Geographic Information Systems with R

Kevin O'Brien

June 11, 2013

Contents

1 Geographic Information Systems

2 Preliminaries

- Working Directory
- Packages

The package ***sp*** provides general purpose classes and methods for defining, importing/exporting and visualizing spatial data.

```
setwd("~/UsingR-GIS")

### Basic packages ###

library(sp)             # classes for spatial data
library(raster)         # grids, rasters
library(rasterVis)      # raster visualisation
library(maptools)
# and the dependencies
```

2.1 The Ozone Example

Let's look at an example. Our dataset, ozone, contains ozone measurements from thirty-two locations in the Los Angeles area aggregated over one month. The dataset includes the station number (Station), the latitude and longitude of the station (Lat and Lon), and the average of the highest eight hour daily averages (Av8top).

This data, and other spatial datasets, can be downloaded from the University of Illinois's Spatial Analysis Lab. By generating a variogram, we will be able to look at the variance of the differences of Av8top among pairs of stations at different distances. We can look at a sample of our data and then a summary of the distances between the stations.

```
ozone<-read.table("http://www.ats.ucla.edu/stat/r/faq/ozone.csv", sep=",", header=T)
head(ozone, n=10)
```

3 Visualisation

```
### VISUALISATION OF GEOGRAPHICAL DATA ###


### RWORLDMAP ###

library(rworldmap)   # visualising (global) spatial data

  # examples:
  newmap <- getMap(resolution="medium", projection="none")
  plot(newmap)

  mapCountryData()
  mapCountryData(mapRegion="europe")
  mapGriddedData()
  mapGriddedData(mapRegion="europe")


### GOOGLEVIS ###

library(googleVis)    # visualise data in a web browser using Google
Visualisation API

  # demo(googleVis)   # run this demo to see all the possibilities

  # Example: plot country-level data
  data(Exports)
  View(Exports)
  Geo <- gvisGeoMap(Exports, locationvar="Country", numvar="Profit",
                    options=list(height=400, dataMode='regions'))
  plot(Geo)
  print(Geo)
  # this HTML code can be embedded in a web page (and be dynamically updated!)

  # Example: Plotting point data onto a google map (internet)
  data(Andrew)
  M1 <- gvisMap(Andrew, "LatLong" , "Tip", options=list(showTip=TRUE,
showLine=F, enableScrollWheel=TRUE,
                           mapType='satellite', useMapTypeControl=TRUE,
width=800,height=400))
  plot(M1)
```

3.1 GoogleMaps

Rgooglemaps

```
### RGOOGLEMAPS ###

library(RgoogleMaps)

  # get maps from Google
  newmap <- GetMap(center=c(36.7,-5.9), zoom =10, destfile = "newmap.png",
maptype = "satellite")
  # View file in your wd
  # now using bounding box instead of center coordinates:
  newmap2 <- GetMap.bbox(lonR=c(-5, -6), latR=c(36, 37), destfile =
"newmap2.png", maptype="terrain")    # try different maptypes
  newmap3 <- GetMap.bbox(lonR=c(-5, -6), latR=c(36, 37), destfile =
"newmap3.png", maptype="satellite")

  # and plot data onto these maps, e.g. these 3 points
   PlotOnStaticMap(lat = c(36.3, 35.8, 36.4), lon = c(-5.5, -5.6, -5.8), zoom=
10, cex=2, pch= 19, col="red", FUN = points, add=F)
```

3.2 The dismo package

```
### GMAP (DISMO) ###

library(dismo)

  # Some examples
  # Getting maps for countries
  mymap <- gmap("France")   # choose whatever country
  plot(mymap)
  mymap <- gmap("Spain", type="satellite")   # choose map type
  plot(mymap)
  mymap <- gmap("Spain", type="satellite", exp=3)  # choose the zoom level
  plot(mymap)
  mymap <- gmap("Spain", type="satellite", exp=8)
  plot(mymap)

  mymap <- gmap("Spain", type="satellite", filename="Spain.gmap")    # save the
map as a file in your wd for future use
```

```
# Now get a map for a region drawn at hand
mymap <- gmap("Europe")
plot(mymap)
select.area <- drawExtent()   # now click on the map to select your region
mymap <- gmap(select.area)
plot(mymap)
# See ?gmap for many other possibilities
```

