Chapter 1

Overview of spatial data problems

1.1 Introduction to spatial data and models

Researchers in diverse areas such as climatology, ecology, environmental health, and real estate marketing are increasingly faced with the task of analyzing data that are:

- highly multivariate, with many important predictors and response variables,
- geographically referenced, and often presented as maps, and
- temporally correlated, as in longitudinal or other time series structures.

For example, for an epidemiological investigation, we might wish to analyze lung, breast, colorectal, and cervical cancer rates by county and year in a particular state, with smoking, mammography, and other important screening and staging information also available at some level. Public health professionals who collect such data are charged not only with surveillance, but also statistical *inference* tasks, such as *modeling* of trends and correlation structures, *estimation* of underlying model parameters, *hypothesis testing* (or comparison of competing models), and *prediction* of observations at unobserved times or locations.

In this text we seek to present a practical, self-contained treatment of hierarchical modeling and data analysis for complex spatial (and spatiotemporal) datasets. Spatial statistics methods have been around for some time, with the landmark work by Cressie (1993) providing arguably the only comprehensive book in the area. However, recent developments in Markov chain Monte Carlo (MCMC) computing now allow fully Bayesian analyses of sophisticated multilevel models for complex geographically referenced data. This approach also offers full inference for non-Gaussian spatial data, multivariate spatial data, spatiotemporal data, and, for the first time, solutions to problems such as geographic and temporal misalignment of spatial data layers.

This book does not attempt to be fully comprehensive, but does attempt to present a fairly thorough treatment of hierarchical Bayesian approaches for handling all of these problems. The book's mathematical level is roughly comparable to that of Carlin and Louis (2000). That is, we sometimes state results rather formally, but spend little time on theorems and proofs. For more mathematical treatments of spatial statistics (at least on the geostatistical side), the reader is referred to Cressie (1993), Wackernagel (1998), Chiles and Delfiner (1999), and Stein (1999a). For more descriptive presentations the reader might consult Bailey and Gattrell (1995), Fotheringham and Rogerson (1994), or Haining (1990). Our primary focus is on the issues of *modeling* (where we offer rich, flexible classes of hierarchical structures to accommodate both static and dynamic spatial data), *computing* (both in terms of MCMC algorithms and methods for handling very large matrices), and *data analysis* (to illustrate the first two items in terms of inferential summaries and graphical displays). Reviews of both traditional spatial methods (Chapters 2, 3 and 4) and Bayesian methods (Chapter 5) attempt to ensure that previous exposure to either of these two areas is not required (though it will of course be helpful if available).

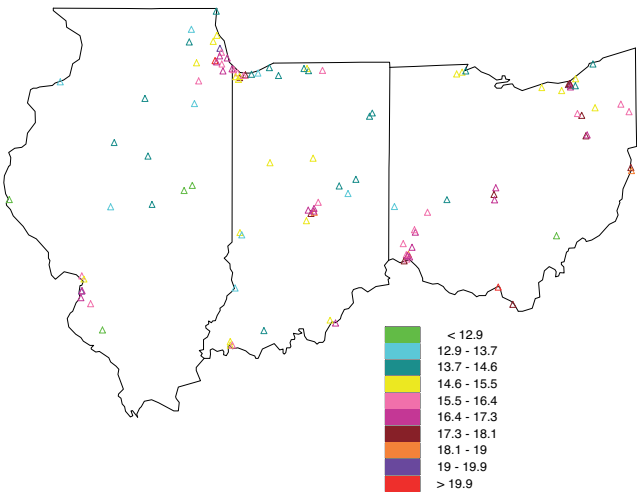


Figure 1.1 *Map of PM2.5 sampling sites over three midwestern U.S. states; plotting character indicates range of average monitored PM2.5 level over the year 2001.*

Following convention, we classify spatial data sets into one of three basic types:

- *point-referenced data*, where $Y(\mathbf{s})$ is a random vector at a location $\mathbf{s} \in \Re^r$, where $\mathbf{s}$ varies *continuously* over D, a fixed subset of $\Re^r$ that contains an r-dimensional rectangle of positive volume;
- *areal data*, where D is again a fixed subset (of regular or irregular shape), but now partitioned into a finite number of areal units with well-defined boundaries;
- *point pattern data*, where now D is itself random; its index set gives the locations of random events that are the spatial point pattern. $Y(\mathbf{s})$ itself can simply equal 1 for all $\mathbf{s} \in D$ (indicating occurrence of the event), or possibly give some additional covariate information (producing a *marked point pattern process*).

The first case is often referred to as *geocoded* or *geostatistical* data, names apparently arising from the long history of these types of problems in mining and other geological sciences. Figure 1.1 offers an example of this case, showing the locations of 114 air-pollution monitoring sites in three midwestern U.S. states (Illinois, Indiana, and Ohio). The plotting character indicates the 2001 annual average PM2.5 level (measured in ppb) at each site. PM2.5 stands for particulate matter less than 2.5 microns in diameter, and is a measure of the density of very small particles that can travel through the nose and windpipe and into the lungs, potentially damaging a person's health. Here we might be interested in a model of the geographic distribution of these levels that account for spatial correlation and perhaps underlying covariates (regional industrialization, traffic density, and the like). The use of colors makes it somewhat easier to read, since the color allows the categories to be ordered more naturally, and helps sharpen the contrast between the urban and rural areas. Again, traditional analysis methods for point level data like this are described in Chapter 2, while Chapter 6 introduces the corresponding hierarchical modeling approach.

The second case above (areal data) is often referred to as *lattice* data, a term we find misleading since it connotes observations corresponding to "corners" of a checkerboard-like grid. Of course, there *are* data sets of this type; for example, as arising from agricultural field trials (where the plots cultivated form a regular lattice) or image restoration (where the data correspond to pixels on a screen, again in a regular lattice). However, in practice most areal data are summaries over an *irregular* lattice, like a collection of county or other

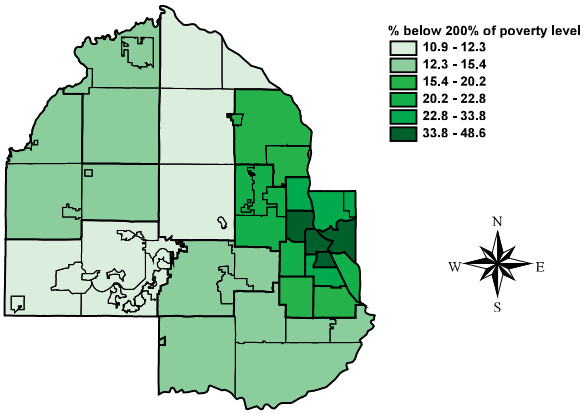


Figure 1.2 `ArcView` *map of percent of surveyed population with household income below 200% of the federal poverty limit, regional survey units in Hennepin County, MN.*

regional boundaries, as in Figure 1.2. Here we have information on the percent of a surveyed population with household income falling below 200% of the federal poverty limit for a collection of regions comprising Hennepin County, MN. Note that we have no information on any single household in the study area, only regional summaries for each region. Figure 1.2 is an example of a *choropleth map*, meaning that it uses shades of color (or greyscale) to classify values into a few broad classes (six in this case), like a histogram (bar chart) for nonspatial data. Choropleth maps are visually appealing (and therefore, also common), but of course provide a rather crude summary of the data, and one that can be easily altered simply by manipulating the class cutoffs.

As with any map of the areal units, choropleth maps *do* show reasonably precise *boundaries* between the regions (i.e., a series of exact spatial coordinates that when connected in the proper order will trace out each region), and thus we also know which regions are adjacent to (touch) which other regions. Thus the "sites" $\mathbf{s} \in D$ in this case are actually the regions (or *blocks*) themselves, which in this text we will denote not by $\mathbf{s}_i$ but by B_i, $i = 1, \ldots, n$, to avoid confusion between points $\mathbf{s}_i$ and blocks B_i. It may also be illuminating to think of the county centroids as forming the vertices of an irregular lattice, with two lattice points being connected if and only if the counties are "neighbors" in the spatial map, with physical adjacency being the most obvious (but not the only) way to define a region's neighbors.

