*Journal of Statistical Software*

*February 2015, Volume 63, Issue 5.* *doi: 10.18637/jss.v000.i00*

# plotKML: Scientific Visualization of Spatio-Temporal Data

**Tomislav Hengl**
ISRIC — World Soil Information
the Netherlands

**Pierre Roudier**
Landcare Research
New Zealand

**Dylan Beaudette**
USDA-NRCS, Soil Science Division
USA

**Edzer Pebesma**
Institute for Geoinformatics
University of Münster, Germany

**Abstract**

**plotKML** is an R package that provides methods for writing the most common R spatial classes into KML files. It builds up on the existing XML parsing functionality (**XML** package), and provides similar plotting functionality as the **lattice** package. Its main objective is to provide a simple interface to generate KML files with a small number of arguments, and allows users to visually explore spatio-temporal data available in R: points, polygons, gridded maps, trajectory-type data, vertical profiles, ground photographs, time series vector objects or raster images, along with the results of spatial analysis such as geostatistical mapping, spatial simulations of vector and gridded objects, optimized sampling designs, species distribution models and similar. A generic `plotKML()` function automatically determines the parsing order and visualizes data directly from R; lower level functions can be combined to allow for new user-created visualization templates. In comparison to other KML writing packages, **plotKML** seems to be more object oriented, it links more closely to the existing R classes for spatio-temporal data (**sp**, **spacetime** and **raster** packages) than the alternatives, and provides users with the possibility to create their own templates.

*Keywords*: space-time objects, scientific visualization, R, KML, geostatistics.

## 1. Introduction

Keyhole Markup Language (KML) is an XML language focused on geographic visualization,

including annotation of maps and images (Wilson 2008; Wernecke 2009). It is the standard adopted by Google and now used by Google Earth™ as its primary exchange format. There is a growing interest in using Google Earth to visualize spatio-temporal data produced in the R environment for statistical computing. We believe that the main reasons responsible for this growth are:

1. Google Earth is one of the most wide-spread geographical data browsers available with >1 billion downloads as of October 2011[1]. It is a largely intuitive software, easily accessible to people without any professional GIS training, and most importantly, a framework which can be used to increase a project reach to the general public (Butler 2006; Patterson 2007; Goodchild 2008).

2. According to McGavra *et al.* (2009), KML is one of the leading Open Geospatial Consortium (OGC) XML encoding standards.

3. Google Earth provides access to high-resolution remote sensing data (Ikonos, GeoEye and Cnes/Spot images, administrative vector data, SRTM DEM and similar), which allows users to qualitatively interpret the analysis results by matching the produced outputs with background imagery (Craglia *et al.* 2008; Fritz *et al.* 2012).

4. As KML provides a diversity of visualization options, it is one of the more attractive platforms for scientific visualization of geographical phenomena — it can enable scientists to detect patterns in their data not visible in other software, but it can also help engaging a wider public into scientific research and decision making (Butler 2006). Goodchild (2008) refers to this as the *'citizen science'*.

Although R is primarily known as a statistical computing environment (R Development Core Team 2013), it is of increasing interest to the field of geoinformatics and applied spatial data analysis due to its extensibility and the growing diversity of spatial and spatio-temporal data structures (Pebesma and Bivand 2005; Pebesma 2012b; Bivand *et al.* 2013). Geo-visualization functions in R, on the other hand, are limited because R was not originally designed for interactive exploration of spatial data or as a GIS platform. Although there are many packages in R already that allow interactive visualization, overlay and animated display of geographical phenomena (via packages **RGGOBI** and **iPlots**; for a review refer to (Cook and Swayne 2007) and/or (Theus and Urbanek 2009)), R has its limitations for interactively exploring geographical data.

There is considerable potential for building connectivity between the sophisticated spatial analysis possible in R and the geo-visualization capacities afforded by Google Earth. However, the export of spatial data from R to KML is not trivial. KML files can be produced using the using GDAL (Geospatial Data Abstraction Library) KML driver, but so far only limited functionality is supported. As the driver description page indicates: *"At this time, only vector layers are handled by the KML driver... limited support is available for fills, line color and other styling attributes."*[2]

Within the R community, other packages produced specifically to allow creation of KML files include: **rgdal** (Keitt *et al.* 2013) and **raster** (Hijmans and van Etten 2012), while truly

[1] `http://www.google.com/earth/connect/newsletter/oct11.html`
[2] `http://www.gdal.org/ogr/drv_kml.html`

specialized KML writing packages include **R2G2** (Arrigo *et al.* 2012) and **RKML**[3]. Our objective with the **plotKML** package was to provide a simple interface to generate KML files with a small number of arguments, and allow users to directly plot spatio-temporal data classes, available in R via the **sp** (Pebesma and Bivand 2005), **spacetime** (Pebesma 2012a), **aqp** (Beaudette and Roudier 2012) and **raster** (Hijmans and van Etten 2012) packages, in a virtual globe type of browser.

This article describes the main functionality and the design of the **plotKML** package and provides a number of examples of how to produce some common KML visualization templates. Some limitations of the package in comparison with other alternative KML writing software and possibilities to improve the package are given in the discussion section. More detailed tutorials including YouTube videos can be found via the package homepage[4].

## 2. Basic concepts

### 2.1. Scientific visualization of geographic phenomena

The purpose of scientific visualization is to *"graphically illustrate scientific data to enable scientists to understand, illustrate, and glean insight from their data."*[5] According to Friendly and Denis (2001), scientific visualization can be primarily connected with visualization of 3D data *"where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component."* In that context, the main purpose of **plotKML** package is to enable interactive scientific visualization of geographic phenomena i.e. scientific geovisualization (Dykes *et al.* 2005).

Interactive geovisualization, i.e. dynamic user-controlled visualization of geographical phenomena, can be today closely connected with 3D computer environments referred to as the *'Virtual globe'*, *'Virtual Earth'* or *'Digital Earth'* (Gore 1998; Bleisch 2011). At the beginning of 21st century, the most known virtual globes are (Bleisch 2011):

- Google Earth
- Microsoft Bing Maps 3D
- AutoDesk LandXplorer
- NASA World Wind
- ESRI's ArcGlobe
- Leica's Virtual Explorer

Open source alternatives to Google Earth and Microsoft Bing Maps 3D are: (KDE) Marble[6], OSSIM Planet[7], and Cesium[8].

[3] http://www.omegahat.org/RKML/
[4] http://plotkml.r-forge.r-project.org/
[5] http://en.wikipedia.org/wiki/Scientific_visualization
[6] http://marble.kde.org/
[7] http://trac.osgeo.org/ossim/
[8] http://cesium.agi.com

According to Elmqvist and Tudoreanu (2007), there are two main reasons for creating 3D virtual globes:

(a) replicate or simulate the real world,

(b) use 3D as canvas for abstract information.

The **plotKML** package offers both. As we will show later, it allows Google Earth to be used not only for visualization of maps, but also as a canvas for 3D or 3D+T visualization of any such data created and analyzed in R (see further Figures 3 and 9).

Although one can argue whether virtual globes should be used more broadly for decision making and land management, two significant developments can not be disputed: (1) ubiquitous access to free high resolution satellite imagery through Google Earth, Yahoo and Bing has revolutionized applications of GIS for both commercial and non-commercial projects, and (2) anyone is today invited to make, edit, mash-up and share geodata (Fritz *et al.* 2012).

## 2.2. Space-time continuum

Everything we measure on Earth can be linked to some space-time *'location'*:

1. *geographic location* (longitude and latitude)
2. *height above the ground surface* (altitude)
3. *time of measurement* (year, month, day, hour, minute, second)
4. *spatio-temporal support* (size of the blocks of material associated with measurements; time interval or duration of measurement).

By attaching spatio-temporal reference to measurements we can dynamically visualize them, but also run spatio-temporal data analysis (Bivand *et al.* 2008; Pebesma 2012b). Many analysts already find it useful to be able to visualize all input and derived maps or results of spatial analysis in a Virtual Earth type of environment such as Google Earth (Patterson 2007). In addition, creating a realistic visualization of observed dynamic phenomena can improve the spatial analysis process, in part, because it can help us make more thoroughly considered interpretations of analysis results (Craglia *et al.* 2008).

