Spatial Autocorrelation Tutorial

Geog 418

October 20, 2024

Introduction

Spatial autocorrelation is a key concept in geography that quantifies the degree to which different spatial objects are correlated based on their geographical proximity. This notion is rooted in the foundational principle of geography: "Everything is related to everything else, but near things are more related than distant things" (Goodchild, 2004). Positive spatial autocorrelation indicates that similar values tend to cluster together in space, while negative autocorrelation suggests that similar values are dispersed across a wider area. This clustering or dispersion can provide insights into underlying spatial patterns and relationships.

Various methods exist to measure spatial autocorrelation, but this tutorial will focus primarily on two: local and global Moran's I. Moran's I is particularly valuable for geographers as it reveals spatial relationships that may not be immediately apparent when examining a map. By applying Moran's I, we can uncover hidden spatial structures within the data, leading to a deeper understanding of the geographical phenomena at play.

A critical aspect of our analysis is the use of census data, which serves as an excellent foundation for studying spatial autocorrelation. Census data is rich in socio-economic information, including variables such as income levels and language spoken. This wealth of data makes it particularly suitable for our analysis, as it not only covers the entire country of Canada but also allows for extensive exploration of spatial relationships across diverse regions. The interconnected nature of the variables in the census data—such as the correlation between income and language proficiency—enables us to conduct robust analyses that reveal significant patterns and insights.

In R, a library is a collection of tools and functions that enhances the functionality of the base language, allowing users to perform complex analyses more efficiently. Each library provides pre-written code that users can leverage, eliminating the need to develop custom functions from scratch. This not only streamlines the coding process but also enables researchers to focus on their analysis rather than on the intricacies of programming.

In this tutorial, we will utilize several libraries that will aid in conducting our analysis and effectively presenting our results. By harnessing these powerful tools, we can enhance our productivity and clarity, making it easier to draw meaningful insights from our data.

```
#install packages if not already installed:
#install.packages("knitr")
#install.packages("rgdal")
#install.packages("tmap")
#install.packages("spdep")
#install.packages("raster")
#install.packages("shinyjs")
#install.packages("e1071")
#install.packages("st")
#install.packages("sf")
```

```r
#Load in libraries:
library("st")
library("sf")
library("e1071")
library("knitr")
library("shiny")
library("tmap")
library("spdep")
library("sp")
library("raster")
```

To analyze our data using R, the first step is to import the datasets into the program. The two lines of code below will read both the CSV file containing the census data and the shapefile representing geographical boundaries. Keep in mind that the file paths provided will differ from yours, so you'll need to replace them with the paths where you downloaded the data.

The CSV file we will be importing contains the complete census data for Canada. While we won't be using the entire dataset for our analysis, we'll demonstrate how to filter and trim it down as we proceed. For now, we'll read in the full dataset to ensure we have all the information at our disposal.

The shapefile we are loading represents the entirety of Canada as divided into census regions, allowing us to visualize and analyze the data geographically.

```r
#From the working dir read in the csv
csv <- read.csv("C:/Users/Turner/Desktop/Geog418/a3/Assignment3_Data/ucgsJQnBVLvP_data.csv")

#Data source is the working dir (where the layer is), layer is the name of the file (without .shp)
shp <- st_read("Assignment3_Data/lda_000a16a_e.shp")
```

```
## Reading layer 'lda_000a16a_e' from data source
##   'C:\Users\Turner\Desktop\Geog418\a3\Assignment3_Data\lda_000a16a_e.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 56590 features and 22 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: 3658201 ymin: 658873 xmax: 9019157 ymax: 6083005
## Projected CRS: PCS_Lambert_Conformal_Conic
```

Next, we want to clean up our data and make it easier to use. First we will create a vector of the column names so we understand what columns refer to what data. Then, we will remove any unwanted rows and merge the result with our spatial polygon data frame, and finally, we will subset to only the city of interest, for this analysis 'Fort St. John'. The last step is to turn any absolute count data into a rate for mapping and analysis.(PROVIDE MAP OF STUDY AREA)

```r
#New column names
cols <- c("GEO UID", "Province code", "Province name", "CD code",
          "CD name", "DA name", "Population", "Land area",
          "Median total income", "Income Sample Size", "French Knowledge",
          "Language Sample Size")

#Apply those names to dataframe
colnames(csv) <- cols
```

```r
#Add column to count number of ID charactors
csv$len <- nchar(csv$`GEO UID`)

#Remove IDs with less than 8 numbers
csv_clean <- subset(csv, csv$len == 8)

#Merge spatial and aspatial data
census_DAs <- merge(shp, csv_clean,
                    by.x = "DAUID",
                    by.y = "GEO UID",
                    all.x = TRUE)

#Subset for Kamloops
Municp <- subset(census_DAs, census_DAs$CMANAME == "Fort St. John")

#Convert to rate
Municp$PercFrench <- (Municp$`French Knowledge`/Municp$`Language Sample Size`)*100
```

Before we can start to analyze our data, we need to be sure that the data we are looking at is relevant. Often, missing data in the form of NA or 0 values can change the results of an analysis. To make sure that the polygons we are looking at actually contain values for our variables of interest. To do this we can remove any polygon that contains an NA value for either median total income or knowledge of French.

```r
#Remove Income NA
Income_noNA <- Municp[which(!is.na(Municp$`Median total income`)),]

#Remove French NA
French_noNA <- Municp[which(!is.na(Municp$`PercFrench`)),]
```

