map3d
3D plotting methods for spatial data

Nikolai Gorte

November 26, 2013

Contents

1 Introduction

The map3d package provides basic methods for plotting spatial data on maps from the OpenStreetMap package using the RGL package. This combination allows the creation of interactive, 3D plots of spatial data on a map.
Spatial data refers to points, lines, polygons and grids each of them with or without attributes as implemented in the spatial data classes in sp.

The motivation for this package was to give the user a nice visualization of spatial data (plotting it on a map) and maybe a way to inetract with it, which has led to the use of RGL.

This vignette describes the methods provided by map3d and also shortly describes some of the packages map3d depends on.

2 OpenStreetMap - package

The OpenStreetMap package provides high resolution map and satellite data for use in R. It currently supports maps from OSM, Bing, Waze, Mapquest, Maptoolkit, MapQuest, stamen, MapBox, ESRI, NPS, Apple, skobbler, OpenCycleMap and CloudMade. A very nice feature which has led to the use of the OpenStreetMap package is the support of transformations between spatial coordinate systems.

3 RGL - package

RGL extends R with a 3D real-time visualization Device System. It is written in C++ using OpenGL.
It offers three-dimensional, real-time visualization functionality and therefore allows the user to generate interactive 3D graphics.

In this package RGL is used to visualize maps and spatial data to give the user a nicer view on the data and the possibility to interact with the graphics e.g. rotate or zoom.

4 Methods

This section describes the methods in map3d and shows some examples.

4.1 map3d

The map3d method creates a 3D RGL plot of a map object.
map3d uses the color matrix and bounding box of the OpenStreetMap map object to create a surface using the surface3d function from RGL.

```
> map <- openmap(c(52.05, 7.5), c(51.9, 7.75),12, type="osm")
> # Reproject if you want. Takes some time!
```

```
> map <- openproj(map, CRS("+init=EPSG:4326"))
> map3d(map)
```

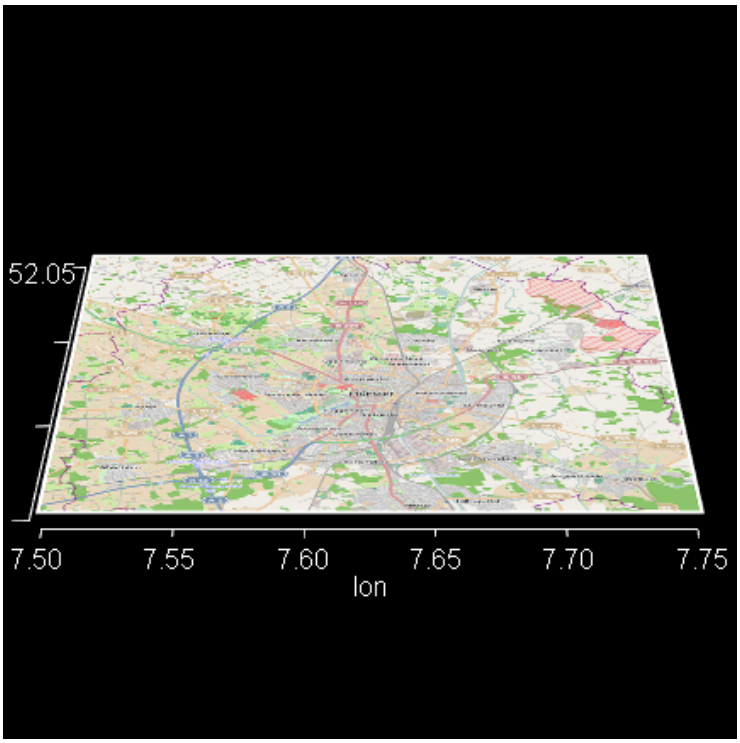

Figure 1: 3D plot of a reprojected map object

4.2 spplot3d

`spplot3d` is the main method of this package. It takes a spatial object and plots it on a map.
The method automatically gets the rigth map section using the bounding box of the spatial object and plots it on the map. It supports spatial data with and without attributes and and also CRS transformation.
Since `RGL` does not include any methods to create legends, the legend is shown as a normal `R` plot.

