
# Becoming a Spatial Data Scientist

Everything you need to know to become a spatial expert

Table of Contents

# Foreword

*Julia Koschinsky, PhD, Executive Director, Center for Spatial Data Science, University of Chicago*

Data scientists usually treat the location attributes of data like any other attribute and apply the same non-spatial methods and tools from the regular data science toolkit. They know how to get the computational aspects to work and how to scale them. On the other side, spatial analysts are more likely to use specialized spatial methods and tools from spatial econometrics, spatial statistics, and geovisualization. But they might not know how to implement new spatial methods computationally and get them to run at scale.

More often than not, data science and spatial analytics are separate worlds with little interaction. Universities are trying to catch up on establishing and modernizing their data science curriculum to meet the growing demand but the traditional separation between computer science and geography departments usually prevails. Data science bootcamps also tend to maintain these divides. Often that leaves data scientists and spatial analysts to fend for themselves in bridging these worlds as they seek to spatially analyze location data at scale.

At the same time, there are spatial data scientists in industry, academia, and other institutions that have been working on integrating the data science and spatial analytics communities. They are moving towards establishing a field of spatial data science. Our Center for Spatial Data Science at the University of Chicago and CARTOCarto have been part of these efforts. Along the same lines, the goal of this ebook is to help data scientists bridge this gap by adding spatial methods to their traditional data science toolkits.

“Spatial data science can be viewed as a subset of generic “data science” that focuses on the special characteristics of spatial data, i.e., the importance of “where.” Data science is often referred to as the science of extracting meaningful information” (Anselin, 2019). Spatial analytics then, is relevant because it helps make sense of spatial data in ways that accounts for its special characteristics, including spatial dependence and spatial heterogeneity (Anselin, 1989), geographic projections, and zonation and scale problems.

To give a few examples, by accounting for spatial structure in data, spatial models can produce more precise and less biased estimates than non-spatial models. They can quantify if there are spillover or interaction effects between neighboring areas and identify if correlations vary across space. Spatial optimization models are useful for solving location-allocation problems such as where to best site new stores. Further, customer segmentation analysis can be improved through spatially constrained cluster methods. And spatial access metrics help identify mismatches in where supply and demand are concentrated.

One of the key current opportunities in spatial data science is the development of a next generation of spatial methods that builds on the lessons learned from earlier methods and finds new ways to model the special characteristics of georeferenced big data (Anselin, 2019; Rey 2019). Hopefully, data scientists who are reading this ebook on their path to becoming spatial data scientists will help take on this challenge.

# Chapter 1: What is Spatial Data Science and Why is it Important?

*"Spatial data science can be viewed as a subset of generic "data science" that focuses on the special characteristics of spatial data, i.e., the importance of "where." Data science is often referred to as the science of extracting meaningful information from data. In this context, it is useful to stress the difference between standard (i.e., non-spatial) data science applied to spatial data on the one hand and spatial data science on the other. The former treats spatial information, such as the latitude and longitude of data points as simply an additional variable, but otherwise does not adjust analytical methods or software tools. In contrast, "true" spatial data science treats location, distance, and spatial interaction as core aspects of the data and employs specialized methods and software to store, retrieve, explore, analyze, visualize and learn from such data. In this sense, spatial data science relates to data science as spatial statistics to statistics, spatial databases to databases, and geocomputation to computation." -- Luc Anselin*

This quote from Professor Luc Anselin, a founding father in the field of spatial data science, provides not only the definition of the field and the background necessary for any data science practitioner to better understand the relationship between spatial and non-spatial data, but provides also the raison d'être for this resource.

Data science is the fastest growing profession in the United States, with opportunities expanding exponentially, year-over-year. These opportunities stem from the realization among corporate and governmental leadership across all sectors, that in order to remain competitive, business and societal decisions must be informed and reenforced by data.

It is more recent though, that these same leaders have come to recognize how impactful spatial analysis can be in this decision-making process, providing an additional level of insight. Nearly every business in the world has a spatial component. And as Professor Anselin noted, unlocking insights from spatial data involves distinct tools and techniques. By arming ourselves with these

methodologies, data scientists can provide greater value and investigate the spatial relationships that underpin every facet of our world.

## Types of Spatial Data

Spatial data is typically categorized into the following types (Cressie, 1993):

### Point-referenced data

Data associated with a spatial index that varies continuously across space (Figure 1). Examples include data from GPS tracking, fixed devices, high resolution satellites. This data is often useful for model inference and prediction at unsampled locations (Banerjee et al., 2014).

### Areal data:

Data associated with a fixed set of locations, with well-defined boundaries (Figure 1). The boundaries can be irregular, as in the case of administrative units (e.g. districts, regions, counties), or can be defined by a regular grid, as in the case of raster data. Typical applications consist of model inference, prediction at unsampled locations, and spatial smoothing (Banerjee et al., 2014).

### Point patterns

Data representing occurrences of events where locations themselves are random. In this context, this data is useful in evaluating possible clustering or inhibition between the observations (Banerjee et al., 2014).

### Network data

Data associated to a set of ordered points, connected by straight lines. Examples include data from mobility networks, internet, and mobile phone networks. Typical applications include the analysis of spatial networks (Barthelemy, 2011) and route optimization (Pillac et al., 2013).

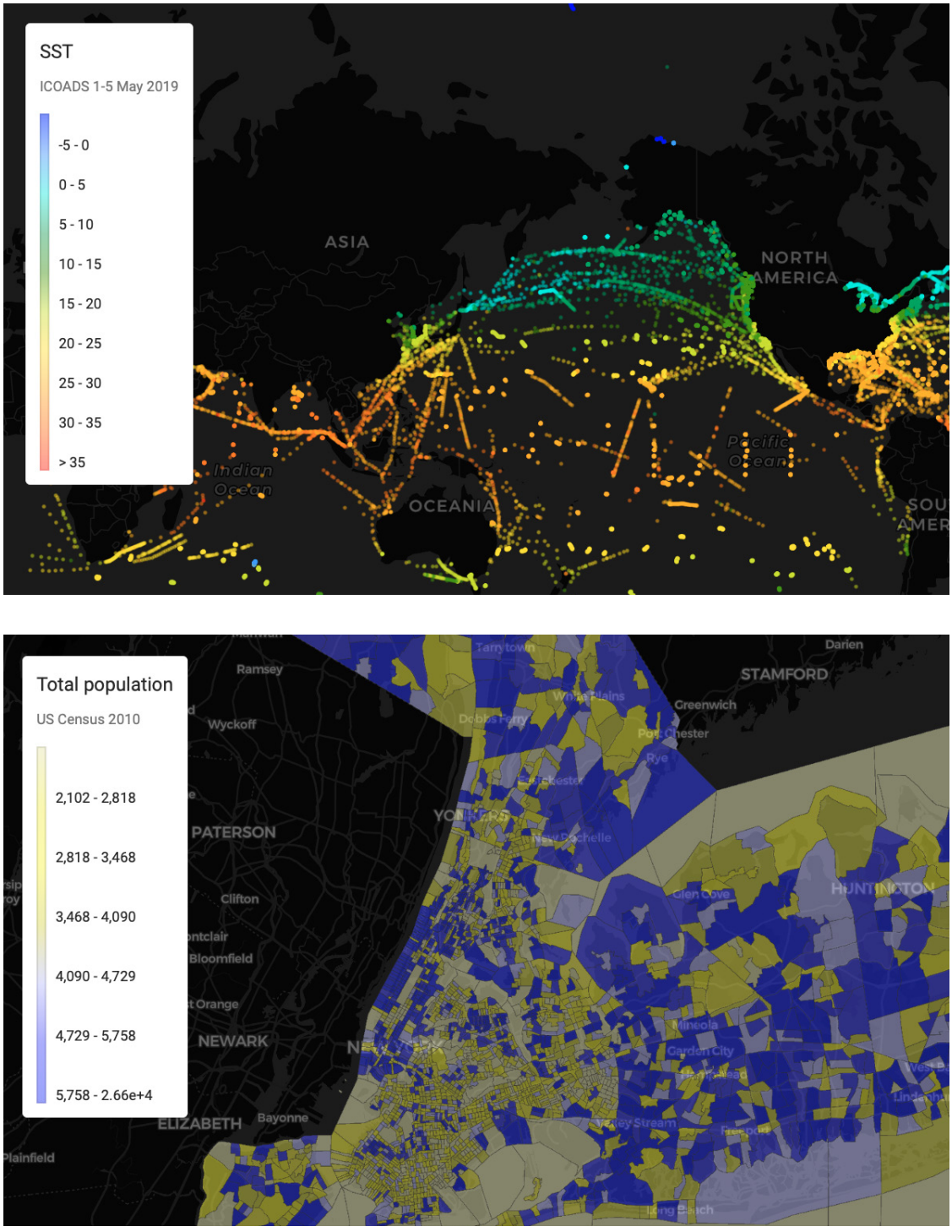



*Figure 1. Types of spatial data. (left): Example of point-referenced data: Sea Surface Temperature in situ observations from the International Comprehensive Ocean-Atmosphere Data Set (Freeman et al., 2016). (right): Example of areal data: total population in the state of NY from US Census data (source: US Census, https://www.census.gov/ ).*

## The Earth is (almost) round

Dealing with spatial data also means that we need to be able to project an uneven spheroid, which is the Earth, on a plane or on a sphere. For a given 3D model of the Earth and an origin relative to its center (a datum), a projection is defined by appropriate functions that map the longitude and latitude coordinates to planar or spherical coordinates. These projected coordinate reference systems may be global or regional, and may have different characteristics, depending amongst other if they better preserve distances, scales, shapes, or seek more of a visualization balance. For example, the Mercator projection, which is the standard map projection for navigation, preserves shapes, while the Mollweide projection preserves area measures. The knowledge of the coordinate reference system (CRS) is critical in order to establish the units of measurements, compute distances and describe the relative position of different regions (e.g. in a neighbourhood structure). A comprehensive list of available CRS' is compiled and updated by several sources, such as http://epsg.io/.

**Dive into our notebook which will be linked throughout this ebook, to learn more about visualizing spatial data using CARTOframes, CARTO's Python package for integrating CARTO maps, analysis, and data services into data science workflows. Why is Spatial Special? Spatial Dependence**

## Why is Spatial Special? Spatial Dependence

Spatial data is geographically referenced data, given at known locations and often represented visually through maps. That geographic reference, or the location component of the data, may be represented using any number of coordinate reference systems, for example, longitude and latitude.

One of the useful properties of spatial data is that data at nearby locations

"tends" to be similar. This was first recognized by Geographer and Cartographer Waldo Tobler in 1970 through his "First Law of Geography," which states that "everything is related to everything else, but near things are more related than distant things." In other words, spatial data is spatially dependent or correlated, and independence between the observations, which is a common assumption for many statistical techniques, is not satisfied.

So how is spatial dependence generated? Spatial dependence can arise for various reasons (Diggle et al., 2013). An observed spatial pattern may be observed in variables strictly depending on the location, or because of direct interactions between the points. In practice, it can be difficult, or even impossible, to distinguish empirically between these different processes.

