Name: Jing Wu
Student ID:29696576

GISC404 GIS software comparison

1. Introduction

Spatial analysis software is software written for the purpose of spatial analysis. Currently, there are a large number of spatial analysis packages, including free software and charging software, which cover most of the spatial data analysis functions.

Each spatial analysis software has its own features, strengths and flaws. This report compared two kinds of spatial analysis software, R and PySAL.

2. An overview of software

2.1 Overview of R

R is a multi-platform open source language and environment for statistical computing and visualization. There are various software packages in R that enable modeling and visualization of geospatial statistics. Rstudio, which is the new integrated development environment (IDE), makes it easier for users to use R. Rstudio simplifies the analysis of spatial data through interactive visual panels (Grosjean, 2018).

When it comes to interacting with R, you can execute commands interactively by typing commands and running (press Enter or Ctrl + Enter), which is called the command line interface (CLI). This type of interaction is very beneficial because it immediately executes the code entered in the console and prints out the results, improving efficiency to a high degree (Grosjean, 2018).

However, in the GIS software packages, interactions are often performed in a graphical user interface (GUI). Users can interact from the system terminal. Some GIS packages have an embedded CLI, but the way they are often used is to click on the control panel to interact. R can also be integrated into GIS software. In spatial analysis, the CLI and GUI options are user-selectable, depending on the user's skills and software availability (Grosjean, 2018).

Rstudio has many advantages in spatial data analysis with the help of the R language:
• Can perform automated tasks (loops)
• User-friendly and efficient, with significant results in data analysis.
• There are many software packages that can be used directly, very convenient
• The software package covers almost every field, and the spatial data analysis model is very comprehensive. Different packages can be used for different types of spatial data analysis.

There are dozens of packages in R that can process spatial data. R analysis of spatial data package mainly includes two parts: reading and writing spatial data, analyzing spatial data.

2.1.1 Reading spatial data

There are three main types of spatial data: vector data, raster data, and geodatabase (gdb file).

a. Vector data
Vector data mainly includes three categories: points, lines and polygons. There are many ways to read vector data in R.

From the point of reading point, line, and polygon, maptools package mainly includes readShapePoints (read points), readShapeLines (read line elements), and readShapePoly (read surface elements). Maptools package also provides another function readShapeSpatial, which can read the above three types of elements.

shp file could be read directly by using the readOGR function in the rgdal package. The incoming parameter is the file name.

The shapefiles package can also read shapefiles. The file name passed in the read.shp function as a parameter can be easily read (DEMPSEY, 2017).

b. Raster data

For raster data, the format is still varied. The raster data reading is mainly based on the rgdal package, and the img file and the tif file can be read directly by readGDAL. Of course, raster data is also more commonly stored in ASCII files. The method of reading the ASCII code file is based on the sp package, and the read.asciigrid function in the sp package can directly read the ASCII code file. This package is the basic package of R language spatial data, specifying the methods and objects of the spatial database (DEMPSEY, 2017).

c. geodatabase

That is to say, Geodatabase is a database proposed by Esri officially. It is impossible to create a Geodatabase without ArcGIS. At present, only the R package given by Esri officially can read the Geodatabase data (ESRI, 2003).

2.1.2 Analyzing spatial data

R can analyze spatial data in many ways. There are many packages in R that can be used for statistical analysis of spatial data. We can divide it into four aspects: visualization, spatial regression, point pattern analysis and geostatistics, and correspondingly list several packages that can be used in R.

The first is visualization. R has powerful statistical calculations and a convenient data visualization system. There are a number of visual methods in the external extensions, such as ggplot2, sp, sf, raster and rasterVis. Visualization can be a great help in ESDA.

The next is spatial autocorrelation and spatial regression. The spdep package provides a comprehensive approach to spatial autocorrelation, such as local Moran's I. This package also provides a diagnostic tool for linear models to determine if there is spatial dependence on the linear regression model. The spatial regression models provided in this package include spatial lag models and spatial error models. Functions for computing geographically weighted regressions are provided in spgwr package.

Point pattern analysis can also be performed very well in R. Spatstat allows free definition of regions of interest and extends marked process and spatial covariates. Its strength lies in fitting models and simulation. It is the only software package that allows users to adapt to non-uniform point process models using point-to-point interaction. The spatgraphs package provides a graphical visualization of spatial point pattern analysis.

Geostatistics also has a lot of package support in R. Geostatistics is very similar to statistics, and the content between spatial statistical analysis and spatial data analysis in geostatistics is often cross-correlated. The gstat package is used for univariate and multivariate geostatistics and is suitable for large data sets. The geoR package can be used to build a Bayesian model, while the geoRglm package uses and builds a linear model. (Bivand, 2019)

2.2 Overview of PySAL

PySAL is an open source cross-platform library for geospatial data science with an emphasis on geospatial vector data written in Python.

PySAL is not a geographic information system. Instead, it is a library in Python that can perform complex spatial analysis in a variety of ways. It ranges from a simple command line interface (CLI) to a program with a graphical user interface (GUI). PySAL can be flexibly integrated with other GUIs to become an external library of GUIs. Since it is a library, PySAL can also be used with the functionality of different GIS or analysis packages. In addition, PySAL can also be used directly in the Python IDE in the way of CLI.

Compared to R, PySAL is more focused on the part of spatial analysis. The main applications of PySAL in spatial analysis are:
• Exploratory spatial data analysis
• Regression and statistical modeling of spatial data
• Visualization of spatial data (detecting spatial clusters and outliers)
• Spatial econometrics

PySAL has four components, explore, viz, model, and lib.
• lib - This component is the most important part of PySAL, such as file reads, spatial weight, and Computational Geometry.

• explore - This component is for Exploratory Spatial Data Analysis (ESDA). The explore component includes Spatial Autocorrelation Analysis (Gamma Statistic, Geary Statistic, Getis-Ord Statistics, Join Count Statistics and Moran Statistics), Geospatial Distribution Dynamics (Markov Methods, Directional LISA, Economic Mobility Indices and Exchange Mobility Methods), and Spatial Inequality Analysis (Theil Inequality Measures and Gini Inequality Measures).

• viz - Visualize spatial data to assist with ESDA, making it easy to observe clusters and outliers, the core components are Choropleth map classification and Lightweight visualization interface.