Some spatial data sets feature *both* point- and areal-level data, and require their simultaneous display and analysis. Figure 1.3 offers an example of this case. The first component of this data set is a collection of eight-hour maximum ozone levels at 10 monitoring sites in the greater Atlanta, GA, area for a particular day in July 1995. Like the observations in Figure 1.1, these were made at fixed monitoring stations for which exact spatial coordinates (say, latitude and longitude) are known. (That is, we assume the $Y(\mathbf{s}_i)$, $i = 1, \ldots, 10$ are random, but the $\mathbf{s}_i$ are not.) The second component of this data set is the number of children in the area's zip codes (shown using the irregular subboundaries on the map) that reported at local emergency rooms (ERs) with acute asthma symptoms on the following day; confidentiality of health records precludes us from learning the precise address of any of the children. These are areal summaries that could be indicated by shading the zip codes, as in Figure 1.2. An obvious question here is whether we can establish a connection between high ozone and subsequent high pediatric ER asthma visits. Since the data are misaligned (point-level ozone but block-level ER counts), a formal statistical investigation of this question requires a preliminary *realignment* of the data; this is the subject of Chapter 7.

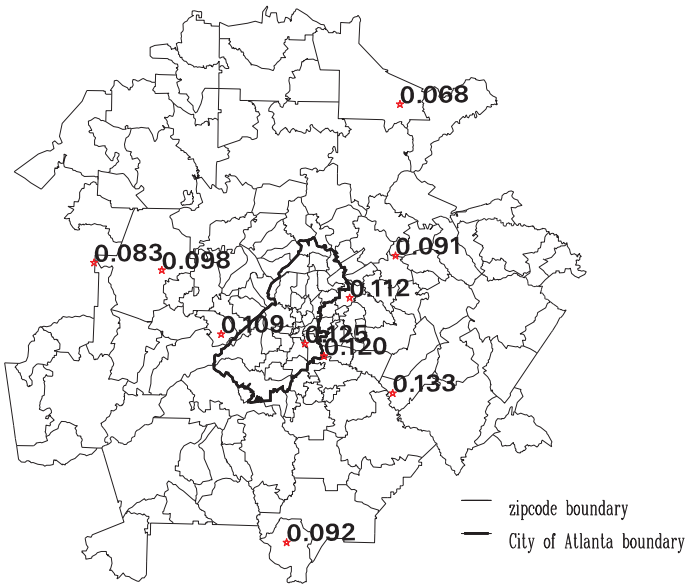


Figure 1.3 *Zip code boundaries in the Atlanta metropolitan area and 8-hour maximum ozone levels (ppm) at 10 monitoring sites for July 15, 1995.*

The third case above (spatial point pattern data) could be exemplified by residences of persons suffering from a particular disease, or by locations of a certain species of tree in a forest. Here the response Y is often fixed (occurrence of the event), and only the locations $\mathbf{s}_i$ are thought of as random. In some cases this information might be supplemented by age or other covariate information, producing a *marked* point pattern). Such data are often of interest in studies of event *clustering*, where the goal is to determine whether an observed spatial point pattern is an example of a clustered process (where points tend to be spatially close to other points), or merely the result of a random event process operating independently and homogeneously over space. Note that in contrast to areal data, where no individual points in the data set could be identified, here (and in point-referenced data as well) precise locations are known, and so must often be protected to protect the privacy of the persons in the set.

In the remainder of this initial section, we give a brief outline of the basic models most often used for each of these three data types. Here we only intend to give a flavor of the models and techniques to be fully described in the remainder of this book.

Even though our preferred inferential outlook is Bayesian, the statistical inference tools discussed in Chapters 2 through 4 are entirely classical. While all subsequent chapters adopt the Bayesian point of view, our objective here is to acquaint the reader with the classical techniques first, since they are more often implemented in standard software packages. Moreover, as in other fields of data analysis, classical methods can be easier to compute, and produce perfectly acceptable results in relatively simple settings. Classical methods often have interpretations as limiting cases of Bayesian methods under increasingly vague prior assumptions. Finally, classical methods can provide insight for formulating and fitting hiearchical models.

1.1.1 Point-level models

In the case of point-level data, the location index $\mathbf{s}$ varies *continuously* over D, a fixed subset of $\Re^d$. Suppose we assume that the covariance between the random variables at two

locations depends on the *distance* between the locations. One frequently used association specification is the exponential model. Here the covariance between measurements at two locations is an exponential function of the interlocation distance, i.e., $Cov(Y(\mathbf{s}_i), Y(\mathbf{s}_{i'})) \equiv C(d_{ii'}) = \sigma^2 e^{-\phi d_{ii'}}$ for $i \neq i'$, where $d_{ii'}$ is the distance between sites s_i and $s_{i'}$, and σ^2 and ϕ are positive parameters called the *partial sill* and the *decay parameter*, respectively ($1/\phi$ is called the *range parameter*). A plot of the covariance versus distance is called the *covariogram*. When $i = i'$, $d_{ii'}$ is of course 0, and $C(d_{ii'}) = Var(Y(\mathbf{s}_i))$ is often expanded to $\tau^2 + \sigma^2$, where $\tau^2 > 0$ is called a *nugget effect*, and $\tau^2 + \sigma^2$ is called the *sill*. Of course, while the exponential model is convenient and has some desirable properties, many other parametric models are commonly used; see Section 2.1 for further discussion of these and their relative merits.

Adding a joint distributional model to these variance and covariance assumptions then enables likelihood inference in the usual way. The most convenient approach would be to assume a multivariate *normal* (or *Gaussian*) distribution for the data. That is, suppose we are given observations $\mathbf{Y} \equiv \{Y(\mathbf{s}_i)\}$ at known locations $\mathbf{s}_i$, $i = 1, \ldots, n$. We then assume that

$$\mathbf{Y} \mid \mu, \boldsymbol{\theta} \sim N_n(\mu \mathbf{1}\ ,\ \Sigma(\boldsymbol{\theta}))\ , \tag{1.1}$$

where N_n denotes the n-dimensional normal distribution, μ is the (constant) mean level, $\mathbf{1}$ is a vector of ones, and $(\Sigma(\boldsymbol{\theta}))_{ii'}$ gives the covariance between $Y(\mathbf{s}_i)$ and $Y(\mathbf{s}_{i'})$. For the variance-covariance specification of the previous paragraph, we have $\boldsymbol{\theta} = (\tau^2, \sigma^2, \phi)^T$, since the covariance matrix depends on the nugget, sill, and range.

In fact, the simplest choices for Σ are those corresponding to *isotropic* covariance functions, where we assume that the spatial correlation is a function solely of the distance $d_{ii'}$ between $\mathbf{s}_i$ and $\mathbf{s}_{i'}$. As mentioned above, exponential forms are particularly intuitive examples. Here,

$$(\Sigma(\boldsymbol{\theta}))_{ii'} = \sigma^2 \exp(-\phi d_{ii'}) + \tau^2 I(i = i'),\ \sigma^2 > 0,\ \phi > 0,\ \tau^2 > 0\ , \tag{1.2}$$

where I denotes the indicator function (i.e., $I(i = i') = 1$ if $i = i'$, and 0 otherwise). Many other choices are possible for $Cov(Y(\mathbf{s}_i), Y(\mathbf{s}_{i'}))$, including for example the powered exponential,

$$(\Sigma(\boldsymbol{\theta}))_{ii'} = \sigma^2 \exp(-\phi d_{ii'}^{\kappa}) + \tau^2 I(i = i'),\ \sigma^2 > 0,\ \phi > 0,\ \tau^2 > 0,\ \kappa \in (0, 2]\,,$$

the spherical, the Gaussian, and the Matérn (see Subsection 2.1.3 for a full discussion). In particular, while the latter requires calculation of a modified Bessel function, Stein (1999a, p. 51) illustrates its ability to capture a broader range of local correlation behavior despite having no more parameters than the powered exponential. We shall say much more about point-level spatial methods and models in Chapters 2, 3 and 6 and also provide illustrations using freely available statistical software.

