Spatial Analysis for Retail Site Selection

Felipe O. Cerezer

2024-08-18

1. Required Libraries

We will use the following R packages: - `sf` for handling spatial data - `raster` for working with raster data - `spatstat` for point pattern analysis - `dplyr` for data manipulation

```
## Install required packages if not already installed
#install.packages(c("sf", "raster", "spatstat", "dplyr"))

## Load libraries
library(sf)
```

```
## Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(raster)
```

```
## Loading required package: sp
```

```
library(spatstat)
```

```
## Loading required package: spatstat.data

## Loading required package: spatstat.univar

## spatstat.univar 3.0-0

## Loading required package: spatstat.geom

## spatstat.geom 3.3-2

##
## Attaching package: 'spatstat.geom'

## The following objects are masked from 'package:raster':
##
##     area, rotate, shift

## Loading required package: spatstat.random

## spatstat.random 3.3-1

## Loading required package: spatstat.explore

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:raster':
##
##     getData
## spatstat.explore 3.3-1

## Loading required package: spatstat.model

## Loading required package: rpart

## spatstat.model 3.3-1

## Loading required package: spatstat.linnet

## spatstat.linnet 3.2-1

##
## spatstat 3.1-1
## For an introduction to spatstat, type 'beginner'
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:nlme':
##
##     collapse

## The following objects are masked from 'package:raster':
##
##     intersect, select, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ape)
```

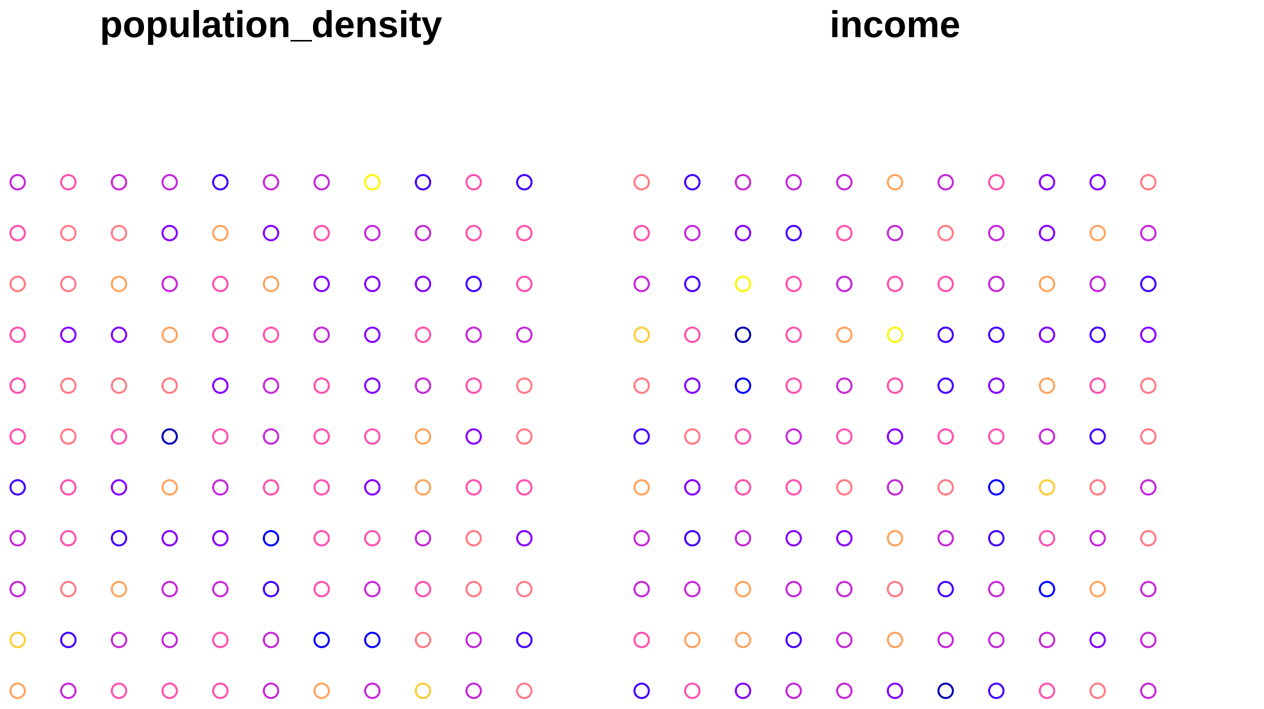
```
##
## Attaching package: 'ape'

## The following object is masked from 'package:dplyr':
##
##     where

## The following objects are masked from 'package:spatstat.geom':
##
##     edges, rotate

## The following objects are masked from 'package:raster':
##
##     rotate, zoom
```

2. Generating Fake Data

First, I will generate some fake spatial and demographic data, including locations of existing stores, population density, and average income.

```r
set.seed(42) # For reproducibility

# Create a simple grid of points representing potential store locations
```

```r
locations <- expand.grid(x = seq(0, 10, by = 1), y = seq(0, 10, by = 1))

# Convert to sf object
locations_sf <- st_as_sf(locations, coords = c("x", "y"), crs = 4326)

# Generate random population density and income data
locations_sf$population_density <- rnorm(nrow(locations_sf), mean = 1000, sd = 300)
locations_sf$income <- rnorm(nrow(locations_sf), mean = 50000, sd = 10000)

# Plot the data
plot(locations_sf)
```



Explanation: Here, I create a grid of potential store locations and generate random values for population density and income at these locations.

