# INTRODUCTION TO SPATIAL AND TABULAR DATA ANALYSIS WITH R

*Setyono Hari Adi*

**ABSTRACT**

Geographic Information System (GIS) has been extensively utilized in water resources management because it features integrated tools to analyze spatial and tabular dataset. Proprietary software for GIS processing is available in the market, but the cost is expensive that only certain organizations are able to make such investment. This article covers technical introductory on data analysis and GIS processing using R Statistics. R is a free statistical software that also support software extensions (packages) for spatial and tabular data manipulation. These features enable R as an integrated tool for big data analysis in GIS to support water resources modeling.

Keywords: R, GIS, modeling

**ABSTRAK**

Sistem Informasi Geografis (SIG) telah banyak digunakan sebagai alat bantu analisis data spasial dan tabular terintegrasi untuk pengelolaan sumber daya air. Perangkat lunak berbayar untuk pemrosesan GIS sudah banyak tersedia, tetapi mahal sehingga hanya organisasi tertentu yang dapat melakukan investasi. Artikel ini mencakup pengantar teknis tentang analisis data dan pemrosesan GIS menggunakan perangkat lunak analisis statistik R. R adalah perangkat lunak statistik gratis yang juga mendukung ekstensi perangkat lunak (paket) untuk manipulasi data spasial dan tabular. Fitur-fitur ini memungkinkan R sebagai alat terintegrasi untuk analisis data besar dalam SIG untuk mendukung pemodelan sumber daya air.

Kata kunci: R, SIG, pemodelan

## OVERVIEW

R is a free statistical software suite for data analysis and graphical data representation that provides a flexible and interactive programming environment and contains large collections of packages, including statistical modeling and data processing (Venables et al., 2016). R Statistics is an interpreter-based software that supports extensions to reduce the data analysis complexity while increasing the processing efficiency. This software tool has been used extensively in research particularly for statistical data analysis. Apart from abundant statistical modeling extensions, R also include packages for spatial data processing, tabular database management, and performance enhancement. This article therefore focused on introduction to spatial and tabular data processing using R, with a brief discussion on how to enhance R processing performance in handling big data analysis.

**Spatial Data Manipulation with R**

Spatial data are commonly used in environmental modeling, especially for distribution analysis over space and time periods. This data can be originated from satellite observations, ground instrumentations, or traditional surveys, in which each value of the parameters is bound into its geographic position. Several dedicated Geographic Information System (GIS) software exist, such as ArcGIS, QGIS, SAGA, and Geographic Resources Analysis Support System (GRASS), although R also provides free packages for GIS-based processing. These packages include software plug-ins for direct manipulation of vector-based and raster-based data and functions that bridge communications between R and other GIS-based software. For the latter, these functions enable R to run certain commands from other GIS software, so that the model flow and parameterization can be controlled within a single R environment.

**Spatial Data Processing**

Spatial data have specific data formats, whether it is raster or vector. The Geospatial Data Abstraction Library (GDAL) is a library that provides translations for raster and vector data under Open Source Geospatial Foundation (OSGEO) standard formats. This library enables supported spatial data files to be imported using the defined standard procedure. Four packages available for handling spatial data include "sp", "rgdal", "raster" and "fields".

The "sp" package provides base classes to store the spatial objects and methods to access the stored spatial objects of the imported spatial data so that the information inside the spatial data can be stored, retrieved, or updated using a particular standard procedure (Bivand et al., 2013; Pebesma and Bivand, 2005).

```
library(rgdal)
library(raster)
library(fields)
## load coordinates file
FL.c = read.table("Coord.txt", header=T)
## converts data.frame to spatial data.frame based on the geolocation
coordinates(FL.c) = c("Longitude", "Latitude")
## define projection (library: raster)
## epsg:4326 >>> WGS 84 geographic projection
projection(FL.c) = CRS("+init=epsg:4326")
## add new parameter (new "random number" column) to existing spatial dataset
new_column = as.data.frame(abs(rnorm(1014, mean = 5, sd = 10)))
names(new_column) = "val"
FL.c@data = cbind(FL.c@data, new_column)
## reproject data (library: sp)
## epsg:3086 >>> albers GDS Florida projection
FL.c.proj = spTransform(FL.c, CRS("+init=epsg:3086"))
## point interpolation
xy = FL.c.proj@coords
vals = FL.c.proj@data$val
## define interpolation model (library: fields)
## Tps = thin plate spline
spline = Tps(xy, vals)
## define template raster for interpolation
e = extent(FL.c.proj)
## define resolution for template raster
r = raster(e, resolution = 1000)
## perform point interpolation
FL.raster = interpolate(r, spline)
## raster data masking
## load shapefile:polygon (library: rgdal)
m = readOGR(".", "FL")
m.proj = spTransform(m, CRS("+init=epsg:3086"))
FL.raster.masked = mask(FL.raster, m.proj)
## raster value extraction (by point)
data = data.frame(FL.c.proj@coords, FL.c@data$SiteID,
                  extract(FL.raster.masked, FL.c.proj@coords))
names(data) = c("x","y","SiteID","Value")
## shapefile to raster
FL.raster.p = rasterize(m.proj, r)
```

