Interactive Web Maps (plotKML, plotGoogleMaps, leaflet)

- *Edited by*: T. Hengl
- Part of: **ISRIC Spring School**

Download this tutorial as **R script**. Even more tutorials with mapview and shiny packages can be found **here** (kindly prepared and shared by Pierre Roudier).

There are several packages now in R that can be used for producing Interactive Web Maps to visualize e.g. spatial predictions (i.e. without a need for a desktop GIS software or similar). In this tutorial we will focus on comparing the following three packages that can be used to share spatial predictions also to non-GIS specialists:

- plotKML package (plots maps via KML files that can be opened in Google Earth),
- plotGoogleMaps package (plots maps via HTML file that you can open in a web-browser),
- leaflet package (plots maps inside RStudio / as HTML files that you can open in a web-browser).

Assuming that your dataset is not very large (e.g. «100,000 pixels and «10,000 points), you can quickly visualize spatial predictions generated using points with the help of the three packages listed above. Consider for example the Geul data set (the floodplain of the Geul river valley, located in the south of the Netherlands, is strongly polluted by heavy metals). This data we can use to generate spatial predictions of Pb concentrations. First we load all packages of interest:

```
> library(GSIF)
> library(plotKML)
> library(RCurl)
> library(sp)
> library(rgdal)
> library(randomForest)
> library(ranger)
> library(raster)
> library(quantregForest)
> library(plotGoogleMaps)
```

```
Loading required package: spacetime
```

```
> library(leaflet)
> nl.rd <- getURL("http://spatialreference.org/ref/sr-org/6781/proj4/")
```

We can load the points and rasters from:

```
> geul <- read.table("geul.dat", header = TRUE)
> coordinates(geul) <- ~x+y
> proj4string(geul) <- CRS(nl.rd)
> grd25 <- readGDAL("dem25.txt")
```

```
dem25.txt has GDAL driver AAIGrid
```

```
and has 52 rows and 44 columns
```

```
> names(grd25) = "dem"
> grd25$dis <- readGDAL("riverdist.txt")$band1
```

```
riverdist.txt has GDAL driver AAIGrid
and has 52 rows and 44 columns
```

```
> grd25 <- as(grd25, "SpatialPixelsDataFrame")
> proj4string(grd25) <- CRS(nl.rd)
```

and extend the available covariates by using e.g. the SAGA GIS DEM analysis toolbox:

```
> saga_cmd = shortPathName("C:/SAGA-GIS/saga_cmd.exe")
> writeGDAL(grd25["dem"], "dem.sdat", "SAGA", mvFlag=-99999)
> system(paste0(saga_cmd, ' ta_hydrology 15 -DEM \"dem.sgrd\" -TWI
\"swi.sgrd\"'))
> grd25$swi <- readGDAL("swi.sdat")$band1[grd25@grid.index]
```

so that finally we can map that variable using the GSIF package fit and predict functions:

```
> grd25.spc2 <- spc(grd25, ~ dem+dis+swi)
```

```
Converting covariates to principal components...
```

```
> m2 = pb ~ PC1+PC2+PC3
> pbm2 <- fit.gstatModel(m2, observations=geul, grd25.spc2@predicted,
method="quantregForest")
```

```
Fitting a Quantile Regression Forest model...
Fitting a 2D variogram...
Saving an object of class 'gstatModel'...
```

```
> pb.rk2 <- predict(pbm2, grd25.spc2@predicted)
```

```
Prediction error for 'randomForest' model estimated using the 'quantreg'
package.
Generating predictions using the trend model (RK method)...
Creating an object of class "SpatialPredictions"
```

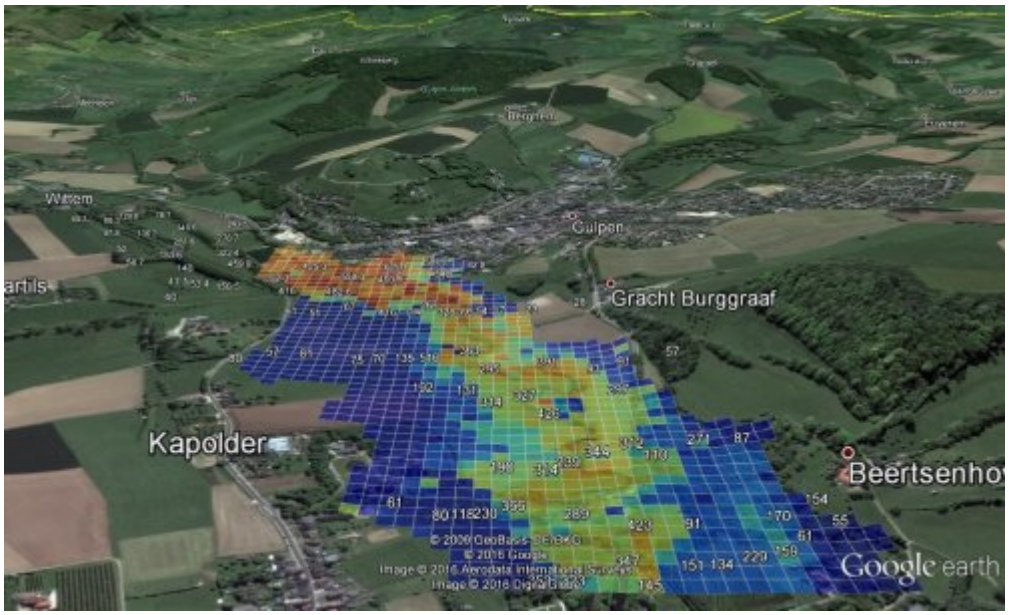
Next, we can visualize the predictions in an interactive environment.