3. Spatial Autocorrelation

Next, I will calculate spatial autocorrelation to understand the spatial dependency of population density and income. I will use Moran's I, a measure of spatial autocorrelation.

```r
# Calculate spatial weights matrix using inverse distance
coords <- st_coordinates(locations_sf)
dist_matrix <- as.matrix(dist(coords))
inv_dist_matrix <- 1 / dist_matrix
diag(inv_dist_matrix) <- 0

# Calculate Moran's I for population density
moran_population <- Moran.I(locations_sf$population_density, inv_dist_matrix)
print(moran_population)
```

```
## $observed
## [1] -0.01228229
```

```
##
## $expected
## [1] -0.008333333
##
## $sd
## [1] 0.008956444
##
## $p.value
## [1] 0.6592809
```

```r
# Calculate Moran's I for income
moran_income <- Moran.I(locations_sf$income, inv_dist_matrix)
print(moran_income)
```

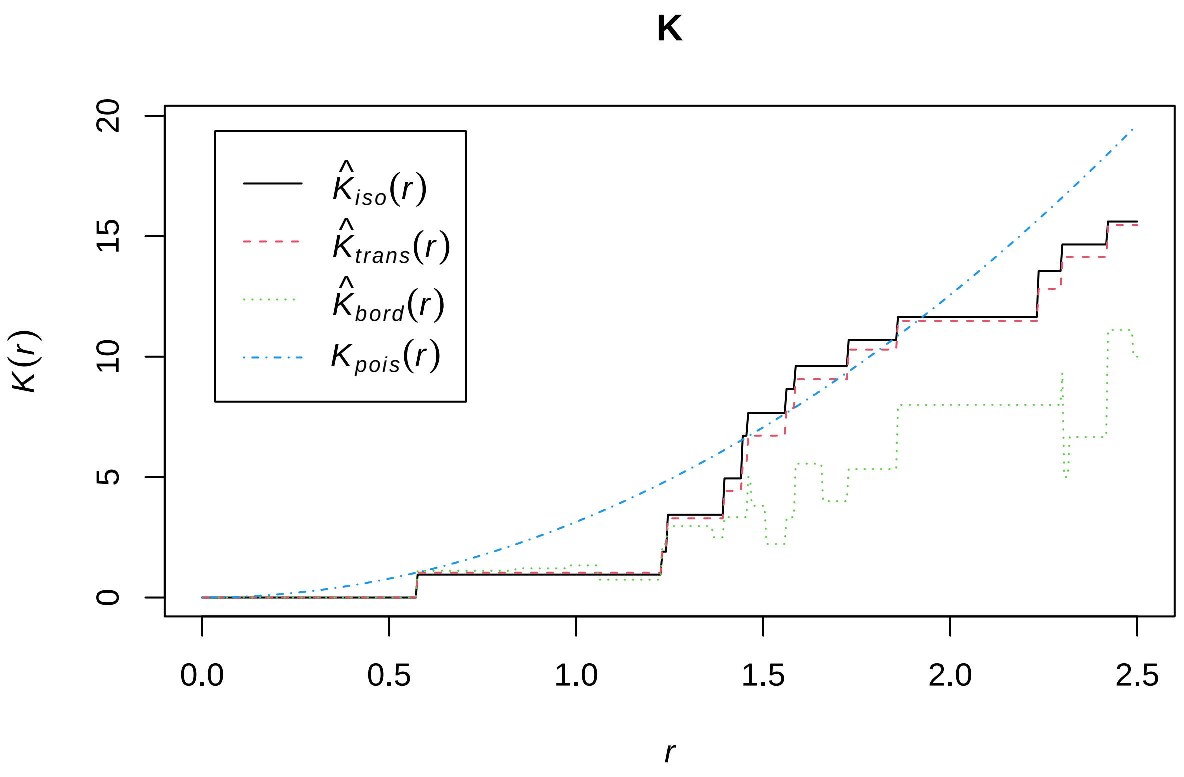
```
## $observed
## [1] -0.02244948
##
## $expected
## [1] -0.008333333
##
## $sd
## [1] 0.008979206
##
## $p.value
## [1] 0.115929
```

Explanation: Moran's I is useful to understand if high or low values of population density and income are clustered in space. A significant Moran's I value indicates that the variable is not randomly distributed but shows a spatial pattern.