Next, we will take a closer look at the two variables we are interested in: Median total income and Percentage of respondents with French language knowledge. We will look at some descriptive stats and do a final check for NA values in the data.

```r
#Calculate descriptive stats for Income
meanIncome <- mean(Income_noNA$`Median total income`)
stdevIncome <- sd(Income_noNA$`Median total income`)
skewIncome <- skewness(Income_noNA$`Median total income`)

#Calculate descriptive stats for French
meanFrench <- mean(French_noNA$`PercFrench`)
stdevFrench <- sd(French_noNA$`PercFrench`)
skewFrench <- skewness(French_noNA$`PercFrench`)

#Create dataframe for display in table
data <- data.frame(Variable = c("Income", "French Language"),
                   Mean = c(round(meanIncome,2), round(meanFrench,2)),
                   StandardDeviation = c(round(stdevIncome,2), round(stdevFrench,2)),
                   Skewness = c(round(skewIncome,2), round(skewFrench,2)))

#Produce table
kable(data, caption = paste0("Descriptive statistics for selected ", 2016, " census variables"))
```

Table 1: Descriptive statistics for selected 2016 census variables

Variable	Mean	StandardDeviation	Skewness
Income	49532.63	7069.35	0.01
French Language	5.27	2.82	0.70

Generating a map in R can involve extensive coding, but the use of libraries significantly simplifies the process. We will now utilize the tmap package to create two maps. One map will illustrate the area surrounding Fort Saint John, highlighting median income levels, while the other will depict the percentage of the population with knowledge of the French language.

We will create both maps using different datasets: the income_noNA dataset for the income map and the French_noNA dataset for the French language map. The steps to generate each map are largely similar. Follow the below steps.

Steps to Create a Map

1) Start by defining the dataset for the map. In our case, we will use the two datasets mentioned above.
2) Next, we will specify the variable that the polygons will represent on the map. This allows us to visualize important geographical distinctions.
3) Each map will have a descriptive title. We will also select a style that best represents the data visually.

Choose Colors: We will pick a color palette for the polygons that enhances readability and a distinct color for the border lines.

Position the Legend: Finally, we will choose the location for the legend on the map, ensuring it is clear and easy to interpret.

If you'd like to experiment with different colors and styles, you can use the following command in your R console:

tmaptools::palette_explorer()

This tool allows you to explore various color palettes and provides the corresponding code to apply your selections to the maps.

By leveraging the tmap package, we can efficiently create visually appealing and informative maps that enhance our understanding of the socio-economic landscape in Fort Saint John.

```
#Choose a pallete
# tmaptools::palette_explorer() #Tool for selecting pallettes

#Map median Income
map_Income <- tm_shape(Income_noNA) +
  tm_polygons(col = "Median total income",
              title = "Median total income",
              style = "jenks",
              palette = "BuGn", n = 6,
              border.alpha = 0,
              colorNA = "grey") +
  tm_layout(legend.position = c("RIGHT", "TOP"))

#Map French Knowledge
map_French <- tm_shape(French_noNA) +
```

```
  tm_polygons(col = "PercFrench",
              title = "Percentage with \n French Knowledge",
              style = "jenks",
              palette = "BuGn", n = 6,
              border.alpha = 0,
              colorNA = "grey") +
  tm_layout(legend.position = c("RIGHT", "TOP"))

#Print maps side by side
tmap_arrange(map_Income, map_French, ncol = 2, nrow = 1)
```