## Measures of Spatial Dependence

Different measures of spatial dependence exist, varying depending on whether you are dealing with continuous spatial processes (the spatial index is assumed to vary continuously), discrete spatial processes (the spatial index only assumes discrete values), or point-pattern processes.

### Measure 1: Covariance Functions and Variograms

A continuous spatial process $u(\boldsymbol{s})$ is often assumed to follow a Gaussian distribution and if so, is then called a Gaussian Process (GP). A GP is parameterized by a mean function and covariance function. In principle, any covariance function $C(\cdot)$ which produces a positive-definite matrix for any input can be used to model the dependence between two observations. In practice, often the covariance function between two points is assumed to be the same if they are shifted in space (i.e. the GP is *stationary*) and to depend only on the distance between them (i.e. the GP is *isotropic*), but non-stationary (and/or *anisotropic*) models are also possible (although more difficult to specify) (Banerjee et al., 2014). An example of a stationary and isotropic covariance model is given by the exponential covariance function

$$C(|\boldsymbol{h}|) = \sigma^2 exp\left(-\frac{|\boldsymbol{h}|}{\rho}\right) \quad (3)$$

where $\sigma^2$ is a scale parameter, and controls the range of the spatial process (small values will imply a fast decay in the correlation with distance). When the stationarity condition is also satisfied by the variance function, we can also define the semivariogram $\gamma(\mathbf{h})$ as

$$Var\big(y(\mathbf{s}+\mathbf{h}) - y(\mathbf{s})\big) =: 2\gamma(\mathbf{h}) = 2\left(C\big(0\big) - C\left(\mathbf{h}\right)\right) \quad (4)$$

An empirical variogram can be constructed by grouping the pairs of observations in bins based on their distance (if isotropy is assumed) or on their distance and direction (for anisotropic processes) and averaging the squared differences from the values for all pairs. Given a semivariogram model, we can then fit this model to the empirical semivariogram and estimate the model parameters.

## Measure 2: Moran's I

For discrete spatial processes, the spatial dependent relationship is characterized in terms of adjacency. Given observations associated with a discrete index $i = 1, , n$ we can construct a neighbourhood structure with entries $w_{ij}$ which connects units $i$ and $j$ in some fashion (e.g. $w_{ij} = 1$ if $i$ and $j$ are neighbours and zero otherwise).

To measure the strength of spatial association for discrete processes, a standard statistics is given by Moran's $I$ coefficient, which is calculated as a ratio of the product of the variable of interest and its spatial lag, with the cross-product of the variable of interest, and adjusted for the spatial weights used (Bivand et al., 2013)

$$I = \frac{n \sum_i \sum_j \left(y_i - \bar{y}\right)\left(y_j - \bar{y}\right) w_{ij}}{\left(\sum_{i \neq j} w_{ij}\right) \sum_j \left(y_i - \bar{y}\right)^2} \quad (5)$$

where $\bar{y}$ represents the mean. By comparing the computed $I$ with the mean and variance of its asymptotic distribution under the null hypothesis that the $y_i$ are IID (which is a normal distribution) we can use this coefficient as an exploratory

measure of spatial association. However, if the aim is to run a test of statistical significance, a Monte Carlo permutation-based approach, in which the values of $y_i$ are randomly assigned to the spatial entities, is typically recommended (Bivand et al., 2013). Moran's $I$ coefficient can be also applied in a local fashion (Anselin, 1995) to identify local clusters and local spatial outliers.

## Measure 3: Spatial Randomness

With point pattern processes, the most basic test of spatial dependence is that of Complete Spatial Randomness (CSR), which consists of assessing if the events are distributed randomly and uniformly over the study area, or alternatively, that there are regions where the events are more likely to occur.

A very basic form of point pattern analysis involves summary statistics such as the mean center and a measure of dispersion given, for example, by the standard deviational ellipse, which separates the distance for each axis.

Testing for CSR, typically involves computing variation of the observations' density across a study area. For example, an empirical sampling distribution from a large number of $\chi^2$ test statistics can be derived comparing, in a rectangular tessellation over the study area, the observed against the expected point counts under the CSR hypothesis, and then calculating a corresponding pseudo p-value (aka Quadrat statistic). To measure the degree of CSR, several test functions are available. For example, the G-function measures the distribution of the distances $d_i$ from an arbitrary point to its nearest point within a range $r$

$$\widehat{G}(r) = \frac{\#\{d_i : d_i < r\}}{n} \tag{6}$$

and compares it to its value under CSR (Bivand et al., 2013)

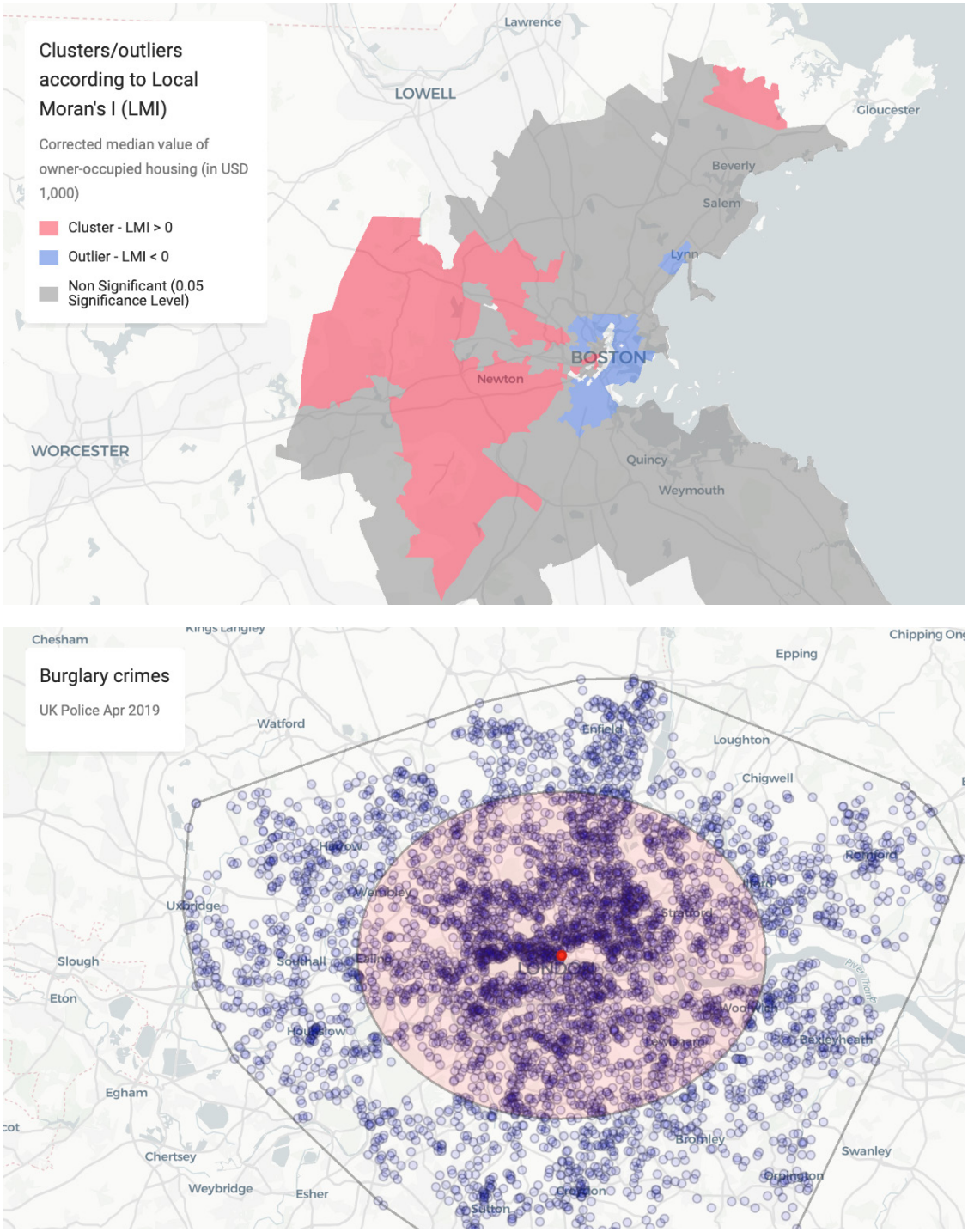



*Figure 2. Measures of spatial dependence for the Boston housing data (Harrison and Rubinfeld, 1978). (left): Local Moran's I plot. (c): Mean center and ellipsoid for the London Police crime data (http://data.police.uk/).*

For those looking to perform their own analysis, the below table includes examples of common packages used for exploratory analysis for measures of spatial dependence:

| Package | Language | Reference | Method |
|---|---|---|---|
| **gstat** | R | https://rdrr.io/cran/gstat | Variogram analysis |
| **scikit-gstat** | Python | https://pypi.org/project/scikit-gstat | Variogram analysis |
| **spdep** | R | https://rdrr.io/cran/spdep | Moran Statistics |
| **PySAL** | Python | https://pysal.readthedocs.io/en/latest/index.html | Moran Statistics, Tests for Spatial Randomness |
| **spatstat** | R | https://rdrr.io/cran/spatstat | Tests for Spatial Randomness |

**Our accompanying notebook has workflow examples of computing measures for spatial dependence**

# Chapter 2: Spatial Modeling - Leveraging Location in Prediction

Spatial modeling consists of the analysis of spatial data (i.e. data that exhibits spatial dependence) to make inferences about the model parameters, to predict at unsampled locations, or for downscaling/upscaling applications (Anselin, 1988; Banerjee et al., 2014).

With new techniques and technologies, increased processing power, and the proliferation of spatial expertise across industries, spatial modeling is evolving to meet new challenges. Areas of applications are many, including climatology, epidemiology, real estate, and marketing, and possible questions that may arise include the following: How does the revenue of my store depend on socio-demographic patterns? Are my clients more likely to churn if their neighbours are also churning? How are the spatial patterns of road incidents related to road and demographic features?

For non-spatial data, a variable of interest can usually be modeled as

$$y = \mu + \varepsilon \qquad (1)$$

where $\mu$ is the mean structure and can depend on some covariates $\boldsymbol{x}$ (also known fixed effects, as e.g. $\mu = x\,\beta$ ) and $\mathcal{E}$ represents an IID process. When dealing with spatial data $y = y(\boldsymbol{s})$, we might add to Equation (1) an extra term $u(\boldsymbol{s})$ representing a spatial random effect

$$y(\boldsymbol{s}) = \mu(\boldsymbol{s}) + \epsilon + u(\boldsymbol{s}) \qquad (2)$$

where represents a spatial index (e.g. longitude/latitude coordinates or the area identifier). effectively acts as a spatial smooth, ensuring that observations that are close in space will be also "close in " and constitutes the added value of the spatial dependence property.

