A plotGoogleMaps tutorial

Milan Kilibarda
University of Belgrade, Faculty of Civil Engineering,
Department of Geodesy and Geoinformatics,
Bulevar kralja Aleksandra 73, 11000 Belgrade, Serbia
kili@grf.bg.ac.rs

April 3, 2013

Contents

1 Introduction

The plotGoogleMaps package provides an interactive plot device for handling the geographic data within web browsers. It is optimized for Google Chrome browser. It is designed for the automatic creation of web maps as a combination of users' data and Google Maps layers. The input data are in form of Spatial-class with associated coordinate reference system. The classes and methods for spatial data and its manipulation is described in book Applied Spatial Data Analysis with R (Bivand et al, 2008). Versions of package higher of 2.0 support plot of spacetime data (STDIF and STFDF) described by Pebesma (2012).

The plotGoogleMaps is based on Google Maps API [1]. Google Maps API is set of predefined JavaScript classes ready to be implemented in any web page, with aim of creation interactive web map Google map mashup. It is possible to create map mashup even if creator is not an expert in web programming, although the basic knowledge in JavaScript programming language, XML, Ajax and XHTML is required.

The plotGoogleMaps enables creation of interactive web map, with the base map supplied by Google, where all map elements and additional functionalities are handled by just one R command from the package. The package provides solution to create and visualize vector and raster data, proportional symbols, pie charts, ellipses and spacetime data. The web map map mashup created by plotGoogleMaps package could be used as a temporary result of spatial visualization generated on the local machine or, published on any web page.

The plotGoogleMaps uses web browser as plotting device instead of default R graphic device. Therefore, it offers more advantages related to R classical plotting device environment; high quality of background Google layers which make better abstraction of geographical reality, spatial data exploration functionality, and map interactivity (navigation control, pan, zoom, attribute info windows, etc). Google Maps API is not suitable for the large data and consequently plotGoogleMaps has the same constrain Kilibarda and Bajat (2012). If you want to cite package in publication cite Kilibarda and Bajat (2012).

This vignette describes functions provided by plotGoogleMaps.

Package plotGoogleMaps is loaded by:

```
library(plotGoogleMaps)
```

[1] `https://developers.google.com/maps/`

2 Plot spatial points

In the following example, it is shown plot of SpatialPointsDataFrame objects of meuse data set. This data set gives locations and topsoil heavy metal concentrations of 155 observations together with soil properties and distance to the river, collected in a flood plain of the river Meuse, in the area around Meers and Maasband (Limburg, the Netherlands) during a fieldwork in the year 1990.

```
# Data preparation
# Point data
data(meuse)
coordinates(meuse)<-~x+y # convert to SPDF
proj4string(meuse) <- CRS('+init=epsg:28992')
# adding Coordinate Referent Sys.
# Create web map of Point data
m<-plotGoogleMaps(meuse,filename='myMap1.htm')
```

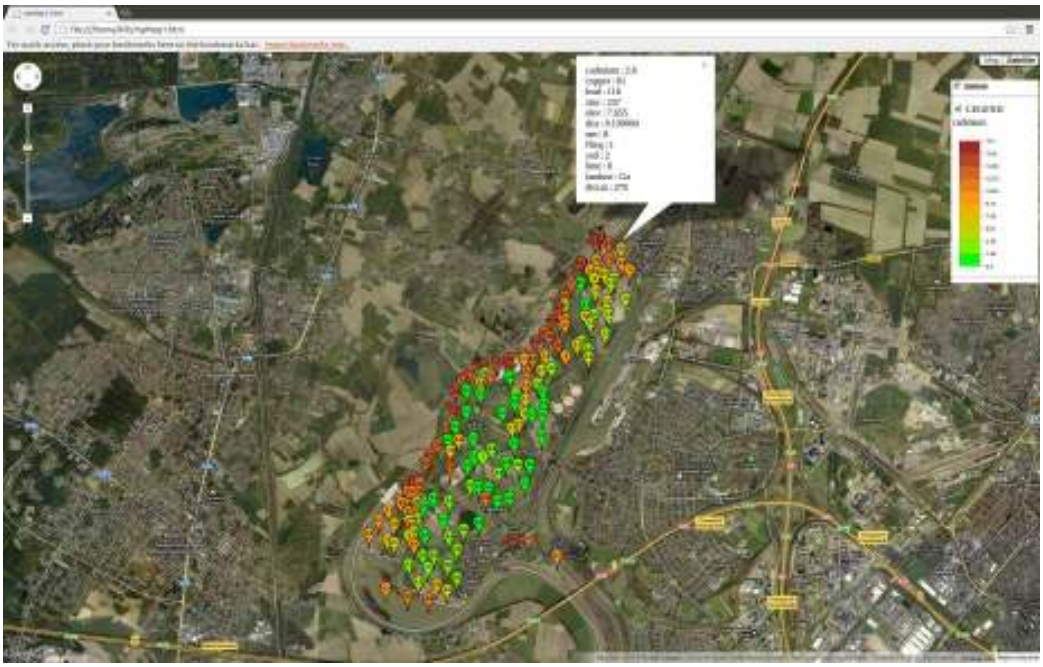

Figure 1: Plot SpatialPointsDataFrame object; meuse data.