Figure 1 for example shows how spatio-temporal data can be visualized in a 2D+T space-time cube. The same data-set is further shown in Figure 9 visualized in Google Earth.

## 2.3. Spatial and spatio-temporal objects

For Erwig *et al.* (1999) spatio-temporal data sets and corresponding databases represent two major groups of features: (1) *moving or dynamic objects* (discrete or vector geometries), and (2) *regions or fields* (continuous). Objects and fields can be further based on *regular* or *irregular* sampling systems and representation models. Many features can be modeled and represented both using discrete (vector) and continuous (raster) GIS models. For example, objects such as a population of animals can be modeled and represented as discrete objects (trajectories or points), but also as densities (*regions*) or polygons representing home ranges.

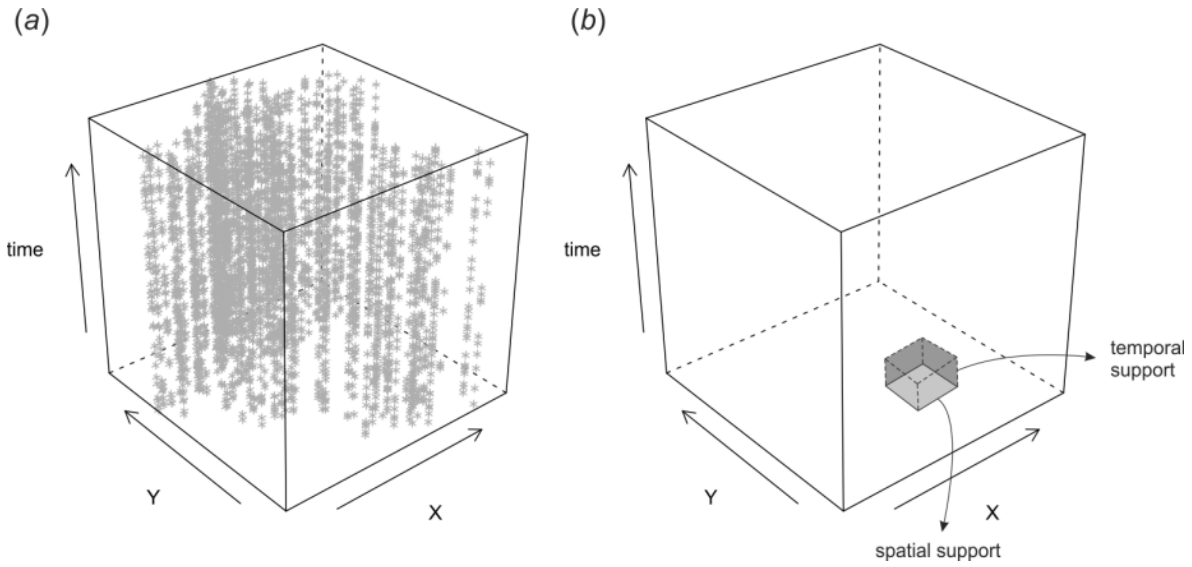


Figure 1: Space-time cube visualized in R: (a) cloud plot showing location of meteorological stations in Croatia (data set from Hengl *et al.* (2012)), (b) illustration of spatial and temporal support in the space-time cube.

Likewise, earthquakes are in their essence *regions* (of sudden and violent shaking of the ground), but are often represented as points. This is just to illustrate the complexity of choosing the right model for representing such dynamic features. For an introduction to the complexity of spatial and spatio-temporal objects and fields refer also to Galton (2004).

Bivand *et al.* (2008, pp.28–55) and Pebesma (2012b) implemented classes for spatial and spatio-temporal data in R via the **sp** and **spacetime** packages. These classes are also highly extendable and are already widely used inside the R spatial data analysis community. The **sp** package (Pebesma and Bivand 2005) is currently one of the top 10 packages with highest number of dependencies on CRAN[9], and has been used as the main starting point for building visualization functionality in **plotKML**.

A schematic overview of 2D, 3D, 2D+T and 3D+T combinations of spatio-temporal object types is given in Table 2.3. Note that not all space-time combinations of 2D/3D+T objects are yet implemented in R, and some are implemented but are still rather experimental. For example, voxels can be constructed by adding the third dimension to object of class `SpatialPixels`, but methods for such type of data are still limited.

Although KML can probably accommodate all spatio-temporal objects listed in Table 2.3, visualization of densely sampled 3D+T objects, e.g. millions of voxels, is probably still not recommended in KML. For example, by making COLLADA (COLLAborative Design Activity; an open standard XML schema) 3D objects one can potentially generate any type of 3D spatio-temporal object, but this is then highly complex and requires good knowledge of COLLADA and KML.

[9]See blog by Dirk Eddelbüttel published at `http://dirk.eddelbuettel.com` on Thursday, 16th August 2012.

| | | *Point / line* | *Regular (block)* | *Irregular (area)* |
|---|---|---|---|---|
| 2D | *Latitude, longitude* | `SpatialPoints` | `RasterLayer` `SpatialGrid` `SpatialPixels` | `SpatialPolygons` |
| 3D | *Latitude, longitude, altitude (thickness)* | `SoilProfile` | Voxels* Monoliths* | Irregular voxels* |
| 2D+T | *Latitude, longitude, time* | `STFDF` `STIDF` `STTDF` | `STFDF` `RasterBrick` | `STFDF` |
| 3D+T | *Latitude, longitude, altitude, time* | Time-series of profiles* | Time-series of voxels* | **Time-series of irregular voxels (space-time prisms)*** |

Table 1: Types of spatio-temporal objects (points and regions) based on the number of dimensions (2D, 3D, 2D+T and 3D+T) and support type and corresponding R classes. `STFDF` stands for a class for spatio-temporal data with full space-time grid, `STIDF` stands for a class for unstructured spatio-temporal or irregular data, and `STTDF` stands for a class for spatio-temporal trajectory data (Pebesma 2012b). $\star$ — classes not yet available in R.

## 2.4. Spatial data structures in R and KML

Consider for example the Google headquarters in Mountain View, CA. The point location can be prepared in R as object of class `SpatialPoints*`, as implemented in the **sp** package, which takes few steps:

```
R> library("sp")
R> lat = 37.423156
R> lon = -122.084917
R> name = "Google headquarters"
R> pnt = data.frame(name, lat, lon)
R> coordinates(pnt) <- ~lon+lat
R> proj4string(pnt) <- CRS("+proj=longlat +datum=WGS84")

R> pnt

class : SpatialPointsDataFrame
features : 1
extent : -122.0849, -122.0849, 37.42316, 37.42316
coord. ref. : +proj=longlat +datum=WGS84
variables : 1
names : name
min values : Google headquarters
max values : Google headquarters
```

The same object in KML can be generated by using the **XML** package (Lang 2013):

```
R> library("XML")
R> pnt.kml <- newXMLNode("kml")
R> h2 <- newXMLNode("Document", parent = pnt.kml)
R> h3 <- newXMLNode("name", "Google headquarters", parent = h2)
R> h4 <- newXMLNode("Folder", parent=pnt.kml[["Document"]])
R> txtc <- sprintf('<Placemark><Point><coordinates>
+      %.5f,%.5f,%.0f</coordinates></Point></Placemark>',
+      coordinates(pnt)[,1], coordinates(pnt)[,2], rep(0, length(pnt)))
R> parseXMLAndAdd(txtc, parent = h4)
R> pnt.kml

<kml>
  <Document>
    <name>Google headquarters</name>
    <Folder>
      <Placemark>
        <Point>
          <coordinates>
    -122.08492,37.42316,0</coordinates>
        </Point>
      </Placemark>
    </Folder>
  </Document>
</kml>
```

where `newXMLNode` creates nodes in an XML object, `sprintf` is the wrapper for the fast C function that returns a character vector, and `saveXML` writes the object to a file (this function is does most of the work in the **plotKML** package). An XML object is basically a hierarchically structured object with nodes of different type. The rules to build and extend such objects are defined via the specific XML scheme, in this case the OGC KML 2.2 scheme (Wilson 2008).

Note that, although both R objects and KML (XML) files are human readable and have similar hierarchical structure, they have some systematic differences. First, KML accepts only geographical coordinates in WGS84 system, which must include altitude i.e. only 3D georeference is acceptable, while in R any proj4 supported projection can be used and the georeference can be 2D or 3D. Secondly, KML is by default used to display objects (*"placemarks"*) on Earth's surface, while the **sp** package does not restrict considering the bounding box and relative position based on land surface. To understand all rules and validity checks of the KML format refer to Wernecke (2009), and for an introduction to the **sp** class objects refer to Bivand *et al.* (2013).