1.1.2 Areal models

In models for areal data, the geographic regions or *blocks* (zip codes, counties, etc.) are denoted by B_i, and the data are typically sums or averages of variables over these blocks. To introduce spatial association, we define a *neighborhood* structure based on the arrangement of the blocks in the map. Once the neighborhood structure is defined, models resembling autoregressive time series models are considered. Two very popular models that incorporate such neighborhood information are the *simultaneously* and *conditionally autoregressive* models (abbreviated SAR and CAR), originally developed by Whittle (1954) and Besag (1974), respectively. The SAR model is computationally convenient for use with likelihood methods. By contrast, the CAR model is computationally convenient for Gibbs sampling used

in conjunction with Bayesian model fitting, and in this regard is often used to incorporate spatial correlation through a vector of spatially varying random effects $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)^T$. For example, writing $Y_i \equiv Y(B_i)$, we might assume $Y_i \overset{ind}{\sim} N(\phi_i, \sigma^2)$, and then impose the CAR model

$$\phi_i | \boldsymbol{\phi}_{(-i)} \sim N\left(\mu + \sum_{j=1}^{n} a_{ij}(\phi_j - \mu)\ ,\ \tau_i^2\right)\ , \tag{1.3}$$

where $\boldsymbol{\phi}_{(-i)} = \{\phi_j : j \neq i\}$, τ_i^2 is the conditional variance, and the a_{ij} are known or unknown constants such that $a_{ii} = 0$ for $i = 1, \ldots, n$. Letting $A = (a_{ij})$ and $M = Diag(\tau_1^2, \ldots, \tau_n^2)$, by Brook's Lemma (c.f. Section 4.2), we can show that

$$p(\boldsymbol{\phi}) \propto \exp\{-(\boldsymbol{\phi} - \mu\mathbf{1})^T M^{-1}(I - A)(\boldsymbol{\phi} - \mu\mathbf{1})/2\}\ , \tag{1.4}$$

where $\mathbf{1}$ is an n-vector of 1's, and I is a $n \times n$ identity matrix.

A common way to construct A and M is to let $A = \rho\, Diag(1/w_{i+})W$ and $M^{-1} = \tau^{-2} Diag(w_{i+})$. Here ρ is referred to as the *spatial correlation* parameter, and $W = (w_{ij})$ is a neighborhood matrix for the areal units, which can be defined as

$$w_{ij} = \begin{cases} 1 & \text{if subregions } i \text{ and } j \text{ share a common boundary, } i \neq j \\ 0 & \text{otherwise} \end{cases}\ . \tag{1.5}$$

Thus $Diag(w_{i+})$ is a diagonal matrix with (i,i) entry equal to $w_{i+} = \sum_j w_{ij}$. Letting $\boldsymbol{\alpha} \equiv (\rho, \tau^2)$, the covariance matrix of $\boldsymbol{\phi}$ then becomes $C(\boldsymbol{\alpha}) = \tau^2[Diag(w_{i+}) - \rho W]^{-1}$, where the inverse exists for an appropriate range of ρ values; see Subsection 4.3.1.

In the context of Bayesian hierarchical areal modeling, when choosing a prior distribution $\pi(\boldsymbol{\phi})$ for a vector of spatial random effects $\boldsymbol{\phi}$, the CAR distribution (1.3) is often used with the 0–1 *weight* (or *adjacency*) *matrix* W in (1.5) and $\rho = 1$. While this results in an *improper* (nonintegrable) prior distribution, this problem is remedied by imposing a sum-to-zero constraint on the ϕ_i (which turns out to be easy to implement numerically using Gibbs sampling). In this case the more general conditional form (1.3) is replaced by

$$\phi_i \mid \boldsymbol{\phi}_{(-i)} \sim N(\bar{\phi}_i\ ,\ \tau^2/m_i)\ , \tag{1.6}$$

where $\bar{\phi}_i$ is the average of the $\phi_{j \neq i}$ that are adjacent to ϕ_i, and m_i is the number of these adjacencies (see, e.g., Besag, York, and Mollié, 1991). We discuss areal models in greater detail in Chapters 4 and 6.

1.1.3 Point process models

In the point process model, the spatial domain D is itself random, so that the elements of the index set D are the locations of random events that constitute the spatial point pattern. $Y(\mathbf{s})$ then normally equals the constant 1 for all $\mathbf{s} \in D$ (indicating occurrence of the event), but it may also provide additional covariate information, in which case the data constitute a marked point process.

Questions of interest with data of this sort typically center on whether the data are *clustered* more or less than would be expected if the locations were determined completely by chance. Stochastically, such uniformity is often described through a *homogeneous Poisson process*, which implies that the expected number of occurrences in region A is $\lambda|A|$, where λ is the *intensity* parameter of the process and $|A|$ is the area of A. To investigate this in practice, plots of the data are typically a good place to start, but the tendency of the human eye to see clustering or other structure in virtually every point pattern renders a strictly graphical approach unreliable. Instead, statistics that measure clustering, and

perhaps even associated significance tests, are often used. The most common of these is *Ripley's K function*, given by

$$K(d) = \frac{1}{\lambda} E[\text{number of points within } d \text{ of an arbitrary point}], \tag{1.7}$$

where again λ is the intensity of the process, i.e., the mean number of points per unit area.

The theoretical value of K is known for certain spatial point process models. For instance, for point processes that have no spatial dependence at all, we would have $K(d) = \pi d^2$, since in this case the number of points within d of an arbitrary point should be proportional to the area of a circle of radius d; the K function then divides out the average intensity λ. However, if the data are clustered we might expect $K(d) > \pi d^2$, while if the points follow some regularly spaced pattern we would expect $K(d) < \pi d^2$. This suggests a potential inferential use for K; namely, comparing an estimate of it from a data set to some theoretical quantities, which in turn suggests whether clustering is present, and if so, which model might be most plausible. The usual estimator for K is given by

$$\widehat{K}(d) = n^{-2}|A| \sum \sum_{i \neq j} p_{ij}^{-1} I_d(d_{ij}) \,, \tag{1.8}$$

where n is the number of points in A, d_{ij} is the distance between points i and j, p_{ij} is the proportion of the circle with center i and passing through j that lies within A, and $I_d(d_{ij})$ equals 1 if $d_{ij} < d$, and 0 otherwise.

We provide an extensive account for point processes in Chapter 8. Other useful texts focusing primarily upon point processes and patterns include Diggle (2003), Lawson and Denison (2002), and Møller and Waagepetersen (2004) for treatments of spatial point processes and related methods in spatial cluster detection and modeling.

1.1.4 Software and datasets

This text extensively uses the `R` (`www.r-project.org`) software programming language and environment for statistical computing and graphics. `R` is released under the GNU open-source license and can be downloaded for free from the Comprehensive R Archive Network (CRAN), which can be accessed from `http://cran.us.r-project.org/`. The capabilities of `R` are easily extended through "libraries" or "packages" that perform more specialized tasks. These packages are also available from CRAN and can be downloaded and installed from within the `R` software environment.

There are a variety of spatial packages in `R` that perform modeling and analysis for the different types of spatial data. For example, the `gstat` and `geoR` packages provide functions to perform traditional (classical) analysis for point-level data; the latter also offers simpler Bayesian models. The packages `spBayes` and `sptimer` have much more elaborate Bayesian functions, the latter focusing primarily upon space-time data. We will provide illustrations using some of these `R` packages in Chapters 2 and 6.

The `spdep` package in `R` provides several functions for analyzing areal-level data, including basic descriptive statistics for areal data as well as fitting areal models using classical likelihood methods. For Bayesian analysis, the `BUGS` language and the `WinBUGS` software is still perhaps the most widely used engine to fit areal models. We will discuss areal models in greater detail in Chapters 4 and 6.

Turning to point-process models, a popular spatial `R` package, `spatstat`, allows computation of K for any data set, as well as the approximate 95% intervals for it so the significance of departure from some theoretical model may be judged. However, full inference likely requires use of the `R` package `Splancs`, or perhaps a fully Bayesian approach with user-specific coding (also see Wakefield and Morris, 2001). We provide some examples of `R` packages for point-process models in Chapter 8.

We will use a number of spatial and spatiotemporal datasets for illustrating the modeling and software implementation. While some of these datasets are included in the `R` packages we will be using, others are available from `www.biostat.umn.edu/~brad/data2.html`. We remark that the number of `R` packages performing spatial analysis is already too large to be discussed in this text. We refer the reader to the CRAN Task View `http://cran.r-project.org/web/views/Spatial.html` for an exhaustive list of such packages and brief descriptions regarding their capabilities.