3.3 Spatial Statistics

```
### SPATIAL STATISTICS ###

## Point pattern analysis
  library(spatial)
  library(spatstat)
  library(spatgraphs)
  library(ecespa)    # ecological focus
  # etc (see Spatial Task View)

  # example
  data(fig1)
  plot(fig1)    # point pattern
  data(Helianthemum)
  cosa12 <- K1K2(Helianthemum, j="deadpl", i="survpl", r=seq(0,200,le=201),
         nsim=99, nrank=1, correction="isotropic")
  plot(cosa12$k1k2, lty=c(2, 1, 2), col=c(2, 1, 2), xlim=c(0, 200),
         main= "survival- death",ylab=expression(K[1]-K[2]), legend=FALSE)
```

3.4 Geostatistics

The ackage ***gstat*** provides a wide range of uni-ariable and multivariable geostatistical modelling, prediction and simulation functions.

```
### Geostatistics ###
  library(gstat)
  library(geoR)
  library(akima)   # for spline interpolation
  # etc (see Spatial Task View)


  library(spdep)   # dealing with spatial dependence
```

When analyzing geospatial data, describing the spatial pattern of a measured variable is of great importance. A common way of visualizing the spatial autocorrelation of a variable is a variogram plot

4 Interacting with other GIS

```
### INTERACTING AND COMMUNICATING WITH OTHER GIS ###

library(spgrass6)   # GRASS
library(RPyGeo)     # ArcGis (Python)
library(RSAGA)      # SAGA
library(spsextante) # Sextante
```

4.1 Other useful packages

```
## Other useful packages ##

library(Metadata)    # automatically collates data from online GIS datasets
(land cover, pop density, etc) for a given set of coordinates

#library(GeoXp)    # Interactive exploratory spatial data analysis
  example(columbus)
  histomap(columbus,"CRIME")

library(maptools)
# readGPS

library(rangeMapper)    # plotting species distributions, richness and traits


# Species Distribution Modelling
library(dismo)
library(BIOMOD)
library(SDMTools)

library(BioCalc)   # computes 19 bioclimatic variables from monthly climatic
values (tmin, tmax, prec)
```

5 Examples

```
### Examples ###

### SPATIAL VECTOR DATA (POINTS, POLYGONS, ETC) ###


# Example dataset: Get "Laurus nobilis" coordinates from GBIF
laurus <- gbif("Laurus", "nobilis")
# get data frame with spatial coordinates (points)
locs <- subset(laurus, select=c("country", "lat", "lon"))

# Making it 'spatial'
coordinates(locs) <- c("lon", "lat")    # set spatial coordinates
plot(locs)

# Define geographical projection
# to look for the appropriate PROJ.4 description look here:
http://www.spatialreference.org/

crs.geo <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84")    # geographical,
datum WGS84
proj4string(locs) <- crs.geo     # define projection system of our data
summary(locs)


# Simple plotting
data(wrld_simpl)
summary(wrld_simpl)     # Spatial Polygons Data Frame with country borderlines
plot(locs, pch=20, col="steelblue")
plot(wrld_simpl, add=T)


### Subsetting
table(locs@data$country)     # see localities by country

locs.gr <- subset(locs, locs$country=="GR")   # select only locs in Greece
plot(locs.gr, pch=20, cex=2, col="steelblue")
plot(wrld_simpl, add=T)
summary(locs.gr)

locs.gb <- subset(locs, locs$country=="GB")    # locs in UK
plot(locs.gb, pch=20, cex=2, col="steelblue")
```

```
plot(wrld_simpl, add=T)
```

6 Making Maps

```
### MAKING MAPS ###

# Plotting onto a Google Map using RGoogleMaps
PlotOnStaticMap(lat = locs.gb$lat, lon = locs.gb$lon, zoom= 10, cex=1.4, pch=
19, col="red", FUN = points, add=F)


# Downloading map from Google Maps and plotting onto it
map.lim <- qbbox (locs.gb$lat, locs.gb$lon, TYPE="all")
mymap <- GetMap.bbox(map.lim$lonR, map.lim$latR, destfile = "gmap.png",
maptype="satellite")
# see the file in the wd
PlotOnStaticMap(mymap, lat = locs.gb$lat, lon = locs.gb$lon, zoom= NULL,
cex=1.3, pch= 19, col="red", FUN = points, add=F)

# using different background
mymap <- GetMap.bbox(map.lim$lonR, map.lim$latR, destfile = "gmap.png",
maptype="hybrid")
PlotOnStaticMap(mymap, lat = locs.gb$lat, lon = locs.gb$lon, zoom= NULL,
cex=1.3, pch= 19, col="red", FUN = points, add=F)

# you could also use function gmap in "dismo"
gbmap <- gmap(locs.gb, type="satellite")
locs.gb.merc <- Mercator(locs.gb)    # Google Maps are in Mercator projection.
This function projects the points to that projection to enable mapping
plot(gbmap)
points(locs.gb.merc, pch=20, col="red")


### Plotting onto a Google Map using googleVis (internet)
points.gb <- as.data.frame(locs.gb)
points.gb$latlon <- paste(points.gb$lat, points.gb$lon, sep=":")
map.gb <- gvisMap(points.gb, locationvar="latlon", tipvar="country",
                  options = list(showTip=T, showLine=F, enableScrollWheel=TRUE,
                           useMapTypeControl=T, width=1400,height=800))
plot(map.gb)
print(map.gb)    # HTML suitable for a web page

##########
```

```
# drawing polygons and polylines
mypolygon <- drawPoly()    # click on the map to draw a polygon and press ESC
when finished
summary(mypolygon)    # now you have a spatial polygon!
```

