High versatility and potential of spatial data analysis with R programming

Nguyen Tien HOANG \(^1\), Nguyen Thi HONG \(^2\) and Katsuaki KOIKE \(^1\)

Abstract: R is a free programming language that has been widely used by statisticians and data miners for statistical software development and data analysis. The number of contributed packages for handling and analyzing spatial data has significantly increased over the last 15 years. This paper reviews the potential for spatial data analysis using R programming. The packages related mainly to geographic information system (GIS), such as \texttt{sp}, \texttt{sf}, \texttt{rgdal}, \texttt{raster}, \texttt{ggmap}, \texttt{tmap}, \texttt{gstat}, and \texttt{RQGIS}, are selected for specific tasks along with useful examples. By referring to these examples, new R users can examine how R handles spatial data and what types of problems it can be applied to. R provides several functions that can import, export, and manipulate vector and raster data. For spatial data analysis, R acts as a GIS tool because it can perform GIS procedures effectively from basic to advanced levels. For visualization and mapping, R can produce various 2D or 3D maps from spatial data using either customized or flexible approaches. A user community needs to be developed to enhance the benefits of R programming for the public and private sectors in Japan, particularly in the field of geoinformatics.

Key words: R package, raster data, vector data, geostatistics, mapping

1. Introduction

The R system is a free software environment for statistical analysis and graphics. It is an execution of the S language that serves the same purpose. For data mining, using a special-purpose language like S can be extremely effective, as compared with using a general-purpose language. Datasets and data analysis approaches can be exchanged and developed using R, and new dedicated components can easily be integrated into R. It is often much harder to achieve these manipulations with programs that require long sequences of mouse clicks to operate (Bivand et al., 2013).

R provides many standard and innovative statistical analysis packages, high-level graphics, and connections to other languages. R supports multicore task distribution, computing through high-performance computer clusters, and handling of large and complex datasets. Because it is open source, R has tremendous learning resources and a worldwide community of volunteer supporters. Therefore, R programming is cost-effective and suitable for developing professional, mission-critical software applications, both for the public and private sectors (Bivand et al., 2013). Ten years after the first R release, developers had built and published more than 200 packages, and the first citation of the “R Project” appeared in 2003 (Tippmann, 2015). At present, more than 13,200 packages are available for all kinds of specialized purposes (CRAN, 2018).

In the field of data mining, R and Python share large parts of the user community (Fig. 1). R is also a programming language preferred by academicians. Muenchen (2017) claimed that the more popular a software package is, the more probable that it will appear in scientific publications as an analysis tool or even an object of study. Figure 2 shows the number of articles found discussing the more popular software packages (those with at least 750 articles) in 2016. R was found to reside in the second place, although it was born after many programming languages and software, such as SPSS, SAS, C, MATLAB, and FORTRAN.

For over 15 years, the number of contributed R packages for handling and analyzing spatial data has increased (Table 1). Although R was not initially designed for geographic information system (GIS), its role has become increasingly critical in this field. This paper aims to highlight the high versatility and potential of spatial data analysis using R with example codes of several GIS-related packages. The codes were produced for the figures, and the examples were selected by the authors. All the resulting figures were exported directly from R without any editing from third party software. In this manuscript, R package name is expressed in a bold typeface such as \texttt{raster} and function’s name and elements and programming code are expressed in the same font such as \texttt{intersect(x, y)}, which is different from the font of main text.

Received: 27 October 2018; Accepted: 17 November 2018

1) Environmental Geosphere Engineering Laboratory, Department of Urban Management, Graduate School of Engineering, Kyoto University, Katsura C1-2, Kyoto 615-8540, Japan. E-mail: nguyen.tienhoang.3x@kyoto-u.ac.jp, koike.katsuaki.5x@kyoto-u.ac.jp

2) Laboratory of Regional Planning, Graduate School of Global Environmental Studies, Kyoto University, Kyoto 606-8501, Japan.
Table 1. Typical R packages for spatial data analysis (Bivand, 2018)

| Main application          | Package name         |
|---------------------------|----------------------|
| Classes for spatial data  | sp, sf               |
| Handling spatial data     | raster, rgdal, rgeos, maptools, shapefiles, rpostgis |
| Visualization and mapping| gmap, tmap, mapview  |
| Geostatistics             | gstat, geoR, geoRglm, geostatsp |
| Spatial regression        | nlme, spdep, sphet   |

Figure 1. Shares of Python, R, both, or other platforms used for analytics, data science, and machine learning in 2016 and 2017. Data source: KDnuggets Poll (Piatetsky, 2017).