The first created map is named myMap1.htm , and it is map of meuse data, mashup with positions of points and attribute data ready to be explored in browser (Fig. 1).

In the next example some additional setting for the plotGoogleMaps is presented. The plotting argument is zinc and icons marker is replaced by labels of zinc. The base map is changed, layer name, and plotting argument is defined (Fig. 2).

```
# zinc labels
ic=iconlabels(meuse$zinc, height=12)
m<-plotGoogleMaps(meuse,zcol='zinc',filename='myMap_z2.htm',
  iconMarker=ic, mapTypeId='ROADMAP',layerName = 'MEUSE POINTS')
```

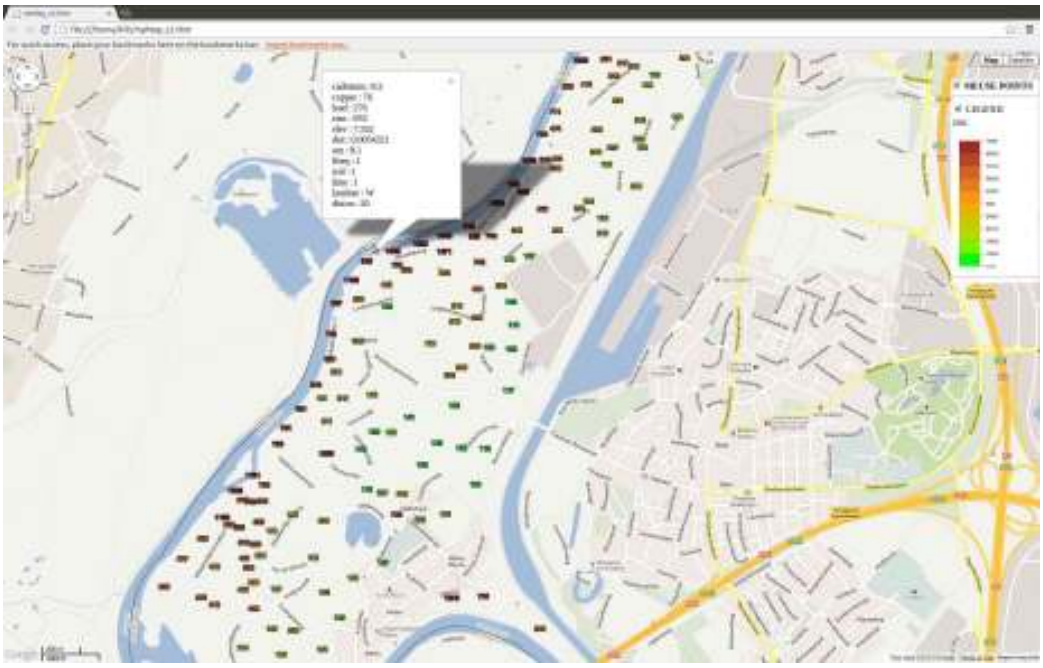

Figure 2: Plot SpatialPointsDataFrame object; meuse data; with additional settings.

The sampled zinc concentration can be plot with proportional symbols and in different colors related to measured concentration. Maximum radius related to maximum concentration is specified in meters (Fig. 3).

.

```
m<-bubbleGoogleMaps(meuse,zcol='zinc',max.radius = 80,filename='myMap3.htm')
# remove outlines
m<-bubbleGoogleMaps(meuse,zcol='zinc',max.radius = 80,
  filename='myMap3.htm',strokeOpacity=0)
```