7 READING AND SAVING DATA

```
### READING AND SAVING DATA

### Exporting KML
writeOGR(locs.gb, dsn="locsgb.kml", layer="locs.gb", driver="KML")

### Reading kml
newmap <- readOGR("locsgb.kml", layer="locs.gb")

### Saving as a Shapefile
writePointsShape(locs.gb, "locsgb")

### Reading (point) shapefiles
gb.shape <- readShapePoints("locsgb.shp")
plot(gb.shape)

# readShapePoly   # polygon shapefiles
# readShapeLines  # polylines
# see also shapefile in "raster"
```

```
### PROJECTING ###

summary(locs)
# define new projection; look parameters at spatialreference.org
crs.laea <- CRS("+proj=laea +lat_0=52 +lon_0=10 +x_0=4321000 +y_0=3210000
+ellps=GRS80 +units=m +no_defs")
locs.laea <- spTransform(locs, crs.laea)
plot(locs.laea)

# Projecting shapefile of countries
country <- readShapePoly("ne_110m_admin_0_countries", IDvar=NULL,
proj4string=crs.geo)     # downloaded from Natural Earth website
plot(country)    # in geographical projection
country.laea <- spTransform(country, crs.laea)  # project

# Plotting
plot(locs.laea, pch=20, col="steelblue")
plot(country.laea, add=T)
# define spatial limits for plotting
plot(locs.laea, pch=20, col="steelblue", xlim=c(1800000, 3900000),
ylim=c(1000000, 3000000))
plot(country.laea, add=T)

#####################

### Overlay

ov <- overlay(locs.laea, country.laea)
countr <- country.laea@data$NAME[ov]
summary(countr)
```

8 Using Raster Data

```
### USING RASTER (GRID) DATA ####


### DOWNLOADING DATA
tmin <- getData("worldclim", var="tmin", res=10)   # this will download global
data on minimum temperature at 10 min resolution
  # can also get other climatic data, elevation, administrative boundaries, etc

### LOADING A RASTER LAYER
tmin1 <- raster("~/UsingR-GIS/wc10/tmin1.bil")   # Tmin for January
fromDisk(tmin1)  # values are stored on disk instead of memory! (useful for
large rasters)
tmin1 <- tmin1/10    # Worldclim temperature data come in decimal degrees
tmin1    # look at the info
plot(tmin1)

?raster    # raster reads many different formats, including Arc ASCII grids or
netcdf files



### CREATING A RASTER STACK (collection of many raster layers with the same
projection, spatial extent and resolution)
library(gtools)
list.ras <- mixedsort(list.files("~/UsingR-GIS/wc10/", full.names=T,
pattern=".bil"))
list.ras   # I have just collected a list of the files containing monthly
temperature values
tmin.all <- stack(list.ras)
tmin.all
tmin.all <- tmin.all/10
plot(tmin.all)

# BRICKS
tmin.brick <- brick(tmin.all)   # a rasterbrick is similar to a raster stack
(i.e. multiple layers with the same extent and resolution), but all the data
must be stored in a single file



### CROP RASTERS
```

```
plot(tmin1)
newext <- drawExtent()    # click on the map
tmin1.c <- crop(tmin1, newext)
plot(tmin1.c)

newext2 <- c(-10, 10, 30, 50)   # alternatively, provide limits
tmin1.c2 <- crop(tmin1, newext2)
plot(tmin1.c2)

tmin.all.c <- crop(tmin.all, newext)
plot(tmin.all.c)
```