4. Point Pattern Analysis

I will analyze the pattern of existing store locations to determine if they are clustered, randomly distributed, or regularly spaced.

```r
# Generate random existing store locations
n_stores <- 15
existing_stores <- data.frame(x = runif(n_stores, 0, 10), y = runif(n_stores, 0, 10))
existing_stores_sf <- st_as_sf(existing_stores, coords = c("x", "y"), crs = 4326)

# Convert to ppp object for spatstat analysis
store_ppp <- as.ppp(st_coordinates(existing_stores_sf), W = owin(c(0, 10), c(0, 10)))

# Perform Ripley's K analysis
K <- Kest(store_ppp)
plot(K)
```



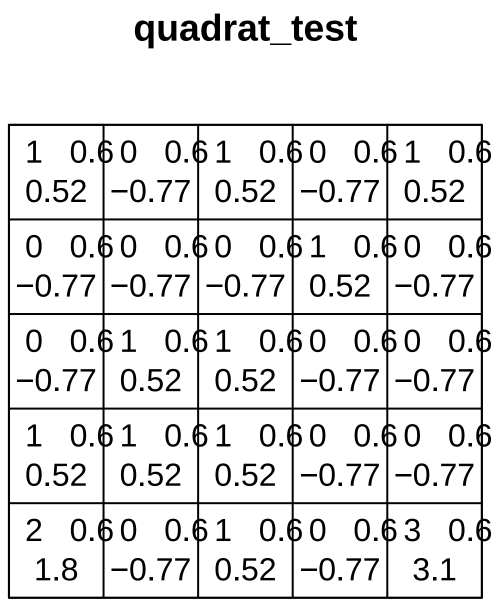
```
# Perform Quadrat test for spatial uniformity
quadrat_test <- quadrat.test(store_ppp, nx = 5, ny = 5)
```

```
## Warning: Some expected counts are small; chi^2 approximation may be inaccurate
```

```
print(quadrat_test)
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  store_ppp
## X2 = 23.333, df = 24, p-value = 0.9996
## alternative hypothesis: two.sided
##
## Quadrats: 5 by 5 grid of tiles
```

```
plot(quadrat_test)
```

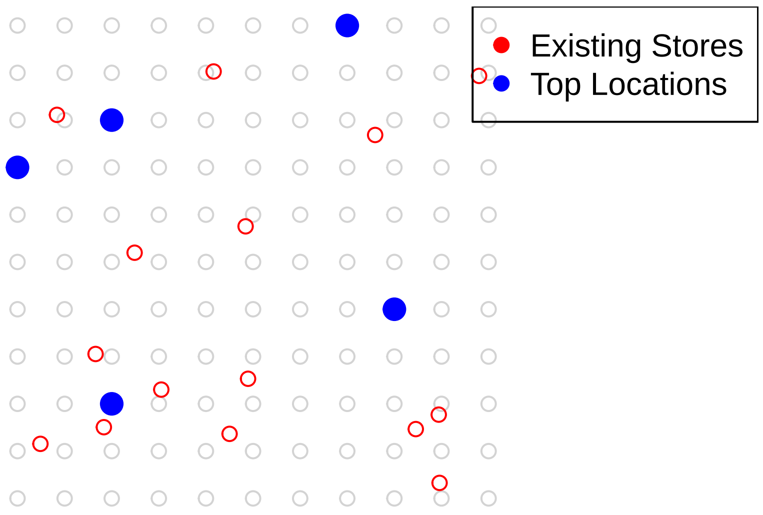


Explanation: Ripley's K function and the Quadrat test allow us to determine the spatial pattern of existing stores. This can be used to understand whether stores are clustered, randomly distributed, or evenly spaced, which can inform where new stores should be placed to avoid supersaturation.

5. Identifying Optimal Retail Locations

Now, I will identify the optimal locations for new retail stores by combining the demographic data (population density and income) with the results of the previous spatial analyses.

```r
# Define a simple scoring system: higher population density and income is better
locations_sf$score <- scale(locations_sf$population_density) + scale(locations_sf$income)

# Select top 5 locations based on the score
top_locations <- locations_sf %>%
  arrange(desc(score)) %>%
  slice(1:5)

# Plot the top locations
plot(st_geometry(locations_sf), col = "lightgrey")
plot(st_geometry(existing_stores_sf), col = "red", add = TRUE)
plot(st_geometry(top_locations), col = "blue", pch = 19, cex = 1.5, add = TRUE)
legend("topright", legend = c("Existing Stores", "Top Locations"), col = c("red", "blue"), pch = 19)
```



Explanation: I combined the population density and income data into a single score that ranks each location based on its suitability for a new store. The top 5 locations with the highest scores are selected as the best options for new stores. This final visualization shows the selected optimal locations on the map (in blue), alongside existing stores (in red).