xSub-GIS Tutorial

February 13, 2019

Merging xSub data with GIS spatial geometries

xSub data are designed for compatibility with Geographic Information Systems (GIS). Each data file contains a spatial unit ID that can be used to merge the event data to GIS spatial geometry files, like shapefiles (.shp), and R SpatialPolygons.

Below are instructions for merging xSub data files with two types of spatial geometries: administrative units and PRIO-GRID cells.

Load packages

We will be using three R packages in this example: `maptools` (for handling spatial geometries and visualization), `rgdal` (for importing shapefiles) and `xSub` (for downloading and importing xSub data files directly from the server).

```
library(maptools)
library(rgdal)
library(xSub)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: TRUE
```

```
## rgdal: version: 1.3-6, (SVN revision 773)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: /usr/share/gdal/2.2
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: (autodetected)
##  Linking to sp version: 1.3-1
```

If you're missing any of these packages, you'll need to install them first, using `install.packages("maptools", dependencies=TRUE)`, etc.

Merge xSub to GADM administrative units

NOTE: xSub is currently compatible with GADM 2.8. Future updates will also be compatible with the current GADM version (3.6), which has a different set of ID variables for matching. In the meantime, we will use the archived versions of GADM 2.8 (https://gadm.org/download_country_v2.html).

Let's begin by loading data from xSub (e.g. UCDP-GED spatial panel data on Afghanistan, district-year level).

```
X <- get_xSub(data_source = "GED",country_iso3 = "AFG",space_unit = "adm2",time_unit = "year")
```

This should create a new object in the R environment, `X`: a data.frame.

Now let's add data on administrative boundaries. We begin by creating temporary file to download GADM boundaries.

```
temp <- tempfile()
```

Download the GADM 2.8 boundaries file for the same country (Afghanistan, or `AFG`) and spatial unit (district, or `adm2`).

```
download.file(url = "https://biogeo.ucdavis.edu/data/gadm2.8/rds/AFG_adm2.rds",destfile = temp)
```

This code downloads the file `"AFG_adm2.rds"` sirectly from GADM's servers, and writes it to a temporary file.

If, instead, you want to create a permanent local copy, just replace the latter argument with the directory/path where you would like to save the file (e.g. `destfile="~/Downloads/AFG_adm2.rds"`). Alternatively, you may use a web browser to manually download the file from GADM's website: https://gadm.org/download_country_v2.html (please use version 2.8 of GADM).

Now let's load the GADM boundaries into R:

```
map <- readRDS(file = temp)
```

The file we downloaded was in .rds format. If you downloaded the shapefile, use `rgdal:readOGR()` to import into R. If you created a permanent local copy, just substitute in the directory/path where you saved the file (e.g. `file="~/Downloads/AFG_adm2.rds"`).

Now we're almost ready to merge. But two quick steps before we can do so.

First, because the `X` object is a panel dataset (same units observed at multiple points in time), we can only visualize one time slice at once. So, let's extract data just for 2015.

```
X_2015 <- X[X$YEAR==2015,]
```

You'll want to modify this if you're working with a different temporal unit (e.g. for months, this could be `X$YRMO==201501` for January of 2015).

Second, let's drop overlapping columns.

```
X_2015 <- X_2015[,c("ID_2",setdiff(names(X_2015),names(map)))]
```

This code elimantes duplicate ID columns that appear in both xSub and GADM, while keeping the column used for merging (`"ID_2"` in the adm2 case). See http://cross-sub.org/about/gis-compatibility for a full list of ID columns for various spatial units.