**Figure 1.** Example of spatial data processing with R

The standard classes and methods are valuable for other R developers to create specific functions for spatial data processing. The "rgdal" package (Bivand et al., 2016) provides a binding mechanism for GDAL and functions to read and write GDAL supported spatial data files. Furthermore, the "raster" package (Hijmans, 2016) is built specifically for raster data analysis, while the "fields" package contains functions for spatial model definition (e.g., spline, kriging) that might be integrated within the raster data analysis (Douglas Nychka et al., 2015). An example of R functions for spatial data processing is presented in Figure 1.

**Communication with Other GIS Software**

Along with packages that enable direct manipulation of spatial data, several other packages within R also provide a standard mechanism to run the function from other GIS software. This mechanism enables parameterization to be programmed within the R environment, passing the variables to be executed outside R, and then getting the execution results back to R for further processing. The three packages used in this research consist of "gdalUtils", "rts" and "RSAGA".

Moreover, OSGEO also provides a free command-line based application for spatial data processing (e.g., gdal_translate() command for spatial data conversion). For the R environment, this utility is wrapped under the "gdalUtils" package (Greenberg and Mattiuzzi, 2015). Because the actual executable functions are located outside the R environment, to use the provided functions, the GDAL utility installation path has to be defined using the gdal_setInstallation(search_path="GDAL installation folder") command. This command automatically sets the "gdalUtils" environment variables that are needed to run GDAL utilities.

The "rts" package (Naimi, 2016) is a notable package for handling Moderate Resolution Imaging Spectroradiometer (MODIS) data in the hierarchical data format (HDF). This package provides the ModisDownload()function to download (and optionally to mosaic) MODIS data tiles for certain observation date ranges. This function runs executable commands from the MODIS Reprojection Tool software (abbreviated as MRT) to perform the data mosaicking process. Therefore, the installation path of the MRT software needs to be defined inside this function. This function also requires the "rcurl" package (Lang, 2016) to perform HyperText Transfer Protocol (HTTP) communications with the NASA website. Before the download process can be initiated, the login information to authorize the download process from the NASA website must be defined. This authentication process is performed using the setNASAauth() command. Furthermore, the list of the MODIS data product can be run using the modisProducts() command. An example of the R script to download, mosaic, mask, and convert HDF to the GeoTIFF format using a combination of "rts" and "gdalUtils" is presented in Figure 2.

```
library("rts")
library("gdalUtils")
library("rcurl")
## set authentication information to login to NASA website
setNASAauth(username = "USER", password = "PASSWORD")
## prepare the date to download
d = "2002/02/16"
d = format(d, "%Y.%m.%d")
## perform download process
for (dt in d) {
  if (dt!= ".") {
    ## MODIS data download and mosaicking (library: rts)
    ## ex. MOD17A2 = MODIS GPP, Florida tiles = h10v05 and h10v06
    ModisDownload(x="MOD17A2", h=10, v=c(5,6), dates = dt, bands_subset = "1 0 0",
                  mosaic = T, delete = T, proj = F, MRTpath = "c:/MRT/bin")
    ## Mask hdf (library: gdalUtils)
    hdf = list.files(pattern="*.hdf$")
    hdf.masked = paste("masked.", hdf, sep="")
    gdalwarp(hdf, dstfile = hdf.masked, cutline = "FL.shp", crop_to_cutline = T)
    ## convert hdf to tif (library: gdalUtils)
    tif = paste(hdf, ".tif", sep="")
    gdal_translate(hdf.masked, dst_dataset = tif)
  }
}
```