Figure 2. Number of scholarly articles published in 2016 for the more popular data science software. To be included, software must be used in at least 750 scholarly articles. Data source: rstats.com (Muenchen, 2017).

2. Handling spatial data
2.1. Vector data

Vector data uses discrete objects, points, lines, and polygons to represent the location and shape of spatial features on the Earth surface. A point is the simplest type, defined by a pair of x- and y-coordinates. Correspondingly, a line or a polygon is represented by a set of x- and y-coordinates. Therefore, basic spatial data can be easily created from R data types, including vectors (numerical, character, and logical), matrices, and data frames.

Because many packages define their own data structures (classes) for spatial data, the sp package was developed to construct shared classes and methods for spatial data (Pebesma, 2018). Other packages, such as rgdal and maptools, depend on the sp classes to provide uncomplicated access ways for vector data (Bivand et al., 2018a, 2018b). R can import shapefiles directly from a local folder or cloud storage. The below code in Box 1 and its product, Figure 3, illustrate an example of reading and writing spatial data by using the rgdal package. The `lines` function plots the shapefile of the river network-polyline type on the shapefile of Hokkaido island-polygon type. R works not only with ESRI shapefiles but also with many other vector data extensions, for instance, MapInfo (.TAB, .MIF), IDRISI (.VCT), GPS (.GPX), and GRASS files.

Figure 3. Shapefiles of the Hokkaido island and its river network plotted using the rgdal package. Data source: http://www.diva-gis.org/gData
High versatility and potential of spatial data analysis with R programming

Box 1. Example code for reading and writing vector data

```r
# load and attach add-on package(s)
> library(rgdal)
# set the working directory
> setwd('H:/')
# read OGR vector maps into Spatial objects
> island <- readOGR("adm.shp")
> river <- readOGR("river.shp")
# plot shapefiles
> plot(island, border = "black", col =="#D1E5F0")
> lines(river, col = "blue", lwd = 1)
# export spatial vector data to shapefile
> writeOGR(island, "hokkaido.shp", "boundary",
driver="ESRI Shapefile")
```

Several packages come with their own samples or general spatial data. One in particular, the `rworldmap` package, consists of global country-level data represented by country boundary polygons and referenced by country names (South, 2011). Such vector data type is beneficial because they can be extracted in R in a straightforward manner for many global-scale applications, such as environmental and climate change modeling. By using three code lines (Box 2), the Environmental Performance Index (EPI) of 180 countries was imported in seconds for further processing (Fig. 4). EPI ranks the nations on 24 performance indicators, covering environmental health and ecosystem vitality.

Box 2. Example code for importing global-scale vector data

```r
> library(rworldmap)
> global <- joinCountryData2Map(countryExData,
joinCode = "ISO3", nameJoinColumn = "ISO3V10")
> mapCountryData(global, nameColumnToPlot="EPI")
```

Table 2. Supported file types for writing raster data.

| File type | Long name          | Default extension | Multiband support |
|-----------|--------------------|-------------------|-------------------|
| raster    | 'Native' raster package format | .grd             | Yes               |
| ascii     | ESRI Ascii         | .asc              | No                |
| SAGA      | SAGA GIS           | .sdat             | No                |
| IDRISI    | IDRISI             | .rst              | No                |
| CDF       | netCDF (requires ncdf4) | .nc              | Yes               |
| GTiff     | GeoTiff (requires rgdal) | .tif             | Yes               |
| ENVI      | ENVI.hdr Labeled   | .envi             | Yes               |
| EHdr      | ESRI.hdr Labeled   | .bil              | Yes               |
| HFA       | Erdas Imagine Images (.img) | .img          | Yes               |