8.1 Projections

```
### DEFINE PROJECTION
crs.geo    # defined above
projection(tmin1.c) <- crs.geo
projection(tmin.all.c) <- crs.geo
tmin1.c    # notice info info at coord.ref.

### CHANGING PROJECTION
tmin1.proj <- projectRaster(tmin1.c, crs="+proj=merc +lon_0=0 +k=1 +x_0=0
+y_0=0 +a=6378137 +b=6378137 +units=m +no_defs")
tmin1.proj   # notice info info at coord.ref.
plot(tmin1.proj)
# can also use a template raster, see ?projectRaster



### PLOTTING
histogram(tmin1.c)
pairs(tmin.all.c)
persp(tmin1.c)
contour(tmin1.c)
contourplot(tmin1.c)
levelplot(tmin1.c)
plot3D(tmin1.c)
bwplot(tmin.all.c)
densityplot(tmin1.c)



### Spatial autocorrelation
```

```
Moran(tmin1.c)    # global Moran's I
tmin1.Moran <- MoranLocal(tmin1.c)
plot(tmin1.Moran)


### EXTRACT VALUES FROM RASTER
View(locs)    # we'll obtain tmin values for our points
locs$tmin1 <- extract(tmin1, locs)    # values are incorporated to the
dataframe
View(locs)

# extract values for a given region
plot(tmin1.c)
reg.clim <- extract(tmin1.c, drawExtent())
summary(reg.clim)

# rasterToPoints
tminvals <- rasterToPoints(tmin1.c)
View(tminvals)

## CLICK: get values from particular locations in the map
plot(tmin1.c)
click(tmin1.c, n=3)   # click n times in the map



### RASTERIZE POINTS, LINES OR POLYGONS
locs2ras <- rasterize(locs.gb, tmin1)
locs2ras
plot(locs2ras, xlim=c(-10,10), ylim=c(45, 60), legend=F)
plot(wrld_simpl, add=T)


### CHANGING RESOLUTION (aggregate)
tmin1.lowres <- aggregate(tmin1.c, fact=2, fun=mean)
tmin1.lowres
tmin1.c     # compare
par(mfcol=c(1,2))
plot(tmin1.c, main="original")
plot(tmin1.lowres, main="low resolution")
dev.off()
```

8.2 Spline Interpolation

```
### SPLINE INTERPOLATION
xy <- data.frame(xyFromCell(tmin1.lowres, 1:ncell(tmin1.lowres)))    # get
raster cell coordinates
View(xy)
vals <- getValues(tmin1.lowres)
require(fields)
spline <- Tps(xy, vals)    # thin plate spline
intras <- interpolate(tmin1.c, spline)
intras
plot(intras)
intras <- mask(intras, tmin1.c)
plot(intras)


# SETTING ALL RASTERS TO THE SAME EXTENT, PROJECTION AND RESOLUTION ALL IN ONE
library(climstats)
?spatial_sync_raster
```

8.3 Elevations

```
### ELEVATIONS: Getting slope, aspect, etc
elevation <- getData('alt', country='ESP')
x <- terrain(elevation, opt=c('slope', 'aspect'), unit='degrees')
plot(x)

slope <- terrain(elevation, opt='slope')
aspect <- terrain(elevation, opt='aspect')
hill <- hillShade(slope, aspect, 40, 270)
plot(hill, col=grey(0:100/100), legend=FALSE, main='Spain')
plot(elevation, col=rainbow(25, alpha=0.35), add=TRUE)


### SAVING AND EXPORTING DATA

# writeraster
writeRaster(tmin1.c, filename="tmin1.c.grd")   # can export to many different
file types
writeRaster(tmin.all.c, filename="tmin.all.grd")
```

```
# exporting to KML (Google Earth)
tmin1.c <- raster(tmin.all.c, 1)
KML(tmin1.c, file="tmin1.kml")
KML(tmin.all.c)     # can export multiple layers
```

9 References

```
### To learn more ###

# Packages help and vignettes, especially
http://cran.r-project.org/web/packages/raster/vignettes/Raster.pdf
http://cran.r-project.org/web/packages/dismo/vignettes/sdm.pdf
http://cran.r-project.org/web/packages/sp/vignettes/sp.pdf

# CRAN Task View: Analysis of Spatial Data
http://cran.r-project.org/web/views/Spatial.html

# R-SIG-Geo mailing list
https://stat.ethz.ch/mailman/listinfo/R-SIG-Geo

# R wiki: tips for spatial data
http://rwiki.sciviews.org/doku.php?id=tips:spatial-data&s=spatial

# book
http://www.asdar-book.org/

############################################################################

Created by Pretty R at inside-R.org
```