1.2 Fundamentals of cartography

In this section we provide a brief introduction to how geographers and spatial statisticians understand the geometry of (and determine distances on) the surface of the earth. This requires a bit of thought regarding cartography (mapmaking), especially map projections, and the meaning of latitude and longitude, which are often understood informally (but incorrectly) as being equivalent to Cartesian x and y coordinates.

1.2.1 Map projections

A map projection is a systematic representation of all or part of the surface of the earth on a plane. This typically comprises lines delineating meridians (longitudes) and parallels (latitudes), as required by some definitions of the projection. A well-known fact from topology is that it is impossible to prepare a distortion-free flat map of a surface curving in all directions. Thus, the cartographer must choose the characteristic (or characteristics) that are to be shown accurately in the map. In fact, it cannot be said that there is a "best" projection for mapping. The purpose of the projection and the application at hand lead to projections that are appropriate. Even for a single application, there may be several appropriate projections, and choosing the "best" projection can be subjective. Indeed there are an infinite number of projections that can be devised, and several hundred have been published.

Since the sphere cannot be flattened onto a plane without distortion, the general strategy for map projections is to use an intermediate surface that can be flattened. This intermediate surface is called a *developable surface* and the sphere is first projected onto the this surface, which is then laid out as a plane. The three most commonly used surfaces are the cylinder, the cone and the plane itself. Using different orientations of these surfaces leads to different classes of map projections. Some examples are given in Figure 1.4. The points on the globe are projected onto the wrapping (or tangential) surface, which is then laid out to form the map. These projections may be performed in several ways, giving rise to different projections.

Before the availability of computers, the above orientations were used by cartographers in the physical construction of maps. With computational advances and digitizing of cartography, analytical formulae for projections were desired. Here we briefly outline the underlying theory for equal-area and conformal (locally shape-preserving) maps. A much more detailed and rigorous treatment may be found in Pearson (1990).

The basic idea behind deriving equations for map projections is to consider a sphere with the geographical coordinate system (λ, ϕ) for longitude and latitude and to construct an appropriate (rectangular or polar) coordinate system (x, y) so that

$$x = f(\lambda, \phi),\ y = g(\lambda, \phi)\ ,$$

where f and g are appropriate functions to be determined, based upon the properties we want our map to possess. We will study map projections using differential geometry concepts, looking at infinitesimal patches on the sphere (so that curvature may be neglected

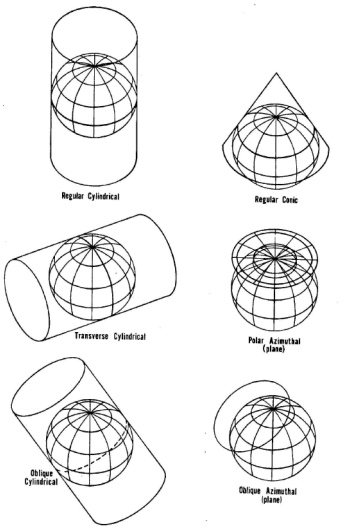


Figure 1.4 *The geometric constructions of projections using developable surfaces (figure courtesy of the U.S. Geological Survey).*

and the patches are closely approximated by planes) and deriving a set of (partial) differential equations whose solution will yield f and g. Suitable initial conditions are set to create projections with desired geometric properties.

Thus, consider a small patch on the sphere formed by the infinitesimal quadrilateral, $ABCD$, given by the vertices,

$$A = (\lambda, \phi),\ B = (\lambda, \phi + d\phi),\ C = (\lambda + d\lambda, \phi),\ D = (\lambda + d\lambda, \phi + d\phi).$$

So, with R being the radius of the earth, the horizontal differential component along an arc of latitude is given by $|AC| = (R\cos\phi)d\lambda$ and the vertical component along a great circle of longitude is given by $|AB| = Rd\phi$. Note that since AC and AB are arcs along the latitude and longitude of the globe, they intersect each other at right angles. Therefore, the area of the patch $ABCD$ is given by $|AC||AB|$. Let $A'B'C'D'$ be the (infinitesimal) image of the patch $ABCD$ on the map. Then, we see that

$$\begin{gathered} A' = (f(\lambda, \phi), g(\lambda, \phi)), \\ C' = (f(\lambda + d\lambda, \phi), g(\lambda + d\lambda, \phi)), \\ B' = (f(\lambda, \phi + d\phi), g(\lambda, \phi + d\phi)), \\ \text{and } D' = (f(\lambda + d\lambda, \phi + d\phi), g(\lambda + d\lambda, \phi + d\phi))\ . \end{gathered}$$

This in turn implies that

$$\overrightarrow{A'C'} = \left(\frac{\partial f}{\partial \lambda}, \frac{\partial g}{\partial \lambda}\right) d\lambda \text{ and } \overrightarrow{A'B'} = \left(\frac{\partial f}{\partial \phi}, \frac{\partial g}{\partial \phi}\right) d\phi\,.$$

If we desire an equal-area projection we need to equate the area of the patches $ABCD$ and $A'B'C'D'$. But note that the area of $A'B'C'D'$ is given by the area of parallelogram formed by vectors $\overrightarrow{A'C'}$ and $\overrightarrow{A'B'}$. Treating them as vectors in the xy plane of an xyz system, we see that the area of $A'B'C'D'$ is the cross-product,

$$(\overrightarrow{A'C'}, 0) \times (\overrightarrow{A'B'}, 0) = \left(\frac{\partial f}{\partial \lambda}\frac{\partial g}{\partial \phi} - \frac{\partial f}{\partial \phi}\frac{\partial g}{\partial \lambda}\right) d\lambda d\phi\,.$$

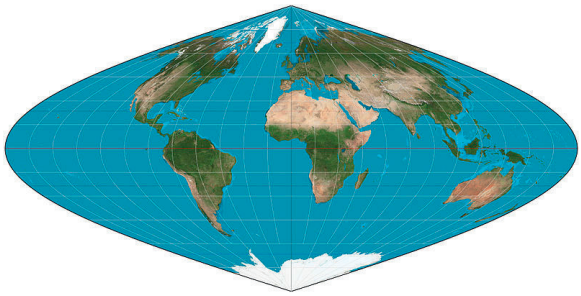


Figure 1.5 *The sinusoidal projection.*

Therefore, we equate the above to $|AC||AB|$, leading to the following partial differential equation in f and g:

$$\left(\frac{\partial f}{\partial \lambda}\frac{\partial g}{\partial \phi} - \frac{\partial f}{\partial \phi}\frac{\partial g}{\partial \lambda}\right) = R^2 \cos\phi\,.$$

Note that this is the equation that must be satisfied by any equal-area projection. It is an underdetermined system, and further conditions need to be imposed (that ensure other specific properties of the projection) to arrive at f and g.

Example 1.1 Equal-area maps are used for statistical displays of areal-referenced data. An easily derived equal-area projection is the sinusoidal projection, shown in Figure 1.5. This is obtained by specifying $\partial g/\partial \phi = R$, which yields equally spaced straight lines for the parallels, and results in the following analytical expressions for f and g (with the 0 degree meridian as the central meridian):

$$f(\lambda, \phi) = R\lambda\cos\phi; g(\lambda, \phi) = R\phi.$$

Another popular equal-area projection (with equally spaced straight lines for the meridians) is the Lambert cylindrical projection given by

$$f(\lambda, \phi) = R\lambda;\ g(\lambda, \phi) = R\sin\phi\,.$$

For conformal (angle-preserving) projections we set the angle $\angle(AC, AB)$ equal to $\angle(A'C', A'B')$. Since $\angle(AC, AB) = \pi/2$, $\cos(\angle(AC, AB)) = 0$, leading to

$$\frac{\partial f}{\partial \lambda}\frac{\partial f}{\partial \phi} + \frac{\partial g}{\partial \lambda}\frac{\partial g}{\partial \phi} = 0$$

or, equivalently, the Cauchy-Riemann equations of complex analysis,

$$\left(\frac{\partial f}{\partial \lambda} + i\frac{\partial g}{\partial \lambda}\right)\left(\frac{\partial f}{\partial \phi} - i\frac{\partial g}{\partial \phi}\right) = 0\,.$$