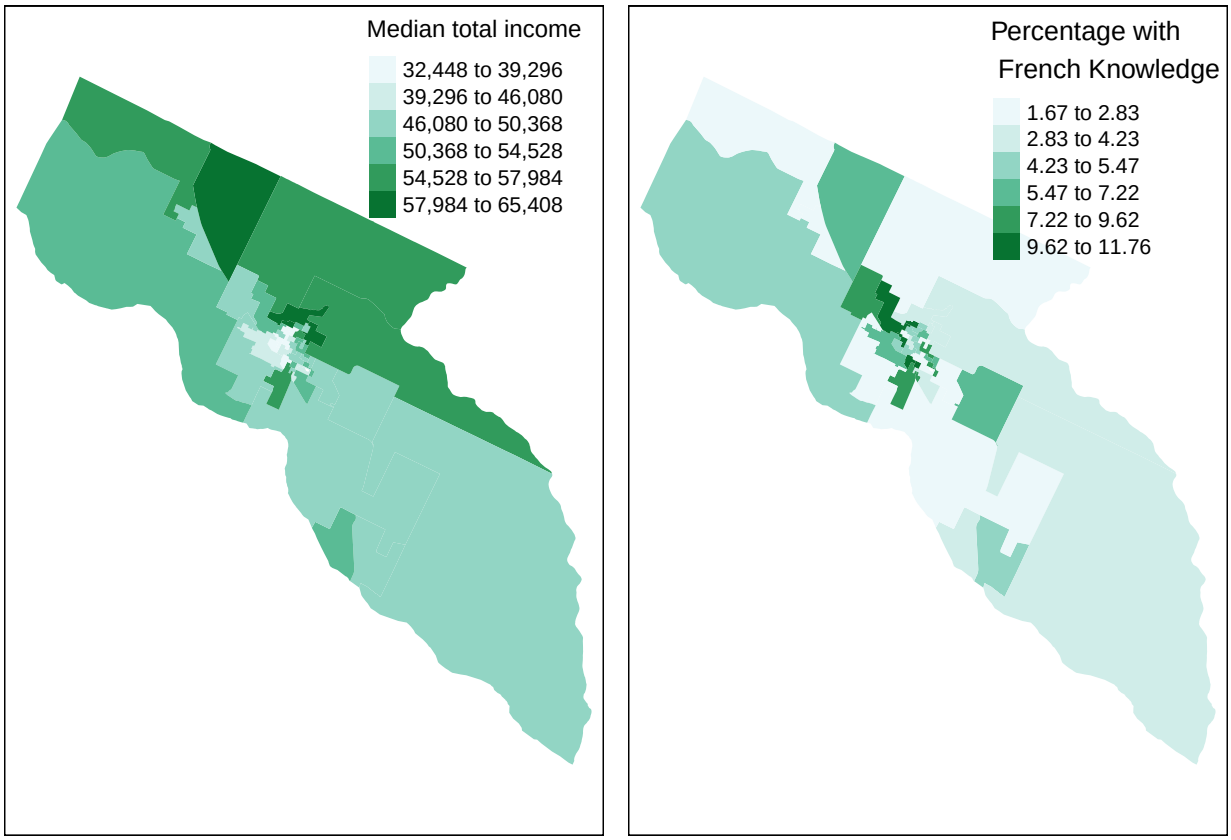

Figure 1: Fort Saint John census dissemination areas showing median total income (left) and percentage of respondants with knowledge of French (right).

Neighbourhood matrix

A weighted neighbourhood matrix is a tool used by geographers to illustrate the spatial relationships among various geographic objects. These matrices play a crucial role in analyzing how one object may be influenced by its neighbouring units. For instance, spatial autocorrelation tests, such as Moran's I, require a nearest neighbour matrix to evaluate the similarity between an object and its neighbours.

There are several types of weight matrices available, each providing different insights. One common example is the K-nearest neighbours method, which constructs a matrix based on a specified number of closest neighbours for each object. This approach can be mathematically represented as follows:

In this tutorial, we will focus on rook and queen adjacency to define our weight matrices.

Rook and queen adjacency draw inspiration from chess to determine neighbour relationships. In chess, a rook can move vertically and horizontally, whereas a queen can move in all directions: vertically, horizontally, and diagonally. Consequently, a rook can have a maximum of four adjacent neighbours, while a queen can have up to eight.

This distinction allows researchers to choose the complexity of their spatial analysis. Rook adjacency is useful for examining straightforward spatial relationships, whereas queen adjacency is better suited for analyzing more intricate neighbourhood interactions.

The code to create a list of neighbours in R is very simple thanks to the poly2nb() function in the 'spdep' package. If we want to change from default queen weighting to rook weighting in our selection, we simply change the 'queen = TRUE' to 'queen = FALSE'.

The code below generates two adjacency matrices for both the income and French-speaking datasets. Each dataset will include both rook and queen adjacency matrices.

```
#Income Neighbours - Queens weight
Income.nb <- poly2nb(Income_noNA)
# Use st_coordinates to get the coordinates
Income.net <- nb2lines(Income.nb, coords=st_coordinates(st_centroid(Income_noNA)))
crs(Income.net) <- crs(Income_noNA)


#Income Neighbours - Rooks weight
Income.nb2 <- poly2nb(Income_noNA, queen = FALSE)
Income.net2 <- nb2lines(Income.nb2, coords=st_coordinates(st_centroid(Income_noNA)))
crs(Income.net2) <- crs(Income_noNA)

#French Neighbours - Queens weight
French.nb <- poly2nb(French_noNA)
French.net <- nb2lines(French.nb, coords=st_coordinates(st_centroid(Income_noNA)))
crs(French.net) <- crs(French_noNA)

#French Neighbours - Rooks weight
French.nb2 <- poly2nb(French_noNA, queen = FALSE)
French.net2 <- nb2lines(French.nb2, coords=st_coordinates(st_centroid(Income_noNA)))
crs(French.net2) <- crs(French_noNA)
```

The code below generates three distinct maps of the area surrounding Fort Saint John, each illustrating spatial connections between different regions. The maps utilize varying line thicknesses to represent different types of adjacency: lines are thinner for queen adjacency and thicker for rook adjacency.

1) The first map, highlighted in green, displays queen adjacency with thin lines connecting adjacent areas.
2) The second map features thicker lines in red, representing rook adjacency and emphasizing direct, edge-sharing connections.
3) The final map combines both types of adjacency, showcasing thick lines for rook connections and thin lines for queen connections, allowing for a comprehensive comparison of the spatial relationships. These visualizations demonstrate how different adjacency criteria influence the connectivity of regions around Fort Saint John.

```
#Make queens map
IncomeQueen <- tm_shape(Income_noNA) + tm_borders(col='lightgrey') +
  tm_shape(Income.net) + tm_lines(col='green')
```

```r
#Make rooks map
IncomeRook <- tm_shape(Income_noNA) + tm_borders(col='lightgrey') +
  tm_shape(Income.net2) + tm_lines(col='blue', lwd = 2)

#Make combined map
IncomeBoth <- tm_shape(Income_noNA) + tm_borders(col='lightgrey') +
  tm_shape(Income.net) + tm_lines(col='yellow', lwd = 2) +
  tm_shape(Income.net2) + tm_lines(col='yellow', lwd = 2)

#Print maps in a three pane figure
tmap_arrange(IncomeQueen, IncomeRook, IncomeBoth, ncol = 3, nrow = 1)
```