2.2. Raster data

Unlike vector data, raster data uses a regular grid, making it a better option for representing continuous phenomena, covering the space. The `raster` package is the most powerful R package for creating, reading, manipulating, and writing raster data (Hijmans et al., 2018). By containing only information about the structure of the data (the number of rows and columns, i.e., the spatial extent), the package can work with large raster files and popular raster file formats (GeoTIFF, ESRI, ENVI, and ERDAS), via the `rgdal` package. Most formats supported for reading are also compatible with exporting. Figure 5 is an
example of importing a digital elevation model (DEM) stored as a GRI file. The writeRaster function can write the GRI file to a GeoTIFF (see Box 3) or any supported raster file type (Table 2).

**Box 3. Example code for reading and writing raster data**

```r
> library(raster)
# create a RasterLayer object by importing a GRI file
> dem <- raster("JPN_alt.gri")
> dem
class: RasterLayer
dimensions: 2604, 2772, 7218288 (nrow, ncol, ncell)
resolution: 0.008333333, 0.008333333 (x, y)
extent: 122.8, 145.9, 23.9, 45.6 (xmin, xmax, ymin, ymax)
coord. ref.: +proj=longlat +ellps=WGS84
names: JPN_alt
values: -110, 3664 (min, max)
> plot(dem, col = terrain.colors(7), axes = F, legend = T)
> mtext("Elevation (m)", side = 4, at = c(145,41), line = 1, las = 2)
# export raster data to a file
> writeRaster(dem, 'dem.tif', overwrite=TRUE)
```

**Figure 5. A DEM dataset (CGIAR SRTM of 3-s resolution) covering Japan. Data source: http://www.diva-gis.org/gData**

3. Spatial data analysis with R

3.1. Geoprocessing and map overlay

All spatial data is built in some coordinate reference system (CRS), which can be specified in many different ways. Matching the coordinate system of different data layers is the first step for any spatial analysis. The rgdal package uses the PROJ.4 library for CRS transformation. A CRS was assigned by an EPSG code and was defined by a projection, datum, and a set of parameters. The projection arguments must be entered exactly as mentioned in the PROJ.4 library. For example, the CRS of the island data from the code below is "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0" (Box 4). This data can be transformed into a new dataset with another CRS using the spTransform function.

**Box 4. Example code for transforming a CRS**

```r
> library(rgdal)
> island <- readOGR("adm.shp")
# get the CRS of a Spatial object
> crs(island)
CRS arguments:
+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
# define the new CRS
> newcrs <- CRS("+proj=utm +zone=54 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0")
# provide transformation between datum(s) and conversion between projections, from one unambiguously specified CRS to another
> hokkaido <- spTransform(island, newcrs)
> crs(hokkaido)
CRS arguments:
+proj=utm +zone=54 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
```

In using R, switching back and forth between spatial and non-spatial data types is effortless. The functions as.vector() and as.matrix() return all values of a raster object as a vector and a matrix, respectively, of cell values. Likewise, the function tidy of the broom package converts a vector object into a tibble, an advanced data frame. The function as(shapefile, "data.frame") can extract the attributes of the shapefile as a data frame. On the other hand, a file in table format containing two columns of longitude and latitude can be imported and converted into a point layer (Box 5 and Fig. 6).
High versatility and potential of spatial data analysis with R programming

Box 5. Example code for importing a point layer generated from coordinate vectors

```r
> library(rworldmap)
# read a file in table format from internet
> airport <- read.csv("https://raw.githubusercontent.com/jpatokal/openflights/master/data/airports.dat", header = F)
> colnames(airport) <- c("ID", "name", "city", "country", "IATA_FAA", "ICAO", "lat", "lon", "altitude", "timezone", "DST")
> country <- getMap(resolution = "low")
# plot a part of world map defined by xlim and ylim
> plot(country, xlim = c(120, 141), ylim = c(9, 51))
# plot a point layer generated from coordinate vectors
> points(airport$lon, airport$lat, pch = 20, col = "blue", cex =.5)
```

