connected points for four of your 1000 n-node configurations corresponding *roughly* to the minimum, mean, median, and maximum values of R_c.

   Note that in order to plot a network, you should keep track of the n-node configurations that correspond to the minimum, mean, median, and maximum values of R_c. For example, if you keep track of the seed value at the start of each simulation, then, when you find that the m^{th} simulation corresponds to the median value for R_c, you can set the seed to the m^{th} seed that you have saved and recreate the nodes.

3 Mathematical Background

4 Useful Resources

Below are some functions that you may find useful:

- `set.seed()` and `.Random.seed` – used to set and save the seed for the random number generator.
- `dist()` – computes the pairwise distances between rows in a matrix or data frame.
- `eigen()` – computes eigenvalues and eigenvectors of a matrix. Be sure to read about all the arguments in its documentation.
- `eigs()` in the `RSpectra` package – calculate the largest k eigenvalues (you may also omit the computations of eigenvectors through `opts`).
- `expand.grid()` – a useful function when creating a grid from vectors, if you need to search over a grid.
- `Mod()` – calculates the magnitude of complex numbers.

Tests for some of the functions that you have been asked to write are available as `testsForAdHocProject.R` on our Canvas page as well. We also provide a `PracticeW.Random.seed.Rmd` file for an example of how to use `.Random.seed` to keep track of and reset the random number generator.

5 Submission Requirements

- You can work in a group with size ≤ 3 for this final project.
- Please submit both the knitted PDF and the original .Rmd file on Canvas. In your submitted files, write down the names of all your group members.
- *Only one member of your group needs to make the submission on Canvas.* If multiple members in your group have made the submission, we will randomly grade one of the writeups and assign the grades to students in this group.
- Everyone should make more or less contributions to the project if you are working in a group. *Highlight at the beginning of your .Rmd and knitted PDF files the contributions of every group member to the project.* If you work on the project by yourself, you can skip this part.

References

Y. Zhang. STAT 302 Lecture Slides, 2023. URL `https://zhangyk8.github.io/teaching/stat302_uw`.