4.2.1 Points

Example plots of a `SpatialPointsDataFrame`.

```
> crs = CRS("+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555
+           +lon_0=5.38763888888889 +k=0.9999079 +x_0=155000
```

```
+           +y_0=463000 +ellps=bessel
+           +towgs84=565.417,50.3319,465.552,-0.398957,0.343988,-1.8774,4.0725
+           +units=m +no_defs")
> ## create SpatialPointsDataFrame
> data("meuse")
> coordinates(meuse) <- ~x+y
> proj4string(meuse) <- crs
> ## plot
> spplot3d(meuse, att = "zinc", grid = TRUE)
```

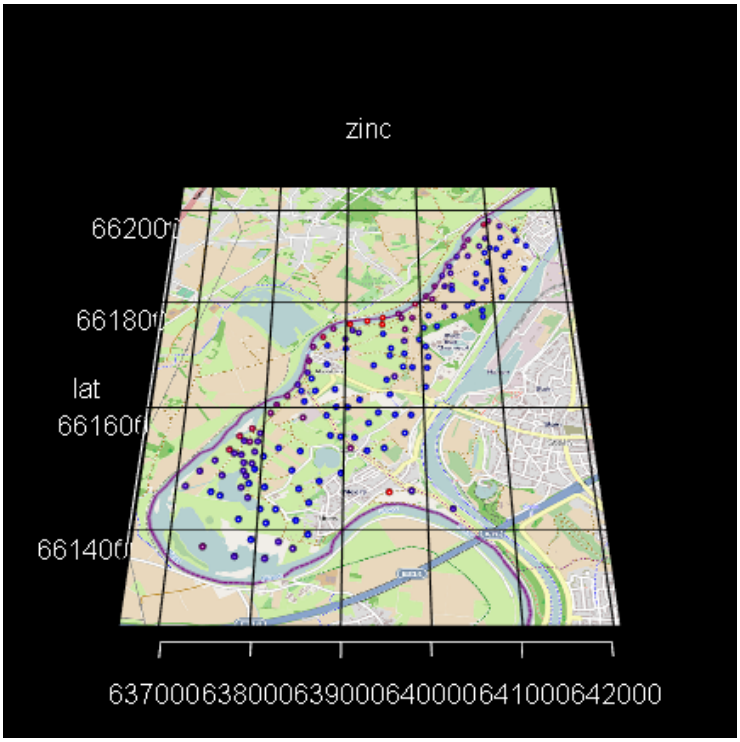

Figure 2: Spatial points with attributes

As you can see in the previous plot, the points are plotted on the map and the color of the points represents the chosen attribute (in this case the zinc concentration). The colors are chosen automatically, but you can also define them by yourself as shown in the following plot.

```
> spplot3d(meuse, att = "zinc", col = c("yellow", "red"), type = "bing")
```

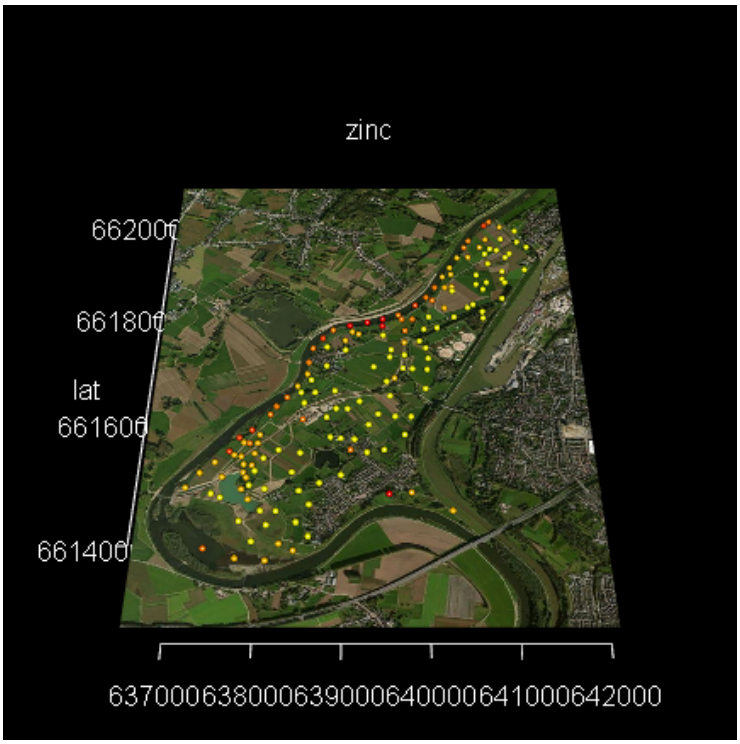

Figure 3: Spatial points with attributes and custom colors

4.2.2 Lines

Example plots of a `SpatialLinesDataFrame`.

```
> data("gewaesserlinien")
> spplot3d(gewaesserlinien, "glName", col = c("blue", "cyan"), CRS = CRS("+init=EPSG:4326"))
```