Figure 5. A layer of airport locations (blue) converted from an online file in table format (.CSV, .DAT, or .TXT extension). Data source: https://openflights.org/

Overlay operations are essential and traditional spatial analyses. An overlay operation combines the geometries and attributes of two input layers to create a new output. R acts as a GIS tool because it can deal with many overlay methods such as union, erase, intersect, cover, and symmetrical difference (Box 6 and Fig. 7). R uses standard arithmetic operators for computations with raster objects and numeric values, or with SpatialPolygon objects. The operators available are: + (addition), - (subtraction), * (multiplication), / (division), ^ (exponentiation), %% (modulus), and %/% (integer division).

Given SpatialPolygon objects \( x \) and \( y \), the following three operators can be replaced by corresponding overlay operations:

- \( x+y \) is the same as \( \text{union}(x, y) \);
- \( x*y \) is the same as \( \text{intersect}(x, y) \);
- \( x-y \) is the same as \( \text{erase}(x, y) \).

Moreover, a function \( \text{fun} \) is flexibly set so that the RasterLayers can be combined as \( \text{overlay}(x, y, \text{fun}, \ldots) \).

Box 6. Example code for operating typical overlay functions

```r
> library(sf)
> library(rgeos)
> library(sp)
> library(raster)
# import the vector data of North Carolina, a state of the United States
> data <- system.file("shape/nc.shp", package = "sf")
> shp <- shapefile(data)
> county1 <- which(shp$NAME == 'Robeson')
> county2 <- which(shp$NAME == 'Bladen')
# read the polygon of Robeson County
> RO <- shp[county1,]
> plot(RO)
# read the polygon of Bladen County
> BL <- shp[county2,]
> plot(BL)
# compute a geometric union of two SpatialPolygons objects. Overlapping polygons are intersected, other spatial objects are appended
> UN <- union(RO, BL)
> plot(UN, col = "#F4A582")
> x <- c(-78.987473, -78.468486, -78.468486, -78.987473)
> y <- c(34.683542, 34.683542, 34.383944, 34.383944)
> coord <- cbind(x, y)
> newarea <- SpatialPolygons(list(Polygons(list(Polygon(coord)), 1)))
> proj4string(newarea) <- crs(UN)
> plot(newarea)
# erase parts of a SpatialPolygons or SpatialLines object with a SpatialPolygons object
> ER <- erase(UN, newarea)
> plot(ER, col = "#FDDBC7")
```
# compute a geometric intersection of two Spatial objects
> IN <- intersect(UN, newarea)
> plot(IN, col = "#D1E5F0")

# replace areas of SpatialPolygons object X that overlap with SpatialPolygons object Y by Y
> CV <- cover(UN, newarea)
> plot(CV, col = "#92C5DE")

# compute the symmetrical difference of SpatialPolygons objects
> DI <- symdif(UN, newarea)
> plot(DI, col = "#4393C3")

## Box 7. Example code for integrating R with QGIS

```r
> library("RQGIS")
> library("raster")
> set_env()
> # read DEM of the Mongón area in northern Peru
> data(dem, package = "RQGIS")
> plot(dem)
> # compute hill shade from slope and aspect layers
> hillshade <- hillShade(terrain(dem), terrain(dem, "aspect"), 40, 270)
> plot(hillshade, col = gray(0:100/100))
> # compute TWI by using module SAGA Wetness Index in QGIS
> wetness <- run_qgis("saga:sagawetnessindex", DEM = dem, TWI = file.path(tempdir(), "wetness.tif"), load_output = TRUE)
> plot(wetness, col = RColorBrewer::brewer.pal(n = 9, name = "Blues"))
```