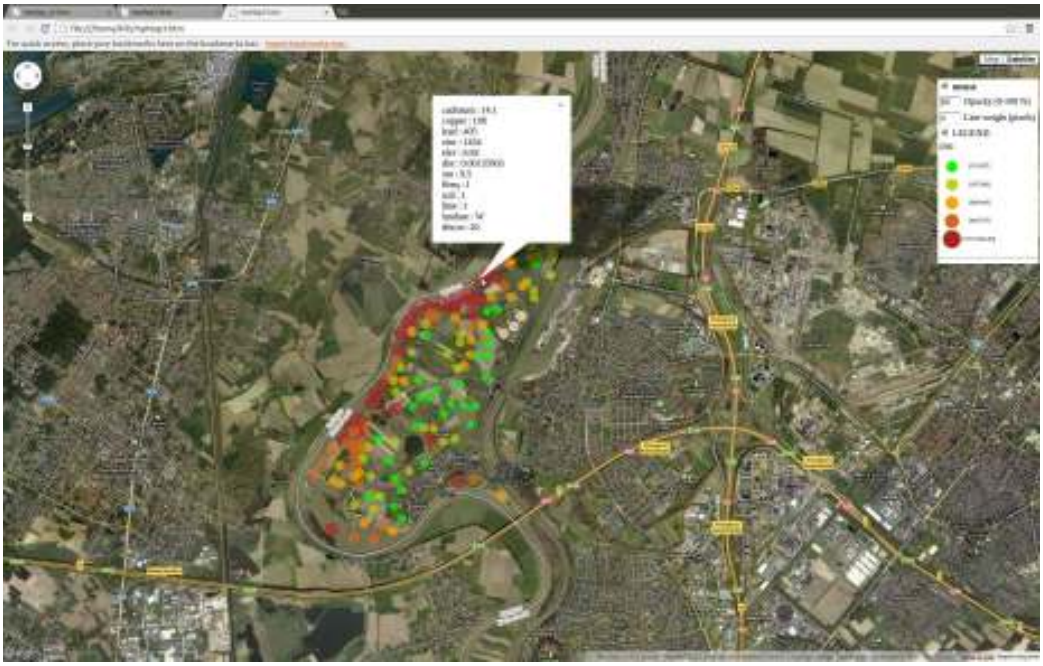

Figure 3: Plot SpatialPointsDataFrame object; meuse data; bubbleGoogleMaps.

The function segmentGoogleMaps produces maps for multivariate mapping. The segmentGoogleMaps creates pie charts or more properly called segmented circles. Pie charts are circles with wedges representing the variables, which are related in some way. In ths example it is presented multivariate plot of heavy metal concentrations from meuse sampling points. Maybe it should be more properly to present some variables that are more related.

```
# colPalette defines colors for plot
m<-segmentGoogleMaps(meuse, zcol=c('zinc','dist.m'),
  mapTypeId='ROADMAP', filename='myMap4.htm',
  colPalette=c('#E41A1C','#377EB8'), strokeColor='black')
```

Fig 4 shows correlation between zinc concentration and distance to river.

Plotting uncertainty of position is provided by ellipseGoogleMaps function. The ellipseGoogleMaps plots standard errors of the computed coordinates, error ellipses describing the uncertainty of a two-dimensional position. Parameters of input spatial points data frame should contain at least three columns: semi-major axis, semi-minor axis, and orientation in degrees. These parameters are product of geodetic least square adjustment or design of a geodetic control network.