A sufficient partial differential equation system for conformal mappings of the Cauchy-Riemman equations that is simpler to use is

$$\frac{\partial f}{\partial \lambda} = \frac{\partial g}{\partial \phi}\cos\phi;\ \frac{\partial g}{\partial \lambda} = \frac{\partial f}{\partial \phi}\cos\phi\,.$$

Example 1.2 The Mercator projection shown in Figure 1.6 is a classical example of a conformal projection. It has the interesting property that rhumb lines (curves that intersect the meridians at a constant angle) are shown as straight lines on the map. This is particularly useful for navigation purposes. The Mercator projection is derived by letting $\partial g/\partial \phi =$

Figure 1.6 *The Mercator projection.*

$R \sec \phi$. After suitable integration, this leads to the analytical equations (with the 0 degree meridian as the central meridian),

$$f(\lambda, \phi) = R\lambda;\ g(\lambda, \phi) = R \ln \tan \left(\frac{\pi}{4} + \frac{\phi}{2} \right) .$$

As is seen above, even the simplest map projections lead to complex transcendental equations relating latitude and longitude to positions of points on a given map. Therefore, rectangular grids have been developed for use by surveyors. In this way, each point may be designated merely by its distance from two perpendicular axes on a flat map. The y-axis usually coincides with a chosen central meridian, y increasing north, and the x-axis is perpendicular to the y-axis at a latitude of origin on the central meridian, with x increasing east. Frequently, the x and y coordinates are called "eastings" and "northings," respectively, and to avoid negative coordinates, may have "false eastings" and "false northings" added to them. The grid lines usually do not coincide with any meridians and parallels except for the central meridian and the equator.

One such popular grid, adopted by The National Imagery and Mapping Agency (NIMA) (formerly known as the Defense Mapping Agency), and used especially for military use throughout the world, is the Universal Transverse Mercator (UTM) grid; see Figure 1.7. The UTM divides the world into 60 north-south zones, each of width six degrees longitude. Starting with Zone 1 (between 180 degrees and 174 degrees west longitude), these are numbered consecutively as they progress eastward to Zone 60, between 174 degrees and 180 degrees east longitude. Within each zone, coordinates are measured north and east in meters, with northing values being measured continuously from zero at the equator, in a northerly direction. Negative numbers for locations south of the equator are avoided by assigning an arbitrary false northing value of 10,000,000 meters (as done by NIMA's cartographers). A central meridian cutting through the center of each 6 degree zone is assigned an easting value of 500,000 meters, so that values to the west of the central meridian are less than 500,000 while those to the east are greater than 500,000. In particular, the conterminous 48 states of the United States are covered by 10 zones, from Zone 10 on the west coast through Zone 19 in New England.

In practice, the UTM is used by overlaying a transparent grid on the map, allowing distances to be measured in meters at the map scale between any map point and the nearest grid lines to the south and west. The northing of the point is calculated as the sum

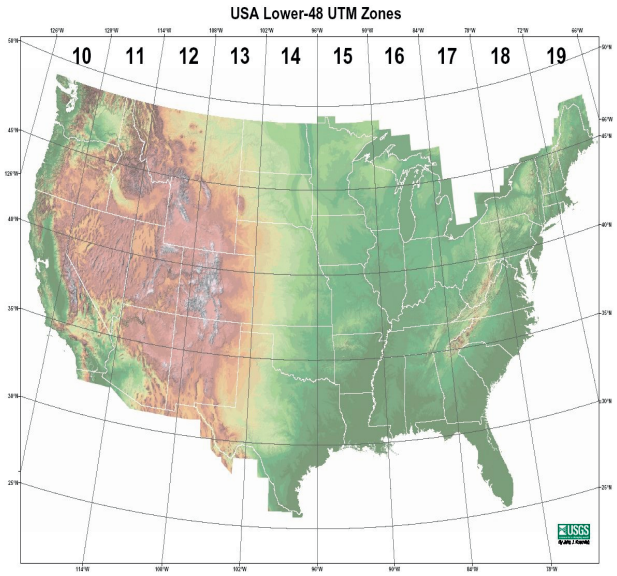


Figure 1.7 *Example of a UTM grid over the United States (figure courtesy of the U.S. Geological Survey).*

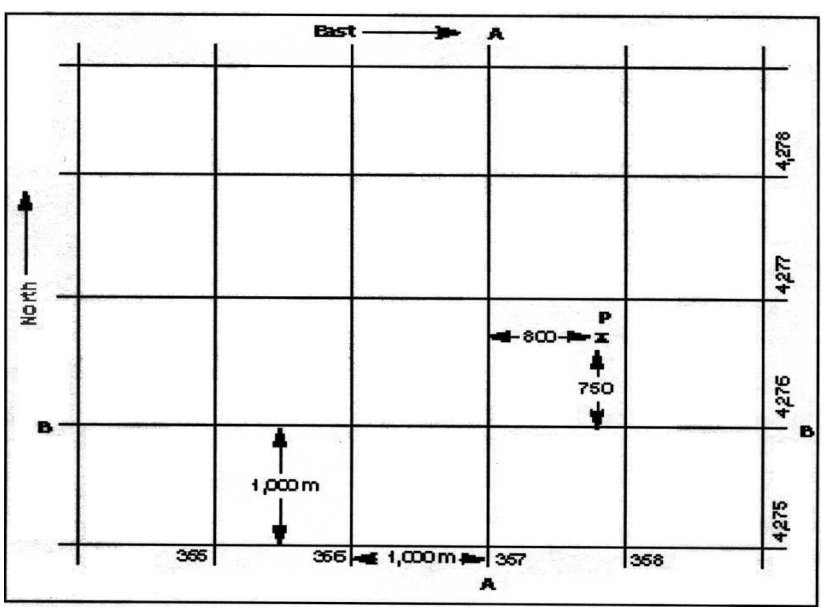


Figure 1.8 *Finding the easting and northing of a point in a UTM projection (figure courtesy of the U.S. Geological Survey).*

of the value of the nearest grid line south of it and its distance north of that line. Similarly, its easting is the value of the nearest grid line west of it added to its distance east of that line. For instance, in Figure 1.8, the grid value of line A-A is 357,000 meters east, while that of line B-B is 4,276,000 meters north. Point P is 800 meters east and 750 meters north of the grid lines resulting in the grid coordinates of point P as north 4,276,750 and east 357,800.

Finally, since spatial modeling of point-level data often requires computing distances between points on the earth's surface, one might wonder about a *planar* map projection, which would preserve distances between points. Unfortunately, the existence of such a map is precluded by Gauss' Theorema Eggregium in differential geometry (see, e.g., Guggenheimer, 1977, pp. 240–242). Thus, while we have seen projections that preserve area and shapes, distances are always distorted. The *gnomonic* projection (Snyder, 1987, pp. 164–168) gives the correct distance from a single reference point, but is less useful for the practicing spatial analyst who needs to obtain complete intersite distance matrices (since this would require not one but many such maps). Banerjee (2005) explores different strategies for computing distances on the earth and their impact on statistical inference. We present a brief summary below.

1.2.2 Calculating distances on the earth's surface

Distance computations are indispensable in spatial analysis. Precise inter-site distance computations are used in variogram analysis to assess the strength of spatial association. They help in setting starting values for the non-linear least squares algorithms in classical analysis (more in Chapter 2) and in specifying priors on the range parameter in Bayesian modeling (more in Chapter 5), making them crucial for correct interpretation of spatial range and the convergence of statistical algorithms. For data sets covering relatively small spatial domains, ordinary Euclidean distance offers an adequate approximation. However, for larger domains (say, the entire continental U.S.), the curvature of the earth causes distortions because of the difference in differentials in longitude and latitude (a unit increment in degree longitude is not the same length as a unit increment in degree latitude except at the equator).

Suppose we have two points on the surface of the earth, $P_1 = (\theta_1, \lambda_1)$ and $P_2 = (\theta_2, \lambda_2)$. We assume both points are represented in terms of latitude and longitude. That is, let θ_1 and λ_1 be the latitude and longitude, respectively, of the point P_1, while θ_2 and λ_2 are those for the point P_2. The main problem is to find the shortest distance (*geodesic*) between the points. The solution is obtained via the following formulae:

$$D = R\phi$$

where R is the radius of the earth and ϕ is an angle (measured in *radians*) satisfying

$$\cos\phi = \sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_2 \cos(\lambda_1 - \lambda_2) \ . \tag{1.9}$$

These formulae are derived as follows. The geodesic is actually the arc of the great circle joining the two points. Thus, the distance will be the length of the arc of a *great circle* (i.e., a circle with radius equal to the radius of the earth). Recall that the length of the arc of a circle equals the angle subtended by the arc at the center multiplied by the radius of the circle. Therefore it suffices to find the angle subtended by the arc; denote this angle by ϕ.