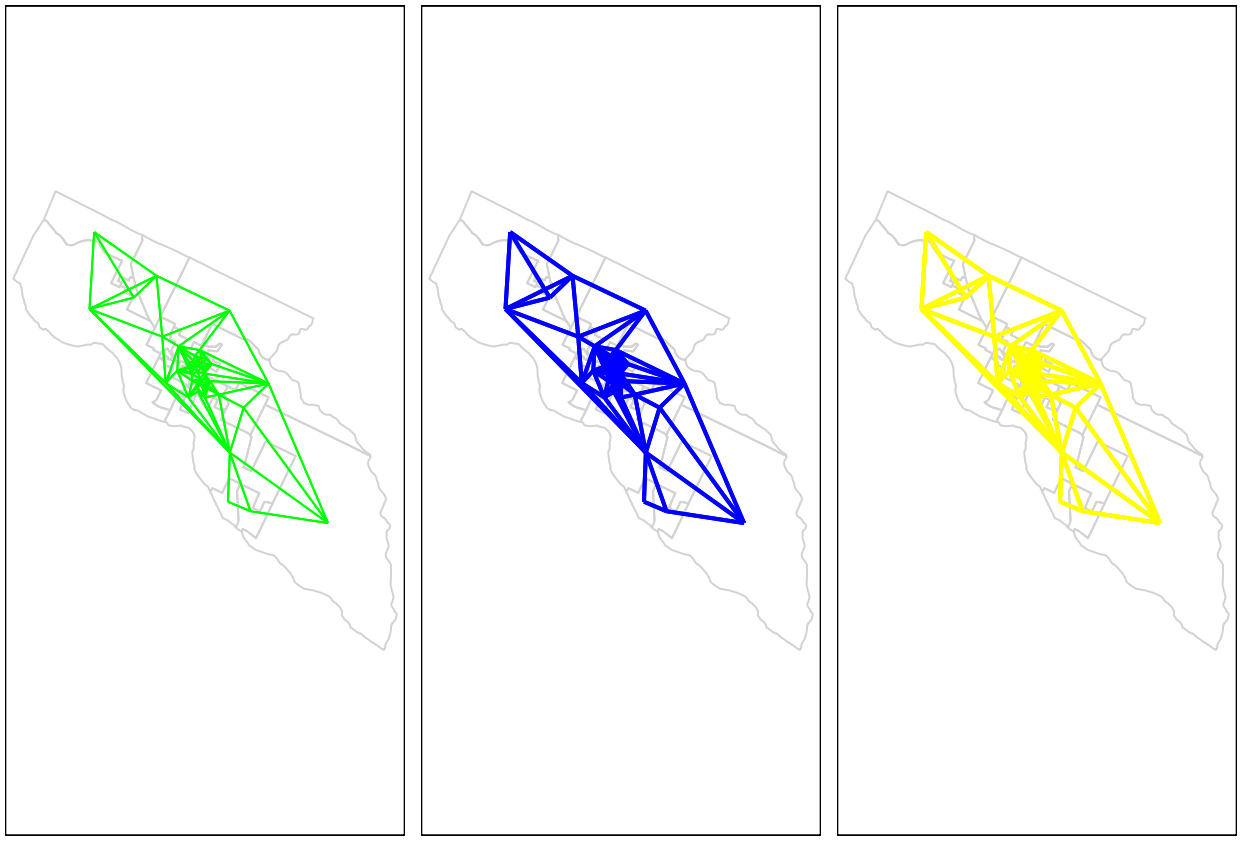

Figure 2: Fort Saint John census dissemination areas showing median total income neighbours queens weight (left) rooks weight (middle) and the combination of the two (right).

In spatial analysis, weights are defined by their "style" (or type) and can include options such as "B," "W," and "C." The B weights matrix employs a binary weighting scheme, assigning a weight of 1 to each neighbour and a weight of 0 to all other polygons. This is the simplest form of weighting, making it easy to identify neighbouring relationships. In contrast, the W weights matrix uses a row-standardized approach, where each neighbour is given equal weight, resulting in weights that sum to 1 for each observation. This allows for more nuanced interpretations of spatial relationships. Lastly, the C weights matrix employs a globally standardized method, assigning equal weights across the entire study area, thus treating all neighbours uniformly regardless of their proximity.

To create a weights matrix in R, the nb2listw function from the spdep package is utilized. This function

converts a neighbourhood object (such as Income.nb or French.nb) into a listw object, which is suitable for further spatial analysis. The function's parameters include zero.policy = TRUE, which ensures that polygons with zero neighbour links are still processed, assigning weights vectors of zero length to these regions. This is crucial for maintaining the integrity of the analysis when some areas lack neighbours.