• model - The spatial regression model in this component comes from the support of the spreg library. This component is mainly used for spatial modelling and is divided into two parts: Spatial Econometrics and Multiscale Geographically Weighted Regression. This component uses a variety of models to simulate spatial relationships in data for spatial data analysis. And

the module contains a series of spatial effects diagnostic methods, tests and estimation methods. (Rey & Anselin, 2010)

3. Perform spatial analysis (Regression) in R and PySAL

When constructing a regression model, we can suspect the inference of the regression model. In essence, it may not be fair to treat individual cases as independent events because adjacent areas are always similar. Therefore, these models need to be diagnosed before the conclusion is reached. The diagnosis is mainly to evaluate the spatial autocorrelation in the residual. If the residual is spatially autocorrelated, then the model is not specified. In this case, you should try to improve the model by adding or removing important variables. If this is not possible, such as no data available or if you don't know what variable to look for, try creating a regression model that controls spatial autocorrelation. (Hijmans & Ghosh, 2019)

I will perform spatial data regression analysis on the map of Manhattan in New York City with Census 2000 data in R and PySAL to compare this two software.

3.1 Regression in R

3.1.1 Reading the spatial data

Use the shapefile function in the raster package to read the required shp file into R.

```
filename <- "/Users/wujing/Desktop/GISC404_RegressionLab_NewYork/NewYork.shp"

p <- shapefile(filename)
```

3.1.2 Spatial data analysis

3.1.2.1 OLS Model
Before building the spatial model, I first construct a traditional linear model to obtain a reference standard for the spatial model. This standard model can be used to determine whether it is necessary to construct a spatial model.

R uses the lm() function to create a regression model, so we could use lm() to build a OLS Model.

Firstly, define the formula

```
f1 <- T0P_POOR~PCTNHW+PCTNHB+T0P_COLL+T0P_UEMP+T0P_FOR+T0_MINC
```

Secondly, build the model with the defined formula

```
OLSmodel <- lm(f1, data = p)
```

Print out the summary of the OLS model

```
summary(OLSmodel)
```

The output of the OLS model summary is provided below:

Name: Jing Wu
Student ID:29696576

```
> summary(OLSmodel)

Call:
lm(formula = f1, data = p)

Residuals:
    Min      1Q  Median      3Q     Max
-32.719  -2.456   0.033   2.724  22.549

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.985e+01  2.188e+00   9.073  < 2e-16 ***
PCTNHW      -6.118e-02  4.511e-02  -1.356 0.176137
PCTNHB       9.829e-02  2.663e-02   3.691 0.000267 ***
T0P_COLL    -1.259e-01  4.429e-02  -2.843 0.004792 **
T0P_UEMP     5.755e-01  5.063e-02  11.366  < 2e-16 ***
T0P_FOR      1.206e-01  3.476e-02   3.470 0.000599 ***
T0_MINC     -4.498e-05  2.460e-05  -1.828 0.068514 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.454 on 290 degrees of freedom
Multiple R-squared:  0.7609,    Adjusted R-squared:  0.756
F-statistic: 153.9 on 6 and 290 DF,  p-value: < 2.2e-16
```

In the OLS model summary, the R square of 0.76 shows that the model fits well. The p-value of the intercepts of PCTNHB, TOP_COLL, TOP_UEMP, TOP_FOR are extremely low, indicate that all predictors are significant except PCTNHW and TO_MINC.

Next, I checked if the errors appear to be randomly distributed in space.

Create a column of OLS model residuals

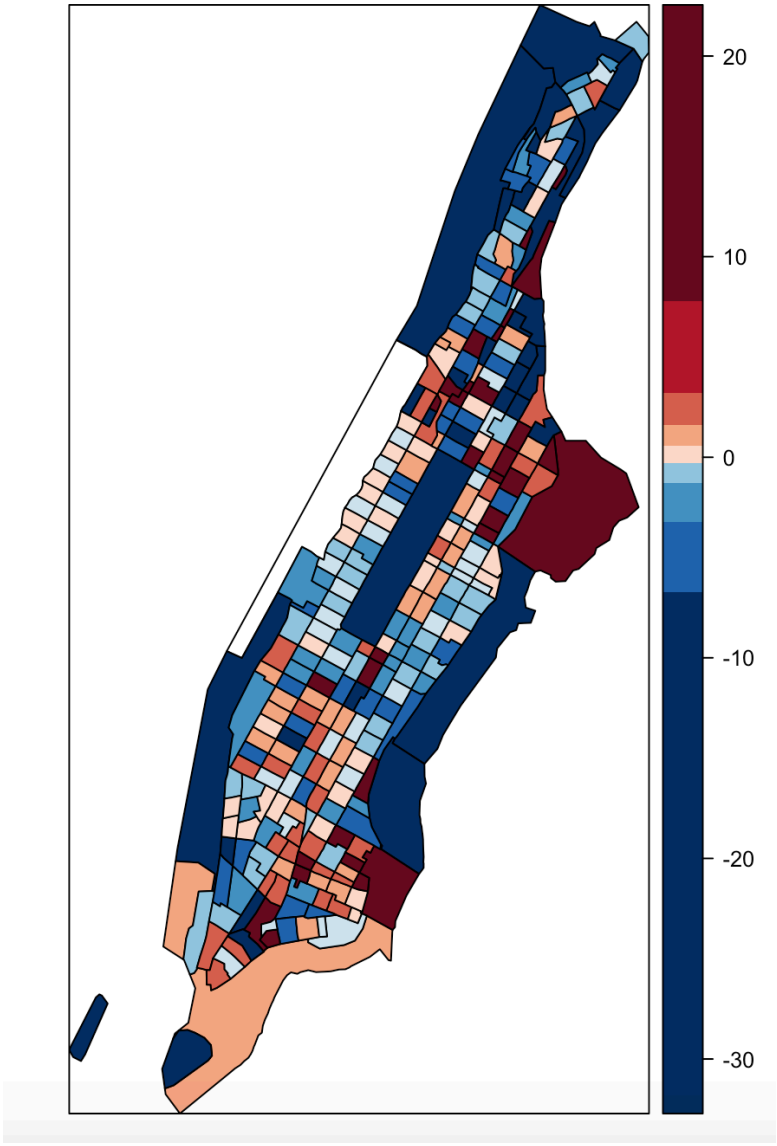
```
p$OLSresiduals <- residuals(OLSmodel)
```