A state-of-the-art package for statistical geo-computing, RQGIS, was introduced in 2017. RQGIS integrates R with QGIS by using the QGIS Python API for console-based geoprocessing. The conceptual model of RQGIS is shown in Figure 8. RQGIS makes it possible to use the thousands of geo-algorithms of QGIS, GRASS, the Orfeo Toolbox, TauDEM, and SAGA within R (Muenchow et al., 2017). Accordingly, more complex, process-oriented terrain attributes can be computed from R. To understand a specific geo-algorithm, the QGIS online help is loaded by the open_help function, and the get_args_man() retrieves the function arguments. It is unnecessary to create new R functions when they are already available from existing dedicated software. An example of using RQGIS is a calculation of the Topographic Wetness Index (TWI), invoked by calling module SAGA Wetness Index, saga:sagawetnessindex and filling the parameters for the argument (Box 7 and Fig. 9).

## 3.2. Spatial interpolation

R can be used as an interpolation tool for predicting values of cells in a raster from a limited number of sample data points. There are several R packages for geostatistical analysis such as gstat, vari diag, geospt, geostatsp, geoR, and geoRglm. The gstat package is the most potent package, offering basic functionality (see Box 8) for univariate and multivariate geostatistics, including variogram modeling and fitting, kriging family (simple, ordinary, indicator, universal, and external drift (co-) kriging), and Gaussian simulation (Pebesma and Graepler, 2018).

Figure 7. Illustrations demonstrating the overlay operations defined by the given codes.
High versatility and potential of spatial data analysis with R programming

Figure 8. Conceptual model of RQGIS (after Muenchow et al., 2017). RQGIS acts as a bridge connecting QGIS and R ecosystems.

Figure 9. (A) DEM, (B) hillshade, and (C) TWI computed by the module SAGA Wetness Index of the Mongón area in northern Peru. The dataset was supplied along with the RQGIS package. The coordinate system used is UTM with the projection units in m.
Figure 10. Bubble plot of spatial data with a scanned topographic map for the background (top) and variogram (below) of topsoil copper concentration (ppm) in a floodplain along the Meuse river. The data was supplied along with the gstat package. The area size is 4 km × 4.5 km. The projected coordinate system is Amersfoort/RD New, and the distance unit is m.

Some functions of the package, for instance, block (co-) kriging, are even more advanced than the geostatistical modules of ArcGIS and QGIS. The meuse dataset, which was chosen for the demonstration of the gstat package, comprises the measurements of four heavy metals (cadmium, copper, lead, and zinc) from the topsoil in a floodplain along the Meuse river in the Netherlands (Fig. 10). Figure 11 compares the spatial prediction of topsoil copper concentration by using ordinary, simple, and universal kriging. There is no significant difference in the copper distribution among the three kriging methods. A small difference is that a continuous boundary of the high copper concentration along the riverbank was predicted by simple kriging.

Box 8. Example code for geostatistical analysis

```r
> library(gstat)
> library(sp)
> library(raster)
# read the scanned topographic map
> topo <- stack("topomap2m.tif")
# read the Meuse river data set
> data(meuse)
> coordinates(meuse) = ~x+y
> data(meuse.grid)
> coordinates(meuse.grid) = ~x+y
> gridded(meuse.grid) = TRUE
# create a bubble plot of spatial data of topsoil copper concentration
> plotRGB(topo, r = 1, g = 2, b = 3)
> points(meuse, pch = 19, cex = meuse$copper/40, col = rgb(red = 0.81, green = 0, blue = 0.92, alpha = 0.7))
# fit ranges and/or sills from a simple or nested variogram model to a sample variogram
> mod <- variogram(log(copper)~1, meuse)
> fit <- fit.variogram(mod, model = vgm(1, "Sph", 900, 1))
> plot(mod, fit)
# compute ordinary kriging
> OK <- krige(log(copper)~1, meuse, meuse.grid, model = fit)
> spplot(OK["var1.pred"])
# compute simple kriging
> SK <- krige(log(copper)~1, meuse, meuse.grid, model = fit, beta = 6)
> spplot(SK["var1.pred"])
# compute universal kriging
> UK <- krige(log(copper)~x+y, meuse, meuse.grid, model = fit, block = c(50,50))
> spplot(UK["var1.pred"])
```
Figure 11. Comparison of spatial prediction of topsoil copper concentrations by ordinary, simple, and universal kriging. The scales in the color index are natural logarithms of copper concentrations (ln ppm).