## 3. Implementation and code examples

### 3.1. Basic principles and main use

The purpose of **plotKML** is to write standard spatio-temporal objects — spatial and space-time points, lines, polygons and grids, trajectories, georeferenced photographs and similar — from R to KML/KMZ files in such a way that they correspond, as much as possible, to standard cartographic plots or standard scientific visualizations. The main philosophy of **plotKML** is thus:

1. *Create a spatio-temporal object of some class*,
2. *Transform coordinates to a coordinate reference system compatible with a virtual globe (geographical coordinates in WGS84; altitude in meters; time in the UTC system)*,
3. *Visualize it using the plotKML function.*

We refer to complete scientific visualization templates in **plotKML** as *'views'*. Views are generated using a combination of low level KML writing functions, which basically coerce the spatio-temporal objects in R to the KML schema.

Visualization-specific generic settings such as icons, color and size of icons (icon styles), labels etc. are referred to as *'aesthetics'* in **plotKML**. These can be set by changing function arguments within each `"kml_layer"` function. Views and their components have been designed to be cartographically *'complete'*, meaning that:

- All spatio-temporal objects are automatically converted to the WGS84 geographical coordinates. Hence the projection system needs to be specified for each input spatial layer.
- Legends for all aesthetics are provided using screen overlays or at least labels are attached to individual plotting objects.
- Missing values (NA), extrapolation areas and/or masked pixels are automatically removed or made transparent.
- Each spatial layer carries some minimum metadata that can be entered via the description tag. This way the distributed KML files can be used as official representations of registered data sets.

The following example shows how to produce a bubble type plot using the **plotKML** package (plot shown in Figure 2). We start by loading the Ebergötzen soil mapping data set (available via the **plotKML** package):

```
R> library("plotKML")
R> data("eberg")
R> eberg <- eberg[runif(nrow(eberg))<.2,]
R> coordinates(eberg) <- ~X+Y
R> proj4string(eberg) <- CRS("+init=epsg:31467")
```

Next, we can reproject this object to the WGS84 coordinate system:

```
R> eberg.ll <- reproject(eberg)
```

so that it can be plotted in Google Earth by using:

```
R> plotKML(eberg.ll["CLYMHT_A"])

Plotting the first variable on the list
KML file opened for writing...
Parsing to KML...
Closing  eberg.ll__CLYMHT_A__.kml
```

and which largely mimics the existing plotting functionality available for spatial data in R, e.g. the `spplot` functionality from the **sp** package:

```
R> spplot(eberg.ll["CLYMHT_A"], edge.col="black",
+    alpha=0.8, cex=seq(.3,3,length=5))
```

Note that the generic `reproject` function available in the **plotKML** package will try to reproject any **sp** class objects to the referent WGS84 system, also within the `plotKML()` function:

```
R> plotKML(eberg["CLYMHT_A"])

Plotting the first variable on the list
KML file opened for writing...
Reprojecting to +proj=longlat +datum=WGS84 ...
Parsing to KML...
Closing  eberg__CLYMHT_A__.kml
```

This means that both `plotKML(eberg)` and `plotKML(eberg.ll)` would work with the same object. It is recommended, however, that the users pre-transform spatial objects into the WGS84 geographic coordinate system as this step can be time consuming when working with large data sets.

By combining multiple aesthetics, **plotKML** can be used also to visualize multivariate data (Figure 3). For example, to visualize two variables at the same time we would run:

```
R> shape = "http://maps.google.com/mapfiles/kml/pal2/icon18.png"
R> kml(eberg.ll, shape = shape, colour = CLYMHT_A, labels = SNDMHT_A,
+    altitude = SNDMHT_A*10, extrude = TRUE)
```

which would use colors to visualize the clay content, and labels and altitude to represent the sand content. Possibility of multivariate visualization makes **plotKML** comparable to the **lattice** package for R (Gabor 2008).

### 3.2. KML building utilities

Aside from the generic `plotKML()` method, the package also contains a number of dedicated methods and functions, which can be referred to as the KML building utilities. The basic KML building utilities are (Figure 4):

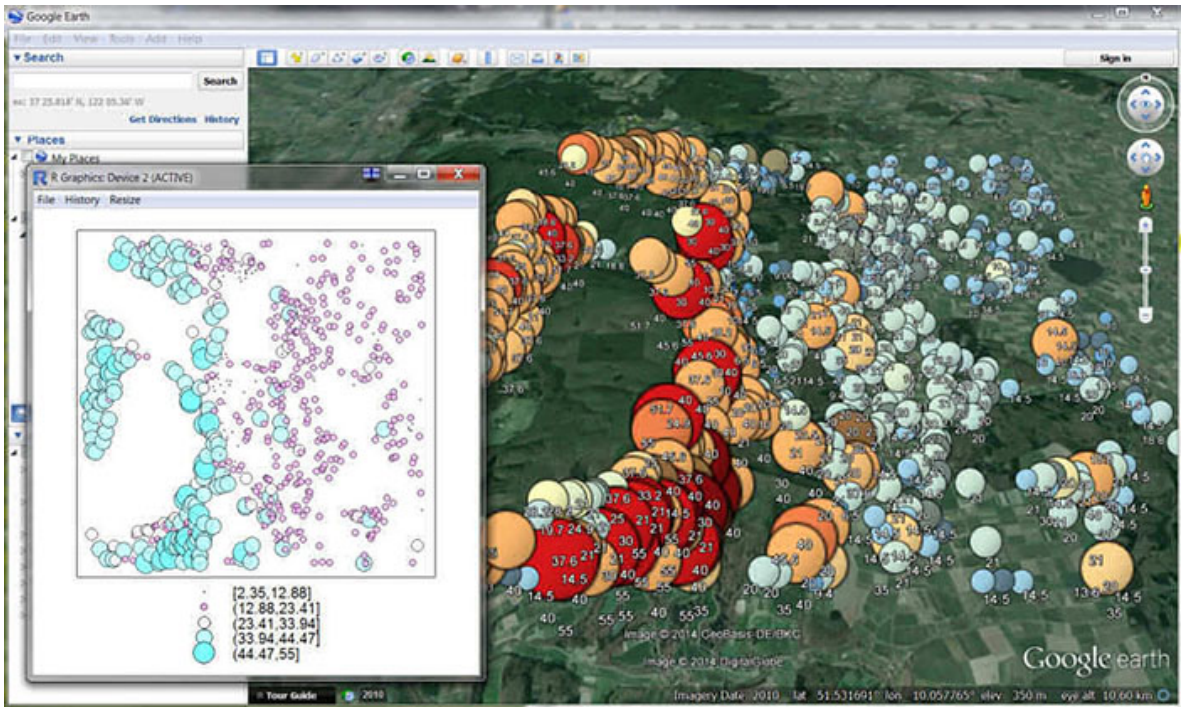


Figure 2: Bubble-type plots in R and the same plot produced using the **plotKML** shown in Google Earth.

- `kml_open()` — opens a KML file in write mode, manages the KML specific URL, and the name of the KML file. It also writes the header of the KML file.
- `kml_close()` — closes the KML file.
- `kml_layer()` — writes any **sp** object into a KML layer (`<Folder>` tag).
- `kml_screenoverlay()` — puts a PNG file on the Google Earth screen; usually a legend attached to the map or a logo image.
- `kml_compress()` — compresses a KML file to a KMZ file.
- `kml_legend()` — generates a legend depending on the type of spatio-temporal object.

These utilities actually provide an advanced mode for KML creation, and allow the user to create multi-layers KML with specific legends. This means that a single KML file can be created that contains all layers of interest (e.g. grids and points), associated legends and explanations of how were the maps derived.