# Continuous Spatial Error Models

Models for continuous spatial processes $y(\boldsymbol{s})$ are typically based on Gaussian Processes and imply a covariance/semivariogram model to describe the spatial dependence structure. Traditional methods used for mapping continuous spatial variables are based on the kriging methodology (Cressie, 1993). In kriging, predictions at new locations come from a weighted average in the neighborhood by assuming a known covariance/semivariogram. Different types of kriging exist depending on the assumptions on the mean and covariance or semivariogram of the process. Ordinary kriging is used to estimate a variable of interest when a variogram is known, while universal and regression kriging are adopted when the model includes some covariates and the variogram is estimated from the model residuals.

Computing the empirical semivariogram and fitting a semivariogram model comes with some uncertainty, both in the construction of the empirical semivariogram and when fitting the semivariogram model to it. In order to properly account for these uncertainties in the estimates/predictions and to enable the development of more complex, realistic models, methods for continuous spatial processes require a Bayesian framework (Banerjee et al., 2014). However, in the general situation, an exact solution does not exist and it must be inferred either by simulation, using Markov Chain Monte Carlo (MCMC) methods, or by approximate methods, for example. using the Integrated Laplace Approximation (INLA) with the Stochastic Partial Differential Equation (SPDE) approach (Bakka et al., 2018).

In order to use a GP in practice, it is necessary to either calculate the determinant or find the inverse of the covariance matrix, and, given $N$ observations computations scale as $\mathcal{O}(N^3)$. This problem, known as the "big $N$ problem," affects both traditional kriging methods and even more MCMC methods, which, despite being very general and applicable to any model, become impractical for problems characterized by large data or very complex structures, due to the computational burden. In response to these computational limitations, scalable approximations have been developed. These include low rank approximations, which decompose the covariance matrix into a smaller rank  matrix, sparse approximation methods, which focus on compactly supported covariance representations, and spectral methods, which appeal to spectral constructions of the covariance matrix (see Chapter 11 in Banerjee et al., 2014).

Point patterns can also be modeled as a continuous spatial process. In this case, the interest lies in modeling the intensity $\lambda(s)$ which varies spatially and may also depend on some covariates. The intensity can be modeled non-parametrically using kernel smoothing (Diggle, 2014), after which, logistic regression can be used to estimate the model coefficients. Alternatively, the intensity can be directly modeled as a Log-Gaussian Cox process and the model parameters estimated using the INLA/SPDE approach (Simpson et al., 2010).

Examples of R and Python packages that can be used in the context of modeling continuous spatial processes are provided in the table below.

| Package | Language | Reference | Method |
|---|---|---|---|
| **gstat** | R | https://rdrr.io/cran/gstat | Kriging |
| **PyKrige** | Python | https://pypi.org/project/PyKrige/ | Kriging |
| **mgcv** | R | https://rdrr.io/cran/mgcv | Kriging with additive models |
| **spBayes** | R | https://rdrr.io/cran/spBayes | Bayesian (MCMC) |
| **RStan** | R | http://mc-stan.org | Bayesian (MCMC) |
| **FKR** | R | https://rdrr.io/cran/fkr | Fixed Rank Kriging (low rank approximation method) |
| **R-INLA** | R | http://www.r-inla.org | Bayesian (INLA/SPDE) (sparse approximation method) |
| **spatstat** | R | https://rdrr.io/cran/spatstat | Intensity estimation by kernel smoothing (point patterns) |

*Examples of common packages used for spatially continuous error models.*

**Our accompanying notebook provides more detail on continuous spatial error models**

# Discrete Spatial Error Models

Discrete spatial models are based on an adjacency matrix $W$, which defines neighbor relationships: entries $w_{ij}$ and $w_{ji}$ are positive when regions and are neighbors, and zero otherwise. Models based on neighbourhood structures are Markovian models: the parameters for the -th area are assumed independent on all the other parameters given the set of its neighbors $N(i)$ , i.e. $p\big(\theta_i|\{\theta_j : j \neq i\}\big) = p\big(\theta_i|\{\theta_j : j \in N(i)\}\big)$. A Gaussian random field that satisfies this conditional independence property is known as a Gaussian Markov Random Field (GMRF). Under the Markovian property, the elements in the precision matrix (the inverse of the covariance matrix, $Q = C^{-1}$) are non-zero only for neighbours and diagonal elements, with the consequence that $Q$ is sparse, which leads to fast computations. Moreover, with GMRFs, since the model directly parametrizes the inverse of the covariance matrix, there is no "big $N$problem" in this context.

In order to use GMRFs, we need to construct sparse $Q$-matrices. A popular choice for this is Conditionally Autoregressive (CAR) models (Anselin, 1988; Banerjee et al., 2014), which rely on the conditional distribution of the spatial error terms

$$p\big(u_i|u_j, j \neq i, \tau_i^{-1}\big) = N\left(\alpha \sum_{j \in N(i)} w_{ij}\, u_j, \tau_i^{-1}\right)$$

i.e. the conditional distribution for the GMRF component for the -th area is normal with a mean that depends, with strength , on the average of its neighbors. The construction of the spatial adjacency matrix determines the class of the CAR model structure: for example, Intrinsic CAR (ICAR) models provide spatial smoothing by averaging measurements of directly adjoining regions. Another common option is to use a Simultaneous Autoregressive (SAR) model, based instead on a spatial autoregressive error term (Banerjee et al., 2014). CAR and SAR models can be also implemented in a Bayesian framework, where they can be used as priors, as part of a hierarchical model (c.f. for example the ICAR specification in the Besag, York, Mollié model, Besag et al., 1991).

Moving away from CAR/SAR models, a different approach consists of adding a GMRF spline-based smooth to our model (Wood, 2010). This has the effect that the model parameters for any unit will vary smoothly over the neighbours of

that unit, with the degree of smoothness controlled by the rank of the GMRF (e.g. full rank corresponds to many knots as units).

Examples of R packages that can be used in the context of modeling discrete spatial processes are provided in the table below:

| Package | Language | Reference | Method |
|---|---|---|---|
| **mgcv** | R | https://rdrr.io/cran/mgcv | GMRF smooths |
| **spdep** | R | https://rdrr.io/cran/spdep | CAR/SAR error models |
| **R-INLA** | R | http://www.r-inla.org | CAR/SAR error models |

*Examples of common packages used for spatially discrete error models*

**Our accompanying notebook provides more detail on discrete spatial error models**

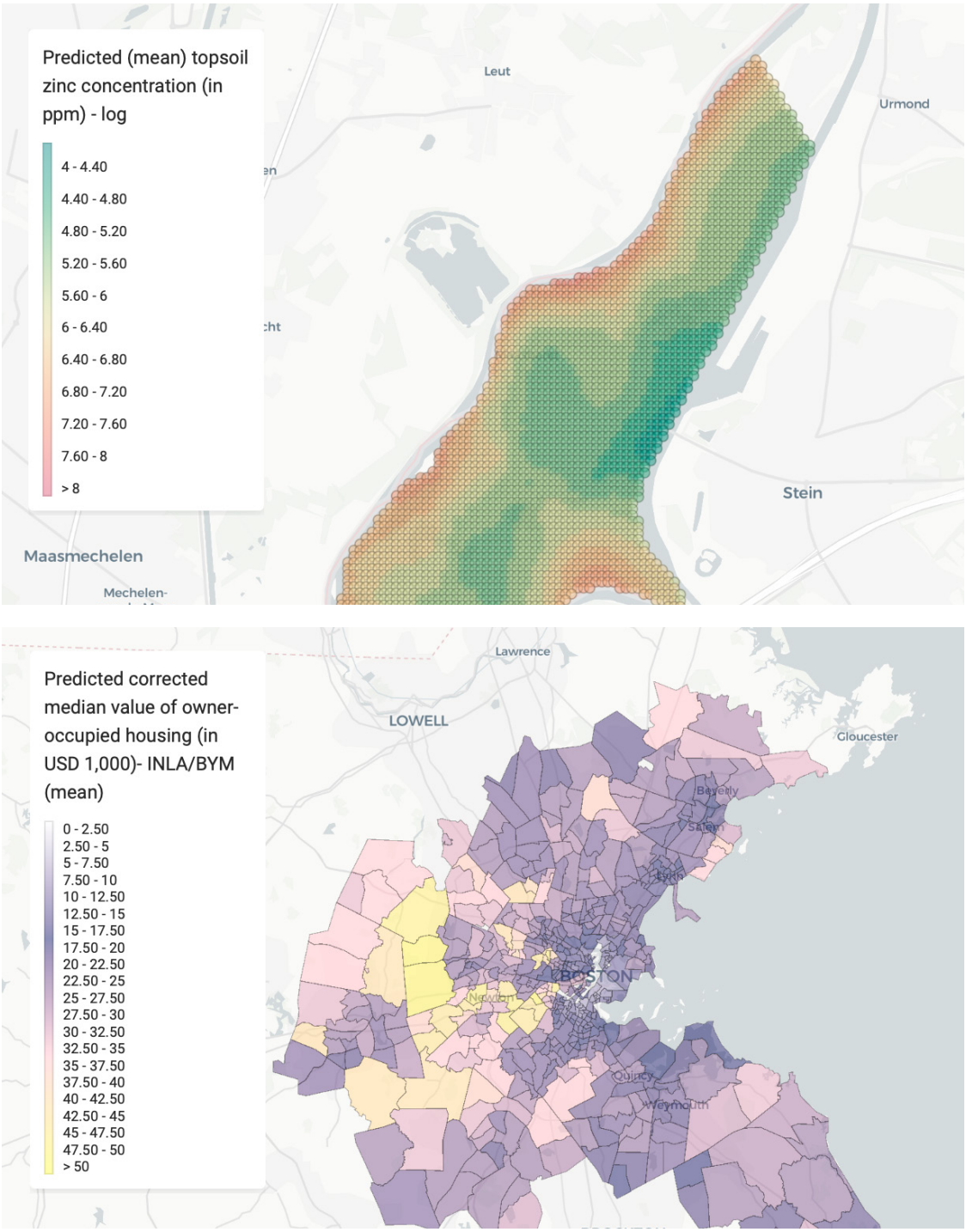



*Figure 3. Predictions (mean) for the concentration of Zinc near the Meuse river in the Netherlands obtained using kriging (left) and predictions (mean) for the owner occupied housing value in Boston obtained using INLA and a Besag, York, Mollié (BYM) model for the spatial random effects (right)*

## Spatially Varying Coefficient Models

In some cases it could be attractive to allow the coefficients in the model to vary by location, envisioning for a particular coefficient a spatial surface, e.g . Geographically weighted regression (GWR; Brunsdon et al., 2010) is the representative approach for spatially varying coefficient (SVC) models for point-referenced data. GWR estimates one set of coefficient values for every observation using all of the data falling within a fixed window (bandwidth) from this location, and giving the most weight to the data that is closest. Because the results tend to depend on the choice of the bandwidth, this method is mainly used as an exploratory technique intended to indicate where non-stationarity is taking place. Better options are represented by spline-based methods (Wood, 2010) and, although more computationally demanding, Bayesian methods (Gelfand et al., 2003; Gamerman et al. 2003). Bayesian SVC models for areal data are also available by using the GMRF specification, for example as implemented in the R package R-INLA (Bakka et al., 2018).

| Package | Language | Reference | Method |
|---|---|---|---|
| **spgwr** | R | https://rdrr.io/cran/spgwr | GWR |
| **PySAL** | Python | https://pysal.readthedocs.io/en/latest/index.html | GWR |
| **mgcv** | R | https://rdrr.io/cran/mgcv | SVC model (point-referenced) |
| **spBayes** | R | https://rdrr.io/cran/spBayes | Bayesian (MCMC) SVC model (point-referenced data) |
| **R-INLA** | R | http://www.r-inla.org | Bayesian (INLA) SVC model (point-referenced and areal data) |

*Examples of common packages used for modeling for spatially varying coefficient models.*

## Spatial Confounding

Spatial confounding occurs when adding a spatially-correlated error term changes the estimates of the fixed-effect coefficients (Hodges, 2010), especially

when the fixed effects are highly correlated with the spatially structured random effect. To avoid this effect, a solution known as restricted spatial regression, is used, consisting of restricting the spatial random effect to the orthogonal space of the fixed effects.