plotKML package

To plot predictions and sampled points in Google Earth we can use the plotKML package:

```
> pb.pol <- grid2poly(pb.rk2@predicted["pb"])
> kml(pb.pol, file.name="pb_predicted.kml", colour=pb, colour_scale =
SAGA_pal[[1]], kmz=TRUE)
```

```
> kml(geul, file.name="pb_points.kml", colour=pb, labels=geul$pb)
> kml_View("pb_points.kml")
> kml_View("pb_predicted.kml")
```



Note that `grid2poly` operation is not recommended for large grids (»10,000 pixels), although using polygons for pixels is possibly the most effective way to show individual predictions in Google Earth (Murray, 2013). For more information about plotKML please refer to Hengl et al. (2015).

plotGoogleMaps package

Somewhat easier-to-use enviroment for visualization of spatial predictions is your web-browser. To visualize raster and point data in a web-browser, you can use the plotGoogleMaps package (Kilibarda and Bajat, 2012), that basically generates an HTML file with all data specified internally. This can be done by running:

```
> mp <- plotGoogleMaps(pb.rk2@predicted, filename='geul.html', zcol='pb',
add=TRUE, colPalette=SAGA_pal[[1]])
> ms <- plotGoogleMaps(geul, filename='geul.html', zcol='pb', add=FALSE,
previousMap=mp)
```

which will automatically open your default browser and show the following:

Note that you can now share that file with your colleagues and they should see exactly the same web-map in their browsers.

leaflet package

Leaflet package is another comparable option for visualizing spatial predictions. It requires, however, that the rasters are prepared via the raster package, and that the palette is attached to the values via the `colorNumeric` function:

```
> r = raster(pb.rk2@predicted["pb"])
> pal <- colorNumeric(SAGA_pal[[1]], values(r), na.color = "transparent")
> leaflet() %>% addTiles() %>%
  addRasterImage(r, colors=pal, opacity=0.6) %>%
  addLegend(pal=pal, values=values(r), title="Pb concentration")
```

To plot only points we would use:

```
> geul.xy <- spTransform(geul, CRS("+proj=longlat +datum=WGS84"))
> leaflet(geul.xy) %>% addTiles() %>%
  addCircleMarkers(radius=geul.xy$pb/50, color="red", stroke=FALSE,
fillOpacity=0.8)
```

Leaflet package can also be customized so that multiple layers can be added above each other. For a more detailed tutorial refer to the leaflet package homepage.

Summary points

In summary, there are multiple ways to plot your predictions in Google Earth, by using Google Maps or OpenStreetMap. For those familiar with basic GIS operations, Google Earth is definitively the best option, especially if you aim at exploring the data in a 3D environment. If you would prefer to use a browser i.e. have the complete data embedded inside an HTML file, then plotGoogleMaps and leaflet are better options. From the three packages, plotGoogleMaps seems to be the easiest to use package for generating combined plots.

Caution: none of the package listed above is really suited for very large data. To visualize large GIS layers you should really look at using Geoserver, OpenLayers and/or Cesium, but this would then require that you set-up and install a server with all necessary software and web-facilities. Some examples of Interactive Web Maps based on OpenLayers you can find here.

References

- Hengl, T., Roudier, P., Beaudette, D. and Pebesma, E., 2015. **plotKML: scientific visualization of spatio-temporal data**. Journal of Statistical Software, 63(5).
- Kilibarda, M. and Bajat, B., 2012. **plotGoogleMaps: The R-based web-mapping tool for thematic spatial data**. Geomatica, 66(1), pp.37-49.
- Murray, S., 2013. **Interactive data visualization for the Web**. O'Reilly Media, Inc.