As for the residuals, using a legend with 10 intervals, and then plotting the residual map.

```
grps <- 10

brks <- quantile(p$OLSresiduals, 0:(grps-1)/(grps-1), na.rm=TRUE)

spplot(p, "OLSresiduals", at=brks, col.regions=rev(brewer.pal(grps, "RdBu")), col="black")
```

The residual map of the OLS model is provided below:

Name: Jing Wu
Student ID:29696576



On the map, the random distribution of residuals is not obvious, so we can perform Moran's I test.

```
# Construct neighbours list from polygon list
w <- poly2nb(p, row.names=p$Id)

# Spatial weights for neighbours lists
ww <- nb2listw(w, style='B',zero.policy=TRUE)

# Moran's I test
moran.test(p$OLSresiduals, ww, 999,zero.policy=T)
```

The Moran's I test result of OLS model residuals is provided below:

```
> moran.test(p$OLSresiduals, ww, 999,zero.policy=T)

        Moran I test under randomisation

data:  p$OLSresiduals
weights: ww  n reduced by no-neighbour observations


Moran I statistic standard deviate = 2.1947, p-value = 0.01409
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
     0.0656252025     -0.0033898305      0.0009888783
```

The p-value is 0.0149, which is less than 0.05, indicates that there is spatial autocorrelation. So, we need to perform spatial regression model.

3.1.2.2 Spatial lag model

Use lagsarlm() of spdep package to build a Spatial lag model. The lagsarlm function provides Maximum likelihood estimation of spatial simultaneous autoregressive lag.

```
Lagmodel <- lagsarlm(f1, data=p, ww, tol.solve=1.0e-30,zero.policy=T)
```

Print out the summary of the Spatial lag model

```
summary(Lagmodel)
```

The output of the spatial lag model summary is provided below:

```
> summary(Lagmodel)

Call:lagsarlm(formula = f1, data = p, listw = ww, zero.policy = T,     tol.solve = 1e-30)

Residuals:
       Min         1Q     Median         3Q        Max
-31.925309  -2.496026  -0.001189   2.696670  22.892520

Type: lag
Regions with no neighbours included:
 295
Coefficients: (asymptotic standard errors)
              Estimate  Std. Error z value  Pr(>|z|)
(Intercept)  2.0444e+01  2.4254e+00  8.4291 < 2.2e-16
PCTNHW      -6.4117e-02  4.5103e-02 -1.4216 0.1551495
PCTNHB       1.0213e-01  2.7545e-02  3.7078 0.0002091
T0P_COLL    -1.2631e-01  4.3742e-02 -2.8876 0.0038821
T0P_UEMP     5.7687e-01  5.0015e-02 11.5340 < 2.2e-16
T0P_FOR      1.2196e-01  3.4556e-02  3.5292 0.0004167
T0_MINC     -4.6459e-05  2.4316e-05 -1.9106 0.0560562

Rho: -0.0037737, LR test value: 0.30442, p-value: 0.58112
Asymptotic standard error: 0.0073942
    z-value: -0.51035, p-value: 0.6098
Wald statistic: 0.26046, p-value: 0.6098

Log likelihood: -1014.314 for lag model
ML residual variance (sigma squared): 54.186, (sigma: 7.3611)
Number of observations: 297
Number of parameters estimated: 9
AIC: 2046.6, (AIC for lm: 2044.9)
LM test for residual autocorrelation
test value: 8.1298, p-value: 0.0043543
```

Name: Jing Wu
Student ID:29696576

We use the same method to detect whether the residuals of spatial lag model are randomly distributed. First draw the residual map and then perform Moran's I test.

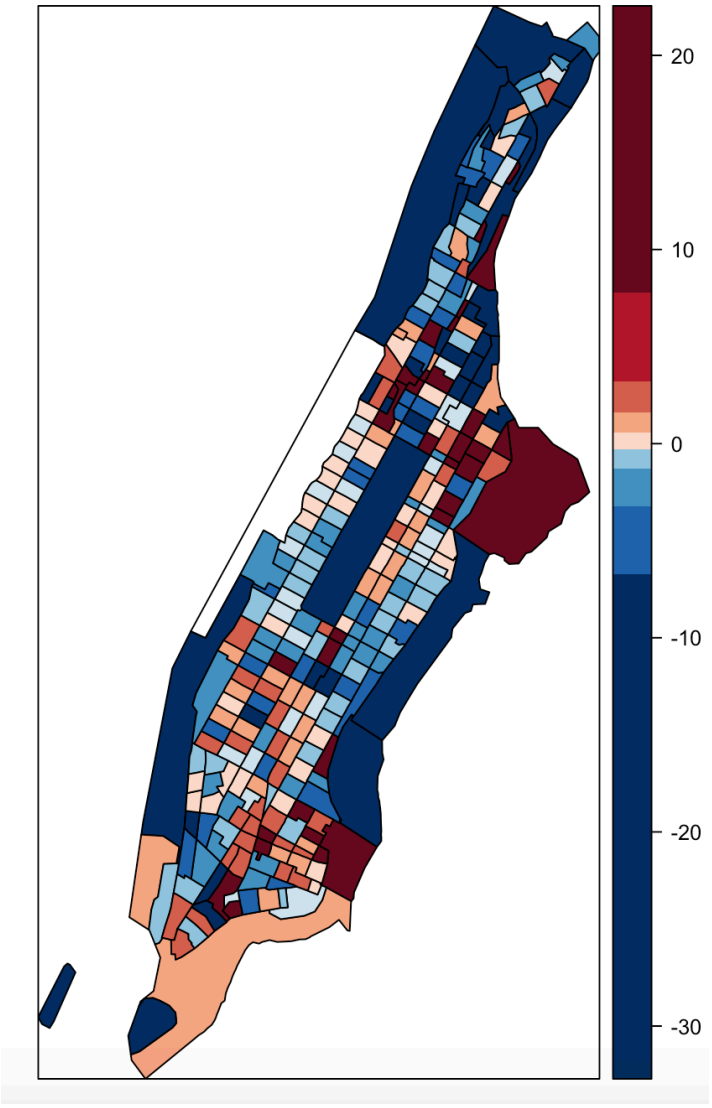
Create a column of spatial lag model residuals

```
p$Lagresiduals <- residuals(Lagmodel)
```

Plot the residual map

```
spplot(p, "Lagresiduals", at=brks, col.regions=rev(brewer.pal(grps, "RdBu")), col="black")
```

The residual map of the spatial lag model is provided below:



Moran's I test for spatial autocorrelation of residuals

```
moran.test(p$Lagresiduals, ww, 999,zero.policy=T)
```

The Moran's I test result of spatial lag model residuals is provided below:

```
> moran.test(p$Lagresiduals, ww, 999,zero.policy=T)

        Moran I test under randomisation

data:  p$Lagresiduals
weights: ww  n reduced by no-neighbour observations


Moran I statistic standard deviate = 2.6836, p-value = 0.003641
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.081024495      -0.003389831       0.000989429
```

The p-value is 0.003641, which is less than 0.05, indicates that there is spatial autocorrelation.