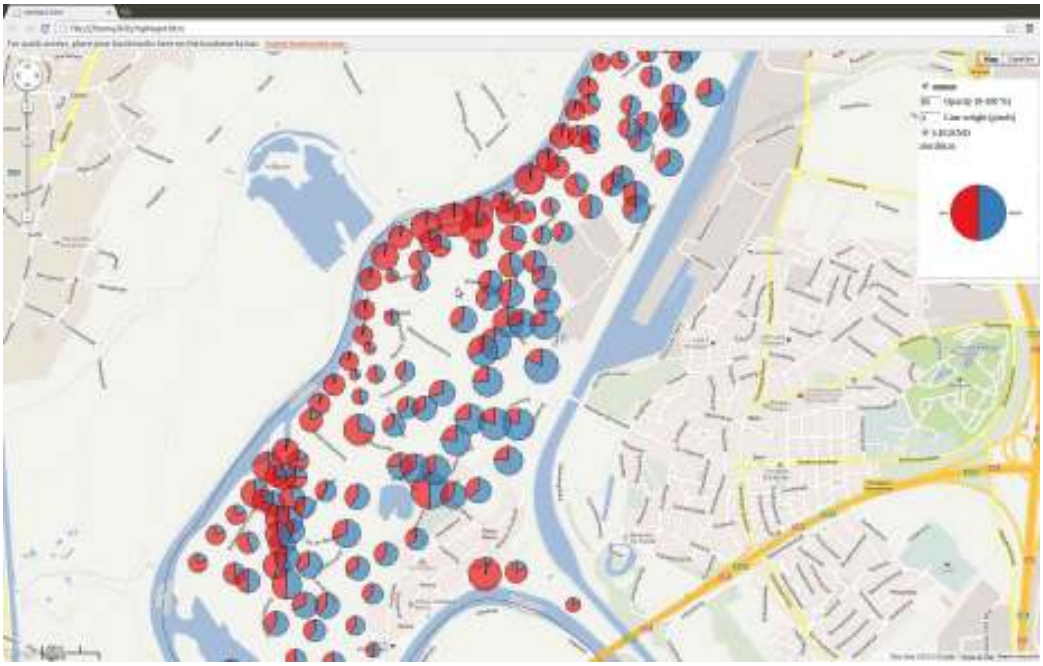

Figure 4: Plot SpatialPointsDataFrame in form of pie charts plot; zinc concentration and distance to river.

In the next example is shown results from geodetic network design results (Fig 5).

```
# Results of least square
ell<- data.frame(E=c(7456263,7456489,7456305,7457415,7457688),
  N=c(4954146 ,4952978, 4952695, 4953038, 4952943),
  Name=c('30T', '31T', '3N', '40T', '41T'),
  A=c(2.960863 ,4.559694, 7.100088, 2.041084 ,3.375919),
  B=c(2.351917, 2.109060, 2.293085, 1.072506, 2.382449),
  teta=c(28.35242, 41.04491, 38.47216, 344.73686, 27.53695))
coordinates(ell) <- ~E+N
proj4string(ell) <- CRS("+proj=tmerc +lat_0=0 +lon_0=21 +k=0.9999
+x_0=7500000 +y_0=0 +ellps=bessel
+towgs84=574.027,170.175,401.545,4.88786,-0.66524,-
13.24673,0.99999311067 +units=m")
# fillOpacity 100 %
m<-ellipseGoogleMaps( ell,filename="Ellipse.htm",zcol=2:4,
  mapTypeId='ROADMAP',fillOpacity=1,strokeOpacity=0)
```

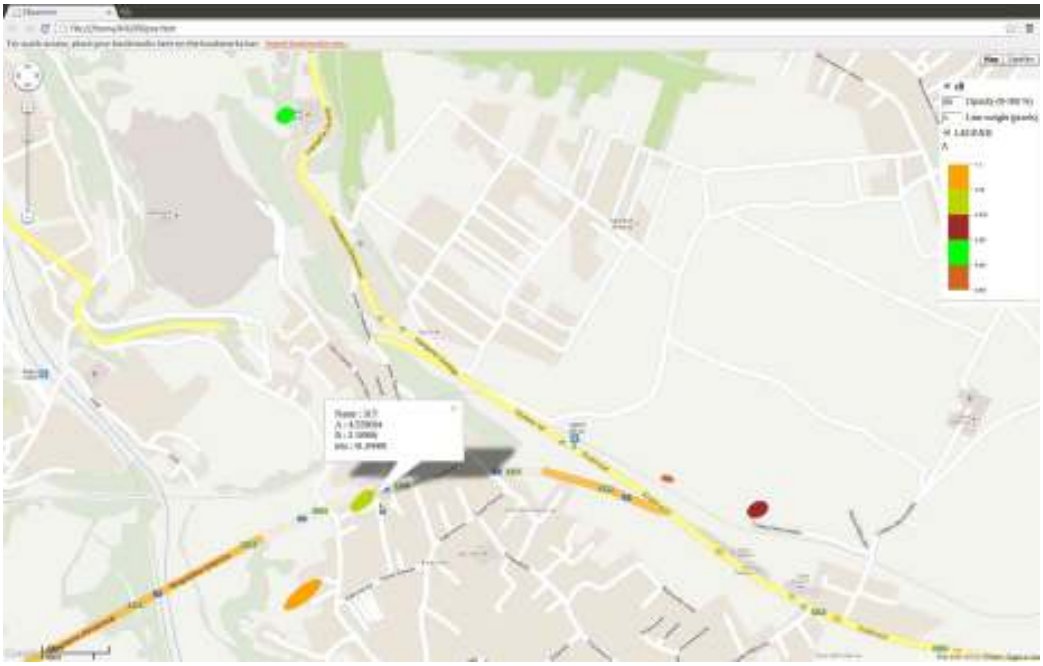

Figure 5: Plot SpatialPointsDataFrame; error ellipses.