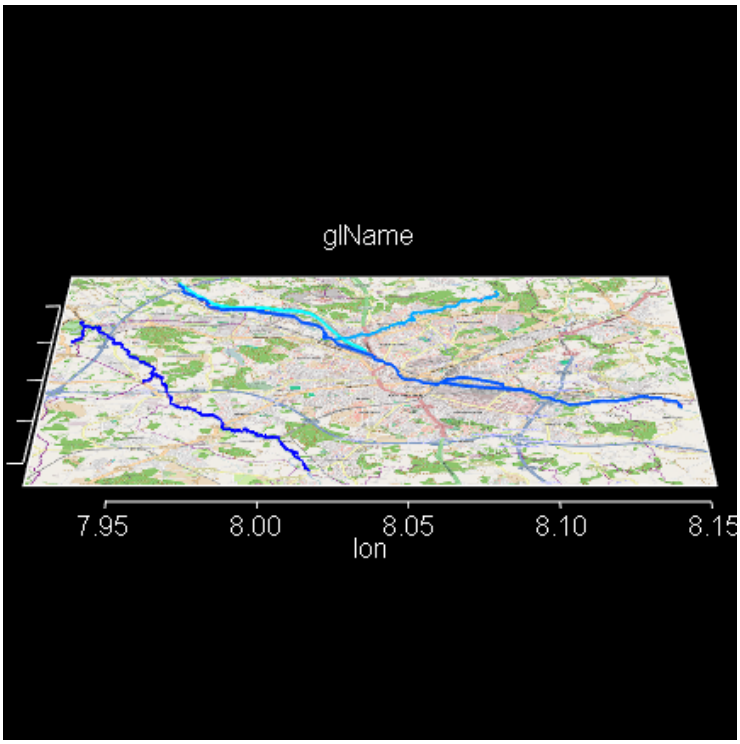

Figure 4: Spatial lines with attributes and custom colors

4.2.3 Polygons

Polygons are supported in `RGL`, but the triangulation of the polygons is extremly time consuming, which is the reason why polygons in this package are plotted using lines. Unfortunately because of this polygons can't be filled and numeric attributes are represented by the height of segments plotted in the centers of the polygons as shown in the following example.

```
> data("vg2500_bld")
> spplot3d(vg2500_bld, "SHAPE_AREA", CRS = CRS("+init=EPSG:4326"))
```

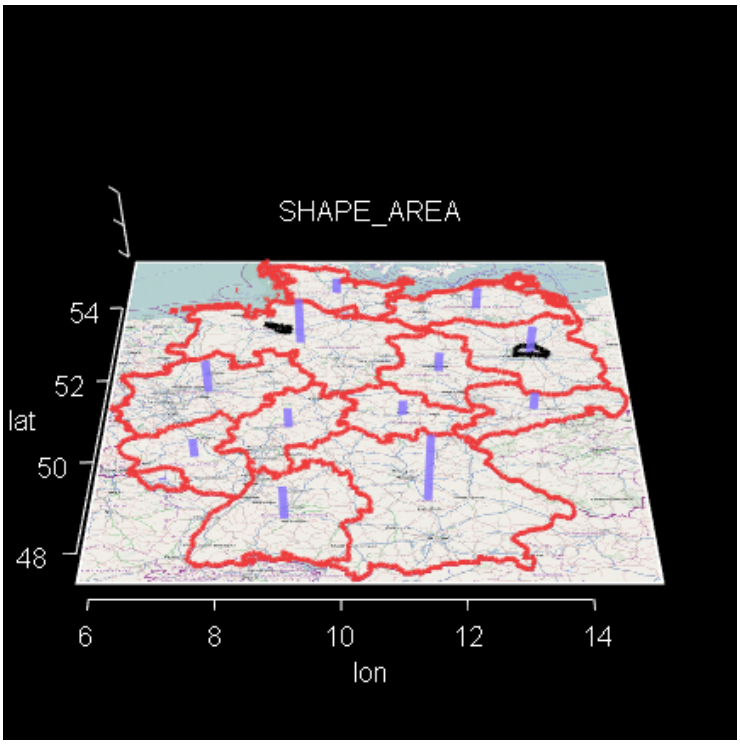

Figure 5: Spatial polygons with attributes

4.2.4 Grids

Example plot of a `SpatialGridDataFrame`.

```
> data("meuse.grid")
> coordinates(meuse.grid) <- ~x+y
> gridded(meuse.grid) <- TRUE
> proj4string(meuse.grid) <- crs
> spplot3d(meuse.grid, att = "ffreq", col = c("red", "blue", "green"),
+          CRS = CRS("+init=EPSG:4326"))
```