Let us form a three-dimensional Cartesian coordinate system (x, y, z), with the origin at the center of the earth, the z-axis along the North and South Poles, and the x-axis on the plane of the equator joining the center of the earth and the Greenwich meridian. Using the left panel of Figure 1.9 as a guide, elementary trigonometry provides the following relationships between (x, y, z) and the latitude-longitude (θ, λ):

$$\begin{aligned} x &= R\cos\theta\cos\lambda, \\ y &= R\cos\theta\sin\lambda, \\ \text{and } z &= R\sin\theta \ . \end{aligned}$$

Now form the vectors $\mathbf{u}_1 = (x_1, y_1, z_1)$ and $\mathbf{u}_2 = (x_2, y_2, z_2)$ as the Cartesian coordinates corresponding to points P_1 and P_2. Hence ϕ is the angle between $\mathbf{u}_1$ and $\mathbf{u}_2$. From

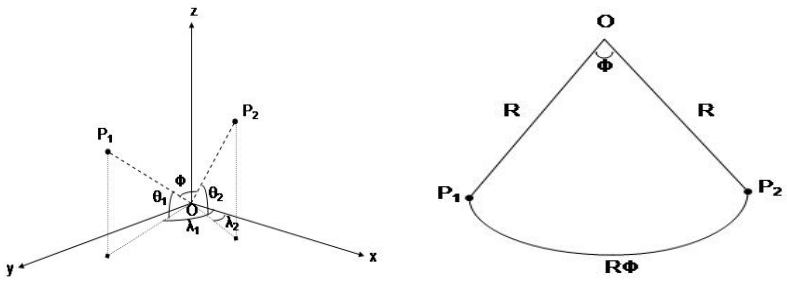


Figure 1.9 *Diagrams illustrating the geometry underlying the calculation of great circle (geodesic) distance.*

standard analytic geometry, the easiest way to find this angle is therefore to use the following relationship between the cosine of this angle and the dot product of $\mathbf{u}_1$ and $\mathbf{u}_2$:

$$\cos\phi = \frac{\langle \mathbf{u}_1, \mathbf{u}_2 \rangle}{||\mathbf{u}_1||\,||\mathbf{u}_2||} \ .$$

We then compute $\langle \mathbf{u}_1, \mathbf{u}_2 \rangle$ as

$$\begin{aligned} &R^2\left[\cos\theta_1\cos\lambda_1\cos\theta_2\cos\lambda_2 + \cos\theta_1\sin\lambda_1\cos\theta_2\sin\lambda_2 + \sin\theta_1\sin\theta_2\right] \\ &\quad = R^2\left[\cos\theta_1\cos\theta_2\cos\left(\lambda_1-\lambda_2\right) + \sin\theta_1\sin\theta_2\right] \ . \end{aligned}$$

But $||\mathbf{u}_1|| = ||\mathbf{u}_2|| = R$, so the result in (1.9) follows. Looking at the right panel of Figure 1.9, our final answer is thus

$$D = R\phi = R\arccos[\sin\theta_1\sin\theta_2 + \cos\theta_1\cos\theta_2\cos\left(\lambda_1-\lambda_2\right)] \ . \tag{1.10}$$

While calculating (1.10) is straightforward, Euclidean metrics are popular due to their simplicity and easier interpretability. More crucially, statistical modeling of spatial correlations proceed from *correlation functions* that are often valid only with Euclidean metrics. For example, using (1.10) to calculate the distances in general covariance functions may not result in a positive definite $\Sigma(\boldsymbol{\theta})$ in (1.1). We consider a few different approaches for computing distances on the earth using Euclidean metrics, classifying them as those arising from the classical spherical coordinates, and those arising from planar projections.

Equation (1.10) clearly reveals that the relationship between the Euclidean distances and the geodetic distances is not just a matter of scaling. We cannot mutiply one by a constant number to obtain the other. A simple scaling of the geographical coordinates results in a "naive Euclidean" metric obtained directly in degree units, and converted to kilometer units as: $\|P_1 - P_2\|\pi R/180$. This metric performs well on small domains but always overestimates the geodetic distance, *flattening out* the meridians and parallels, and stretching the curved domain onto a plane, thereby stretching distances as well. As the domain increases, the estimation deteriorates.

Banerjee (2005) also explores a more natural metric, which is along the "chord" joining the two points. This is simply the Euclidean metric $||\mathbf{u}_2 - \mathbf{u}_1||$, yielding a "burrowed through the earth" distance — the chordal length between P_1 and P_2. The slight underestimation of the geodetic distance is expected, since the chord "penetrates" the domain, producing a straight line approximation to the geodetic arc.

The first three rows of Table 1.1 compare the geodetic distance with the "naive Eucidean" and chordal metrics. The next three rows show distances computed by using three planar projections: the Mercator, the sinusoidal and a centroid-based data projection, which is developed in Exercise 10. The first column corresponds to the distance between the farthest

Methods	Colorado	Chicago-Minneapolis	New York-New Orleans
geodetic	741.7	562.0	1897.2
naive Euclidean	933.8	706.0	2172.4
chord	741.3	561.8	1890.2
Mercator	951.8	773.7	2336.5
sinusoidal	742.7	562.1	1897.7
centroid-based	738.7	562.2	1901.5

Table 1.1 *Comparison of different methods of computing distances (in kms). For Colorado, the distance reported is the maximum inter-site distance for a set of* 50 *locations.*

points in a spatially referenced data set comprising 50 locations in Colorado (we will revisit this dataset later in Chapter 11), while the next two present results for two differently spaced pairs of cities. The overestimation and underestimation of the "naive Euclidean" and "chordal" metrics respectively is clear, although the chordal metric excels even for distances over 2000 kms (New York and New Orleans). We find that the sinusoidal and centroid-based projections seem to be distorting distances much less than the Mercator, which performs even worse than the naive Euclidean.

This approximation of the chordal metric has an important theoretical implication for the spatial modeler. A troublesome aspect of geodetic distances is that they are *not* necessarily valid arguments for correlation functions defined on Euclidean spaces (see Chapter 2 for more general forms of correlation functions). However, the excellent approximation of the chordal metric (which is Euclidean) ensures that in most practical settings valid correlation functions in $\Re^3$ such as the Matérn and exponential yield positive definite correlation matrices with geodetic distances and enable proper convergence of the statistical estimation algorithms.

Schoenberg (1942) develops a necessary and sufficient representation for valid positive-definite functions on spheres in terms of normalized Legendre polynomials P_k of the form:

$$\psi(t) = \sum_{k=0}^{\infty} a_k P_k(\cos t),$$

where a_k's are positive constants such that $\sum_{k=0}^{\infty} a_k$ converges. An example is given by

$$\psi(t) = \frac{1}{\sqrt{1+\alpha^2-2\alpha\cos t}}, \quad \alpha \in (0,1),$$

which can be easily shown to have the Legendre polynomial expansion $\sum_{k=0}^{\infty} \alpha^k P_k(\cos t)$. The chordal metric also provides a simpler way to construct valid correlation functions over the sphere using a sinusoidal composition of any valid correlation function on Euclidean space. To see this, consider a unit sphere ($R = 1$) and note that

$$\|\mathbf{u}_1 - \mathbf{u}_2\| = \sqrt{2 - 2\langle \mathbf{u}_1, \mathbf{u}_2 \rangle} = 2\sin(\phi/2).$$

Therefore, a correlation function $\rho(d)$ (suppressing the range and smoothness parameters) on the Euclidean space transforms to $\rho(2\sin(\phi/2))$ on the sphere, thereby *inducing* a valid correlation function on the sphere. This has some advantages over the Legendre polynomial approach of Schoenberg: (1) we retain the interpretation of the smoothness and decay parameters, (2) it is simpler to construct and compute, and (3) it builds upon a rich legacy of investigations (both theoretical and practical) of correlation functions on Euclidean spaces (again, see Chapter 2 for different correlation functions).