3 Plot spatial lines

The plotGoogleMaps produces plot of SpatialLinesDataFrame similary like plotting SpatialPointsDataFrames. In the next example coloring is used by default and border width is set related to line attribute (Fig 6). The lines used in this case representing distance to Meuse River.

```
# Line data
data(meuse.grid)
coordinates(meuse.grid)<-c('x','y')
meuse.grid<-as(meuse.grid,'SpatialPixelsDataFrame')
im<-as.image.SpatialGridDataFrame(meuse.grid['dist'])
cl<-ContourLines2SLDF(contourLines(im))
proj4string(cl) <- CRS('+init=epsg:28992')
mapMeuseCl<- plotGoogleMaps(cl,zcol='level',strokeWeight=1:9 ,
  filename='myMap5.htm',mapTypeId='ROADMAP')
```

The strokeWeight argument defines line width corresponding to line attribute level, distance to river.

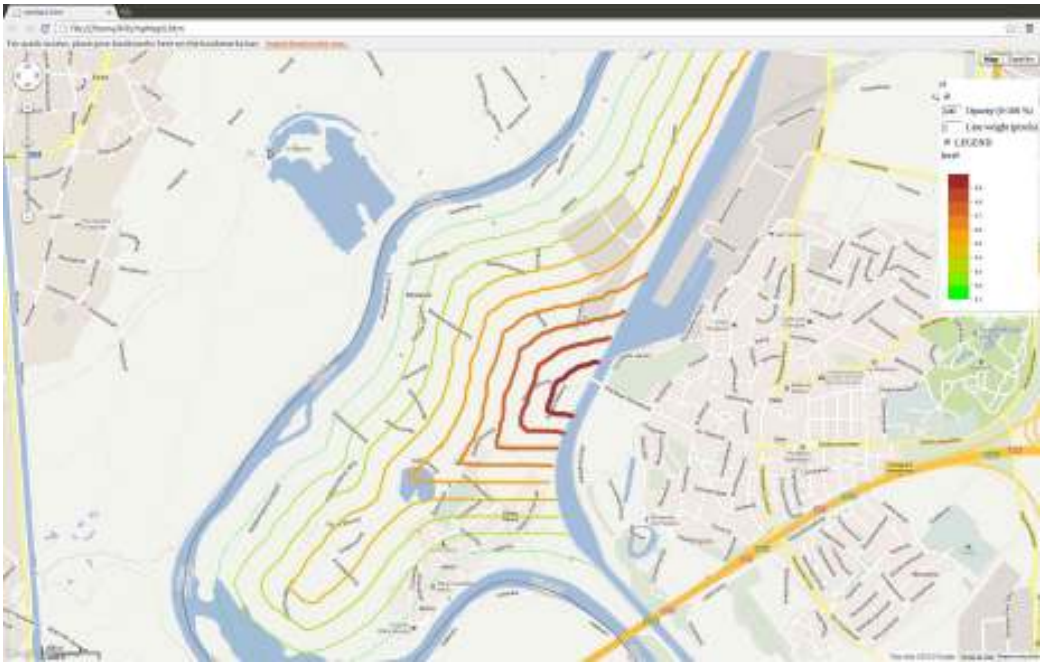

Figure 6: Plot SpatialLinesDataFrame.

4 Plot spatial polygons

For plotting spatial polygon data frame is used shapefile provided by maptools package. This is for the 100 counties of North Carolina, and includes counts of numbers of live births (also non white live births) and numbers of sudden infant deaths, for the July 1, 1974 to June 30, 1978 and July 1, 1979 to June 30, 1984 periods (Bivand, 2011).

For the colour coding RColorBrewer package is used.

Next command plots nc data with colour sheme obtained from RColorBrewer for the polygons and white border is set to county border (Fig 7). The color scheme relates to plotting attribute named BIR74.

```
nc <- readShapeSpatial( system.file("shapes/sids.shp",package="maptools")[1],
  proj4string=CRS("+proj=longlat +datum=NAD27"))
library(RColorBrewer)
m<-plotGoogleMaps(nc,zcol="NWBIR74",filename='MyMap6.htm',
  mapTypeId='TERRAIN',colPalette= brewer.pal(7,"Reds"),
  strokeColor="white")
```