Figure 12. Choropleth map of polygon perimeters in degree unit of the state of North Carolina, U.S.A. The dataset was supplied along with the sf package.
Other traditional interpolation methods, such as inverse distance weighting (IDW) and cubic spline algorithms, are also available in R. The `idw` function of the `phylin` package transforms a set of samples with location and value to a table of coordinates, based on the IDW algorithm with three different methods available for weight calculation (Shepard, modified Shepard, and Shepard with neighbors). The `splinefun` of the `stats` package performs cubic spline interpolation of given data points, returning either a list of interpolated points or an interpolation function. In addition, several packages try to solve specific problems. The `rtop` package provides geostatistical interpolation functions of data with irregular spatial support. The `mps` package illustrates some theoretical aspects of multiple-point statistics. The number of geostatistical packages from the R community continues to grow.

3.3. Visualization and mapping

R has several professional packages for spatial data visualization and mapping such as `ggmap`, `tmap`, and `sf`. The `sf` package provides modern alternative options for parts of the `sp`-family of packages to handle spatial vector data in R (Pebesma, 2018). Plotting a map of simple features represented by spatial vector data is presently very fast and simple by using the map templates in `sf` (Box 9 and Fig. 12). A new package, `tmap` can be used to create many types of thematic maps, including choropleth, categorical, proportional symbol, isopleth or contour, cartogram, raster, and dasymetric maps (Tennekes, 2018). It was built for both flexible and fixed approaches based on the following layered grammar of graphics.

Box 9. Example code for creating a choropleth map

```r
> library(sf)
# import the data
> NC = read_sf(system.file("gpkg/nc.gpkg", package="sf"))
# create a choropleth map
> plot(NC[, 2], key.pos = 1, axes = TRUE, graticule = TRUE, pal = terrain.colors(15, alpha = 1), nbreaks = 15)
```

The `plot3d` and `rgl` packages provide useful functions for 3D graphics in R. A recently developed package, `rayshader`, produces hill-shaded maps of elevation matrices with ray tracing and spherical texture mapping (Morgan-Wall, 2018) and also supports 3D mapping, as it can be used to pass a texture map into the `plot_3d` function. Box 10 is an example code in which a water layer was added onto a bathymetric/topographic 3D map (Fig. 13). The appearance and transparency of the water layer can be customized by adjusting the function arguments.

Box 10. Example code for plotting a 3D map

```r
> library(rayshader)
> library(rgl)
> data <- montereybay
# compute a global shadow map for a elevation matrix
> globalshadow <- ray_shade(data, zscl = 50, lambert = FALSE)
# compute Ambient Occlusion shadow map
> ambient <- ambient_shade(data, zscl = 50)
# plot a 3D map of a bathymetric/topographic data and add a water layer on it
> data %>% sphere_shade(zscl = 10, texture = "imhof1") %>% add_shadow(globalshadow, 0.5) %>% add_shadow(ambient) %>%
plot_3d(data, zscl = 50, fov = 0, theta = -45, phi = 45, windowsize = c(800,800), zoom = 0.9, water = TRUE, waterdepth = 6,
wateralpha = 0.5, watercolor = "lightblue", waterlinecolor = "white", waterlinealpha = 0.3)
# save the 3D map as png file
> rgl.snapshot("fig.png", fmt = "png", top = TRUE)
```

A tool, `ggmap`, produces mapping and visualization styles by combining the spatial information of static maps from Google Maps, OpenStreetMap, Stamen Maps, or CloudMade Maps (Kahle and Wickham, 2013). This tool is used in conjunction with `ggplot2`, which can be used to plot a downloaded map image as a context layer and then, plot additional content layers of data, statistics, or models on the top of the map. With the extensive map database from online services, `ggmap` provides several useful tools to obtain mapping components easily and fast for any place around the world. Recently, Google has changed its API requirements, such that users need to provide an API key to load the Google Maps data. The `register_google()` function can be used to set the API key in `ggmap` (Box 11). Figure 14 is a product of adding a vector layer of the three campuses of Kyoto University (Yoshida main campus in the north, Uji in the south, and Katsura in the west) onto the Google Maps background. This vector layer was defined by the coordinates of the campuses as a data frame.
High versatility and potential of spatial data analysis with R programming