The code provided below demonstrates the creation of weights matrices for both income and French-speaking datasets:

For this tutorial, both weights matrices will be of type W, which allows for row-standardized weights. This enables more meaningful comparisons in subsequent spatial analyses, such as autocorrelation tests like Moran's I. To explore the distribution of weights across observations and their neighbours, the print.listw function can be employed. For more details on different styles of spatial weighting, you can refer to the documentation here

```
#Create Income weights matrix
Income.lw <- nb2listw(Income.nb, zero.policy = TRUE, style = "W")

#Create French weights matrix
French.lw <- nb2listw(French.nb, zero.policy = TRUE, style = "W")

head(Income.lw[["weights"]])[c(1:3)]
```

```
## [[1]]
## [1] 0.3333333 0.3333333 0.3333333
##
## [[2]]
## [1] 0.5 0.5
##
## [[3]]
##  [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

Global Moran's I

Now that we have determined how to choose and weight our neighbours, we can calculate the Global Moran's I statistic. This method of testing for spatial autocorrelation looks across the entire study area for every location simultaneously [14]. The equation for this statistic is

$$I = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j}(x_i - \bar{x})(x_j - \bar{x})}{(\sum_{i=1}^{n}\sum_{j=1}^{n} W_{i,j})\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Here, if x is the variable being assessed, x_i is the variable value at a point of interest (i) and x_j represents a neighbour to x_i (here determined by the queen weighting scheme). The spatial weighting applied to the weighting matrix $W_{i,j}$ is multiplied by both the differences of x_i and the mean value of variable x, and x_j and the mean value of variable x.

The denominator in this case is used to standardize our values, and therefore relatively high values of I correspond with positive spatial autocorrelation, and relatively low values of I correspond with negative spatial autocorrelation. Remember that the global Moran's I statistic provides an indication of how spatially autocorrelated our data is over the entire dataset, thus representing a spatial pattern at the global scale [15].

```
#Calculate Global Moran's I for Income

miIncome <- moran.test(Income_noNA$`Median total income`, Income.lw, zero.policy = TRUE)
```

```r
#Extract Global Moran's I results for Income
mIIncome <- miIncome$estimate[[1]]
eIIncome <- miIncome$estimate[[2]]
varIncome <- miIncome$estimate[[3]]

#Calculate Global Moran's I for French
miFrench <- moran.test(French_noNA$PercFrench, French.lw, zero.policy = TRUE)

#Extract Global Moran's I results for French
mIFrench <- miFrench$estimate[[1]]
eIFrench <- miFrench$estimate[[2]]
varFrench <- miFrench$estimate[[3]]
```

The results of the code above provide the Global Moran's I statistics for the median income and French knowledge datasets, with key values extracted into three components: mi, ei, and var.

mi: This represents the Global Moran's I statistic itself. A higher value indicates a stronger tendency for the data to cluster spatially. In this case, a Moran's I of approximately 0.25 suggests a moderate level of clustering for median income, implying that areas with similar income levels are likely to be grouped together.

ei: This denotes the expected Moran's I under the null hypothesis of no spatial autocorrelation, which is typically set at 0. This value serves as a baseline for comparison with the observed mi value.

var: This component describes the variance of the Moran's I statistic, providing insight into the variability of spatial relationships within the dataset.

```r
#Function to calculate the range of global Moran's I
moran.range <- function(lw) {
  wmat <- listw2mat(lw)
  return(range(eigen((wmat + t(wmat))/2)$values))
}

#Calculate the range for the Income variable
range <- moran.range(Income.lw)
minRange <- range[1]
maxRange <- range[2]
```

The outputs reveal a Moran's I of about 0.25 for median income, indicating positive spatial autocorrelation, while the Moran's I for French knowledge is approximately 0.0086, close to zero. This suggests that there is minimal spatial autocorrelation for French speakers.

In summary, these results imply that individuals with similar incomes tend to reside in close proximity to one another, demonstrating a tendency toward clustering. Conversely, the distribution of French speakers appears to be more uniform across the area, indicating that language knowledge does not follow the same spatial patterns as income.

However, we can still go a step further and figure out whether these patterns are statistically significant. To do so, we can use a Z-test. Here our null hypothesis is that we have no spatial autocorrelation, and the alternate hypothesis is that there is signifigant spatial autocorrelation. Using an α value of 0.05, if our Z-score falls above or below 1.96, we can say our results are significantly different from the null hypothesis. A value greater than +1.96 would imply positive spatial autocorrelation, and a value less than -1.96 would imply signifigant negative autocorrelation.

We can calculate a Z-test using the following code:

```r
#Calculate z-test for Income
zIncome <- (mIIncome - eIIncome) / (sqrt(varIncome))