3.1.2.3 Spatial error model

Use errorsarlm() of spdep package to build a Spatial error model. The errorsarlm function provides Maximum likelihood estimation of spatial simultaneous autoregressive error.

```
Errormodel <- errorsarlm(f1, data=p, ww, tol.solve=1.0e-30,zero.policy=T)
```

Print out the summary of the Spatial error model

```
summary(Errormodel)
```

The output of the spatial error model summary is provided below:

```
> summary(Errormodel)

Call:errorsarlm(formula = f1, data = p, listw = ww, zero.policy = T,     tol.solve = 1e-30)

Residuals:
       Min         1Q     Median         3Q        Max
-33.677334  -2.784204  -0.044178   2.604654  24.150498

Type: error
Regions with no neighbours included:
 295
Coefficients: (asymptotic standard errors)
             Estimate  Std. Error z value  Pr(>|z|)
(Intercept)  1.9426e+01  2.2423e+00  8.6636 < 2.2e-16
PCTNHW      -6.9717e-02  4.4742e-02 -1.5582 0.1191887
PCTNHB       1.0437e-01  2.9037e-02  3.5945 0.0003250
T0P_COLL    -1.2661e-01  4.5277e-02 -2.7964 0.0051682
T0P_UEMP     5.6238e-01  5.0001e-02 11.2472 < 2.2e-16
T0P_FOR      1.3204e-01  3.4834e-02  3.7906 0.0001503
T0_MINC     -2.8317e-05  2.5009e-05 -1.1323 0.2575212

Lambda: 0.030569, LR test value: 3.9871, p-value: 0.045849
Asymptotic standard error: 0.014526
    z-value: 2.1044, p-value: 0.035342
Wald statistic: 4.4286, p-value: 0.035342

Log likelihood: -1012.473 for error model
ML residual variance (sigma squared): 53.177, (sigma: 7.2922)
Number of observations: 297
Number of parameters estimated: 9
AIC: 2042.9, (AIC for lm: 2044.9)
```

Name: Jing Wu
Student ID:29696576

Check if the errors appear to be randomly distributed in space

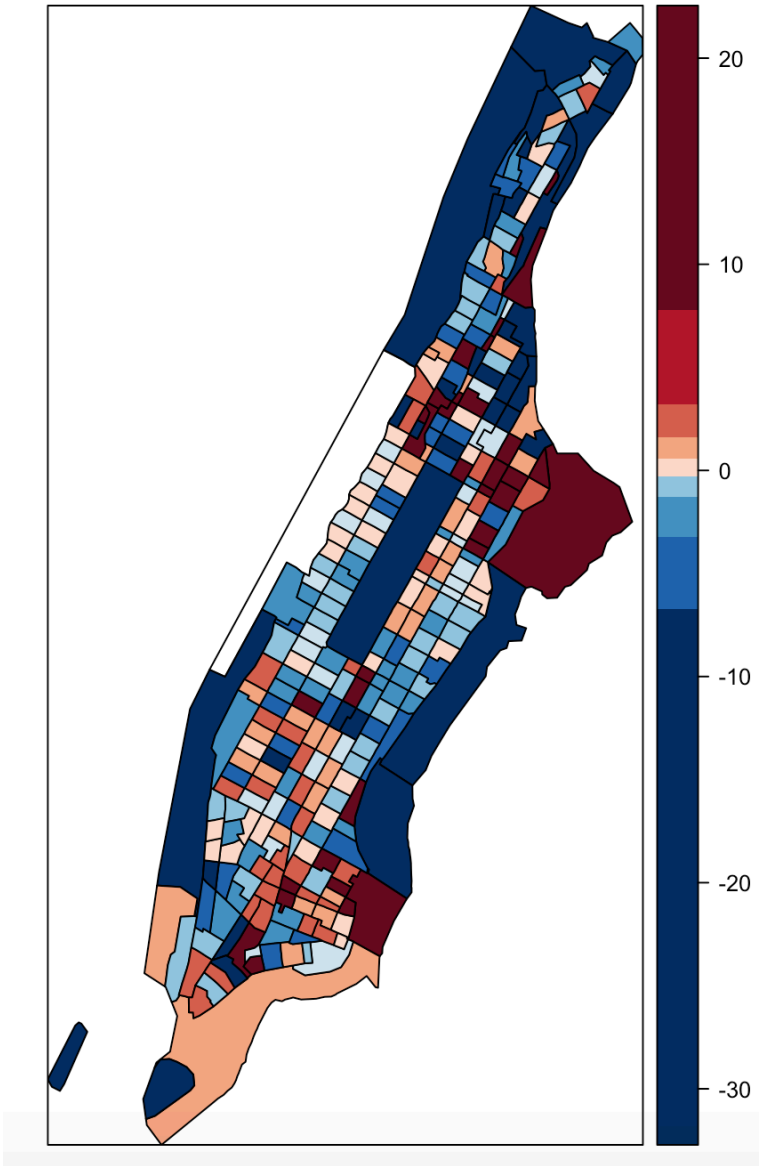
Create a column of spatial error model residuals

```
p$Errorresiduals <- residuals(Errormodel)
```

Plot the residual map

```
spplot(p, "Lagresiduals", at=brks, col.regions=rev(brewer.pal(grps, "RdBu")), col="black")
```

The residual map of the spatial error model is provided below:



Moran's I test for spatial autocorrelation of residuals

```
moran.test(p$Errorresiduals, ww, 999,zero.policy=T)
```

The Moran's I test result of spatial error model residuals is provided below:

```
> moran.test(p$Errorresiduals, ww, 999,zero.policy=T)

        Moran I test under randomisation

data:  p$Errorresiduals
weights: ww  n reduced by no-neighbour observations


Moran I statistic standard deviate = -0.47356, p-value = 0.6821
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
    -0.0182753806     -0.0033898305      0.0009880618
```

The p-value is 0.6821, which is more than 0.05, indicates that there is no spatial autocorrelation.