**Figure 2.** R script to download, mosaic, mask, and covert MODIS HDF files to GeoTIFF

"RSAGA" is the R package to communicate with the System for Automated Geoscientific Analyses (SAGA) software (Brenning, 2008). This package provides functions to run SAGA geoprocessing modules from the R environment. The SAGA installation path is defined using the rsaga.env(path = "SAGA-installation-path") command. The SAGA geoprocessing modules are classified into libraries, where the list of this libraries can be retrieved using the rsaga.get.libraries(saga.env$modules) command. Furthermore, geoprocessing modules inside any particular library (e.g., hydrology library under "ta_hydrology") can be retrieved using the rsaga.get.modules(libs = "ta_hydrology", env = saga.env) command, while the usage of certain modules (e.g., "Topographic Wetness Index (TWI)" module under the "ta_hydrology" library) is run using the rsaga.get.usage("ta_hydrology", "Topographic Wetness Index (TWI)", env=saga.env) command. The result of this command is the list of required parameters for the SAGA module. Finally, the geoprocessing module execution (e.g., TWI) is governed using the rsaga.geoprocessor("ta_hydrology", "Topographic Wetness Index (TWI)", env=saga.env, param=list(TWI-module-parameters)) command.

**Database Management with R**

R also provides packages to work with the Database Management System (DBMS) software. This research used the DBMS software to manage large, millions of tabular data records, where data are structured in the form of relational tables to simplify the data query processes using the SQL language. The DBMS software was built specifically for big data management; therefore, it offers better performance and efficiency compared to traditional data handling. This research used the MySQL database server as the DBMS software, where the "RMySQL" package is used to bridge the communications between R and MySQL.

"RMySQL" is an R package that provides an interface and driver for the MySQL database (Ooms et al., 2016). This package provides basic functions to connect and pass the SQL query to be executed by the MySQL server. The result of the select query is returned to R in the form of the data frame, which later can be used for further analysis. The general syntax to execute the MySQL query using tahe "RMySQL" package is provided in Figure 3.

```
library(RMySQL)
## connect to MySQL server
con=dbConnect(MySQL(),user="USERNAME", password="PWD",
              dbname="DATABASE-NAME", host="localhost")
## define sql query
sql = "SELECT * FROM table_name" ## SELECT query (example)
#sql = "UPDATE table_name SET columnA = new_value" ## UPDATE query (example)
## send query to MySQL, and return the result to rs variable
rs = dbSendQuery(con, sql)
## fetch data from recordset
data = fetch(rs, n=-1)
## clear data in the recordset
dbClearResult(rs)
## disconnect current connection
dbDisconnect(con)
```

**Figure 3.** The general syntax of query execution in "RMySQL"

**R Software Performance Enhancement**

R is an interpreter-based software that provides an interactive programming environment. However, due to the nature of the interpreter-based software, the execution of large amounts of command-scripts (especially for-loop) could significantly decelerate the overall processes. This bottleneck process occurs due to the nature of the interpreter-based program architecture, in which the command-scripts are executed (interpreted) on-the-fly during the process. This back-and-forth interpretation processes create massive communication between R and the computer processing unit that decelerate the overall performance. Furthermore, because data processing is performed in the computer Random Access Memory (RAM), analyzing large datasets could also potentially trigger memory problems. Therefore, two approaches are available for large data management, which include parallel processing and virtual memory management.

**Parallel Processing**

Two packages to enable parallel processing in R are "foreach" (Weston, 2015a) and "doParallel" (Weston, 2015b). This combination is specially designed to enable parallel processing in asynchronous processes that are wrapped into a for-loop. These packages utilize the advances of the multi-core processor technology, in which asynchronous commands can be performed simultaneously (virtually parallel) on different processor cores. At the end of the loop, the results of these processes are merged and returned to the main command. The general syntax for parallel processing using these two packages is presented in Figure 4.

```
library(foreach)
library(doParallel)
## define clusters for available cores
cores = detectCores()-1
clusters = makeCluster(pcores)
## register cluster processing
registerDoParallel(clusters)
## define number of processes
processes = vector(mode="numeric", length=n)
## perform parallel processing
results = foreach (i=iter(processes)) %dopar% {
  ## defines asynchronous process that is controlled by index i
}
## unregister the clusters
stopCluster(clusters)
```

**Figure 4.** The general syntax for parallel processing in R

**Virtual Memory Management**

The virtual memory management in R is performed using the "ffbase" package (Jonge et al., 2016), which is the basic statistical function for the "ff" package (Adler et al., 2014). This library enables atomic data structures virtualization where the processed data are stored on disk, rather than in the memory, and only a section of its structure are mapped into the RAM. This strategy enables R to efficiently process large datasets without memory limitation. To use this function, the variable only needs to be defined as "ff" or "ffdf" for vector or data frame, respectively, using as.ff(vector( )) or as.ffdf(data.frame( )) commands. Furthermore, the name of the function to manipulate the ff (or ffdf) variables is generally similar to the built-in basic function of R with the additional ".ff" (or ".ffdf") suffix. For example, sum.ff to obtain the vector sum and subset.ffdf to obtain the subset data frame. The complete list of the supported functions is available in the help menu by executing the help("ffdf") command within the R environment.