#Calculate z-test for French
zFrench <- (mIFrench - eIFrench) / (sqrt(varFrench))
```

The zscore(.1571) confirms that french knowlege lacks significant spatial clustering or dispertion. The French pattern is fairly random.

The zscore(3.1399) confirms that median income is significantly clustered.

Local spatial autocorrelation

Explain local spatial autocorrelation

Local spatial autocorrelation operates on principles similar to those of global spatial autocorrelation, but with a crucial distinction: it focuses on a bounded area defined by distance. While global spatial autocorrelation assesses the relationships between all objects across the entire study area, local spatial autocorrelation narrows the scope to specific locations and their immediate neighbours.

This localized approach enables the identification of significant clusters or dispersions that might not be evident when analyzing the dataset as a whole. By examining spatial relationships within defined neighbourhoods, local spatial autocorrelation reveals patterns of spatial dependence that provide valuable insights into localized phenomena, allowing for a more nuanced understanding of how characteristics are distributed across space.

The calculation for Local Moran's I has many of the same features as our global calculation, although arranged in a different way.

$$I_i = \frac{x_i - \bar{x}}{S_i^2} \sum_{j=1}^{n} W_{i,j}(x_j - \bar{x}) where S_i^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

Instead of manually performing all the calculations, we can leverage the localmoran() function to handle the complex computations for us, provided we input our variable and weighting scheme.

The code below calculates Local Indicators of Spatial Association (LISA) for both the median income and French knowledge datasets. This analysis enables us to understand how each location relates spatially to its neighbours.

As with the previous global tests, we will extract key values from our outputs. For both datasets, we retrieve the following components:

Li: The Local Moran's I value, indicating the degree of spatial autocorrelation for each location.

E: The expected Local Moran's I value under the null hypothesis of no spatial autocorrelation.

Var: The variance of the Local Moran's I, reflecting the variability of the estimates.

Z: The Z-score of the Local Moran's I, which standardizes the Local Moran's I value for statistical significance testing.

P: The p-value, indicating the statistical significance of the Local Moran's I results.

```r
#Calculate LISA test for Income
lisa.testIncome <- localmoran(Income_noNA$`Median total income`, Income.lw)

#Extract LISA test results for Income
```

```
Income_noNA$Ii <- lisa.testIncome[,1]
Income_noNA$E.Ii<- lisa.testIncome[,2]
Income_noNA$Var.Ii<- lisa.testIncome[,3]
Income_noNA$Z.Ii<- lisa.testIncome[,4]
Income_noNA$P<- lisa.testIncome[,5]

#Calculate LISA test for Income
lisa.testFrench <- localmoran(French_noNA$PercFrench, French.lw)

#Extract LISA test results for Income
French_noNA$Ii <- lisa.testFrench [,1]
French_noNA$E.Ii<- lisa.testFrench [,2]
French_noNA$Var.Ii<- lisa.testFrench [,3]
French_noNA$Z.Ii<- lisa.testFrench [,4]
French_noNA$P<- lisa.testFrench [,5]
```

Now going back to our basic mapping template we can visualize some of these results to understand what this test is doing.

```
#Map LISA z-scores for Income
map_LISA_Income <- tm_shape(Income_noNA) +
  tm_polygons(col = "Z.Ii",
              title = "Income Local Moran's I Z-Scores",
              style = "fixed",
              border.alpha = 0.1,
              midpoint = NA,
              colorNA = NULL,
              breaks = c(min(Income_noNA$Z.Ii),-1.96,1.96,max(Income_noNA$Z.Ii)),
              palette = "-RdBu", n = 3)+
  tm_compass(position=c("left", "top"))+
  tm_scale_bar(position=c("left", "bottom"))+
  tm_legend(position = c("right", "top"))

#Map LISA z-scores for French
map_LISA_French <- tm_shape(French_noNA) +
  tm_polygons(col = "Z.Ii",
              title = "French Local Moran's I Z-Scores",
              style = "fixed",
              border.alpha = 0.1,
              midpoint = NA,
              colorNA = NULL,
              breaks = c(min(French_noNA$Z.Ii),-1.96,1.96,max(French_noNA$Z.Ii)),
              palette = "-RdBu", n = 3)+
  tm_compass(position=c("left", "top"))+
  tm_scale_bar(position=c("left", "bottom"))+
  tm_legend(position = c("right", "top"))