## Validation Tools

To assess the predictive performance of a spatial model, traditional validation tools are typically adopted. These rely on graphical methods (e.g. graphical inspection of the residuals) or computing some discrepancy measures such as the Root Mean Square Error (RMSE), the pseudo-, the Logarithmic and the Continuous Ranked Probability Score, either splitting the data into a train and a test subset or using k-fold cross validation (Hastie et al., 2017). However, extra care must be taken with spatial data since, in this case, observations that are closer in space have stronger dependencies, which can result in biased measures of discrepancy. Equivalent spatial cross-validation and bootstrap strategies based on spatial resampling-based methods can be implemented, using, for example, the R package sperrorest (Brenning, 2012). In practice, both non- and spatial cross-validation methods can be very computationally expensive when working with spatial models, and their use is still not very common.

## Spatio-temporal Models

A temporal dimension when working with spatial data is also common, as for example in the case of moving devices such as sensors, vehicles, or mobile phones. Methods for analysing spatio-temporal data model the field accounting both for a spatial dimension, as described in the previous sections, and the temporal dimension, which fundamentally differs because time flows only in one direction. For a review of spatio-temporal models see Banerjee et al. (2014) and Cressie and Wikle (2011).

# Chapter 3: Spatial Clustering and Regionalization

Like clustering in traditional data science, spatial clustering covers a wide range of methods and applications. Some traditional clustering methods can easily be adapted to spatial problems while others require a reformulation to account for the spatial relationships inherent in spatial data. Others are not well-suited for spatial problems.

Within clustering, there exists a special subclass where additional spatial constraints can be added to ensure that base geographies are contiguous. This type of clustering, which yields regions built from sub-geographies, is known as regionalization. This class of clustering methods is known as regionalization.

This chapter will cover some powerful spatial clustering methods and show some of the more common and/or powerful methods used in spatial data science. While the landscape of methods is large, many clustering algorithms don't neatly fall into existing methods and need to be custom programmed using linear programming, graphs (e.g., min cost flow), heuristic methods (e.g., genetic algorithms, tabu search, etc.), or other methods. We will not be covering these here

## Methods for Spatial Clustering

### 1. K-means for Spatial Clustering

One of the more common methods used for spatial clustering, k-means, uses data attributes to create a predefined number of categories or classes that, via those attributes are different from one another while staying alike within that category, has a couple of advantages. (Arribas-Bel)

First, the number of clusters is set a priori, as opposed to DBSCAN (described below) which requires bandwidth tweaking to establish the number of clusters

desired. Additionally this method is fast and robust, even when working on higher dimensional datasets.

K-means is useful for clustering in a parameter space where geographies are assigned categories, commonly used in segmentation (Singleton, Spielman, 2013) as well as for spatial variants where each cluster forms a spatially connected component.

| Language/Platform | Reference |
|---|---|
| **Python** | scikit-learn - https://scikit-learn.org/stable/ |
| **R** | stats - https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html |
| **PostGIS** | https://postgis.net/docs/ST_ClusterKMeans.html |

*How and where to find/use k-means*

While k-means can be used as a quick and dirty method for direct use on latitude and longitude, that should be discouraged, as it may not yield reliable results, especially if there is a strong geographical boundary such as a river. K-means works by minimizing the variance of inter-cluster values and maximizing intra-cluster values, not by minimizing distance. That is, k-means minimizes the variances of 'distance' instead of minimizing the distance within the cluster.

As such, it is generally fine for spatially close clusters (within a city for example), but distances should be computed using the haversine formula instead of straight latitude-longitude since geographical distances change with latitude. Again, this can be okay for rough clusters, but algorithm-based strategies can actually work against the goal of creating good clusters.

For example, if clusters are imbalanced (for example, one cluster has more samples than another), the cluster that has a higher number of samples will tend to be randomly selected more, resulting in more seeds centers within that area. Meanwhile, areas that have fewer samples are more likely to be assigned with the areas that are sampled more. The result is that significant clusters that

are spatially separated but have fewer samples tend to be grouped with other clusters which have more samples and are farther away. In this case, algorithms like DBSCAN are a better approach. (Boeing, G.)

## 2. DBSCAN for Spatial Clustering

DBSCAN, which is a standard piece of the data scientist toolkit, identifies clusters by grouping entities together that are within a distance r from each other such that a cluster has m points in it. If a cluster fails to meet the m points threshold, entities are classified as noise. Otherwise, they are either classified as a cluster border or as part of the cluster core. In addition to meeting the limitations listed above, an advantage of this method is that cluster shapes can be arbitrary as opposed to k-means where cluster shapes are convex. An example of using DBSCAN for clustering can be seen here.

Variants of DBSCAN have further functionality. OPTICS is a generalization of DBSCAN that is more tolerant to distances and therefore different cluster densities. HDBSCAN is another variant that takes a hierarchical approach to finding dense clusters. HDBSCAN was used in the CARTO project A Million Walks in the Park:


*Learn even more about this project on the CARTO blog*

DBSCAN was also used in a CARTO blog post about Safegraph's data. Follow along with the example notebook: https://github.com/CartoDB/data-science-book/blob/master/Chapter%202/dbscan.ipynb

Another variant of DBSCAN is Generalized DBSCAN which takes into account spatial features. See the research paper for more information. As of the time of writing, there are not implementations in common data science languages, although a Java version exists in the ELKI project.

| Language/Platform | Reference |
|---|---|
| **Python** | scikit-learn - https://scikit-learn.org/stable/ |
| **R** | https://rdrr.io/cran/dbscan/ |
| **PostGIS** | https://postgis.net/docs/ST_ClusterDBSCAN.html |

*How and where to find/use DBScan*

## Clustering with umap

Umap is a newer technique and its use in clustering is still at the stage where results are to be taken with a larger grain of salt than other clustering methods. This methodology is interesting as the data is embedded in a multi-dimensional manifold.

| Language/Platform | Reference |
|---|---|
| **Python** | https://umap-learn.readthedocs.io |
| **R** | https://rdrr.io/cran/umap/ |

*Learn more about umap*

# Regionalization

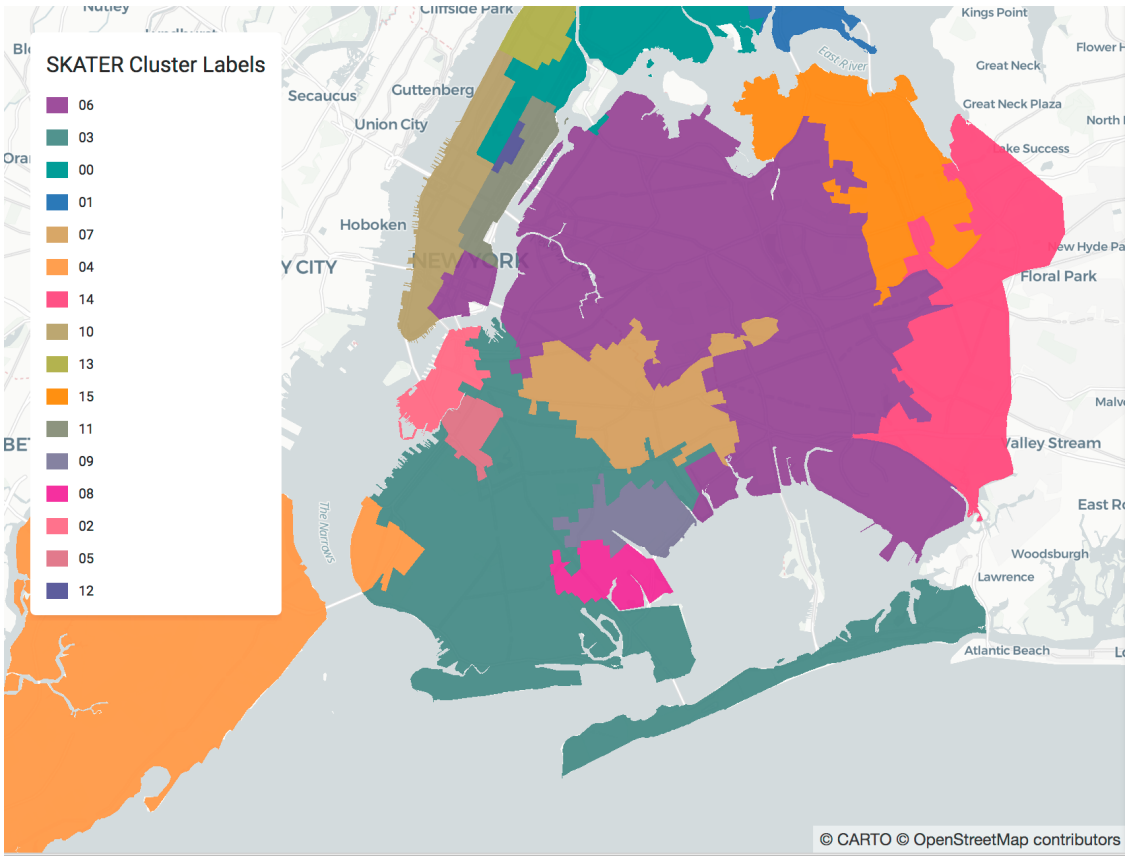
Regionalization is a type of clustering that enforces contiguity constraints on the geographies. That means that smaller geographies can be put together to form larger, contiguous regions that are constructed to optimize for qualities such as similar populations, homogenous measures (e.g., similar socio-demographic characteristics), and compactness among others. Regionalization techniques can be used to construct sales territories that are roughly fair between all sales reps, build fair voting districts, ...

## SKATER

The SKATER algorithm enables regionalization by constructing a contiguity-based minimum spanning tree that ensures homogeneity within trees by minimizing costs that are the inverse of the similarity of joined regions. (Assuncao et al, 2006). This means that a cost is associated with each neighbor pair. This can be one or more standardized attribute values that are reduced by calculating by some distance metric (e.g., manhattan, euclidean, etc.). Larger distances in the attribute space are less disimilar so are less likely to be paired. The contiguity is represented as a minimum spanning tree where cuts are made to ensure that individual regions are homogenous.

These properties of SKATER allow one to construct regions that are similar within a cluster and dissimilar from other nearby clusters.