The following example demonstrates how several layers can be put together in the same KML file. We start by loading some gridded layers:

To write a gridded layer with a point layer on the top we can run:

```
R> kml_open("eberg.kml")
R> kml_layer(eberg_grid, colour=TWISRT6)
```

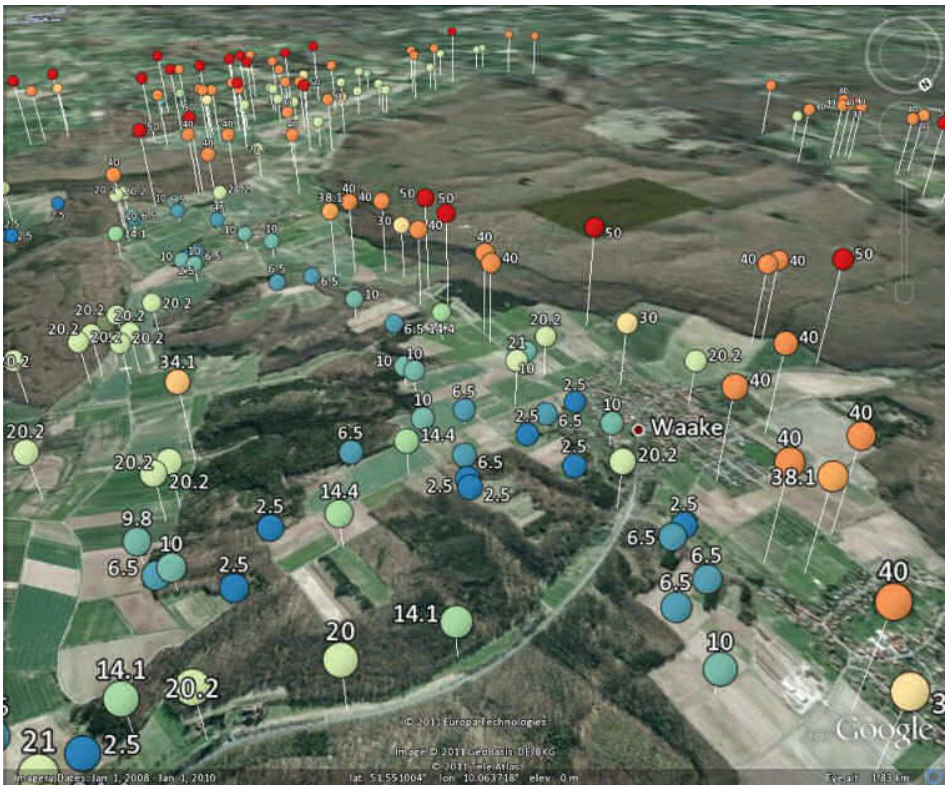


Figure 3: Multivariate visualization using three aesthetics parameters (in the case above: color, labels and elevation). The plot shows changes in sand content in the soil for the Ebergötzen case study.

```
R> kml_layer(eberg.ll[1,], colour=CLYMHT_A)
R> kml_close("eberg.kml")
```

The resulting plot is shown in Figure 5.

### 3.3. plotKML specific classes

In addition to the existing classes already available in R, we have constructed several new **plotKML** classes to provide even richer visualization possibilities:

- `"SpatialMetadata"` — *a class to store and exchange spatial metadata.*
- `"SpatialPhoto"` — *a class to store spatial location, image (photograph) and its geometric properties.*
- `"SpatialPredictions"` — *a class to store results of geostatistical mapping.*
- `"SpatialVectorsSimulations"` — *a class to store list of equiprobable simulations of point, line and polygon features.*

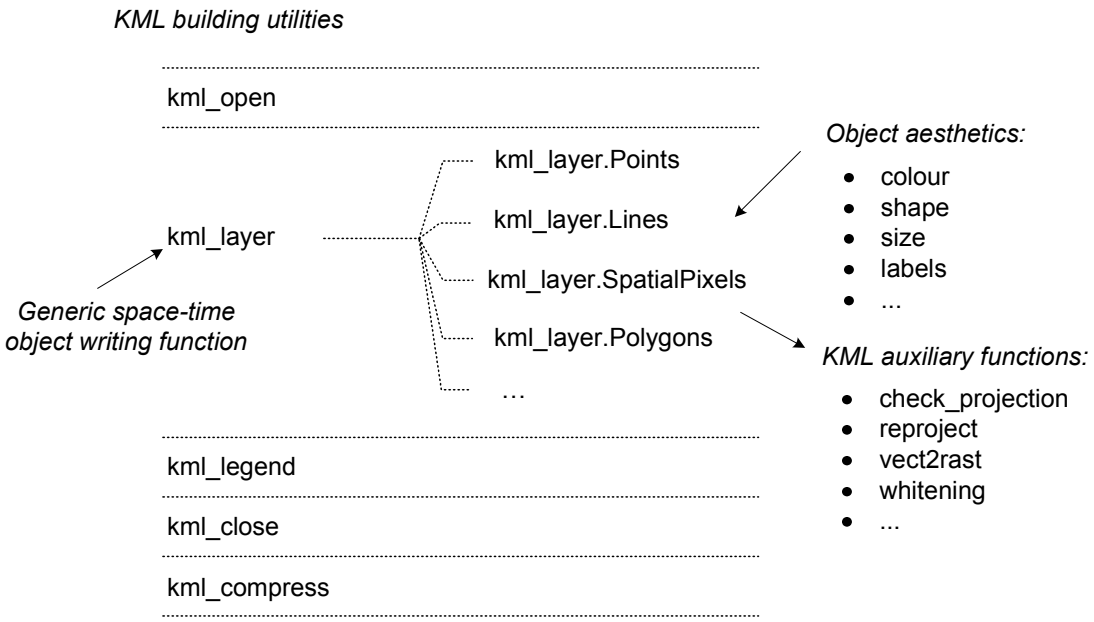


Figure 4: Overview of the KML building utilities available in the **plotKML** package.

- `"RasterBrickSimulations"` — *a class to store a list of equiprobable realizations of the same feature.*
- `"RasterBrickTimeSeries"` — *a class to store a time series of grids representing the same feature.*

Most of the classes listed above extend the basic **sp** and **raster** based classes by attaching the inputs and outputs of the spatial analysis. For example, the `"SpatialPredictions"` class contains both the input sampled values, the results of model fitting (regression and the variogram model), predictions and the results of cross validation. By plotting such object via the `plotKML-method`, one can prepare complete scientific visualization for Google Earth (see Figure 6).

The process of getting from input data (R) to a map in Google Earth is now straightforward and requires only four steps:

(*1*) *load/format the data*,

(*2*) *fit the geostatistical model*,

(*3*) *make predictions*,

(*4*) *visualize maps*,

Consider for example the Meuse data set commonly used for geostatistical exercises (Pebesma 2004):