1.3 Maps and geodesics in R

The R statistical software environment today offers excellent interfaces with Geographical Information Systems (GIS) through a number of libraries (also known as packages). At the core of R's GIS capabilities is the maps library originally described by Becker and Wilks (1993). This maps library contains the geographic boundary files for several maps, including county boundaries for every state in the U.S. For example, creating a map of the state of Minnesota with its county boundaries is as simple as the following line of code:

```
> library(maps)
> mn.map <- map(database="county", region="minnesota")
```

If we do not want the county boundaries, we simply write

```
> mn.map <- map("state", "minnesota"),
```

which produces a map of Minnesota with only the state boundary. The above code uses the boundaries from R's own maps database. However, other important regional boundary types (say, zip codes) and features (rivers, major roads, and railroads) are generally not available, although topographic features and an enhanced GIS interface is available through the library RgoogleMaps . While in some respects R is perhaps not nearly as versatile as ArcView or other purely GIS packages, it does offer a rare combination of GIS and statistical analysis capabilities.

It is possible to import shapefiles from other GIS software (e.g. ArcView) into R using the maptools package. We invoke the readShapePoly function in the maptools package to read the external shapefile and store the output in minnesota.shp. To produce the map, we apply plot to this output.

```
> library(maptools)
> minnesota.shp <- readShapePoly("minnesota.shp",
+                     proj4string=CRS("+proj=longlat"))
> plot(minnesota.shp).
```

For the above to work, you will need three files with extensions ".shp", ".shx" and ".dbf". They must have the same name and differ only in the extension. The "minnesota.shp" file contains the geometry data, the "minnesota.shx" file contains the spatial index, and the "minnesota.dbf" file contains the attribute data. These are read using the readShapePoly() function to produce a spatial polygon object.

The above is an example of how to draw bare maps of a state within the USA using either R's own database or an external shapefile. We can also draw maps of other countries using the mapdata package, which has some world map data, in conjunction with maps. For example, to draw a map of Canada, we write

```
> library(maps)
> library(mapdata)
> map("worldHires","Canada",
+     xlim=c(-141,-53), ylim=c(40,85),
+     col="gray90", fill=TRUE)
```

We leave the reader to experiment further with these examples.

In practice, we are not interested in bare maps but would want to plot spatially referenced data on the map. Let us return to the counties in Minnesota. Consider a new file newdata.csv that includes information on the population of each county of Minnesota along with the number of influenza A (H1N1) cases from each county. We first merge our new dataset with the minnesota.shp object already created using the county names.

```
> newdata <- read.csv("newdata.csv")
> minnesota.shp@data <- merge(minnesota.shp@data, newdata,
+                             by="NAME").
```

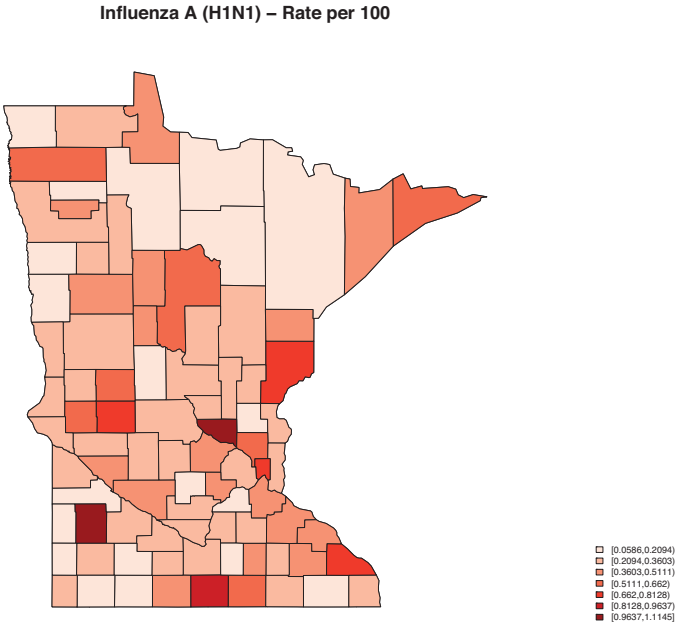


Figure 1.10 *Map showing Influenza A (H1N1) rates (cases/population)×100 in different counties of Minnesota for 1999.*

To plot the data in a visually appealing way, we use two additional packages: `RColorBrewer`, which creates nice color schemes, and `classInt`, which facilitates classifying the data into classes which will correspond to the plotting colors. Below, we present the code to plot H1N1 influenza rates per 100 after classifying the data into equal intervals.

```
> library(RColorBrewer)
> library(classInt)
> CASES <- minnesota.shp@data$cases
> POP1999 <- minnesota.shp@data$POP1999
> var <- (CASES/POP1999)*100
> nclr <- 7
> plotclr <- brewer.pal(nclr,"Reds")
> class <- classIntervals(var, nclr, style="equal", dataPrecision=4)
> colcode <- findColours(class, plotclr)
> plot(minnesota.shp)
> plot(minnesota.shp, col=colcode, add=T)
> title(main="Influenza A (H1N1) - Rate per 100")
> legend("bottomright", legend=names(attr(colcode, "table")),
+    fill=attr(colcode, "palette"), cex=0.6, bty="n").
```

The resulting map is shown in Figure 1.10 and is called a *choropleth map*.

Map projections in R can be performed using the `mapproj` package. For example, we can create a variety of cartographic map projections of any part of the world using `xlim` and `ylim` to specify the bounds (using longitudes and latitudes) of the plotting region and use `projection` to specify the type of projection we desire. In fact, the `maps` package uses `mapproj` for its map projections. Therefore, loading `maps` automatically invokes `mapproj`.

```
> library(maps) ## automatically loads mapproj
> sinusoidal.proj = map(database= "world", ylim=c(45,90), xlim=c(-160,-50),
+     col="grey80", fill=TRUE, plot=FALSE, projection="sinusoidal")
```

```
> map(sinusoidal.proj)
```

produces a Sinusoidal map projection (Example 1.1) between latitudes 45 and 90 degrees and longitudes −160 and −150 degrees. (Run the above code and test your geography to see what part of the world this is!) Repeating the above with `projection` set to `"mercator"` will result in a Mercator projection (Example 1.2). For higher resolution maps, one can load the `mapdata` package and use `"worldHires"` instead of `"world"` in the database.

For distance computations using a map projection, as was done for the Mercator and sinusoidal in Table 1.1, it is convenient to use the `mapproject` function in package `mapproj`. This simply converts latitude and longitude to rectangular coordinates. Suppose `LON` and `LAT` are two vectors containing longitudes and latitudes on the earth's surface to be projected. Then, a simple command such as

```
> xy.sinusoidal <- mapproject(LON, LAT, projection="sinusoidal")
```

produces the projected coordinates in Euclidean space. The coordinates are accessed by `xy.sinusoidal$x` and `xy.sinusoidal$y`, respectively. For distance computations that will approximate the great circle distance, these coordinates need to be multiplied by the radius of the earth.

While `mapproject` does not offer the UTM projections, the `rgdal` package can be used to construct UTM projections. This is particularly useful when plotting point-level data through the `RgoogleMaps` package, another exciting GIS interface offered by `R`. It is especially useful for plotting points such as GPS (Global Positioning System) locations on maps. For example, the Colorado data used in Table 1.1 is the file `ColoradoS-T.dat` and can be downloaded from `www.biostat.umn.edu/~brad/data2.html`. We can read the file in `R` as

```
> coloradoST <- read.table("ColoradoS-T.dat", header = TRUE).
```

Next, we use the `GetMap.bbox` function in the `RgoogleMaps` package to set the region within which our coordinates will be plotted. A ‘‘`maptype`’’ argument provides some options for the type of background map we want. The region can also be plotted using the `PlotOnStaticMap` function. This is sometimes useful to obtain an idea whether the underlying map will be useful with reference to our plotting coordinates.