In the skater.ipynb notebook, we demonstrate a use case to construct clusters of areas in New York City that have similar median incomes.




SKATER has advantages over similar methods in that it is relatively efficient.

| Language/Platform | Reference |
|---|---|
| **Python** | https://github.com/pysal/spopt in the near future |
| **R** | https://cran.r-project.org/web/packages/spdep/index.html |

*Resources to perform SKATER analysis*

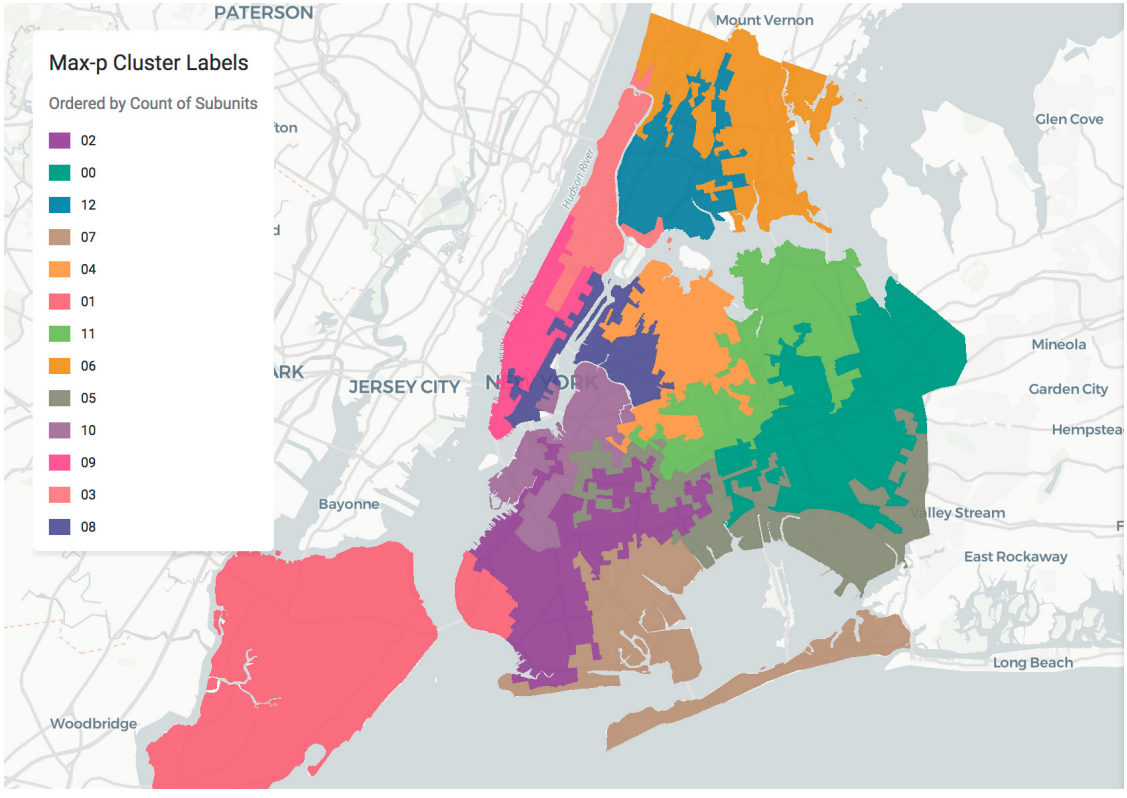
## Max-p

The Max-p-regions algorithm provides spatially constrained regions that are homogenous and meet a minimum threshold requirement. For example,

finding regions constructed from census tracts that are similar median incomes but ensuring that each region has a minimum number of households.

Max-p allows one to construct regions without pre-defining the number of regions desired, but allows for some control by establishing a minimum threshold that all regions must meet. Effectively, this can be used to force regions to be a minimum size in terms of base units (e.g., each region must contain 10 counties) or population (e.g., each region must have 10k people). This property can be useful for aggregating sub-units so that the region has a statistically significant sample when doing, for example, polling.

The drawbacks to using this method are that max-p can be slower to run for larger datasets. For this reason, heuristic-based solutions have been developed so that approximate solutions can be calculated. Max-p also can lead to the construction of non-compact regions as can be seen in the map below. This can be a problem for some applications as having spatially compact regions is important for efficiency of travel within a region depends can depend strongly on the shape of the constructed region.



## Agglomerative Clustering

Agglomerative clustering is a type of hierarchical clustering where clusters are built from the bottom up. This algorithm starts building clusters where each object is in its own cluster, then clusters are recursively merged (agglomerated) using a "linkage strategy" such as minimizing the sum of squared distances within a cluster. Similar to k-means, the cluster number is specified and initial random seeds are selected at the beginning of a run.

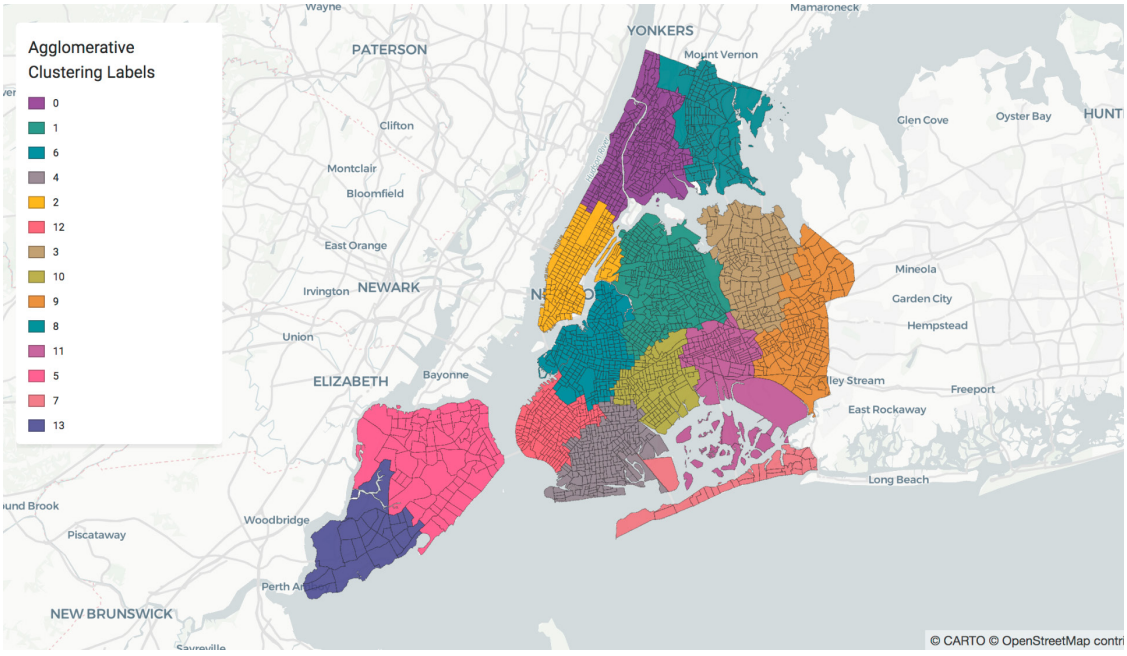
The linkage strategy in agglomerative clustering depends on the use case, but four main ones are used:

- Minimize the sum of squared distances within clusters (minimize distance variance)
- Minimize average distance within clusters
- Minimize the closest distance within clusters
- Minimize the maximum distance within clusters

What makes agglomerative clustering well suited for spatial problems is that the clusters can be built with a pre-defined connectivity graph, such that only connected clusters can be joined into larger clusters, and distances between units can be pre-calculated according to different metrics (e.g., euclidean or an arbitrary distance from an external service like a routing engine). Such graphs can created using PySAL's weights interface.

Different linkage strategies lead to clusters with different compactness. For example, given that `ward` minimizes the sum of squared distances within a cluster, it tends to create more compact clusters.

In our example in this notebook, we use PySAL's weight objects to find contiguous clusters of shoreline-clipped census tracts in New York City, and we manually add a half dozen bridge connections to demonstrate how the clusters permeate based on connected nodes.




Click here to view the interactive map

| Language/Platform | Reference |
|---|---|
| **Python** | https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html |
| **R** | https://rdrr.io/cran/cluster/ |

# Chapter 4: Logistics Optimization with Spatial Analysis

Optimization is a discipline that combines mathematical methods with computer science, and whose goal is to find the best possible element with regard to some criterion and a set of available alternatives.

It is widely used across many different sectors from logistics, to production planning, to scheduling, where it is used to make optimal decisions regarding where to open new distribution centers, to determine the inventory policy of a specific product, or to design the routes for delivering e-commerce products, to name just a few examples.

Optimization is especially useful in contexts where decision making is difficult due to the scarcity of a resource, the high investment required to run a business, or simply because taking into consideration all the relevant data and information, together with the complex interactions among the different elements of the problem is impossible for a person. This is why many decision support systems include some optimization component, allowing for the evaluation of millions of solutions in a matter of seconds.

One of the sectors where optimization is most commonly used is Logistics. Some examples of its applications in this sector include:

- Strategic decision making: Should we open a new cross-dock center? Where should we open it?
- Tactical decision making: Should distribution centers deliver everywhere or should they have specific delivery areas? What should the delivery areas of each DC be?
- Operational decision making: Given a set of pickups/deliveries: what is the optimal route? What vehicle should we use? What driver should perform each action?
- Real-time decision making: If there is an incident on-route, how should we react?

All of these logistics problems have a very strong spatial component that must be considered as part of any optimization solution.

## What does data for optimization look like?

The three most commonly used types of data in Logistics are points, lines, and distance/time.

### Points

Represent locations. They can be customers, distribution centers, stores, etc.

### Lines

Represent the transportation network. They are mainly used for visualization purposes, and in order to store information on the network's characteristics (connections between points, distances, etc.) matrix or graph structures are used.

### Distance/time

For any logistics problem, it is essential to represent its network characteristics (mainly times/distances between points, but it can be anything: costs, road safety, etc.). The most natural way to represent this information is with matrices. Since these matrices are usually very sparse, triplet or linked matrix representations are used instead

# Building our optimization model

An optimization problem consists of two main components, the model and the search.

## The Model

The model is the formulation of the problem. It can be a traditional mathematical formulation with equations, or a more conceptual formulation not necessarily expressed in mathematical terms.

A typical optimization model consists of the following components:

### 1) Decision Variables

Decision variables represent the decisions that need to be made and that will lead to an optimal solution. Depending on the logistics problem at hand, our decision variables could be whether to open a distribution center (DC) at a specific location, whether a zip code is served by a DC, or which truck will serve one customer and when.

The most frequently used variables are those with integer, binary, and continuous domains. For some specific problems, such as task assignment problems, it can also be useful to work with set variables.

### 2) Objective Function

The objective function is the measurement/metric that will allow us to determine objectively whether one solution is better than another. The typical cost function includes costs, but it can consist of anything measurable: cost, service level, etc.

The objective function can be a mathematical expression or more complex expressions. For example, in case of having a multiobjective optimization problem, we may be interested in applying the lexicographic method in which the second order goals are only evaluated if there is a tie between two or more solutions when evaluating the first order goals.