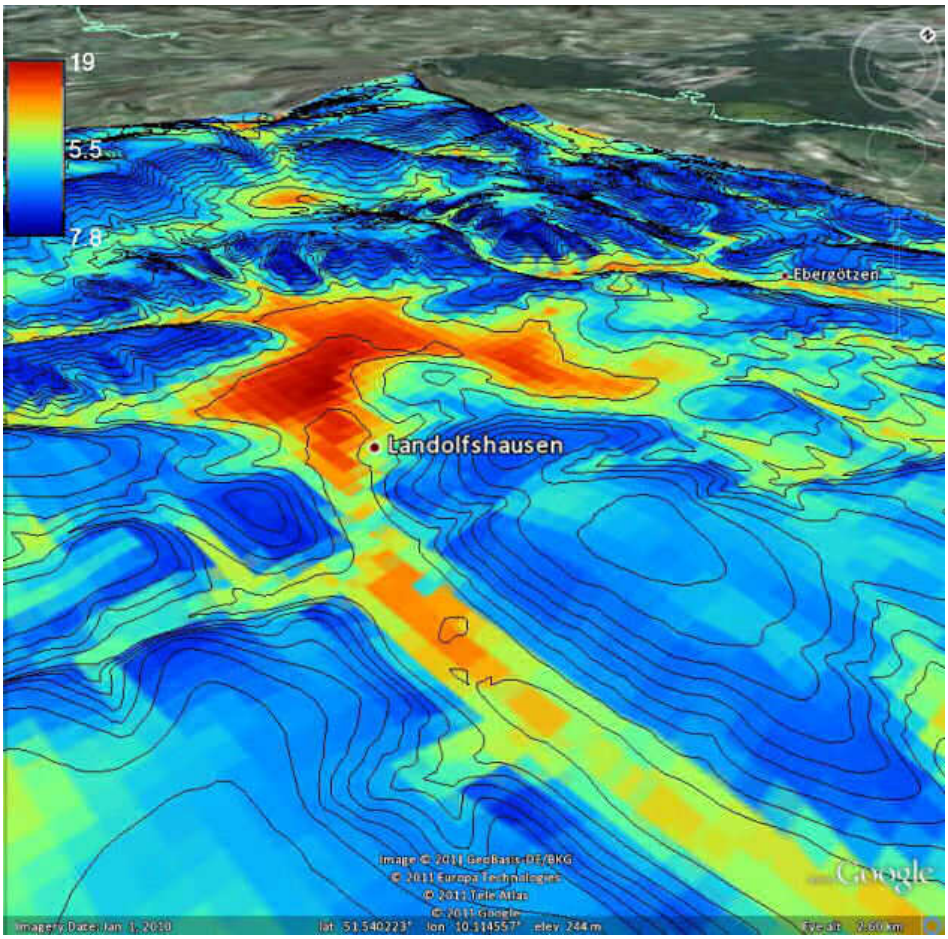


Figure 5: Example of a multi-layer KML file produced using **plotKML**: Topographic Wetness Index derived in SAGA GIS (raster image with legend) and contour lines overlaid. This visualization template has been largely inspired by the SAGA GIS software (Conrad 2007).

```
R> library("sp")
R> demo(meuse, echo=FALSE)
```

Via the **GSIF** package we can automate the model fitting and prediction:

```
R> library("GSIF")
R> omm <- fit.gstatModel(meuse, om~dist+ffreq, meuse.grid,
+     family = gaussian(log))
R> om.rk <- predict(omm, meuse.grid)

[using ordinary kriging]

100% done
```

here the generic function `fit.gstatModel` from the **GSIF** package (Hengl 2013) tries to fit a linear regression-kriging model following the input samples (`meuse`) and gridded maps

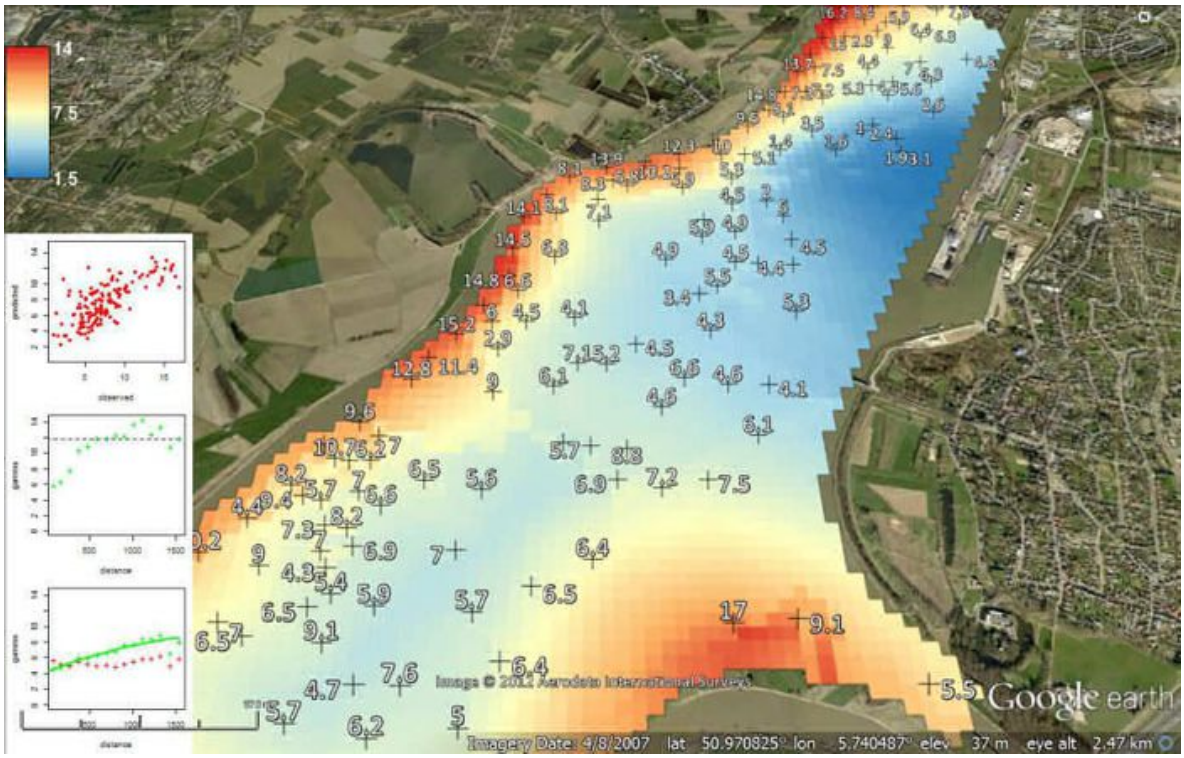


Figure 6: Scientific visualization of the results of geostatistical mapping (percent organic matter in top-soil) in Google Earth™: a combination of visualization of gridded map with a legend and map showing the sampling locations and fitted regression and variogram models.

(`meuse.grid`). The output (object of class `"SpatialPredictions"`) can be plotted in Google Earth by running:

```
R> plotKML(om.rk)

KML file opened for writing...
Reprojecting to +proj=longlat +datum=WGS84 ...
Writing to KML...
Reprojecting to +proj=longlat +datum=WGS84 ...
Parsing to KML...
Loading required package: gstat
Closing  om.rk.kml
```

which shows most of elements of geostatistical mapping of interest: sampling locations, resulting spatial predictions, but also success of regression model fitting, variogram fitting and cross-validation. Note also that wrapping model fitting and export to KML allows full automation of the mapping process, so that the process can be run on a server (Pebesma *et al.* 2011).

### 3.4. Visualization of spatio-temporal classes

From 2012, there are several implementations of spatio-temporal classes in R. Pebesma (2012b) shows how purely spatial classes can be extended to space-time classes (via the **spacetime** (Pebesma 2012a) package): spatio-temporal full data frame (STFDF), sparse spatio-temporal data frame (STSDF), spatio-temporal irregular data frames (STIDF). **plotKML** works with most of the space-time classes implemented in R, especially the ones extending the **sp** classes, but the package should also work with the **raster** package class objects.

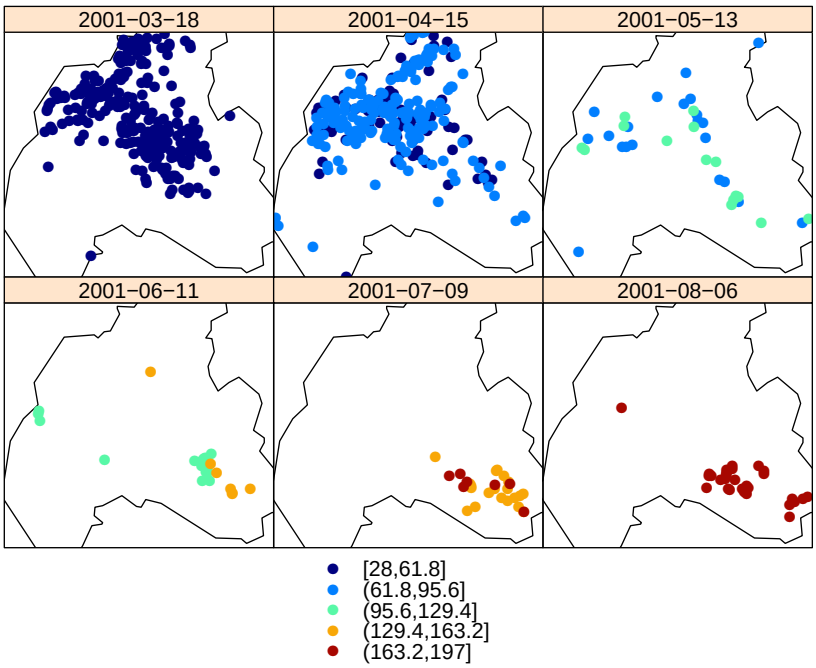


Figure 7: Spatio-temporal plot of point pattern in time: foot-and-mouth epidemic data from north Cumbria (UK) (Diggle *et al.* 2005).

