Some notes on the SP package, mapping in R

Jooil Kim

A little about my personal adventure in spatial data

- The need for processing gridded data
    - Importing gridded data from model output or data sources on the web
    - Creating gridded data for model input
    - Manipulating the gridded data (ex. Changing data magnitudes or replacing data, resizing the grid)
    - Creating visual maps of the gridded data
    - Usually global grids

Short note on gridded data formats

- **“Matrix” representation**
    - Large m-by-n matrix representing a m-by-n grid
    - Each data entry in the matrix represents the corresponding data in the original grid
    - Conceptually easy, but only 1 data per matrix, and only for complete grids
    - Ex. ASCII grid format, raster(?)
- **“Table” representation**
    - Table of coordinates and corresponding data
    - More efficient, as you only need lines for which data are present
    - Flexible, complex representation possible. Ex) multiple data in one grid file
    - Ex. Net-CDF, SP data formats

The SP package

- One of many classes for storing spatial data in R
- Well-supported within R
  - Most popular? Lots of examples
- “Requires” rgdal to be installed
- Good resource from the UCSD library: “Applied spatial data analysis with R”, available as e-book
- Another good resource: http://www.rspatial.org
- Under continuous development (ex. code that used to work breaks after update, cryptic messages about end-of-life, etc.)
- To be replaced by SF (F = “Feature”), to be more compatible with GIS?

About rgdal

- GDAL: Geospatial Data Abstraction Library (http://gdal.org)
- rgdal: A way to use GDAL within R
- Compiled rgdal library available for Mac, but outdated? Not sure for other OS's

```
setRepositories(ind=1:2)
install.packages("rgdal")
```

The data types of SP (see vignette for more info)

	Spatial Info. Only	+ Data
points, lines	SpatialPoints, SpatialLines	SpatialPointsDataFrame, SpatialLinesDataFrame
polygons	SpatialPolygons	SpatialPolygonsDataFrame
pixels	SpatialPixels	SpatialPixelsDataFrame
grids	SpatialGrid	SpatialGridDataFrame

From the sp package vignette:
Pixels: Can be partial, unordered, stores explicit coordinates
Grids: Full grids

Importing spatial data

- `readAsciiGrid` (*maptools* package)
- `read.asc, sp.from.asc, etc.` (*SDMTools* package)
- `nc_open, ncvar_get, nc_close` (*ncdf4* package)
- Importing data into data.frame with *lat*, *lon*, [*value*] columns, and creating your own SP object

```
emissions_all <- read.table(filedir_psource, sep = ",",
stringsAsFactors = F, header = T, colClasses = c("numeric",
"numeric", "numeric"), skip = 2, blank.lines.skip = T)

names(emissions_all) <- c("lat", "lon", "likely.low",
“likely.high")

coordinates(emissions_all) <- c("lon", "lat")
proj4string(emissions_all) <- " +proj=longlat +datum=WGS84
+ellps=WGS84 +towgs84=0,0,0”
```

Useful spatial data resources

- **Socioeconomic Data and Applications Center, NASA/CIESIN**
  - http://sedac.ciesin.columbia.edu/data/sets/browse
  - Ex. Gridded Population of the World (GPW) product, National Identifier Grid maps, Land/Water area maps
  - ASCII grid
- **GADM**
  - https://gadm.org/index.html
  - Administrative boundaries in spatial data format, spanning multiple layers (ex. United states > State > County)
  - Data can be downloaded in R format (sp or sf)
  - Careful not to click on the banner ads!

“over” function

- Finding overlapping data between two sp objects (ex. sGDF of global population + sPolygon of border of country)
- `over` only works on spatial points

```
input_sPointsDF <- as(input_sGDF, "SpatialPointsDataFrame")
matchindex_c <- over(input_sPointsDF, poly_region)

matchindex_c[which(is.na(matchindex_c))] <- 0 # Get rid of NAs
matchindex_c <- which(matchindex_c > 0)

Total_region <- sum(input_sPointsDF$att[matchindex_c], na.rm = T)
```

Mapping sGDF’s

- *rworldmap*: `mapGriddedData`, can be useful if it provides what you need, works on sGDF’s
- *lattice* package: `levelplot`, `contourplot`, works on sGDF’s
- *ggplot*: Need to convert sGDF into data.frame
- *ggmap*: Functions to bring in map data from Google Maps, etc.

Mapping sGDF's

```
df_i <- as.data.frame(sGDF_emis[[attr]])
names(df_i) <- c("value")

df_i$long <- coordinates(sGDF_emis)[,1]
df_i$lat <- coordinates(sGDF_emis)[,2]

p <- ggplot(df_i, aes(x = long, y = lat)) +
       theme_bw() +
       theme(legend.key.height = unit(4, "line")) +
       coord_cartesian(xlim = xlim, ylim = ylim, expand = FALSE) +
       scale_fill_gradientn(colours = map_colourPalette, na.value = NA, name =
scale_caption, trans = "log10", limits = c(L_limit, H_limit))

p <- p + geom_raster(aes(fill = value), interpolate = F)

#### Draw map land contour border on map
#library(mapdata)
#p <- p + borders("world", color = “grey70")
#p <- p + borders("worldHires", colour = "grey70")

worldmap <- readShapeSpatial("/Users/jooil/Downloads/ne_110m_land/
ne_110m_land.shp") # need to probably fix this...
worldmap <- fortify(worldmap)
p <- p + geom_path(data = worldmap, aes(x = long, y = lat, group = group), colour =
"grey70")
```