### 3) Constraints:

Constraints define what solutions are feasible from different points of view. We can have, for example, physical constraints (a truck cannot transport more than

its capacity), and business constraints (every client should not be further than 20 miles away from the closest DC).

Constraints can be algebraic expressions $x + y \leq 1$, or logical expressions (if A, then B).

Given these elements, a solution to an optimization problem consists of a set of pairs variable-value $\{(x_i, a_i)\}_{i=1,,N}$ such that if every variable is set to its paired value, $x_i = a_i$ all constraints are fulfilled.

All optimization problems have decision variables, but not all of them necessarily have an objective function or constraints. For example, when scheduling tasks, finding a solution is so complicated, that just finding a solution which satisfies all constraints is considered to be a success.

## The Search

The search is responsible for finding the best possible solution. It is called search as a reference to the exploration of the solution space (defined by the problem's constraints) performed seeking the optimal solution.

The search then consist of an algorithm or a combination of algorithms whose goal is to find a solution. Search algorithms are usually classified into two categories:

### Exact Algorithms

Exact algorithms are those which solve a problem to optimality, i.e., they find the actual optimal solution. Ideally, we would always like to use an exact algorithm to be sure we have the best possible solution. However, we are constrained by time and computational capacity, so not all problems can be solved with exact algorithms.

One of the best known exact algorithms in Optimization is the Simplex Algorithm. In Logistics, there are some interesting problems related to network design that can be solved with the Simplex Algorithm such as distribution

center (DC) opening/closing, and distribution area definition.

There are several solvers with the Simplex Algorithm and some of its variants implemented, both commercial and open source. A good choice is Google OR-Tools, an open source software suite for optimization. This suite provides you with an API to model your optimization problem, and later connect to different solvers, so you can compare their performance on your specific problem.

### Approximate algorithms:

Due to the complexity of many optimization problems, often times it is impossible to find the optimal solution in a suitable amount of time. This is the case of combinatorial optimization problems, from which the routing problems that apply in Logistics are a good example.

The goal of an approximation algorithm is to come as close as possible to the optimum value in a reasonable amount of time. Two of the best known approximation algorithms applied in routing problems are Simulated Annealing, and Tabu Search. The implementation of these algorithms can be found in Python packages such as simanneal, or Google OR-Tools that allows you to test different metaheuristics.

It is important to note that approximation algorithms cannot tell whether a solution is optimal or not, so a stopping criteria needs to be set. The most commonly used criteria is to set a time limit. A combination of a time + improvement criteria is also common. One such criteria would look like "Stop after t seconds without finding a solution which improves the last best solution found in more than a p% since it was found".

## Algorithms for Routing Problems

Focusing specifically on routing problems, algorithms can also be classified in two categories: Graph vs. Non-graph ones. This is because routing components can be represented as graphs, where the graph nodes are the places to visit (they can be distribution centers, stores, customer's addresses, etc.) and the

edges are the possible connections between pairs of nodes. Nodes and edges can be assigned weights representing elements of the problem. For example, a node can be assigned the value of its demand, and an edge can be assigned the time from the origin to the destination nodes. Some very well known graph algorithms applied for routing are Dijkstra's algorithm for finding the shortest path between two nodes, or the Christofides algorithm for solving the Traveling Salesman Problem, as seen in the next chapter.

## Optimization in Action: Solving the Traveling Salesman Problem

Perhaps the most famous and prominent routing optimization problem, with multiple methods developed for finding a solution, the Traveling Salesman Problem is defined thusly:

*"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"*

The Traveling Salesman Problem is a combinatorial optimization problem. This means it gets very hard to solve with a rather low number of cities as the number of possible solutions for *n* cities is *n!*

Because of this combinatorial complexity, exact algorithms are rarely the best approach. There is a very powerful iterative algorithm that uses integer linear programming (an exact technique) at each iteration, which ensures optimality and that has been proven to work very efficiently with instances of up to 1000 cities from one of the very well known TSP benchmarks. You can find the formulation using Gurobi here. However, when your business problem requires additional constraints, this algorithm is no longer an option. This is why the most common approach is to use approximate algorithms.

Among the different approximate algorithms, the most common ones are the family of local search algorithms, and genetic algorithms. One example of a local search algorithm is Simulated Annealing. Ant Colony Optimization (ACO) is one of the best known genetic algorithms. The main difference between

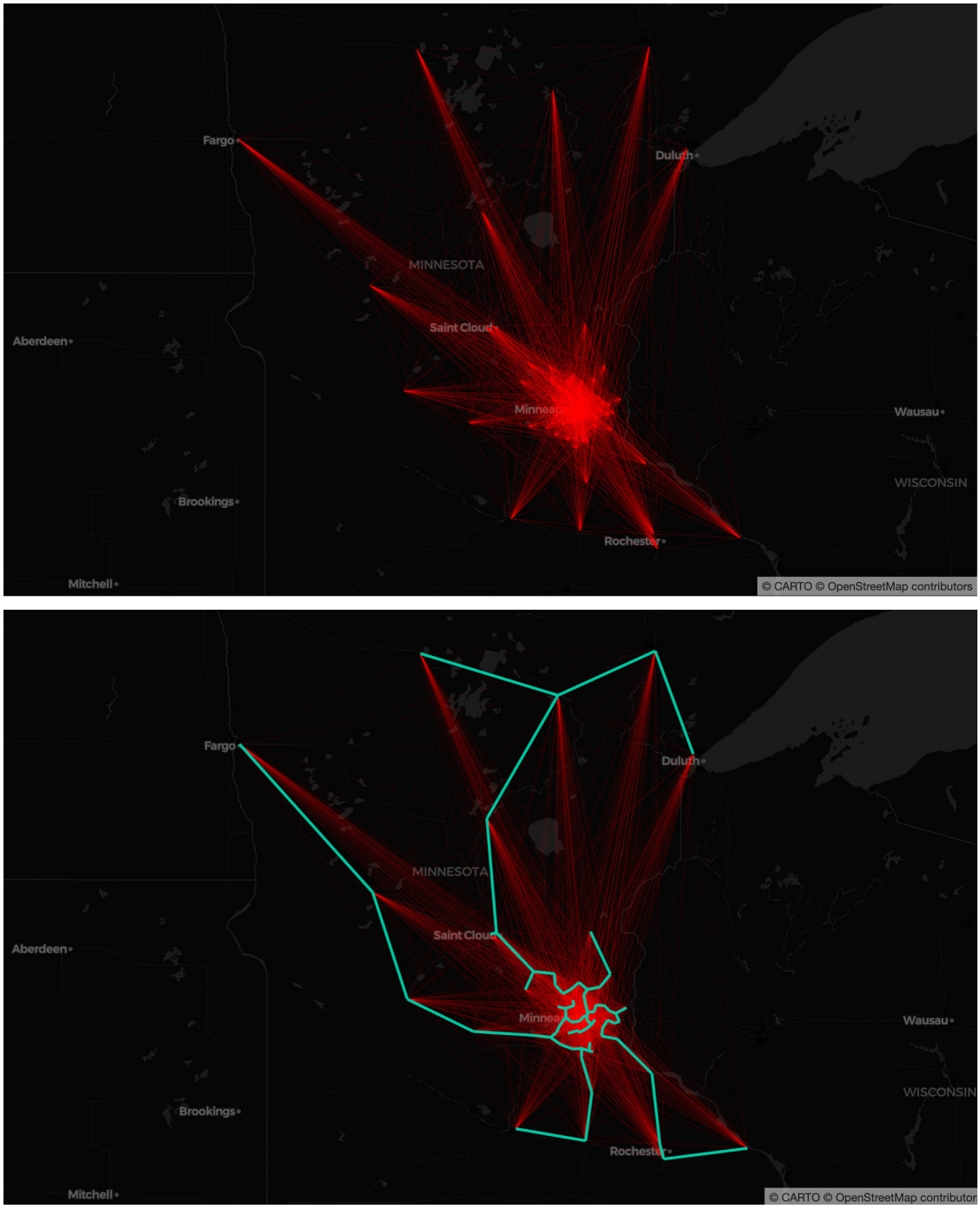
these two families of algorithms is on how the problems are formulated, and, of course, the logic behind them. While Simulated Annealing starts from one solution and keeps moving to neighboring solutions with some randomness, Ant Colony Optimization can be seen as a simulation technique in which artificial ants (simulated agents) move through the graph.

Both families of algorithms have been proved to be very powerful with routing problems, and in particular with the TSP. The main criterion for choosing one or the other usually depends on the data scientist's proficiency with each of them, and the requirements in terms of open source vs. commercial software. With local search algorithms, it is very common to find libraries with several of this family's algorithms implemented, so normally, two or three of these techniques will be tested to find the one that suits our problem best.

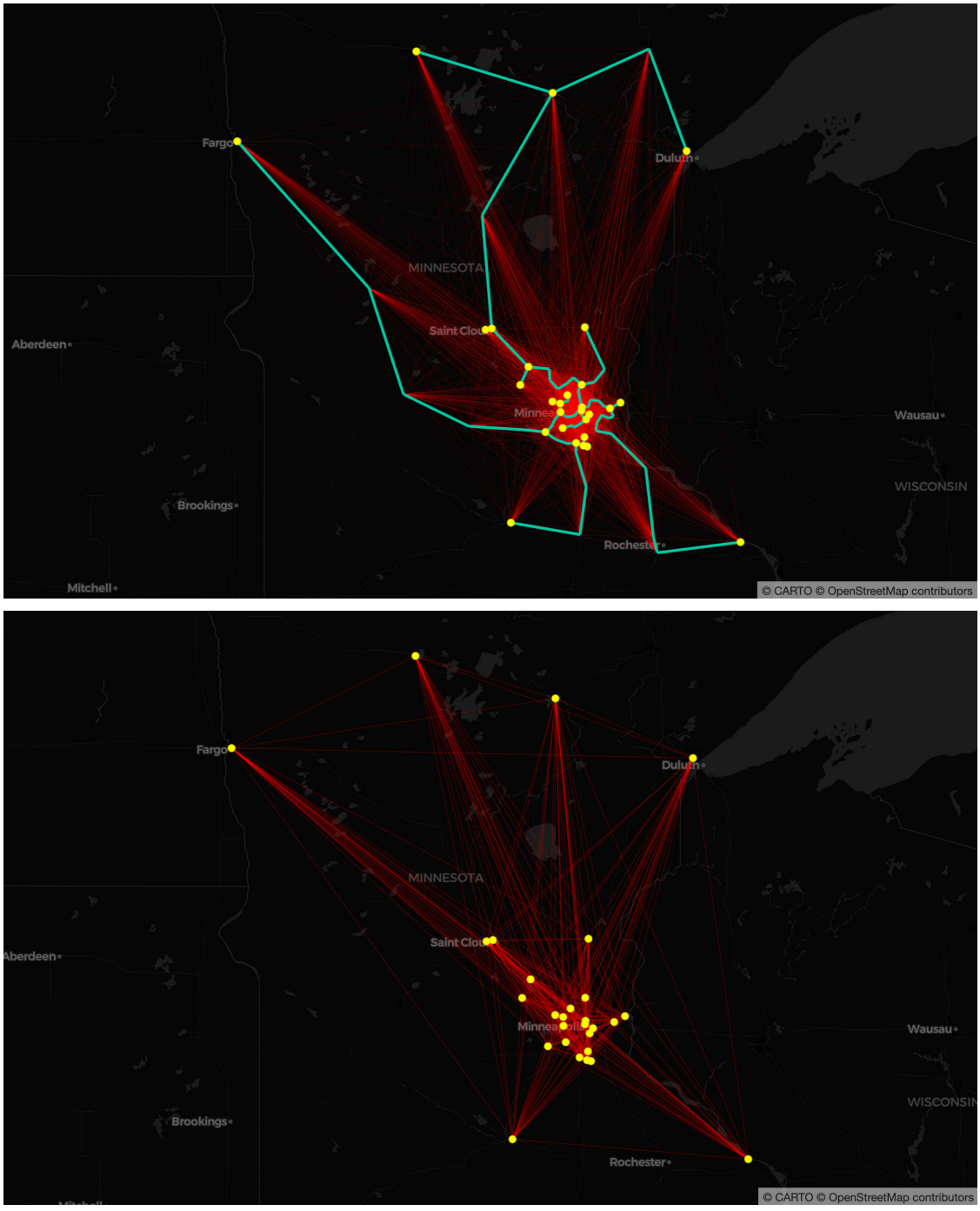
Finally, many of the algorithms used to solve the Traveling Salesman Problem require an initial solution to start from. The quality of this first solution (understanding quality as the proximity to the optimal solution) can save us many hours of testing our algorithms. A very well known algorithm for finding this first solution is the Christofides Algorithm (explained in detail below). This algorithm guarantees that its solutions will be within a factor of 3/2 of the optimal solution, so often times its solution is good enough and no further improvement is performed with local search algorithms.