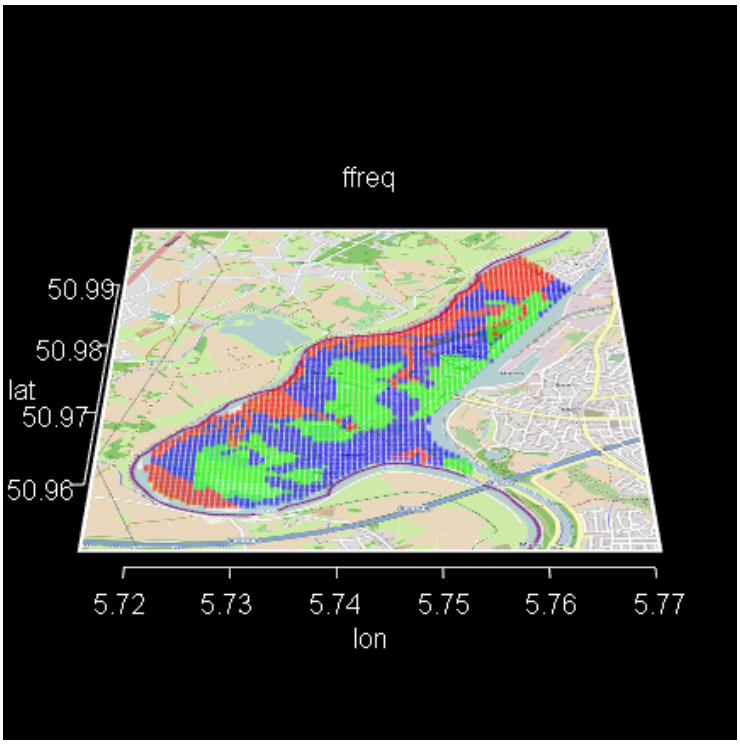

Figure 6: Spatial grid with attributes and CRS transformation

4.3 ADD - functionality

The `ADD` functionality allows to add spatial objects to an existing plot. To do this you have to specify the `CRS` used in the plot you want to add objects to. Following example shows this procedure.

```
> data("meuse.riv")
> meuse.riv <- SpatialPolygons(list(Polygons(list(Polygon(meuse.riv)),"meuse.riv")))
> proj4string(meuse.riv) <- crs
> ## First plot, ADD = FALSE
> spplot3d(meuse.riv, col = "blue", CRS = CRS("+init=EPSG:4326"))
> ## Add SpatialPoints and SpatialGrid
> spplot3d(meuse, "zinc", ADD = TRUE, CRS = CRS("+init=EPSG:4326"))
> spplot3d(meuse.grid, "dist", col=c("black", "white"), ADD = TRUE,
+          CRS = CRS("+init=EPSG:4326"))
```

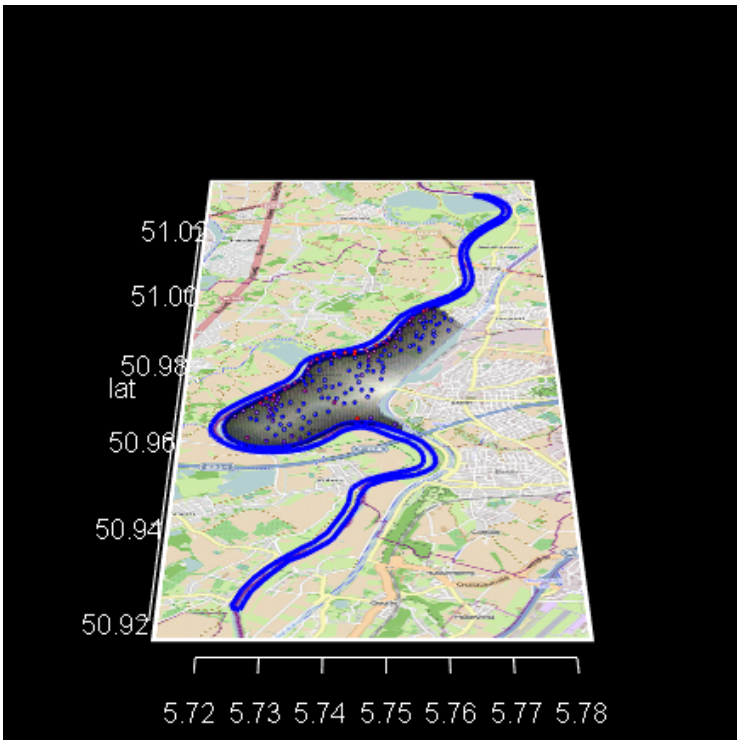

Figure 7: Polygon, points and grid in one plot.