An example on how these techniques enhance the execution process in R is presented in Table 1. Note that this result cannot be used as a reference since the actual performance is also influenced by the algorithm efficiency and the computer specification.

Table 1. Comparison of time to process 10 parameters numerical weather data with more than 3.5 million of rows

| No. | Technique | Time elapsed (s) |
|---|---|---|
| 1. | Standard for-loop with standard variable | 263.76 |
| 2. | Standard for-loop with ff variable | 87.02 |
| 3. | Foreach with doParallel (7 cores) and standard variable | 35.74 |
| 4. | Foreach with doParallel (7 cores) and ff variable | 27.34 |

Note: Processes were run on Intel Core™ i7-3770 Core Processing Unit (CPU) at 3.4 GHz with 16 GB Random Access Memory (RAM) machine

Table 1 shows that the highest execution efficiency was achieved by combining both the virtual memory management and parallel processing techniques. However, combining these two approaches might also significantly increase the program's complexity.

## REFERENCES

Adler, D., Gläser, C., Nenadic, O., Oehlschlägel, J., & Zucchini, W. (2014). *ff: Memory-efficient storage of large data on disk and fast access functions*. Retrieved from https://CRAN.R-project.org/package=ff

Bivand, R., Keitt, T., & Rowlingson, B. (2016). *rgdal: Bindings for the Geospatial Data Abstraction Library*. Retrieved from https://CRAN.R-project.org/package=rgdal

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2013). *Applied spatial data analysis with R* (2nd ed.). New York: Springer.

Brenning, A. (2008). Statistical geocomputing combining R and SAGA: The example of landslide susceptibil ity analysis with generalized additive models. In *J. Böhner, T. Blaschke, L. Montanarella (Eds.), SAGA — Seconds Out. Hamburger Beiträge zur Physischen Geographie und Landschaftsökologie* (2nd ed., Vol. 19, pp. 23–32).

Douglas Nychka, Reinhard Furrer, John Paige, & Stephan Sain. (2015). *fields: Tools for spatial data*. Re trieved from www.image.ucar.edu/fields

Greenberg, J. A., & Mattiuzzi, M. (2015). *gdalUtils: Wrappers for the Geospatial Data Abstraction Library (GDAL) Utilities*. Retrieved from https://CRAN.R-project.org/package=gdalUtils

Hijmans, R. J. (2016). *raster: Geographic Data Analysis and Modeling*. Retrieved from https://CRAN.R-pro ject.org/package=raster

Jonge, E. de, Wijffels, J., & Laan, J. van der. (2016). *ffbase: Basic Statistical Functions for Package "ff."* Retrieved from https://CRAN.R-project.org/package=ffbase

Lang, D. T., & team, the C. (2016). *RCurl: General Network (HTTP/FTP/...) Client Interface for R*. Re trieved from https://CRAN.R-project.org/package=RCurl

Naimi, B. (2016). *rts: Raster Time Series Analysis*. Retrieved from https://CRAN.R-project.org/package=rts

Ooms, J., James, D., DebRoy, S., Wickham, H., & Horner, J. (2016). *RMySQL: Database Interface and "MySQL" Driver for R*. Retrieved from https://CRAN.R-project.org/package=RMySQL

Pebesma, E. J., & Bivand, R. S. (2005, October 2). Classes and methods for spatial data in R. *R News*, p. 5.

Ooms, J., James, D., DebRoy, S., Wickham, H., Horner, J., 2016. RMySQL: Database Interface and "MySQL" Driver for R.

Pebesma, E.J., Bivand, R.S., 2005. Classes and methods for spatial data in R. R News 5.

Venables, W., Smith, D., & the R. Core Team. (2016). *An Introduction to R: A Programming Environment f or Data Analysis and Graphics Version 3.3.2*. Bristol: Network Theory.

Weston, S., 2015a. foreach: Provides Foreach Looping Construct for R.

Weston, S., 2015b. doParallel: Foreach Parallel Adaptor for the "parallel" package.