Consider for example the foot-and-mouth epidemic data from north Cumbria (UK) (Diggle *et al.* 2005). This is a spatio-temporal point pattern represented with a matrix containing $(x, y, t)$ coordinates of the 648 observations. To visualize this data in R (**spacetime** package) we would run:

```
R> data("fmd")
R> fmd0 <- data.frame(fmd)
R> coordinates(fmd0) <- c("X", "Y")
R> proj4string(fmd0) <- CRS("+init=epsg:27700")
R> fmd_sp <- as(fmd0, "SpatialPoints")
R> dates <- as.Date("2001-02-18")+fmd0$ReportedDay
R> library("spacetime")
R> fmd_ST <- STIDF(fmd_sp, dates, data.frame(ReportedDay=fmd0$ReportedDay))
R> data("northcumbria")
R> ln <- Line(northcumbria)
R> NC <- SpatialLines(list(Lines(list(ln), ID="NC")))
```

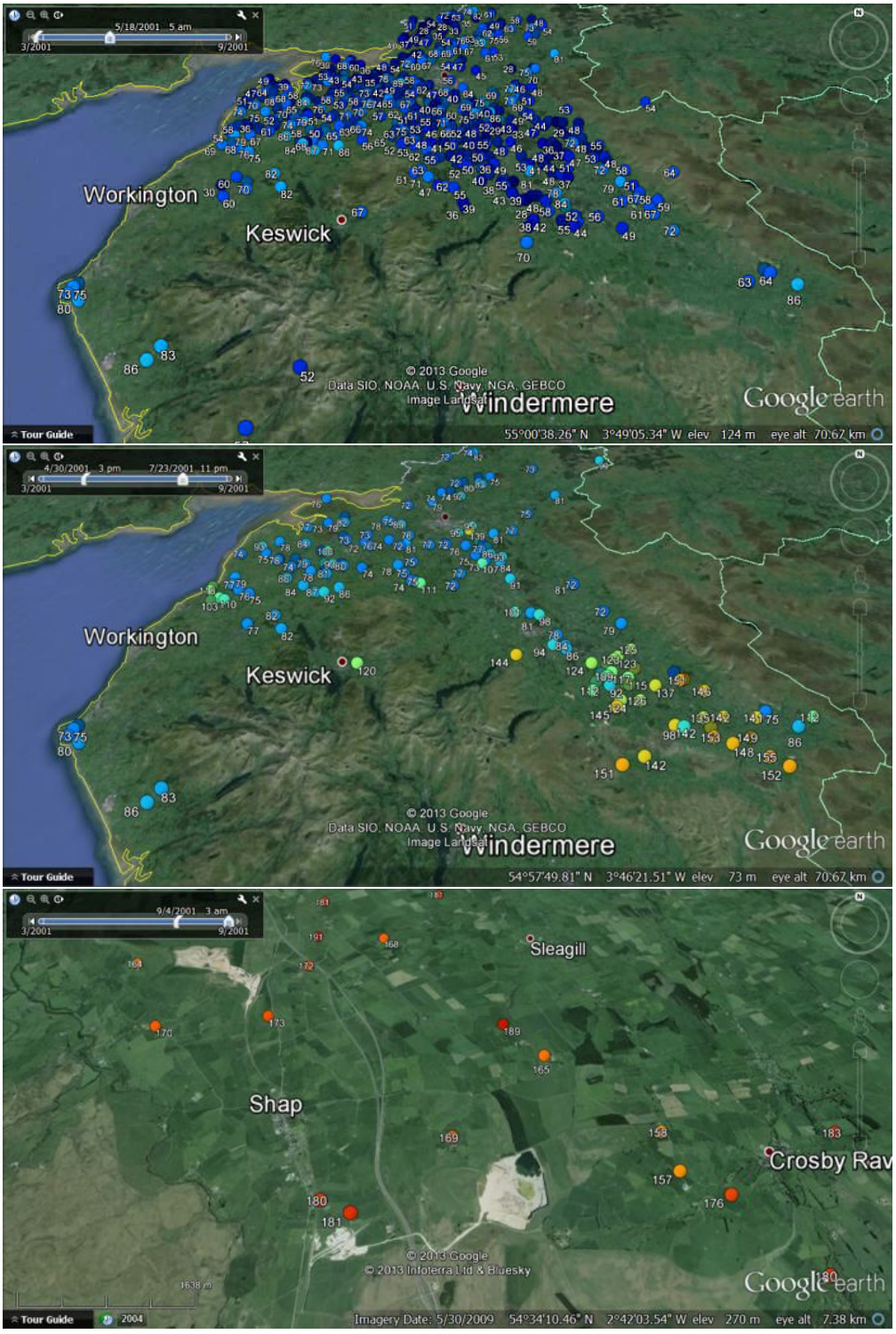


Figure 8: Visualization of the food-and-mouth epidemic data from north Cumbria (UK) in Google Earth (compare with Figure 7). Note that Google Earth allows users to slide through an event, but also to set the temporal support (width in the slide bar).

```
R> proj4string(NC)  <- CRS("+init=epsg:27700")
R> stplot(fmd_ST, sp.layout=list("sp.lines", NC), col.regions=SAGA_pal[[1]])
```

which produces a plot shown in Figure 7.

To plot this data in Google Earth we can do (Figure 8):

```
R> plotKML(fmd_ST, colour_scale=SAGA_pal[[1]])

Plotting the first variable on the list
KML file opened for writing...
Reprojecting to +proj=longlat +datum=WGS84 ...
Parsing to KML...
Closing  fmd_ST.kml
```

Some advantages of getting this data to Google Earth, as compared to using the `stplot()` function (Figure 7), are:

1. We can animate spread of disease by moving the time slider.
2. We can observe the event with different temporal support size.
3. We can zoom in into the specific points and locate the actual farm.
4. We can try to analyze whether the spread of disease has anything to do with the proximity to roads, type of land cover etc.
5. We do not necessarily need to prepare any administrative data for the study area as these are already available in Google Earth.

### 3.5. Visualization of time series of rasters

**plotKML** can also be used to visualize raster stacks i.e. time series of rasters. Raster stacks or raster bricks (Hijmans and van Etten 2012) can be different type of remote sensing data or predicted or modelled values. The plot shown in Figure 9 is based on generic functions for plotting of `"RasterBrickTimeSeries"` class data. It shows a time series of MODIS Land Surface Temperature images for a small area in Istria. We start by building an spatio-temporal object from table data:

```
R> data("LST")
R> gridded(LST) <- ~lon+lat
R> proj4string(LST) <- CRS("+proj=utm +zone=33 +datum=WGS84 +units=m")
```

Next, format dates:

```
R> dates <- sapply(strsplit(names(LST), "LST"), function(x){x[[2]]})
R> datesf <- format(as.Date(dates, "%Y_%m_%d"), "%Y-%m-%dT%H:%M:%SZ")
```