3.1.2.4 GWR model

Use gwr.sel() function of spgwr package to get the optimal bandwidth.

```
bw <- gwr.sel(f1, data=p)
```

Use gwr() function of spgwr package to get the GWR parameters with the best bandwidth

```
gwr_result <- gwr(f1, data=p, bandwidth=bw, hatmatrix = TRUE)
```

Print the GWR parameters

```
gwr_result
```

The GWR parameters with the optimal bandwidth is provided below:

```
> gwr_result
Call:
gwr(formula = f1, data = p, bandwidth = bw, hatmatrix = TRUE)
Kernel function: gwr.Gauss
Fixed bandwidth: 1.322856
Summary of GWR coefficient estimates at data points:
                   Min.     1st Qu.      Median     3rd Qu.        Max.  Global
X.Intercept.  2.5937e-02  1.0066e+01  2.3624e+01  4.6240e+01  5.6770e+01 19.8513
PCTNHW       -3.5188e-01 -1.9770e-01 -6.9245e-02  1.5766e-01  7.7990e-01 -0.0612
PCTNHB       -2.3451e-01 -8.0887e-02  2.2029e-01  7.1473e-01  1.8887e+00  0.0983
T0P_COLL     -6.0896e-01 -3.0605e-01 -1.2394e-01 -6.9270e-02  2.0408e-01 -0.1259
T0P_UEMP     -6.0794e-02  2.3457e-01  3.4231e-01  4.8935e-01  6.8906e-01  0.5755
T0P_FOR      -3.2328e-01 -1.6963e-01  1.3171e-01  3.1824e-01  5.6126e-01  0.1206
T0_MINC      -8.9707e-04 -3.4159e-04 -8.5834e-05 -4.6668e-06  4.3184e-04  0.0000
Number of data points: 297
Effective number of parameters (residual: 2traceS - traceS'S): 59.76897
Effective degrees of freedom (residual: 2traceS - traceS'S): 237.231
Sigma (residual: 2traceS - traceS'S): 4.590141
Effective number of parameters (model: traceS): 46.62696
Effective degrees of freedom (model: traceS): 250.373
Sigma (model: traceS): 4.468049
Sigma (ML): 4.102358
AICc (GWR p. 61, eq 2.33; p. 96, eq. 4.21): 1795.22
AIC (GWR p. 96, eq. 4.22): 1727.944
Residual sum of squares: 4998.314
Quasi-global R2: 0.9258371
```

Name: Jing Wu
Student ID:29696576

The AIC of GWR model is less than both spatial lag model and spatial error model which means that the GWR model in this case perform the best.

We could alse use global R square to check the model performance

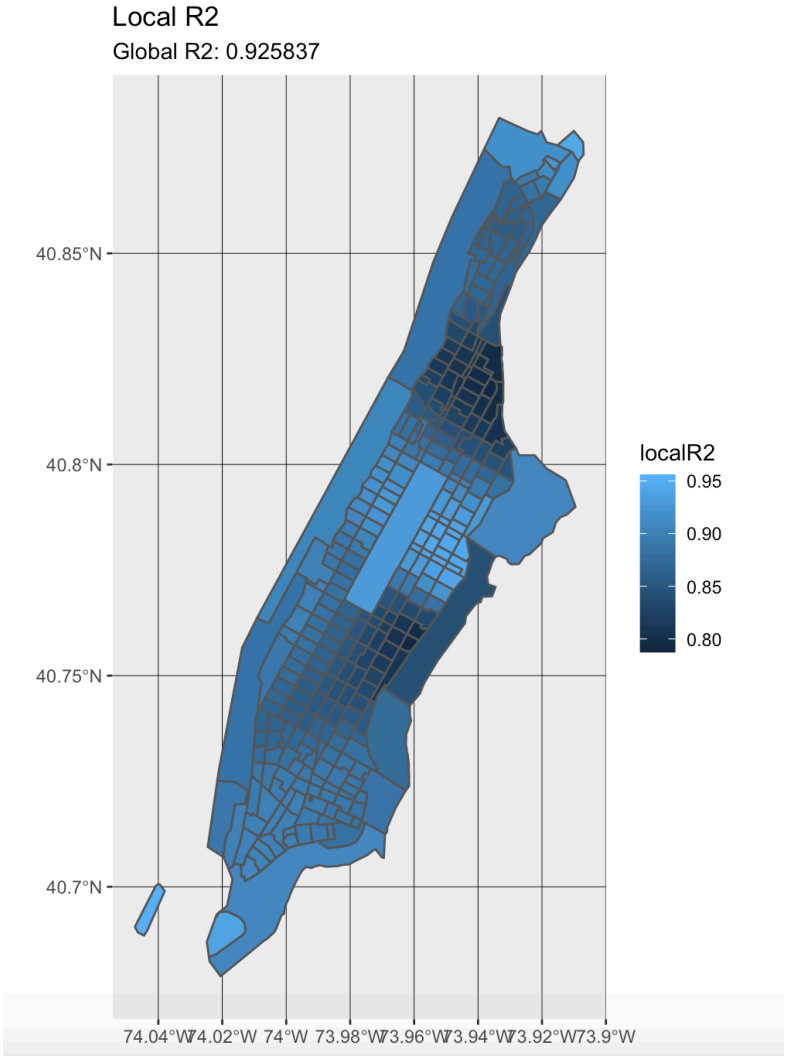
Get the global R square

```
globalR2 <- (1 - (gwr_result$results$rss/gwr_result$gTSS))

sp <- gwr_result$SDF

sf <- st_as_sf(sp)
```

Plot out the global R square map

```
ggplot() + geom_sf(data = sf, aes(fill=localR2)) +
  coord_sf() +
  theme(panel.grid.major = element_line(color = "black")) +
  ggtitle(paste("Local R2")) +
  labs(subtitle = paste("Global R2:", round(globalR2, 6)))
```

The global R square map of the GWR model is provided below:



Name: Jing Wu
Student ID:29696576

3.2 Regression in PySAL

3.2.1 Reading the spatial data

Use pysal.lib.io.open class in PySAL to load the shp file and dbf file. The shp file defines the shape and dbf file contains the data for analysis.

```
# Load shp file in PySAL
shp_path = "/Users/wujing/Desktop/GISC404_RegressionLab_NewYork/NewYork.shp"

f = pysal.lib.io.open(shp_path)
```

```
# Load dbf file for spatial data analysis
dbf_path = "/Users/wujing/Desktop/GISC404_RegressionLab_NewYork/NewYork.dbf"

dbf = pysal.lib.io.open(dbf_path)
```

3.2.2 spatial data analysis

3.2.2.1 OLS model

Extract the T0P_POOR column from the dbf file and make it the dependent variable for the regression.

```
y = np.array(dbf.by_col("T0P_POOR"))

y.shape = (len(dbf.by_col("T0P_POOR")),1)
```

Extract PCTNHW, PCTNHB, T0P_COLL, T0P_UEMP, T0P_FOR, T0_MINC columns from the dbf file to be used as independent variables in the regression.

```
X = []
cols = ["PCTNHW","PCTNHB","T0P_COLL","T0P_UEMP","T0P_FOR","T0_MINC"]
for item in cols:
    X.append(dbf.by_col(item))
X = np.array(X).T
```

Create the OLS model by using pysal.model.spreg.OLS class in PySAL. The minimum parameters needed to run an ordinary least squares regression are the two numpy arrays containing the independent variable and dependent variables respectively.

```
ols = pysal.model.spreg.OLS(y, X, name_y='POVERTY', name_ds='columbus', white_test=True)
```

We can easily obtain a full summary of all the results nicely formatted and ready to be printed

```
print(ols.summary)
```

Name: Jing Wu
Student ID:29696576

The output of the OLS model summary is provided below:

```
REGRESSION
----------
SUMMARY OF OUTPUT: ORDINARY LEAST SQUARES
-----------------------------------------
Data set            :    columbus
Weights matrix      :        None
Dependent Variable  :     POVERTY                Number of Observations:         297
Mean dependent var  :     20.3580                Number of Variables   :           7
S.D. dependent var  :     15.0894                Degrees of Freedom    :         290
R-squared           :      0.7609
Adjusted R-squared  :      0.7560
Sum squared residual:   16111.305                F-statistic           :    153.8535
Sigma-square        :      55.556                Prob(F-statistic)     :   4.752e-87
S.E. of regression  :       7.454                Log likelihood        :   -1014.466
Sigma-square ML     :      54.247                Akaike info criterion :    2042.932
S.E of regression ML:      7.3652                Schwarz criterion     :    2068.788

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     t-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT      19.8513017       2.1878452       9.0734488       0.0000000
               var_1      -0.0611754       0.0451128      -1.3560524       0.1761374
               var_2       0.0982933       0.0266341       3.6905062       0.0002673
               var_3      -0.1258921       0.0442870      -2.8426451       0.0047918
               var_4       0.5755096       0.0506348      11.3658991       0.0000000
               var_5       0.1206185       0.0347573       3.4703067       0.0005989
               var_6      -0.0000450       0.0000246      -1.8284148       0.0685144
------------------------------------------------------------------------------------

REGRESSION DIAGNOSTICS
MULTICOLLINEARITY CONDITION NUMBER           17.140

TEST ON NORMALITY OF ERRORS
TEST                             DF        VALUE           PROB
Jarque-Bera                       2         139.635          0.0000

DIAGNOSTICS FOR HETEROSKEDASTICITY
RANDOM COEFFICIENTS
TEST                             DF        VALUE           PROB
Breusch-Pagan test                6          76.685          0.0000
Koenker-Bassett test              6          30.011          0.0000

SPECIFICATION ROBUST TEST
TEST                             DF        VALUE           PROB
White                            27         214.924          0.0000
================================ END OF REPORT =====================================
```

The R-squared is 0.76 which is the same in the R OLS model.

Create the OLS model with spatial diagnostics, which need to set the spat_diag=True, moran=True when we build the OLS model.

```
DIAG_ols = pysal.model.spreg.OLS(y, X, QueenWeights, spat_diag=True, moran=True, name_y='POVERTY', name_ds='columbus', white_test=True)
```

Now we could print the model summary with spatial diagnostics. These include Lagrange multiplier tests and Moran's I of the residuals.

```
print(DIAG_ols.summary)
```

The output of the OLS model summary spatial diagnostics part is provided below:

```
DIAGNOSTICS FOR SPATIAL DEPENDENCE
TEST                             MI/DF       VALUE           PROB
Moran's I (error)                0.0658          2.483           0.0130
Lagrange Multiplier (lag)           1            0.355           0.5515
Robust LM (lag)                     1            3.018           0.0824
Lagrange Multiplier (error)         1            4.219           0.0400
Robust LM (error)                   1            6.882           0.0087
Lagrange Multiplier (SARMA)         2            7.236           0.0268
```

3.2.2.2 Spatial lag model

Firstly, we need to specify the spatial weights matrix that includes the spatial configuration of the observations into the error component of the model.

```
QueenWeights = pysal.lib.weights.Queen.from_shapefile(shp_path)
```

Create the Spatial lag model by using pysal.model.spreg.ML_Lag class in PySAL

```
lagModel = pysal.model.spreg.ML_Lag(y, X, w = QueenWeights, method="ORD",
spat_diag=True)
```

Print out the Spatial lag model summary

```
print(lagModel.summary)
```

The output of the spatial lag model summary is provided below:

```
print(lagModel.summary)
REGRESSION
----------
SUMMARY OF OUTPUT: MAXIMUM LIKELIHOOD SPATIAL LAG (METHOD = ORD)
-----------------------------------------------------------------
Data set            :     unknown
Weights matrix      :     unknown
Dependent Variable  :     dep_var                Number of Observations:         297
Mean dependent var  :     20.3580                Number of Variables   :           8
S.D. dependent var  :     15.0894                Degrees of Freedom    :         289
Pseudo R-squared    :      0.7612
Spatial Pseudo R-squared:  0.7607
Sigma-square ML     :      54.186                Log likelihood        :   -1014.299
S.E of regression   :       7.361                Akaike info criterion :    2044.598
                                                 Schwarz criterion     :    2074.148

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     z-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT      20.5007603       2.6814965       7.6452683       0.0000000
               var_1      -0.0644013       0.0463091      -1.3906834       0.1643214
               var_2       0.1025021       0.0280097       3.6595272       0.0002527
               var_3      -0.1263485       0.0437430      -2.8884277       0.0038717
               var_4       0.5769998       0.0502306      11.4870254       0.0000000
               var_5       0.1220857       0.0351110       3.4771320       0.0005068
               var_6      -0.0000466       0.0000243      -1.9160195       0.0553626
           W_dep_var      -0.0041380       0.0680536      -0.0608048       0.9515147
------------------------------------------------------------------------------------
================================ END OF REPORT =====================================
```