#Plot maps in a 2 pane figure
tmap_arrange(map_LISA_Income, map_LISA_French, ncol = 2, nrow = 1)
```

Remember, Z-scores above 1.96 indicate positive spatial autocorrelation, scores of 0 suggest no autocorrelation, and scores below -1.96 indicate negative spatial autocorrelation.

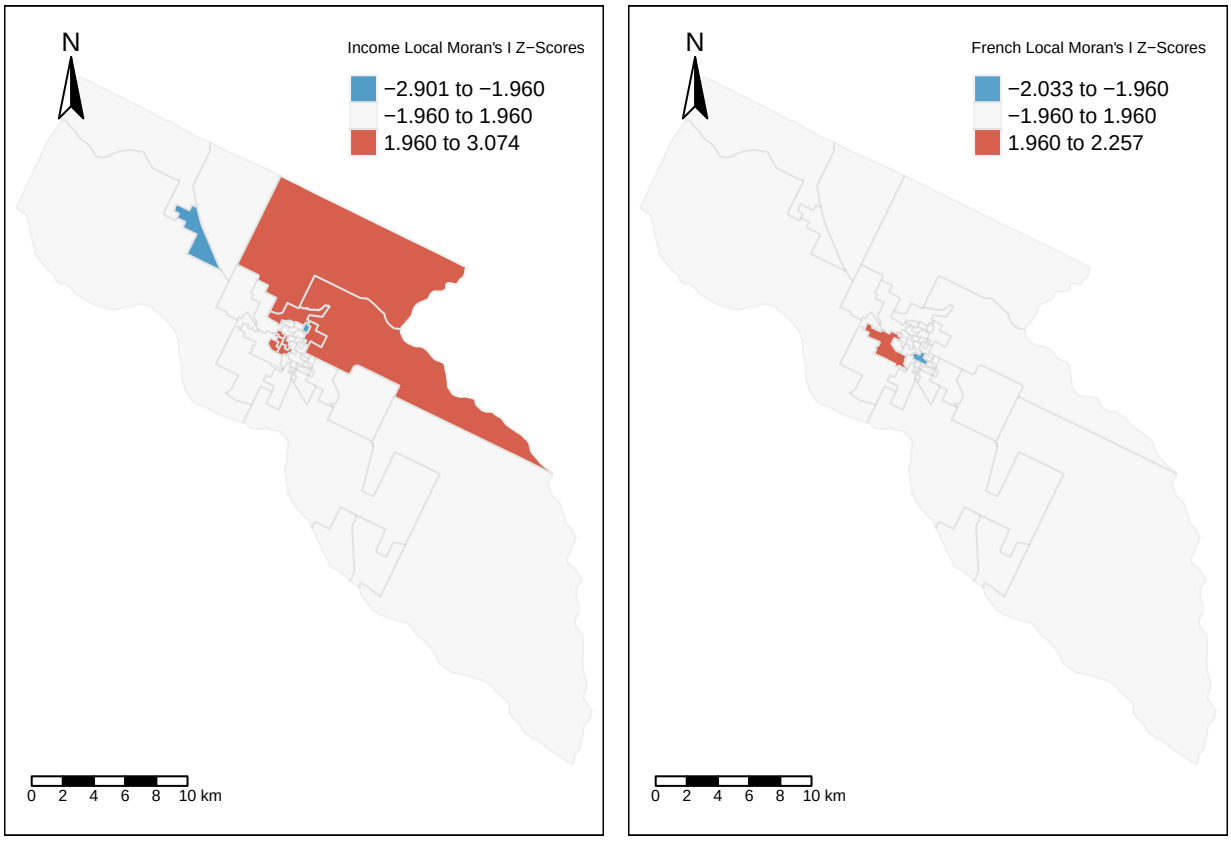

Figure 3: Kamloops census dissemination areas showing LISA z-scores for median total income (left) and percentage of respondants with knowledge of French (right).

The maps above display the Z-scores for various neighbourhoods around Fort Saint John, highlighting areas of significant autocorrelation.

Starting with the median income map, we observe distinct areas north of the city that exhibit significant differences from their neighbouring regions, marked in red. Conversely, areas to the west of the town show a strong similarity to their neighbours, represented in blue.

Turning to the French knowledge map, we find that there are few neighbourhoods that significantly differ from or resemble their neighbours; these instances are primarily located within the city itself. This reinforces our earlier analysis, indicating that French speakers are more uniformly distributed across the area.

While these maps are great for visualizing where the data is and getting a rough idea of how many polygons are significantly positively or negatively spatially autocorrelated, it can be even more informative to graph these trends.

```
#Create Moran's I scatter plot for Income
moran.plot(Income_noNA$`Median total income`, Income.lw, zero.policy=TRUE, spChk=NULL, labels=NULL, xlab
           ylab="Spatially Lagged Median Total Income ($)", quiet=NULL)
```

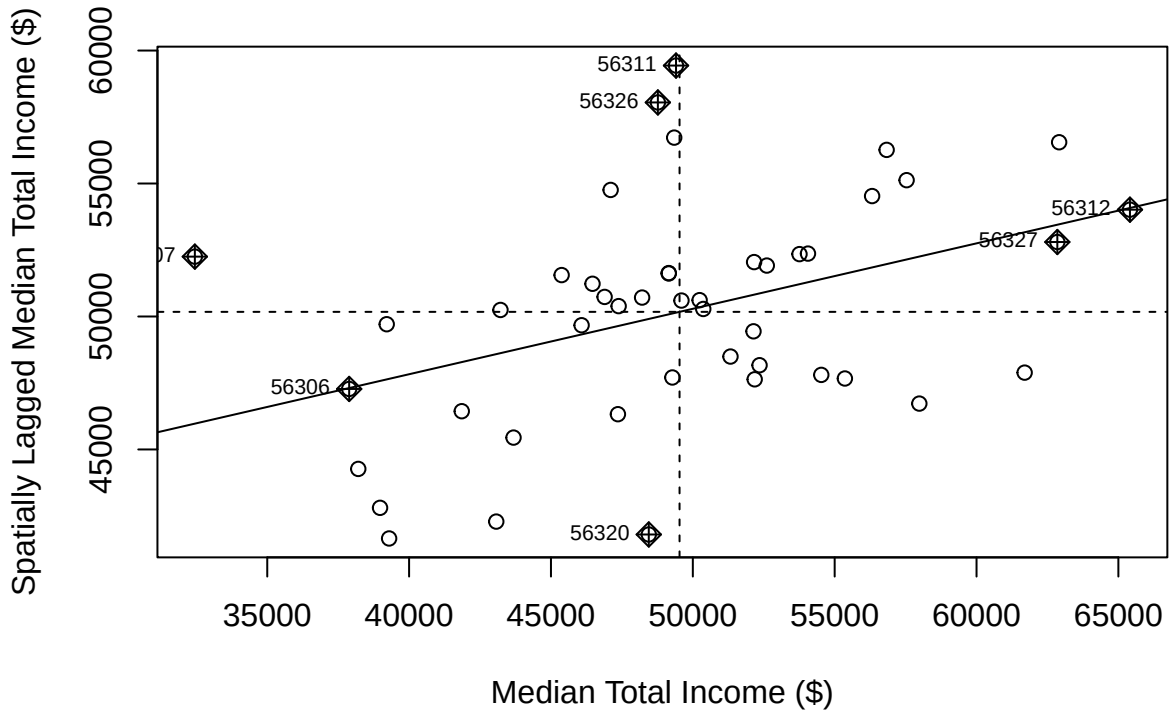

Figure 4: Moran's I scatter plot for median total income.