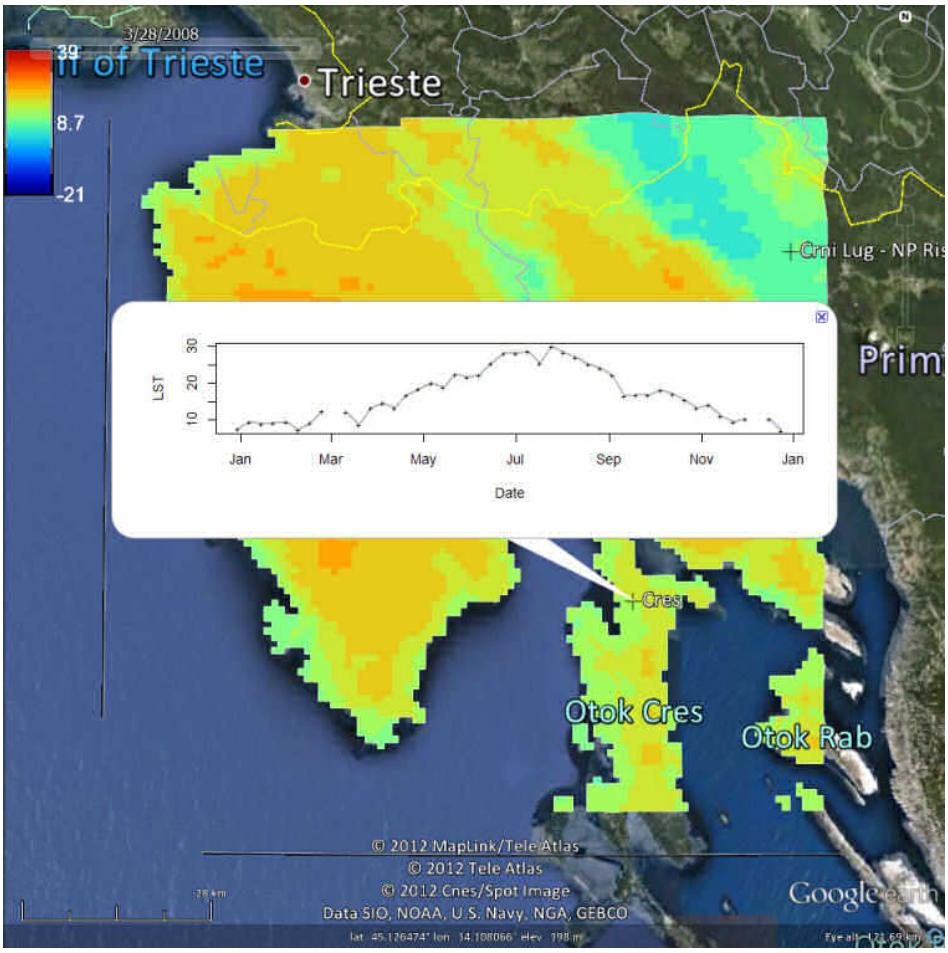


Figure 9: Example of a time series of MODIS images (`RasterBrickTimeSeries` class) plotted in Google Earth. The values of MODIS Land Surface Temperature at ground stations can be interactively explored by clicking at the points of interest. This way both changes in 2D+T and T can be visually explored.

we subtract and add ±4 days to derive begin and end period:

```
R> TimeSpan.begin = as.POSIXct(unclass(as.POSIXct(datesf))-4*24*60*60,
+    origin="1970-01-01")
R> TimeSpan.end = as.POSIXct(unclass(as.POSIXct(datesf))+4*24*60*60,
+     origin="1970-01-01")
R> LST_ll <- reproject(LST)

Reprojecting to +proj=longlat +datum=WGS84 ...
Reprojecting to +proj=longlat +datum=WGS84 ...
...
```

Next, we can select three climatic stations in the area (to allow for visualization of time-series data) and prepare an object of class `"RasterBrickTimeSeries"`:

```
R> st.names <- c("Pazin", "Crni Lug - NP Risnjak", "Cres")
R> pnts <- HRtemp08[which(HRtemp08$NAME==st.names[1])[1],]
R> pnts <- rbind(pnts, HRtemp08[which(HRtemp08$NAME==st.names[2])[1],])
R> pnts <- rbind(pnts, HRtemp08[which(HRtemp08$NAME==st.names[3])[1],])
R> coordinates(pnts) <- ~Lon + Lat
R> proj4string(pnts) <- CRS("+proj=longlat +datum=WGS84")
R> LST_ll <- brick(LST_ll)
R> LST_ll@title = "Time series of MODIS Land Surface Temperature"
R> LST.ts <- new("RasterBrickTimeSeries", variable = "LST", sampled = pnts,
+         rasters = LST_ll, TimeSpan.begin = TimeSpan.begin[1:5],
+         TimeSpan.end = TimeSpan.end[1:5])
```

so that we can plot the MODIS images in Google Earth by running:

```
R> plotKML(LST.ts, colour_scale=SAGA_pal[[1]])

KML file opened for writing...
Parsing to KML...
Writing to KML...
Closing  LST.ts.kml
```

The resulting plot allows us to explore the changes in values both in space (by using the slide bar in Google Earth) and time (by browsing the individual time-series plots). Again, temporal support can be controlled by the width of the time slider. Setting up a wider temporal support Google Earth will interpolate the displayed colors automatically and display patterns that we would otherwise not be able to see.

### 3.6. plotKML in comparison to other KML writing software

Table 3.6 list some common KML writing software and provides our quick assessment of their functionality. In comparison to other KML writing packages in R and python (e.g. **R2G2** and **pyKML**), we see the potential of **plotKML** primarily within its ambition to provide a larger family of lower-level and wrapper functions that allow both fast generation of complete plots, and customization of KML plots by binding several lower level functions (Table 3.6).

Although spatial objects in R can be converted to KML formats using a range of different approaches, the **plotKML** package tries to simplify the process considerably. It has a potential particularly for visualising time series of spatial layers and outputs of spatial analysis (as shown in sections 3.4 and 3.5), which is a task that is always rather challenging.

A possible limitation of using **plotKML** will be visualization of large data sets, which is at the moment not recommended. Instead of attempting direct export of large GIS layers into a single KML file, we advice the user to consider some of the following strategies to optimize export and visualization of large data sets:

1. Tile large study areas into regular blocks.
2. Consider increasing the grid cell size in the raster images and/or remove redundant nodes in the vector layers by various thinning procedures.

| *Aspect* | GDAL/OGR | SAGA GIS | GRASS GIS | Autodesk | Geomedia | ArcGIS | Mapping Toolbox | **pyKML** | **R2G2** | **plotKML** |
|---|---|---|---|---|---|---|---|---|---|---|
| Programming language | C++ / Python | C++ / Python | C++ / Python ... | – | SQL ... | AML / Python ... | MatLab | Python | R | R |
| Open Source software | ✓ | ✓ | ✓ | – | – | – | – | ✓ | ✓ | ✓ |
| Writes vector data (points, lines, polygons) | ⋆ | – | ⋆ | ★ | ★ | ★ | ★ | ⋆ | ⋆ | ★ |
| Writes rasters (single raster layer, raster stacks) | – | ⋆ | ⋆ | – | – | ★ | ⋆ | ⋆ | – | ★ |
| Writes spatio-temporal data (spatio-temporal classes) | – | – | – | – | – | ⋆ | ⋆ | – | – | ★ |
| Can efficiently handle large data (Gigapixel images and large vectors) | ★ | ⋆ | ★ | ★ | ★ | ★ | ★ | ★ | ⋆ | ⋆ |
| Allows creation of visualization templates (KML writing utilities) | ⋆ | – | ⋆ | – | – | ⋆ | ⋆ | ★ | ⋆ | ★ |
| Allows styling (icons size and colors, line width, transparency etc.) | – | – | – | – | – | ⋆ | ★ | ⋆ | ⋆ | ★ |
| Supports visualization of scientific data (statistical plots and charts) | – | – | – | – | – | ⋆ | ★ | ⋆ | – | ★ |
| Coding efficiency (ease of writing the spatial objects to KML files) | ★ | – | ★ | – | ⋆ | ★ | ★ | ⋆ | ★ | ★ |
| Provides user-friendly GUI | – | ★ | ⋆ | ★ | ★ | ★ | ⋆ | – | – | – |

Table 2: A matrix comparison of some KML writing software. ★ — full capability, ⋆ — possible but with limitations, – — not possible in this package. Functionality subjectively approximated by the authors (subject to discussion).

3. Consider putting the data into a database and then generate KML in chunks.

## 4. Discussion

We have described the design of the R package **plotKML** in comparison to the other existing packages and software for visualization of similar data. The `plotKML()` method is suggested as a straightforward approach to get the results of spatial analysis quickly from R to Google Earth, but this of course does not alleviate the need for a comprehensive understanding of (geo)statistical analysis and spatial data object structures.

Indirectly, **plotKML** promotes KML as a standard to use for scientific visualization of spatio-temporal data. Can this be justified? We think that there are several attractive features of using KML schema for scientific visualization. First of all, KML often goes beyond what we typically use within a desktop GIS. KML files can contain both spatial, textual and multimedia information, hence anyone can virtually explore geographical data as standing about the Earth's surface, but then also read all required information about how was the data produced and where to find more information via the embedded HTML content (Wernecke 2009). Second, as indicated in the introduction already, KML is the format used by Google Earth, and this is likely to remain the most popular geographical browser in the years to come. We believe that packages that connect statistical and geographical programming environments with Google Earth type of browsers are also critical for engaging the wider public in project outcomes. Both researchers and applied users like to see patterns in data clearly and they like to explore data in user-friendly Virtual Earth type of environment. It is also worth mentioning that, Elsevier[10] now encourages authors to attach to attach KML files to their article submission. This option is now available for over 80 of their journals in earth sciences, life sciences, and social sciences, so it is in a way already a standard for scientific visualization.