## Christofides Algorithm

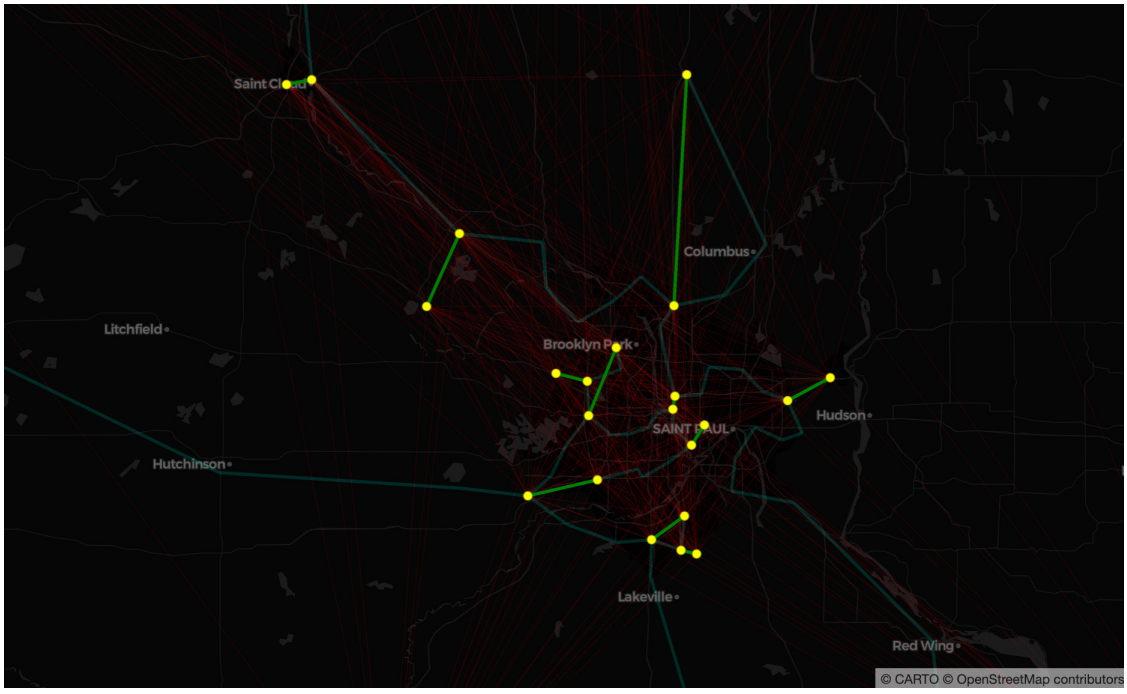
One method for solving this problem is the Christofides algorithm. Step-by-step instructions for this method are as follows:



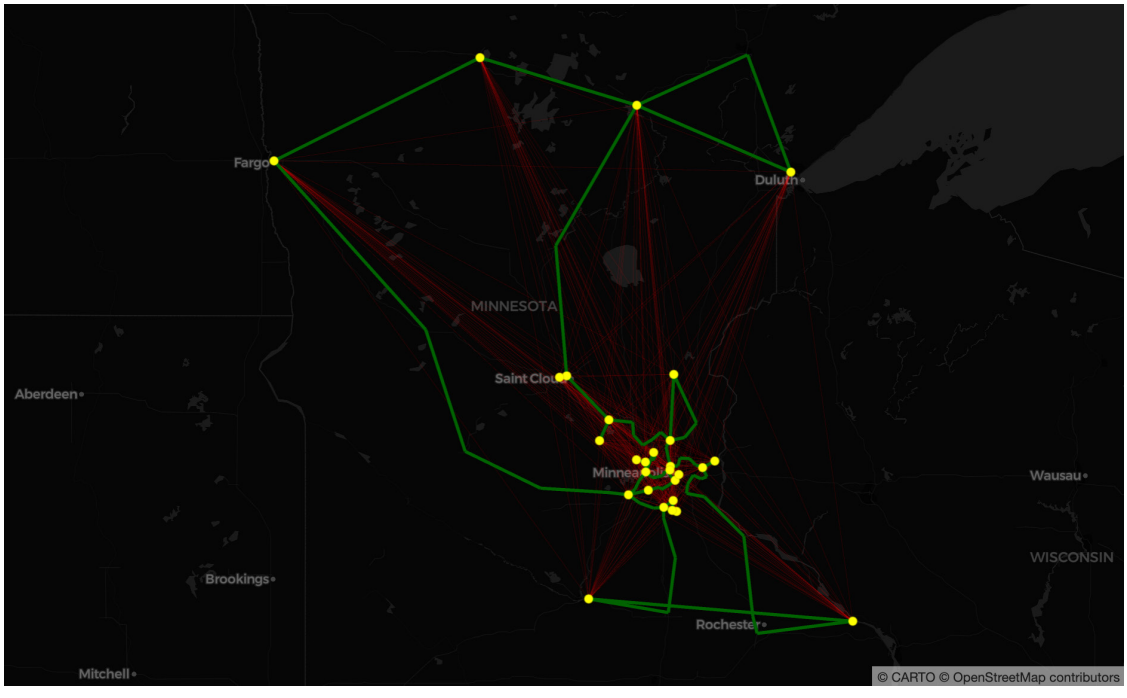

*The example aims to apply Christofides Algorithm to find the shortest path of visiting 73 retail stores in Minnesota. Create a minimum spanning tree T (right) of Complete Graph G (left)*



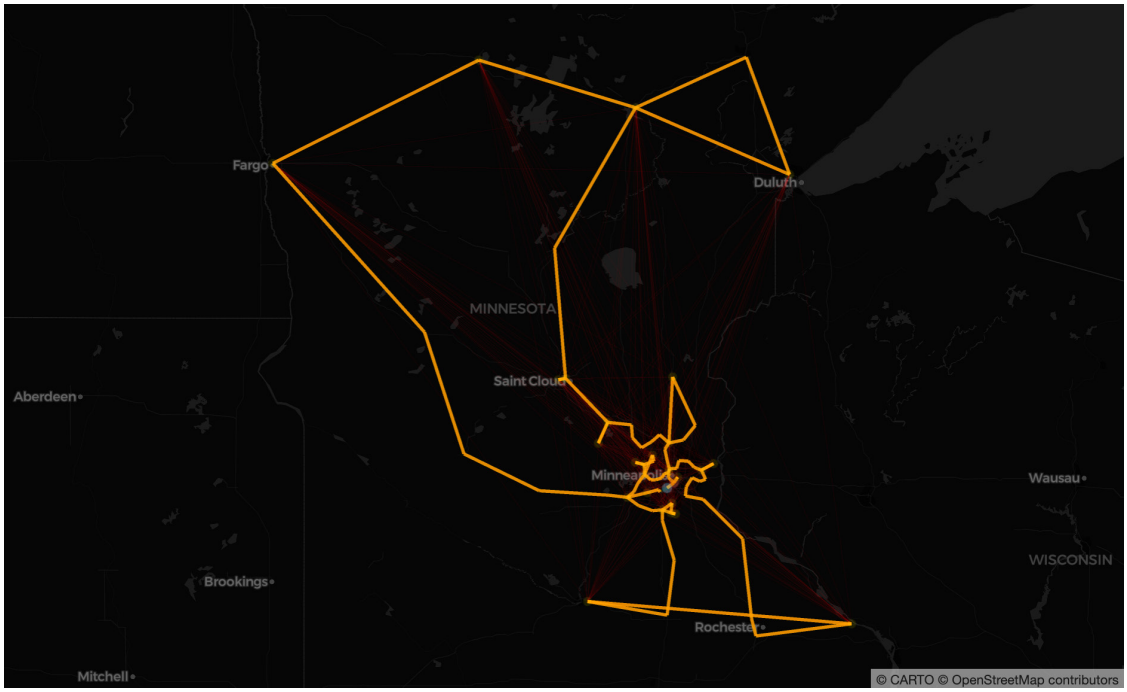

*Vertices with odd degree O (left) and subgraph G' (right) of G using only the vertices of O*