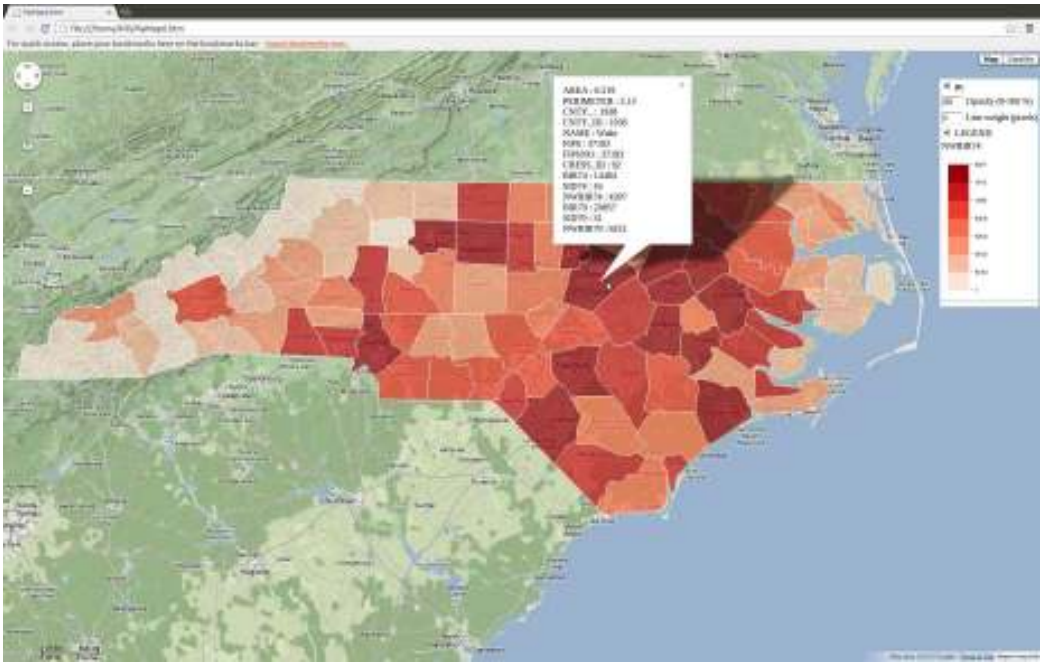

Figure 7: Plot SpatialPolygonsDataFrame.

5 Plot SpatialPixelsDataFrame

In the next example is shown plot SpatialPixelsDataFrame (Fig 8).

```
data(meuse.grid)
coordinates(meuse.grid)<-c('x','y')
meuse.grid<-as(meuse.grid,'SpatialPixelsDataFrame')
proj4string(meuse.grid) <- CRS('+init=epsg:28992')
m=plotGoogleMaps(meuse.grid,zcol='dist',
  at=seq(0,0.9,0.1),colPalette= brewer.pal(9,"Reds"))
```

6 Plot several layers

A map becomes more readable when is combined several layers. The plotGoogleMaps functions could be use to create map mashup with several layers, the function should contain argument add=TRUE. The next plot should have the name of previous map the argument previousMap =*name of saved map produced by functions from plotGoogleMaps package* (Fig 9).

Figure 8: Plot SpatialPixelsDataFrame.

```
m1<- plotGoogleMaps(cl,zcol='level',
  strokeWeight=1:9 ,
  colPalette='grey',
  add=TRUE)
m2<-bubbleGoogleMaps(meuse,zcol='zinc',
  colPalette= brewer.pal(5,"Accent"),
  max.radius = 80,
  previousMap= m1,
  filename='comb.html')
```

7 Plot STFDF

Produc data used for this plot is described in spacetime vignette (Pebesma, 2012), example is from vignette (Fig 10).

```
#STFDF  data from spacetime vignette
# spacetime: Spatio-Temporal Data in R
library("maps")
states.m = map('state',plot=FALSE,fill=TRUE)
```