Box 11. Example code for plotting a point vector layer on the Google Map background

```r
> library(ggmap)
> library(tidyverse)

# register an API key to load Google Maps data
> register_google(key = "your_API_key_here", account_type = "premium", day_limit = 100000)

# create a data frame of point coordinates
> x <- c(135.780828, 135.800648, 135.679666)
> y <- c(35.026218, 34.910104, 34.981831)
> label <- c("Yoshida campus","Uji campus", "Katsura campus")
> coord <- data.frame(x, y, label)

# download Google Maps data of an identified area
> kyoto <- c(left = 135.650680, bottom = 34.885248, right = 135.842272, top = 35.044070)
> bg <- get_stamenmap(kyoto, zoom = 14)

# plot the point vector layer on the Google Maps background
> ggmap(bg) + geom_point(aes(x = x, y = y), data = coord, alpha =.5, col = "red", size = 5) + geom_text(data = coord, aes(x = x, y = y, label = label), size = 3, vjust = 1.8, hjust = 0.5, col = "red")
```

4. Conclusion

R is an excellent choice for spatial data analysis due to its excellent capability in analyzing and visualizing data. This paper provides a brief accessible introduction of spatial data analysis with R. Generally, R can implement most tasks that other GIS software can perform. Base R includes many functions that can be used for reading, visualizing, and analyzing spatial data. It has much more depth than what is presented here. With a huge developmental and operational community, many R packages have been developed and used over the years. There are various powerful tools, such as raster, RQGIS, and gstat packages, which enable R to handle advanced GIS processing. It is necessary indeed to learn R and participate in the developing R community in the field of geoinformatics.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers 18H01924 and 18H06496. The authors wish to express their sincere gratitude to Dr. Yukari Kido for valuable comments and suggestions that helped improve the clarity of the manuscript.

References

Bivand, R. (2018) CRAN Task View: Analysis of Spatial Data. https://CRAN.R-project.org/view=Spatial (accessed: 2018/10/18)

Bivand, R., Keitt, T., Rowlingson, B., Pebesma, E., Sumner, M., Hijmans, R., Rouault, E., Warmerdam, F., Ooms, J., and Rundel, C. (2018a) rgdal: Bindings for the "Geospatial" Data Abstraction Library. https://CRAN.R-project.org/package=rgdal (accessed: 2018/10/18)
要旨

Rプログラミングによる空間データ解析の高い汎用性と可能性

ホアン ティアン・ホン ティ・小池 克明

Rはフリーのプログラミング言語であり、統計ソフト開発やデータ解析のために統計学、データマイニングの分野で広く用いられている。ここ15年間で空間データの処理や解析用に作成されたパッケージ数は大幅に増加している。本論文では、sp, sf, rgdal, raster, ggmap, tmap, gstat、RQGISのような地理情報システム(GIS)関連の主要パッケージに注目し、様々な有用な例を付けてRを用いた空間データ解析のポテンシャルをレビューする。この実例によってRがどのように空間データを扱い、どのような種類の問題に応用できるかを把握できる。Rはインテロープ、エクスポート、およびベクターとラスターデータの処理用に複数のシナリオを提供する。空間データ解析に対しては、Rは基本から発展レベルまでのGIS処理を効率的に実行できるので、GISツールのように作動する。Rの可視化とマッピングでは、カスタマイズあるいはフレキシブルのいずれかのアプローチで、空間データから2次元、あるいは3次元マップが作成される。Rプログラミングの利点をさらに向上させるために、日本でも、特に情報地質学の分野でユーザー-コミュニティの形成が望まれる。

キーワード：Rパッケージ、ラスターデータ、ベクターデータ、地球統計学、マッピング