Cornell STSCI 4520/5520 - Statistical Computing

Final Project Spring 2022 - A Decentralized Air Travel Network

Before doing anything, sit down and read this entire document from start to finish.

Introduction - They Spoke of Hubs and Point to Points

The largest US airlines operate "hub and spoke" air travel networks, maintaining "hubs" at a few major airports. Travelers originating at hub airports enjoy a wide selection of non-stop destinations. Travelers from smaller airports usually must travel along a "spoke" to connect to a hub before taking a second flight to their intended destination.

A "point to point" air travel network attempts to provide travelers with more non-stop destinations to and from smaller airports.

Your final project will analyze the existing air travel network using real flight data, and attempt to decentralize the network in order to provide more non-stop travel options to people who live far from hub airports.

Crucially, you will consider spatial information in your analysis. Like travelers in many rural areas, travelers from upstate New York routinely search for flights from several nearby airports with the hope of finding an itinerary whose cost or convenience outweighs the inconvenience of having to travel for an hour or more by bus or car to the airport. You may be aware that there are several airports within driving distance of Ithaca–Syracuse, Elmira, and Binghamton–in addition to our beautiful Ithaca Tompkins Airport.

Project Parameters

You may work individually or in teams of 2. If you choose to work with a team member, you must notify both of the TAs by Friday, April 22.

You may discuss the project with other teams, but you are not allowed to show your code to anyone not on your team.

Make sure your code is readable. This means

1. including comments, giving enough, but not too much information about what the code is doing,
2. using informative variable names,
3. using functions to avoid repetitive code,
4. using indentation where appropriate (4 spaces preferred),
5. keeping line widths below 80 spaces,
6. using whitespace to break up long strings of code.

See the code below for examples of readable code.

The project is due on **Monday, May 16, 2022** on canvas.

Products

R package

You will turn in an R package in tar.gz format (more on that next week).

Part of your grade will depend on whether we can install and run the code in your R package with minimal effort, so make sure you test that it works!

Excessive use of external R packages will be penalized. Feel free to make judicious use of a handful of packages (e.g. `sp`, `maps`, and `fields`). A file that loads 10 different packages will be viewed unfavorably.

Each function in your package should be documented with Roxygen comments.

Each function in your package should have a meaningful unit test, written with `testthat`, and all of the tests should pass.

The R package should, at a minimum, contain the following components:

1. A data frame called `airport_data` with at least three columns:

   - all of the unique airport codes in `datasets/airline_2019-07-01.csv`, sorted alphabetically
   - the longitude coordinates of each airport
   - the latitude coordinates of each airport

   You can use the information in `datasets/all-airport-data.csv` to help build your data frame, but note that a few airports are missing, so you'll have to track down the coordinates of those airports.

2. A function to compute the *great circle distance* between pairs of longitude/latitude coordinates. It should take the following form:

   ```r
   # lonlat1 - an n1 x 2 matrix, first column longitudes, second column latitudes
   # lonlat2 - an n2 x 2 matrix, first column longitudes, second column latitudes
   # distmat - an n1 x n2 matrix of distances. distmat[i,j] is the distance between
   #           lonlat1[i,] and lonlat2[j,]
   distfun <- function( lonlat1, lonlat2 ){
       # insert code here to compute distances
       return(distmat)
   }
   ```

3. A function to find all airports in `airport_data` within `x` miles of a longitude latitude point

   ```r
   # lonlat - a length-2 vector or a 1 x 2 matrix containing a longitude and latitude
   # airport_data - your airport data frame
   # x - distance
   # ports - character vector of airport codes within x miles of lonlat
   airports_nearby <- function( lonlat, airport_data, x = 75 ){
       # insert code here to find airports
       return( ports )
   }
   ```

4. Using the data in `datasets/airline_2019-07-01.csv`, create a list containing information about direct connections from each airport. A direct connection between airports A and B exists if there is at least one flight from A to B in the dataset. The object should be a list of character vectors called `direct_from`. The `j`th list element of the list should contain a character vector of the nonstop destinations reachable from the `j`th airport in `airport_data`. The list should be indexable via the airport code (`direct_from[["ITH"]]` are the airports you can fly to from Ithaca). It should also be symmetric; if you can fly from airport A to airport B, you should also be able to fly from airport B to airport A (see next problem). Achieve symmetry by adding the opposite connection when there is an asymmetry.