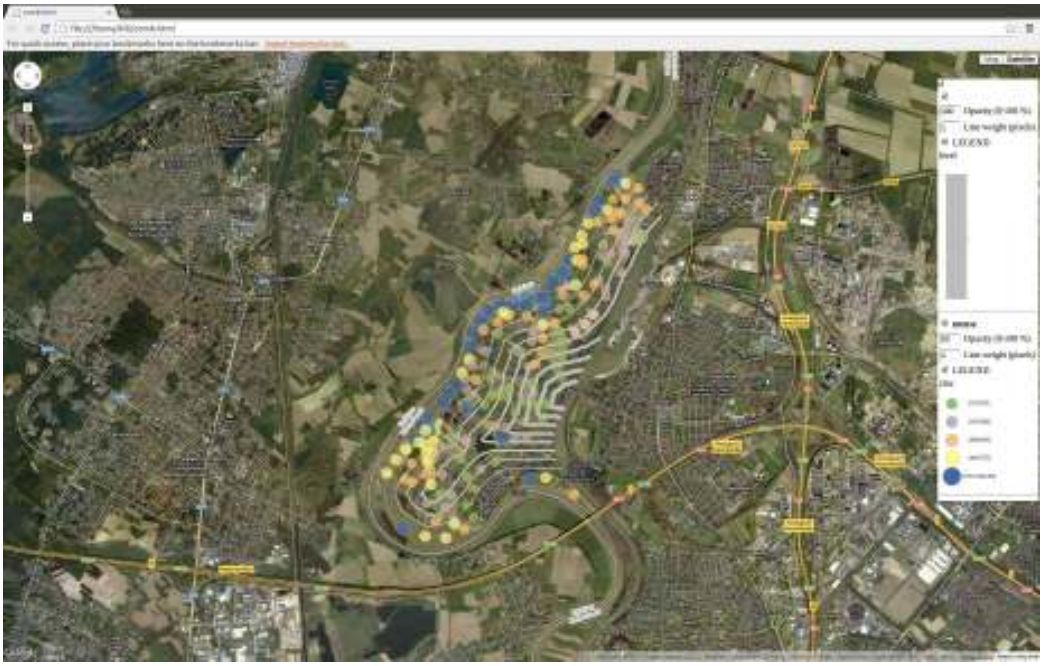

Figure 9: Plot multiple layers.

```
IDs <- sapply(strsplit(states.m$names, ":"),function(x) x[1])
library("maptools")
states = map2SpatialPolygons(states.m,IDs=IDs)
yrs = 1970:1986
time = as.POSIXct(paste(yrs, "-01-01", sep=""),tz = "GMT")
library("plm")
data("Produc")
Produc.st = STFDF(states[-8], time,
  Produc[order(Produc[2], Produc[1]),])
Produc.st@sp@proj4string=CRS('+proj=longlat +datum=WGS84')
library(RColorBrewer)
ee= stplotGoogleMaps(Produc.st,zcol='unemp',
  stfilename='USA.htm',colPalette=brewer.pal(9, "YlOrRd"),
  mapTypeId='ROADMAP',w='49%',h='49%', fillOpacity=0.85)
 # without control
ee= stplotGoogleMaps(Produc.st,zcol='unemp',
  stfilename='USA2.htm',colPalette=brewer.pal(9, "YlOrRd"),
  mapTypeId='ROADMAP',w='33%',h='25%',
  fillOpacity=0.85, control.width=0)
```

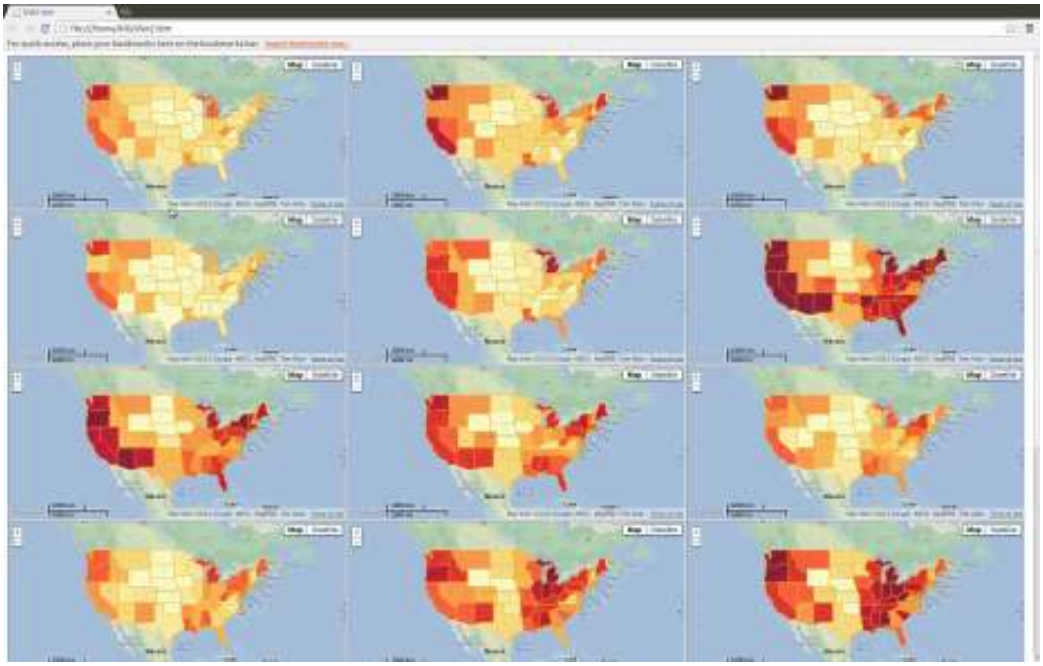

Figure 10: Plot STFDF.