But there are still much room for improvements. As we have demonstrated, both R and KML/Google Earth are fairly flexible and can be used to represent spatial objects and features in 4D (geographical coordinates, altitude, time), nevertheless, both are not ideal for specific tasks. For example, KML is not (yet) suitable to represent complex geological structures, although some authors (e.g. De Paor and Whitmeyer (2011) and Blenkinsop (2012)) have suggested ways around the problem.

We foresee that the future of our package will be in linking our scripts with Java scripts and/or PHP scripts i.e. implementing the functionality that subsets and tiles data from large databases on the fly. We also hope to constantly update and expand the number of supported classes in **plotKML** (especially following the matrix in Table 2.3) and then add more and more functionality for visualizing distributed data via network links and web mapping services.

## Acknowledgments

This package has been developed as a part of the Global Soil Information Facilities initiative — tools for collating and serving global soil data developed jointly by the ISRIC — World Soil Information institute and collaborators. ISRIC is a non-profit organization with a mandate

[10] `http://www.elsevier.com/googlemaps`

to serve the international community as custodian of global soil information and to increase awareness and understanding of the role of soils in major global issues.

The authors would like to thank the two anonymous reviewers for their suggestions and R code chunks that were largely incorporated in this article.

## References

Arrigo N, Albert LP, Mickelson PG, Barker MS (2012). "Quantitative visualization of biological data in Google Earth using **R2G2**, an R CRAN package." *Molecular Ecology Resources*, **12**(6), 1177–1179.

Beaudette D, Roudier P (2012). ***aqp**: Algorithms for Quantitative Pedology*. R package version 1.4, URL `http://CRAN.R-project.org/package=aqp`.

Bivand R, Pebesma E, Rubio V (2008). *Applied Spatial Data Analysis with R*. Use R Series. Springer-Verlag, Heidelberg.

Bivand R, Pebesma E, Rubio V (2013). *Applied Spatial Data Analysis with R*. Use R Series, 2nd edition. Springer-Verlag, Heidelberg.

Bleisch S (2011). *Evaluating the Appropriateness of Visually Combining Quantitative Data Representations with 3D Desktop Virtual Environments Using Mixed Methods*. Phd thesis, City University London.

Blenkinsop T (2012). "Visualizing structural geology: From Excel to Google Earth." *Computers & Geosciences*, **45**(0), 52–56.

Butler D (2006). "Virtual globes: The web-wide world." *Nature*, **439**, 776–778.

Conrad O (2007). *SAGA — Entwurf, Funktionsumfang und Anwendung eines Systems für Automatisierte Geowissenschaftliche Analysen*. Ph.D. thesis, University of Göttingen, Göttingen.

Cook D, Swayne DF (2007). *Interactive and Dynamic Graphics for Data Analysis*. Springer-Verlag, New York. ISBN 978-0-387-71761-6.

Craglia M, Goodchild M, Annoni A, Camara G, Gould M, Kuhn W, Mark D, Masser I, Maguire D, Liang S, Parsons E (2008). "Next-Generation Digital Earth: A position paper from the Vespucci Initiative for the Advancement of Geographic Information Science." *International Journal of Spatial Data Infrastructures Research*, **3**(6), 146–167.

De Paor DG, Whitmeyer SJ (2011). "Geological and geophysical modeling on virtual globes using KML, COLLADA, and Javascript." *Computers & Geosciences*, **37**(1), 100–110.

Diggle P, Rowlingson B, Su Tl (2005). "Point process methodology for on-line spatio-temporal disease surveillance." *Environmetrics*, **16**(5), 423–434.

Dykes J, MacEachren A, Kraak J (2005). *Exploring Geovisualization*. International Cartographic Association. Elsevier Science. ISBN 9780080531472.

Elmqvist N, Tudoreanu ME (2007). "Occlusion Management in Immersive and Desktop 3D Virtual Environments: Theory and Evaluation." *International Journal of Virtual Reality*, **6**(2), 21–32.

Erwig M, Güting RH, Schneider M, Vazirgiannis M (1999). "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases." *GeoInformatica*, **3**, 269–296.

Friendly M, Denis DJ (2001). "Milestones in the history of thematic cartography, statistical graphics, and data visualization." URL `http://www.datavis.ca/milestones/`.

Fritz S, McCallum I, Schill C, Perger C, See L, Schepaschenko D, van der Velde M, Kraxner F, Obersteiner M (2012). "Geo-Wiki: An online platform for improving global land cover." *Environmental Modelling & Software*, **31**, 110–123.

Gabor G (2008). "lattice: Multivariate Data Visualization with R." *Journal of Statistical Software, Book Reviews*, **25**(2), 1–3. ISSN 1548-7660. URL `http://www.jstatsoft.org/v25/b02`.

Galton A (2004). "Fields and objects in space, time, and space-time." *Spatial cognition and computation*, **4**(1), 39–68.

Goodchild MF (2008). "Commentary: whither VGI?" *GeoJournal*, **72**(3-4), 239–244. ISSN 0343-2521. `doi:10.1007/s10708-008-9190-4`.

Gore A (1998). "The digital earth: understanding our planet in the 21st century." *Australian surveyor*, **43**(2), 89–91.

Hengl T (2013). ***GSIF**: Global Soil Information Facilities — tools (standards and functions) and sample datasets for global soil mapping*. R package version 0.3-6, URL `http://CRAN.R-project.org/package=GSIF`.

Hengl T, Heuvelink G, Perčec Tadić M, Pebesma E (2012). "Spatio-temporal prediction of daily temperatures using time-series of MODIS LST images." *Theoretical and Applied Climatology*, **107**, 265–277.

Hijmans RJ, van Etten J (2012). ***raster**: Geographic data analysis and modeling*. R package version 2.0-41, URL `http://CRAN.R-project.org/package=raster`.

Keitt TH, Bivand R, Pebesma E, Rowlingson B (2013). ***rgdal**: Bindings for the Geospatial Data Abstraction Library*. R package version 0.8-3, URL `http://CRAN.R-project.org/package=rgdal`.

Lang DT (2013). ***XML**: Tools for parsing and generating XML within R and S-PLUS*. R package version 3.98-1.1, URL `http://CRAN.R-project.org/package=XML`.

McGavra G, Morris S, Janée G (2009). *Technology Watch Report: Preserving Geospatial Data*. DPC Technology Watch Series Report 09-01. Digital Preservation Coalition, York, UK.

Patterson TC (2007). "Google Earth as a (Not Just) Geography Education Tool." *Journal of Geography*, **106**(4), 145–152.

Pebesma E (2012a). ***spacetime**: classes and methods for spatio-temporal data*. R package version 1.0-3, URL http://CRAN.R-project.org/package=spacetime.

Pebesma E (2012b). "**spacetime**: Spatio-Temporal Data in R." *Journal of Statistical Software*, **51**, 1–30.

Pebesma E, Cornford D, Dubois G, Heuvelink G, Hristopulos D, Pilz J, Stöhlker U, Morin G, Skøien JO (2011). "INTAMAP: the design and implementation of an interoperable automated interpolation web service." *Computers & Geosciences*, **37**(3), 343–352.

Pebesma EJ (2004). "Multivariable geostatistics in S: the **gstat** package." *Computers & Geosciences*, **30**(7), 683–691.

Pebesma EJ, Bivand RS (2005). "Classes and methods for spatial data in R." *R news*, **5**(2), 9–13.

R Development Core Team (2013). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.

Theus M, Urbanek S (2009). *Interactive graphics for data analysis: principles and examples*. Series in computer science and data analysis. CRC Press. ISBN 9781584885948.

Wernecke J (2009). *The KML handbook: geographic visualization for the Web*. Addison-Wesley. ISBN 9780321525598.

Wilson T (2008). "OGC KML." *OGC Standard OGC 07-147r2*, Open Geospatial Consortium Inc.

**Affiliation:**

T. Hengl
OpenGeoHub foundation
Roghorst 206, 6708 KT Wageningen, the Netherlands
Tel.: +31- 317-427537
E-mail: tom.hengl@opengeohub.org

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
February 2015, Volume 63, Issue 5 *Submitted:* 2013-01-22
doi:10.18637/jss.v000.i00 *Accepted:* 2014-07-08