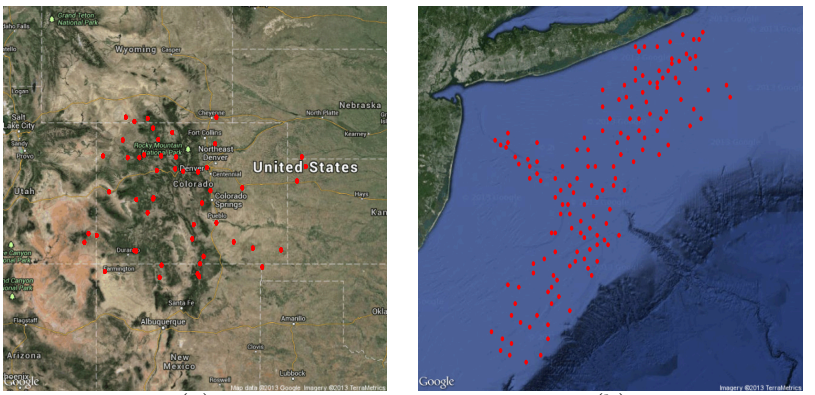
```
> library(RgoogleMaps)
> MyMap <- GetMap.bbox(lonR = range(coloradoST$Longitude),
+                      latR = range(coloradoST$Latitude),
+                      size=c(640,640), maptype = "hybrid")
> PlotOnStaticMap(MyMap).
```

We now convert the longitude and latitude to the same coordinate system as in the `MyMap` object we created above.

```
> convert_points <- LatLon2XY.centered(MyMap,
+                       coloradoST$Latitude,
+                       coloradoST$Longitude)
> points(convert_points$newX, convert_points$newY,
+        col = 'red', pch=19).
```

Finally, we convert our points to UTM coordinates using the `rgdal` package. The code below performs the conversion from latitude-longitude to UTM coordinates and then plots these coordinates on the map created above. This will be achieved in two steps. First, we convert out longitudes and latitudes in the Colorado dataset into a special data type known as `SpatialPoints`. We store this in `SP_longlat`, which is converted by `spTransform` to UTM coordinates. The `zone` parameter specifies the zone of the UTM needs to be supplied explicitly.

```
> library(sp)
> library(rgdal)
```



(a) (b)

Figure 1.11 *(a) A plot of the* 50 *locations in Colorado; (b) a plot of locations in the New York/New Jersey Bight reporting scallop catches.*

```
> SP_longlat <- SpatialPoints(coords =
+                  cbind(coloradoST$Longitude,
+                        coloradoST$Latitude),
+            proj4string = CRS("+proj=longlat +ellps=WGS84"))
> SP_utm <- spTransform(SP_longlat,
+               CRS("+proj=utm +zone=13 +datum=WGS84"))
> plot(SP_utm).
```

We repeat the above exercise with another well known spatial data set involving the catches of scallops in the New York/New Jersey Bight. This data set also has coordinates in terms of latitude and longitude and can be downloaded from `www.biostat.umn.edu/~brad/data/scallops.txt`. The resulting plots are presented in Figure 1.11. The map features, including the background, can be altered by changing the parameter ‘‘`maptype`" in the function `GetMap.bbox`. The options are ‘‘`roadmap`’’, ‘‘`mobile`’’, ‘‘`satellite`’’, ‘‘`terrain`" and ‘‘`hybrid`".

Finally, we mention the `fields` package in `R`, which offers several useful functions for spatial analysis. In particular, it includes two functions `rdist` and `rdist.earth` that conveniently compute inter-site distances. Let `X1` and `X2` be two matrices representing two different sets of locations. Then,

```
> library(fields)
> euclidean.dist = rdist(X1, X2)
> spherical.dist = rdist.earth(X1,X2)
```

computes the inter-site distance matrices between the locations in `X1` and `X2`. The function `rdist` uses the Euclidean distance, while `rdist.earth` uses the spherical or geodetic distance. The latter should be used only when `X1` and `X2` contain latitude-longitude coordinates.

1.4 Exercises

1. What sorts of areal unit variables can you envision that could be viewed as arising from point-referenced variables? What sorts of areal unit variables can you envision whose mean could be viewed as arising from a point-referenced surface? What sorts of areal unit variables fit neither of these scenarios?

2. What sorts of sensible properties should characterize association between point-referenced measurements? What sorts of sensible properties should characterize association between areal unit measurements?
3. Suggest some regional-level covariates that might help explain the spatial pattern evident in Figure 1.2. (*Hint:* The roughly rectangular group of regions located on the map's eastern side is the city of Minneapolis, MN.)
4. (a) Suppose you recorded elevation and average daily temperature on a particular day for a sample of locations in a region. If you were given the elevation at a new location, how would you make a plausible estimate of the average daily temperature for that location?
   (b) Why might you expect spatial association between selling prices of single-family homes in this region to be weaker than that between the observed temperature measurements?
5. For what sorts of point-referenced spatial data would you expect measurements across time to be essentially independent? For what sorts of point-referenced data would you expect measurements across time to be strongly dependent?
6. For point-referenced data, suppose the means of the variables are spatially associated. Would you expect the association between the variables themselves to be weaker than, stronger than, or the same as the association between the means?
7. (a) Write your own `R` function that will compute the distance between two points P_1 and P_2 on the surface of the earth. The function should take the latitude and longitude of the P_i as input, and output the geodesic distance D given in (1.10). Use $R = 6371$ km.
   (b) Use your program to obtain the geodesic distance between Chicago (87.63W, 41.88N) and Minneapolis (93.22W, 44.89N), and between New York (73.97W, 40.78N) and New Orleans (90.25W, 29.98N).
8. A "naive Euclidean" distance may be computed between two points by simply applying the Euclidean distance formula to the longitude-latitude coordinates, and then multiplying by $(R\pi/180)$ to convert to kilometers. Find the naive Euclidean distance between Chicago and Minneapolis, and between New York and New Orleans, comparing your results to the geodesic ones in the previous problem.
9. The *chordal* ("burrowing through the earth") distance separating two points is given by the Euclidean distance applied to the Cartesian spherical coordinate system given in Subsection 1.2.2. Find the chordal distance between Chicago and Minneapolis, and between New York and New Orleans, comparing your results to the geodesic and naive Euclidean ones above.
10. A two-dimensional projection, often used to approximate geodesic distances by applying Euclidean metrics, sets up rectangular axes along the centroid of the observed locations, and scales the points according to these axes. Thus, with N locations having geographical coordinates $(\lambda_i, \theta_i)_{i=1}^N$, we first compute the centroid $(\bar{\lambda}, \bar{\theta})$ (the mean longitude and latitude). Next, two distances are computed. The first, d_X, is the geodesic distance (computed using (1.10) between $(\bar{\lambda}, \theta_{\min})$ and $(\bar{\lambda}, \theta_{\max})$, where $\theta_{\min}$ and $\theta_{\max}$ are the minimum and maximum of the observed latitudes. Analogously, d_Y is the geodesic distance computed between $(\lambda_{\min}, \bar{\theta})$ and $(\lambda_{\max}, \bar{\theta})$. These actually scale the axes in terms of true geodesic distances. The projection is then given by

$$x = \frac{\lambda - \bar{\lambda}}{\lambda_{\max} - \lambda_{\min}} d_X; \text{ and } y = \frac{\theta - \bar{\theta}}{\theta_{\max} - \theta_{\min}} d_Y \ .$$

Applying the Euclidean metric to the projected coordinates yields a good approximation to the inter-site geodesic distances. This projection is useful for entering coordinates in spatial statistics software packages that require two-dimensional coordinate input and uses Euclidean metrics to compute distances (e.g., the variogram functions in `geoR`, the `spatial.exp` function in `WinBUGS`, etc.).

(a) Compute the above projection for Chicago and Minneapolis ($N = 2$) and find the Euclidean distance between the projected coordinates. Compare with the geodesic distance. Repeat this exercise for New York and New Orleans.

(b) When will the above projection fail to work?

11. Use the `sp`, `rgdal` and `RgoogleMaps` packages to create an UTM projection for the locations in the scallops data and produce the picture in Figure 1.11(b).

12. Use the `fields` package to produce the inter-site distance matrix for the locations in the scallops data. Compute this matrix using the `rdist.earth` function, which yields the geodetic distances. Next project the data to UTM coordinates and use the `rdist` function to compute the inter-site Euclidean distance matrix. Draw histograms of the inter-site distances and comment on any notable discrepancies resulting from the map projection.