8 Plot STIDF

Time series of measurements at meteorological stations data is used for STIDF plot (Fig 11). Data set is obtained from plotKML package, description of data and package is available on package tutorial page [2].

```
[samepage=true]

# Data preparation
# Point data
# data from plotKML package and plotKML tutorial
library(plotKML)
data(HRtemp08)
HRtemp08$ctime <- as.POSIXct(HRtemp08$DATE,format="%Y-%m-%dT%H:%M:%SZ")
library(spacetime)
sp <- SpatialPoints(HRtemp08[,c("Lon","Lat")])
proj4string(sp) <- CRS("+proj=longlat +datum=WGS84")
HRtemp08.st <- STIDF(sp, time = HRtemp08$ctime,
   data = HRtemp08[,c("NAME","TEMP")])
   HRtemp08_jan <- HRtemp08.st[1:500]
```

[2] http://gsif.isric.org/doku.php?id=wiki:tutorial_plotkml

```
#str(HRtemp08_jan)
# plot STDIF
stplotGoogleMaps(HRtemp08_jan,zcol='TEMP',
  mapTypeId='ROADMAP',w='49%',h='49%')
# plot STDIF bubble
stplotGoogleMaps(HRtemp08_jan,zcol='TEMP',
   stfilename='HR_temp.html',
   mapTypeId='ROADMAP',w='49%',h='49%',
   strokeOpacity = 0,
  do.bubble=T, bubble= list(max.radius=15000,
  key.entries =quantile(HRtemp08_jan@data[,'TEMP'],(1:5)/5, na.rm=T),
  do.sqrt = F) )
```

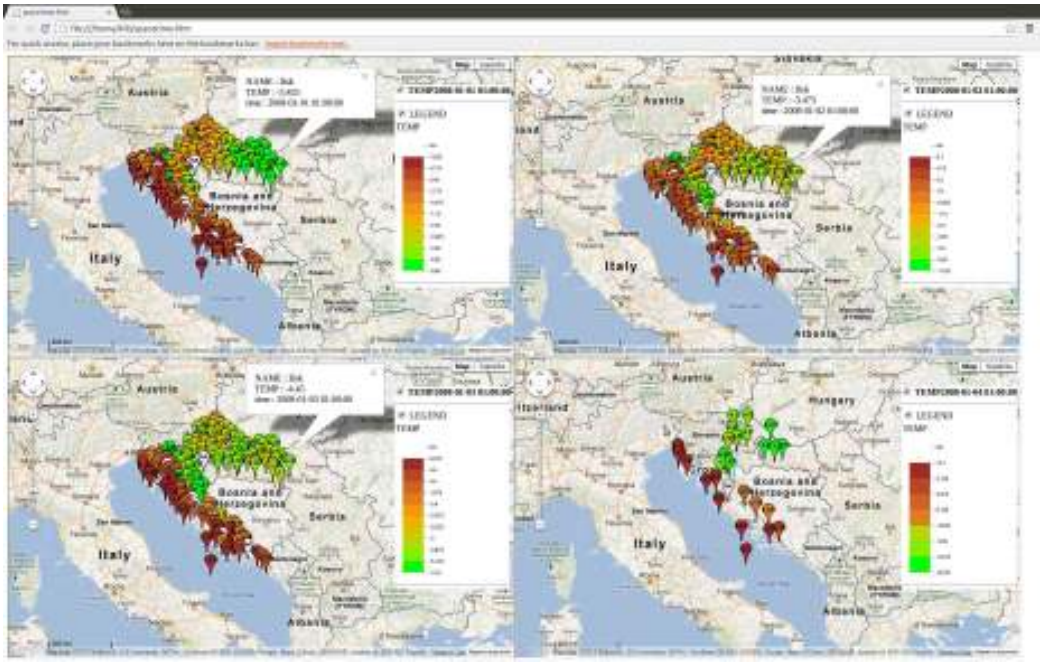

Figure 11: Plot STIDF.

References

Bivand R (2011) Introduction to the north carolina sids data set (revised). Tech. rep., URL http://cran.rproject.org/web/packages/spdep/vignettes/sids.pdf

Bivand R, Pebesma E, Rubio V (2008) Applied spatial data analysis with R. Use R Series, Springer, Heidelberg

Kilibarda M, Bajat B (2012) plotgooglemaps: The r-based web-mapping tool for thematic spatial data. GEOMATICA 66:37–49, DOI 10.5623/cig2012-007

Pebesma E (2012) spacetime: Spatio-temporal data in r. Journal of Statistical Software 51(7), URL http://www.jstatsoft.org/v51/i07