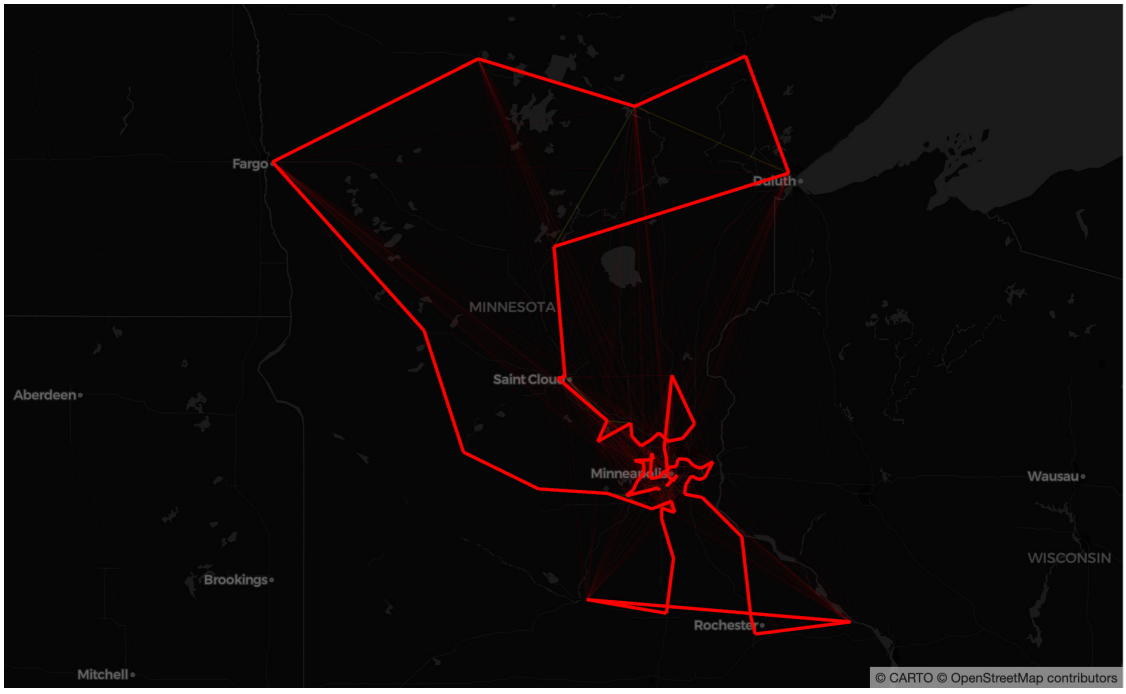

*Find a minimum-weight perfect matching $M$ (above) in the induced subgraph given by the vertices from $O$.*




*Find a minimum-weight perfect matching $M$ (above) in the induced subgraph given by the vertices from $O$.*




*Form an Eulerian circuit E (above) in H.*




*WWMake the circuit found in previous step into a Hamiltonian circuit by skipping repeated vertices (shortcutting)*

**Explore our notebook for further details on the Christofides Algorithm and alternative methods for solving the Traveling Salesman Problem**

# Chapter 5: Continue Your Spatial Education

The need for spatial data science professionals and practitioners across governments, organizations, companies, and cities is growing as these institutions continue to further recognize the importance of deriving new insights from growing location data assets. To meet this demand, many programs have sprung up, from higher education to tech bootcamps, that seek to train the next generation of spatial experts

### The University of Chicago

The Center for Spatial Data Science

- Undergraduate Study
- Post-grad study
- Research

### The University of Southern California

Spatial Sciences Institute

- Undergraduate Study
- Post-grad study
- Research

**The University of Wisconsin-Madison**

Geospatial Data Science Lab

- Post-grad study
- Research


**The University of Liverpool**

Geographic Data Science Lab

- Post-grad study
- Research


**University College London**

CASA: The Bartlett Centre for Advanced Spatial Analysis

- Post-grad study
- Research

## Arizona State University

SPARC: Spatial Analysis Research Center

- Undergraduate Study
- Post-grad study
- Research

## The University of Oregon

Spatial Cognition, Computation, and Complexity Lab

- Undergraduate Study
- Research

## The Pratt Institute

SAVI: The Spatial Analysis and Visualization Initiative

- Post-grad study
- Research

## References


Anselin, Luc. (Forthcoming). “Spatial Data Science”, in *The International Encyclopedia of Geography: People, the Earth, Environment, and Technology.*

Anselin, L. (1989). “What is Special About Spatial Data? Alternative Perspectives on Spatial Data Analysis” (89-4). *UC Santa Barbara: National Center for Geographic Information and Analysis.* Retrieved from https://escholarship.org/uc/item/3ph5k0d4.

Arribas-Bel, D and Rey, S. “Geographic Data Science with PySAL and the pydata stack” Retrieved from http://darribas.org/gds_scipy16/

AssunÇão, RM, Neves, MC, Câmara, G & Da Costa Freitas, C. (2006) “Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees”, in *International Journal of Geographical Information Science.*

Bakka, Haakon, et al. “Spatial Modeling with R-INLA: A Review.” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 10, no. 6, 2018, doi:10.1002/wics.1443.

Banerjee, Sudipto, et al. *Hierarchical Modeling and Analysis for Spatial Data*. Chapman & Hall/CRC, 2014.

Barthélemy, Marc. “Spatial Networks.” *Physics Reports*, vol. 499, no. 1-3, 2011, pp. 1–101., doi:10.1016/j.physrep.2010.11.002.

Besag, Julian, et al. “Bayesian Image Restoration, with Two Applications in Spatial Statistics.” *Annals of the Institute of Statistical Mathematics*, vol. 43, no. 1, 1991, pp. 1–20., doi:10.1007/bf00116466.

Bivand, Roger S., et al. “Applied Spatial Data Analysis with R.” 2013, doi:10.1007/978-1-4614-7618-4.

Boeing, G. (2018, March 22). Clustering to Reduce Spatial Data Set Size. https://doi.org/10.31235/osf.io/nzhdc

Brenning, Alexander. “Spatial Cross-Validation and Bootstrap for the Assessment of Prediction Rules in Remote Sensing: The R Package Sperrorest.” *2012 IEEE International Geoscience and Remote Sensing Symposium*, 2012, doi:10.1109/igarss.2012.6352393.

Brunsdon, Chris, et al. “Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity.” *Geographical Analysis*, vol. 28, no. 4, 2010, pp. 281–298., doi:10.1111/j.1538-4632.1996.tb00936.x.

Cressie, Noel A. C. *Statistics for Spatial Data*. John Wiley, 1993.

Cressie, Noel, and Christopher K. Wikle. *Statistics for Spatio-Temporal Data*. Wiley, 2011.

Diggle, Peter J., et al. “Spatial and Spatio-Temporal Log-Gaussian Cox Processes: Extending the Geostatistical Paradigm.” *Statistical Science*, vol. 28, no. 4, 2013, pp. 542–563., doi:10.1214/13-sts441.

Diggle, Peter J. *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. CRC Press, 2014.

Finley, Andrew O., et al. “SpBayes: AnRPackage for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models.” *Journal of Statistical Software*, vol. 19, no. 4, 2007, doi:10.18637/jss.v019.i04.

Freeman, Eric, et al. “ICOADS Release 3.0: a Major Update to the Historical Marine Climate Record.” *International Journal of Climatology*, vol. 37, no. 5, 2016, pp. 2211–2232., doi:10.1002/joc.4775.

Gamerman, Dani, et al. “Space-Varying Regression Models: Specifications and Simulation.” *Computational Statistics & Data Analysis*, vol. 42, no. 3, 2003, pp. 513–533., doi:10.1016/s0167-9473(02)00211-6.

Gelfand, Alan E, et al. “Spatial Modeling With Spatially Varying Coefficient Processes.” *Journal of the American Statistical Association*, vol. 98, no. 462, 2003, pp. 387–396., doi:10.1198/016214503000170.

Harrison, David, and Daniel L Rubinfeld. “Hedonic Housing Prices and the

Demand for Clean Air." *Journal of Environmental Economics and Management*, vol. 5, no. 1, 1978, pp. 81–102., doi:10.1016/0095-0696(78)90006-2.

Hastie, Trevor, et al. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2017.

Hodges, James S., and Brian J. Reich. "Adding Spatially-Correlated Errors Can Mess Up the Fixed Effect You Love." *The American Statistician*, vol. 64, no. 4, 2010, pp. 325–334., doi:10.1198/tast.2010.10052.

Kammann, E. E., and M. P. Wand. "Geoadditive Models." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 52, no. 1, 2003, pp. 1–18., doi:10.1111/1467-9876.00385.

Pillac, Victor, et al. "A Review of Dynamic Vehicle Routing Problems." *European Journal of Operational Research*, vol. 225, no. 1, 2013, pp. 1–11., doi:10.1016/j.ejor.2012.08.015.

Rey, S. (2019). "Geographical Analysis: Reflections of a Recovering Editor." *Geographical Analysis*, 0, 1–9. Retrieved from https://onlinelibrary.wiley.com/doi/abs/ 10.1111/gean.12193.

Simpson, D., et al. "Going off Grid: Computationally Efficient Inference for Log-Gaussian Cox Processes." *Biometrika*, vol. 103, no. 1, 2016, pp. 49–70., doi:10.1093/biomet/asv064.

Singleton, Alexander D. & Spielman, Seth E. "The Past, Present, and Future of Geodemographic Research in the United States and United Kingdom," *The Professional Geographer*, 66:4, 558-567, 2014. DOI: 10.1080/00330124.2013.848764

Wood, Simon N. "Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 1, 2010, pp. 3–36., doi:10.1111/j.1467-9868.2010.00749.x.

**Links for helpful packages and tools:**

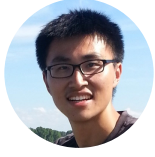
- Geopandas - http://geopandas.org/
- cartoframes==1.0b1 - https://carto.com/developers/cartoframes/
- Carto-print - https://github.com/CartoDB/carto-print
- Matplotlib - https://matplotlib.org/
- Seaborn - https://seaborn.pydata.org/
- Pandas - https://pandas.pydata.org/
- Dask - https://dask.org/
- netCDF4 - https://pypi.org/project/netCDF4/
- Jupyter - https://jupyter.org/
- NumPy - https://www.numpy.org/
- SciPy - https://www.scipy.org/
- Sklearn - https://pypi.org/project/sklearn/
- Shapely - https://pypi.org/project/Shapely/
- Fiona - https://pypi.org/project/Fiona/
- Scikit-gstat - https://scikit-gstat.readthedocs.io/en/latest/
- Pyproj - https://pypi.org/project/pyproj/
- Utm - https://pypi.org/project/utm/
- PySAL - https://pysal.org/
- Pointpats - https://pypi.org/project/pointpats/
- Rpy2 - https://pypi.org/project/rpy2/
- Ipywidgets - https://ipywidgets.readthedocs.io/en/latest/
- GeoPy - https://github.com/geopy/geopy
- Simanneal - https://pypi.org/project/simanneal/

## Authors

**Miguel Alvarez** is a Data Scientist at CARTO. His work focus has been on solving problems with a strong spatial component by enriching data, and applying spatial analysis and spatial statistics, combined with Machine Learning techniques.

**Giulia Carella** is a Data Scientist at CARTO. She holds a Ph.D. in Statistical climatology from the University of Southampton (UK) and previously she worked as a researcher at the Le Laboratoire des Sciences du Climat et de l'Environnement (France).

**Andy Eschbacher** is a Senior Data Scientist at CARTO, where he integrates data science solutions into CARTO's infrastructure, solves spatial data science problems for clients, and builds out tools to better enable people working at the intersection of data science and GIS.



Data Scientist **Dongjie Fan** has a background in Statistics & Mathematics, as well as a Master's degree in Urban Informatics from New York University. He has worked on a wide range of data science projects in different areas, including spatial analysis and image processing.

**Steve Isaac** is CARTO's Content Marketing Manager, in charge of content strategy, content creation, and copyediting across multiple channels including social media, the CARTO blog, newsletters, and more.

**Julia Koschinsky** is the Executive Director of the Center for Spatial Data Science at the University of Chicago and has been part of the GeoDa team for over ten years. She has been conducting and managing research funded through federal awards of over $8 million to gain insights from the spatial dimensions of urban challenges in housing, health, and the built environment.