```
#Create Moran's I scatter plot for French
moran.plot(French_noNA$PercFrench, French.lw, zero.policy=TRUE, spChk=NULL, labels=NULL, xlab="Responda
           ylab="Spatially Lagged knowledge of French (%)", quiet=NULL)
```

In these plots, the points marked with diamonds are deemed statistically significant, while the regression line illustrates the overall trend. For the median income plot, we observe a positive trend, indicating a strong

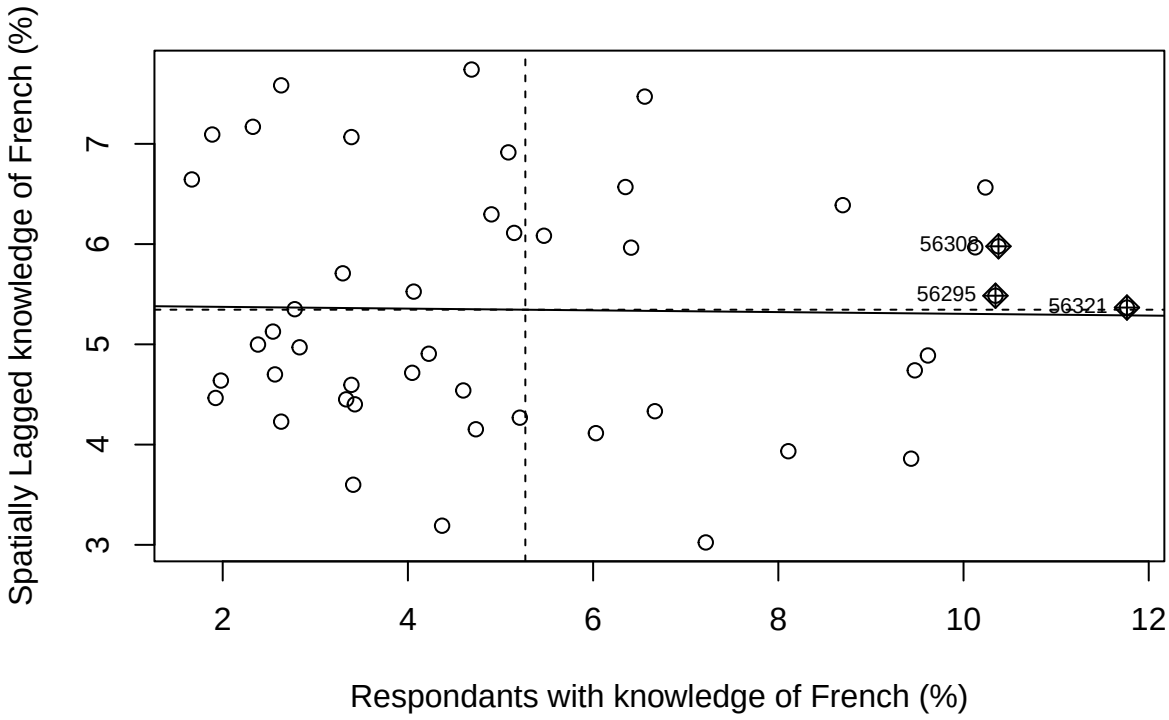

Figure 5: Moran's I scatter plot for percentage of respondants with knowledge of French.

spatial autocorrelation among income levels. This finding is consistent with our earlier analysis, reinforcing the notion that individuals with similar incomes tend to cluster together geographically.

In contrast, the French language plot reveals a weak trend, suggesting minimal spatial autocorrelation among French speakers. When we compare this to the median income plot, it's clear that the two datasets exhibit distinct patterns. This divergence further supports our earlier conclusions that the distribution of French speakers is more uniform across the area, unlike the clustering observed in income levels.

Summary

This tutorial delved into the spatial relationships of median income and French speakers in the vicinity of Fort Saint John, BC, using R. We began by importing our data and segmenting it into two distinct datasets: one for income and another for French language speakers. We introduced the concept of spatial autocorrelation, subsequently calculating the Global Moran's I for both datasets. This analysis revealed a positive spatial autocorrelation for median income while indicating no significant autocorrelation for French speakers.

We then explored local patterns through the LISA test, extracting key metrics to visualize areas of significant clustering and dispersion. The results demonstrated that similar median incomes tended to cluster together, whereas French speakers were distributed more uniformly across the region, consistent with our earlier findings.

Finally, we generated Moran's I scatter plots to offer a visual representation of these spatial relationships. The plots reaffirmed our previous conclusions, highlighting a positive autocorrelation for income and a weak autocorrelation for French language knowledge.

I hope this tutorial has provided valuable insights into the use of spatial data in R and enhanced your understanding of the patterns associated with different socioeconomic variables.

References

Goodchild, M. F. (2004). Geographic information science. Progress in Human Geography, 28(3), 334–350. https://doi.org/10.1191/0309132504ph498pr