5. A function for testing whether a `direct_from` object is symmetric. It should return a list whose first element is TRUE or FALSE (whether or not it is symmetric), and whose second element is an `n x 2` character matrix giving the `n` asymmetric connections.

   ```r
   # direct_from - air travel network list
   # symm - TRUE or FALSE
   # conn - n x 2 matrix of asymmetric connections.
   ```

```
    check_symmetric <- function( direct_from ){
        # insert code here to check for symmetry
        return( list(symm = symm, conn = conn ) )
    }
```

Analysis - Rmarkdown vignette

You should use your package to conduct an analysis of the air travel network, and design a different network meeting certain criteria. This analysis should go in the package's `vignettes/` directory in an Rmarkdown file.

Vignettes are supposed to run quickly. However, some of the steps below may take hours to run. For those parts of the analysis, run them in separate scripts. Those scripts should save objects that you load in the Rmarkdown file. The separate scripts should go in the package's `inst/scripts/` directory.

1. Make a map of the airport data. Try to include information about the airport codes and the number of non-stop destinations from each airport in your map. You might find the `maps` package or `ggplot2` useful for this.
2. Below is code that defines a grid of points covering the continental USA and figures out which points are inside the continental USA.

   Given a point on the grid, figure out which other points are reachable by

   a. Traveling a distance of `2x` (as the crow flies), OR
   b. Traveling less than `x` miles to an airport, taking one flight, and traveling less than `x` miles to another point.

   Plot the points reachable from the gridpoint closest to Ithaca. Use `x = 75` miles and `x = 100` miles.

   What proportion of grid points are reachable from the Ithaca gridpoint?

```
# get the polygon defining the continental USA
library("maps")
library("sp")
usa_poly <- map("usa")
usa_poly$x <- c(NA,usa_poly$x,NA)
usa_poly$y <- c(NA,usa_poly$y,NA)
nai <- which( is.na( usa_poly$x ) )

# define a grid of longitude and latitude points
n1 <- 180
n2 <- 90
lo <- seq(usa_poly$range[1], usa_poly$range[2], length.out = n1)
la <- seq(usa_poly$range[3], usa_poly$range[4], length.out = n2)
lonlat_grid <- as.matrix( expand.grid( lo, la ) )

# figure out which points are inside USA
in_usa <- rep(FALSE, nrow(lonlat_grid))
for(j in 1:(length(nai)-1)){

    in_this <- sp::point.in.polygon(
        lonlat_grid[,1],
        lonlat_grid[,2],
        usa_poly$x[ (nai[j]+1):(nai[j+1]-1) ],
        usa_poly$y[ (nai[j]+1):(nai[j+1]-1) ]
    )
    in_usa <- in_usa | in_this
}

# subset to the points in USA
lonlat_usa <- as.matrix( lonlat_grid[ in_usa, ] )
```

3. Repeat the previous exercise for every point in `lonlat_usa`. Make plots of the proportion of reachable grid points from every point in `lonlat_usa` for `x = 75` miles and `x = 100` miles. What is the average proportion of reachable grid points for `x = 75` and `x = 100`?

   This part may take longer. If so, do the analysis in a separate script, save the results, and then load them into your vignette.

4. Now your task is to redesign the airline network to improve the average proportion of reachable grid points. This means creating new `direct_from` objects. Your new `direct_from` objects should remain symmetric. This part will definitely take longer to run, so put it in one or more separate scripts.

   Redesign networks separately under the following 3 scenarios:

   a. The total number of direct connections must remain constant. In other words, if you add a connection, you must remove a different connection.

   b. The total number of connections to and from each airport must remain constant.

   c. You can add or subtract up to two connections from each airport, leaving the other connections unchanged, and keeping the total number of connections constant.

   Your redesigned networks should be saved in your R package as data objects with the names `direct_from_a`, `direct_from_b`, and `direct_from_c`, corresponding to the scenarios above.

   For each redesigned network, make a map of the reachable grid points from Ithaca, make a map of the proportion of reachable grid points from every grid point, and report the average number of reachable grid points.

   Some hints:

   It's wasteful if both Ithaca and Elmira have flights to Detroit because a resident living in the area could drive to either airport if they want to go to eastern Michigan. It would be better if the list of non-stop destinations from Ithaca and Elmira did not overlap.

   It's also wasteful if you can fly from Ithaca to both O'Hare and Midway airports, since they are both in Chicago.

   Since you have a computer, your best strategy might be to propose random changes to the airport network, evaluate whether the change improves the statistic, and keep the change if it does. This can be iterated many times to search for an improved network.