3.2.2.3 Spatial error model

Create the Spatial error model model by using pysal.model.spreg.ML_Error class in PySAL

```
errorModel = pysal.model.spreg.ML_Error(y, X, w = QueenWeights, method="ORD",
spat_diag=True)
```

Print out the Spatial lag model summary

```
print(errorModel.summary)
```

The output of the spatial error model summary is provided below:

```
print(errorModel.summary)
REGRESSION
----------
SUMMARY OF OUTPUT: MAXIMUM LIKELIHOOD SPATIAL ERROR (METHOD = ORD)
-----------------------------------------------------------------
Data set            :     unknown
Weights matrix      :     unknown
Dependent Variable  :     dep_var                Number of Observations:         297
Mean dependent var  :     20.3580                Number of Variables   :           7
S.D. dependent var  :     15.0894                Degrees of Freedom    :         290
Pseudo R-squared    :      0.7607
Sigma-square ML     :      53.541                Log likelihood        :   -1013.034
S.E of regression   :       7.317                Akaike info criterion :    2040.068
                                                 Schwarz criterion     :    2065.924

------------------------------------------------------------------------------------
            Variable     Coefficient       Std.Error     z-Statistic     Probability
------------------------------------------------------------------------------------
            CONSTANT      19.3837917       2.2274220       8.7023438       0.0000000
               var_1      -0.0635864       0.0447152      -1.4220289       0.1550179
               var_2       0.1049873       0.0283013       3.7096271       0.0002076
               var_3      -0.1259017       0.0448952      -2.8043442       0.0050419
               var_4       0.5645432       0.0502033      11.2451429       0.0000000
               var_5       0.1308031       0.0348691       3.7512636       0.0001759
               var_6      -0.0000369       0.0000249      -1.4793046       0.1390589
              lambda       0.1484298       0.0992353       1.4957359       0.1347225
------------------------------------------------------------------------------------
================================ END OF REPORT =====================================
```

We could see from the summary of the OLS model, spatial lag model and spatial error model, the AIC of spatial error model is 2040.068, which is the least.

3.2.2.4 GWR model

Get the coordinates of each polygon

```
coords = []
for i in f:
    coords.append(i.centroid)
```

Use pysal.model.mgwr.sel_bw.Sel_BW class in PySAL to get the optimal bandwidth. In this case, we choose the golden section search AIC - adaptive Gaussian.

```
bw = pysal.model.mgwr.sel_bw.Sel_BW(coords, y, X, kernel='gaussian').search(criterion='AIC')
```

Create the GWR model with the optimal bandwidth by using pysal.model.mgwr.gwr.GWR class in PySAL

```
GWR = pysal.model.mgwr.gwr.GWR(coords, y, X, bw=bw, fixed=False, kernel='bisquare')
```

Name: Jing Wu
Student ID:29696576

Print out the summary of the model. We could get the summary of GWR model by using fit() method in pysal.model.mgwr.gwr.GWR class.

```
print(GWR.fit().summary())
```

The output of the GWR model summary is provided below:

```
print(GWR.fit().summary())
```

```
===========================================================================
Model type                                                         Gaussian
Number of observations:                                                 297
Number of covariates:                                                     7

Global Regression Results
---------------------------------------------------------------------------
Residual sum of squares:                                         16111.305
Log-likelihood:                                                  -1014.466
AIC:                                                              2042.932
AICc:                                                             2045.432
BIC:                                                             14460.122
R2:                                                                  0.761
Adj. R2:                                                             0.756

Variable                       Est.         SE  t(Est/SE)    p-value
------------------------------ ---------- ---------- ---------- ----------
X0                            19.851      2.188      9.073      0.000
X1                            -0.061      0.045     -1.356      0.175
X2                             0.098      0.027      3.691      0.000
X3                            -0.126      0.044     -2.843      0.004
X4                             0.576      0.051     11.366      0.000
X5                             0.121      0.035      3.470      0.001
X6                            -0.000      0.000     -1.828      0.067

Geographically Weighted Regression (GWR) Results
---------------------------------------------------------------------------
Spatial kernel:                                     Adaptive bisquare
Bandwidth used:                                                76.000
```

```
Diagnostic information
---------------------------------------------------------------------------
Residual sum of squares:                                          4364.122
Effective number of parameters (trace(S)):                          52.998
Degree of freedom (n - trace(S)):                                  244.002
Sigma estimate:                                                      4.229
Log-likelihood:                                                   -820.510
AIC:                                                              1749.015
AICc:                                                             1773.559
BIC:                                                              1948.470
R2:                                                                  0.935
Adj. alpha (95%):                                                    0.007
Adj. critical t value (95%):                                         2.736

Summary Statistics For GWR Parameter Estimates
---------------------------------------------------------------------------
Variable                   Mean        STD        Min     Median        Max
-------------------- ---------- ---------- ---------- ---------- ----------
X0                       25.524     22.006    -11.459     22.293     72.125
X1                        0.099      0.330     -0.456      0.008      0.898
X2                        0.376      0.509     -0.593      0.322      1.537
X3                       -0.191      0.207     -0.951     -0.150      0.149
X4                        0.321      0.223     -0.551      0.302      0.870
X5                        0.145      0.270     -0.349      0.138      0.551
X6                       -0.000      0.000     -0.001     -0.000      0.000
===========================================================================

None
```

We could see from the summary that the AIC of the OLS model is 2042.932, while the AIC of the GWR model is 1749.015, so the GWR model perform better.

4. Comparison

4.1 Consistency

The four linear models constructed in R and PySAL each contain many parameters. Here, we only compare the AIC values of each model to determine whether the regression analysis performed by the two software packages is consistent.

