

Chapter 48: spatial analysis

Section 48.1: Create spatial points from XY data set

When it comes to geographic data, R shows to be a powerful tool for data handling, analysis and visualisation.

Often, spatial data is available as an XY coordinate data set in tabular form. This example will show how to create a spatial data set from an XY data set.

The packages `rgdal` and `sp` provide powerful functions. Spatial data in R can be stored as `Spatial*DataFrame` (where `*` can be `Points`, `Lines` or `Polygons`).

This example uses data which can be downloaded at [OpenGeocode](#).

At first, the working directory has to be set to the folder of the downloaded CSV data set. Furthermore, the package `rgdal` has to be loaded.

```
setwd("D:/GeocodeExample/")  
library(rgdal)
```

Afterwards, the CSV file storing cities and their geographical coordinates is loaded into R as a `data.frame`

```
xy <- read.csv("worldcities.csv", stringsAsFactors = FALSE)
```

Often, it is useful to get a glimpse of the data and its structure (e.g. column names, data types etc.).

```
head(xy)  
str(xy)
```

This shows that the latitude and longitude columns are interpreted as character values, since they hold entries like "-33.532". Yet, the later used function `SpatialPointsDataFrame()` which creates the spatial data set requires the coordinate values to be of the data type `numeric`. Thus the two columns have to be converted.

```
xy$latitude <- as.numeric(xy$latitude)  
xy$longitude <- as.numeric(xy$longitude)
```

Few of the values cannot be converted into numeric data and thus, NA values are created. They have to be removed.

```
xy <- xy[!is.na(xy$longitude),]
```

Finally, the XY data set can be converted into a spatial data set. This requires the coordinates and the specification of the Coordinate Reference System (CRS) in which the coordinates are stored.

```
xySPoints <- SpatialPointsDataFrame(coords = c(xy[,c("longitude", "latitude")]),  
proj4string = CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"),  
data = xy  
)
```

The basic plot function can easily be used to sneak peak the produced spatial points.

```
plot(xySPoints, pch = ".")
```



Section 48.2: Importing a shape file (.shp)

rgdal

ESRI shape files can easily be imported into R by using the function `readOGR()` from the `rgdal` package.

```
library(rgdal)
shp <- readOGR(dsn = "/path/to/your/file", layer = "filename")
```

It is important to know, that the `dsn` must not end with `/` and the `layer` does not allow the file ending (e.g. `.shp`)

raster

Another possible way of importing shapefiles is via the `raster` library and the `shapefile` function:

```
library(raster)
shp <- shapefile("path/to/your/file.shp")
```

Note how the path definition is different from the `rgdal` import statement.

tmap

`tmap` package provides a nice wrapper for the `rgdal::readOGR` function.

```
library(tmap)
sph <- read_shape("path/to/your/file.shp")
```