Model	AIC in R	AIC in PySAL
OLS model	2044.9	2042.9
Spatial lag model	2046.6	2044.6
Spatial error model	2042.9	2040.1
GWR model	1727.9	1749.0

We can report that there are some differences in the numerical values between the analysis results, but there is consistency in the final selection and evaluation of the model. When R and PySAL are used to implement spatial regression analysis on the same dataset, there are differences in specific values due to differences in parameter settings in the two softwares and the choice of techniques used in the implementation. The results are very close.

4.2 Software functionality

Compared with PySAL, R has more expansion packs to implement spatial data analysis. For different types of analysis, R has a special package to support, and there are still a lot of R packages are being developed, and its functionality will be more and more comprehensive. PySAL is just an extension of Python, so the limitations in spatial analysis will be greater than R.

4.3 Complexity

Since the spatial regression analysis of R and PySAL in this report is performed in the IDE, it is necessary to be familiar with R language and Python programming. Based on the understanding of R language and Python programming, it is very convenient to use extended package for spatial data analysis in R or PySAL in Python environment. Both methods can obtain relevant information by querying the corresponding API.

5. Answer the following questions based on your experience in 300 - 400 words per question

a. Which packages did you choose and why?

In the comparison between R and PySAL, I prefer to use R for spatial analysis.
First, R itself is built specifically for statistical analysis applications. It has packages available in all areas, and for spatial statistics, R has many powerful and complete packages. If PySAL is mapped to the R locale, it is just one of many R-packages.

Second, the R language simplifies many programming grammars in terms of programming syntax. A large number of package functions allow people who are not proficient in programming to complete complex statistical analysis processes by writing small amounts of code. Because today's industries have great needs for data analysis, such as data mining, machine learning, social networking, bioinformatics, financial data analysis, geographic data analysis. Some users in these fields do not have a strong programming foundation, and R's simplification of programming is easier for them to use. For some statisticians who have no programming basis, the interface is friendlier. PySAL requires the user to have some knowledge of the Python programming language before it can be used.

Again, the R package is only 70M, and the installation is extremely simple. And R has multiple platform adaptability, linux, window can be used, the required operating environment is very random. PySAL is a package in Python. It has certain requirements for the use environment. You need to configure the Python environment first.

Another advantage of R is that there are all-encompassing statistical functions that can be called; in contrast, PySAL can only use its own functions, which has limitations. PySAL has a much smaller choice of spatial data functions than R. And the use of R is mainly statisticians and data analysts. Once there is a new theory, the corresponding library will be developed for use. Since R has more functions that can be used directly, there are more options for visualization than PySAL.

b. Briefly describe your experience. Which product was more useful? Did you experience any problems (e.g. installation issues, access to tools/functions, hanging-up etc.)? Was there a tutorial and tutorial data?

Name: Jing Wu
Student ID:29696576

If we only compare the R language with the PySal package, then there is no doubt that the R language is more practical. Through the above question, the advantages of R can be summarized as follows:
1. R has more spatial analysis packages and functions than PySal
2. R's programming language is easy to learn and suitable for all fields
3. R's software package is small, easy to install and compatible, and can be applied to various operating systems.
4. Each package in R can complement each other.

However, when we can use PySal, it means that we already have a Python environment, so we can also install other Python libraries for auxiliary analysis by using the pip command. First of all, Python, like R, has lots of libraries to use. Python's standard library is huge, also Python has a definable third-party library that can be imported, which can help with the work of various fields, just like R. All standard and third-party libraries are available as long as Python is installed.

Second, Python is very scalable and embeddable. If you need a piece of critical code to run faster or if you want some algorithms not to be exposed, you can write some programs in C or C++ and then use them in a Python program to be safer.

Finally, compared to general programming languages, Python is easier to learn. Although Python is written in C language, it eliminates the complicated programming part of c and simplifies the syntax of Python.

In my opinion, both R and PySal are very useful. If the comparison is based solely on R and PySal, R is more useful. If you extend the comparison to R and Python, both have advantages and disadvantages.

There are many libraries for spatial analysis in Python, such as shapefile, osgeo, shapely, and geopandas. By combining these libraries, spatial data analysis can be performed efficiently in Python.

However, some Python libraries have specific requirements for Python versions and environments. When I installed PySal, I was running into error while installation, that was because I was using Python 2. When I upgraded to Python 3.6, I was able to install PySal smoothly.

I used an online tutorial to learn how to do the regression model in R, the link is http://michaelminn.net/tutorials/gis-spatial-regression/

As for PySal, I complete the entire code part by querying its API.

c. Describe how you could apply this product to a GIS project

In the report, I have used R and PySal for spatial regression.

First, use the lm() method in R to establish the OLS model. After adding the residual column in the original data, draw the residual map to determine whether the residual is randomly

distributed. If it is not possible to visually judge on the residual map, Moran's I test can be performed and analyzed from the Moran's I index.

If we determine that there is spatial autocorrelation, we need to construct a spatial regression model. Use the lagsarm() method in the spdep package to build the Spatial lag model and errorsarlm() to build the Spatial error model. Then use the residual map and Moran's I test to determine if there is spatial autocorrelation.

R also provides a way to build the GWR model. The best bandwidth can be obtained using the gwr.sel() method in the spgwr package. Then use the best bandwidth and gwr() methods to construct the GWR model to obtain GWR parameters. Finally, visualize the global R square map to determine the performance of the model.

6. References

Bivand, R. (2019, 04 06). *CRAN Task View: Analysis of Spatial Data*. Retrieved from r: https://cran.r-project.org/web/views/Spatial.html

DEMPSEY, C. (2017, 05 01). *Types of GIS Data Explored: Vector and Raster*. Retrieved from GIS LOUNGE: https://www.gislounge.com/geodatabases-explored-vector-and-raster-data/

ESRI. (2003, 08 0). *The Complete Geographic Information System.* Retrieved from esri: http://downloads.esri.com/support/whitepapers/ao_/420arcgis8.pdf

Grosjean, P. (2018, 04 23). *Introduction to svGUI: Manage R GUIs in a central place*. Retrieved from R: https://cran.r-project.org/web/packages/svGUI/vignettes/svGUI.html

Hijmans, R. J., & Ghosh, A. (2019, 05 26). *Spatial Data Analysis with R.* Retrieved from rspatial: https://www.rspatial.org/analysis/analysis.pdf

Rey, S. J., & Anselin, L. (2010). PySAL: Python Spatial Analysis Library. *Handbook of applied spatial analysis*, 175-193. Retrieved from pysal: https://pysal.readthedocs.io/en/latest/#