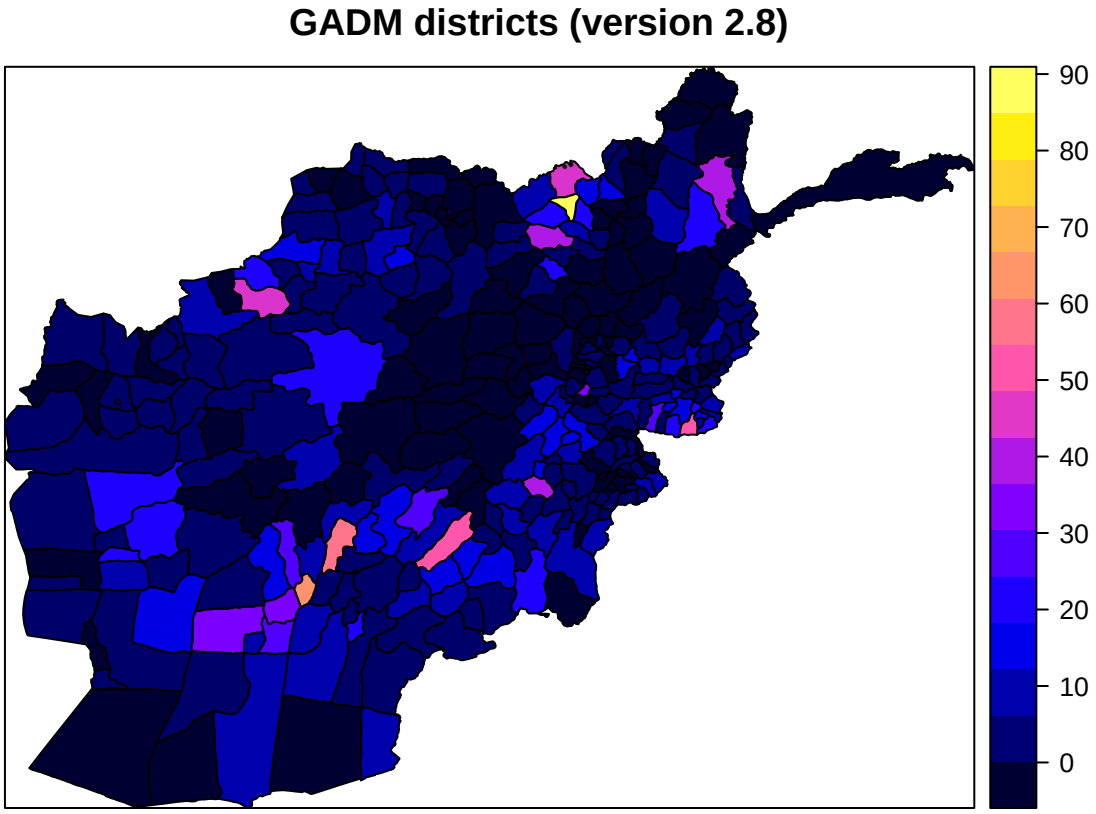
Now we can finally merge by the common ID variable `"ID_2"`:

```
map_2015 <- merge(map,X_2015,by="ID_2")
```

Hooray! If all went according to plan, this should have created a new SpatialPolygonsDataFrame object `map_2015`, which contains both the polygons from `map` and the covariates from `X_2015`.

Let's plot this to make sure everything makes sense:

```
spplot(map_2015,zcol="ACTION_ANY",main="GADM districts (version 2.8)")
```



This function plots the object `map_2015`, visualizing the variable `"ACTION_ANY"`.

Try testing this out with other years, or other countries and data sources.

Merge xSub to PRIO-GRID

Now we will attempt the same thing, but with PRIO-GRID cells as units of analysis. This involves a couple extra steps.

We will begin by downloading data from xSub as before, but with `"priogrid"` instead of `"adm2"` as the spatial unit:

```
X <- get_xSub(data_source = "GED",country_iso3 = "AFG",space_unit = "priogrid",time_unit = "year")
```

Create a new temporary file and directory to download PRIO-GRID's basegrid shapefile (zipped):

```
temp <- tempfile()
temp.dir <- tempdir()
```

Download and unzip the PRIO-GRID shapefile:

```
download.file(url = "http://file.prio.no/ReplicationData/PRIO-GRID/priogrid_shapefiles.zip",
              destfile = temp)
unzip(temp,exdir = temp.dir)
```

This code downloads the file `"priogrid_shapefiles.zip"` directly from PRIO's servers, and writes it to a temporary file, and unzips it to a temporary directory. If, instead, you want to create a permanent local copy, just replace the latter argument with the directory/path where you would like to save the

file (e.g. `destfile="~/Downloads/priogrid_shapefiles.zip"`) and where you'd like to extract it (e.g. `exdir="~/Downloads"`). Alternatively, you may paste the url directly into a web browser to manually download and unzip the file.

Let's now read the shapefile into R, using the `readOGR()` function from the `rgdal` library. If you have trouble installing this library, you may also use `sf::st_read()` or `readShapePoly()` (although the latter is deprecated).

```
map <- rgdal::readOGR(dsn = temp.dir,layer = "priogrid_cell")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "/tmp/RtmpRJQpWI", layer: "priogrid_cell"
## with 259200 features
## It has 5 fields
## Integer64 fields read as strings:  gid col row
```

Rename PRIO-GRID variables to same format as xSub:

```
names(map) <- paste0("PRIO_",toupper(names(map)))
```

Subset global grid by spatial extent of xSub data:

```
map <- map[map$PRIO_GID%in%X$PRIO_GID,]
```

This last step is necessary because the original `map` object was global in extent, and we're only looking at one country here.

As before, we will extract xSub data just for 2015:

```
X_2015 <- X[X$YEAR==2015,]
```

As before, we also drop overlapping columns, except for the column used for merging (`"PRIO_GID"`).

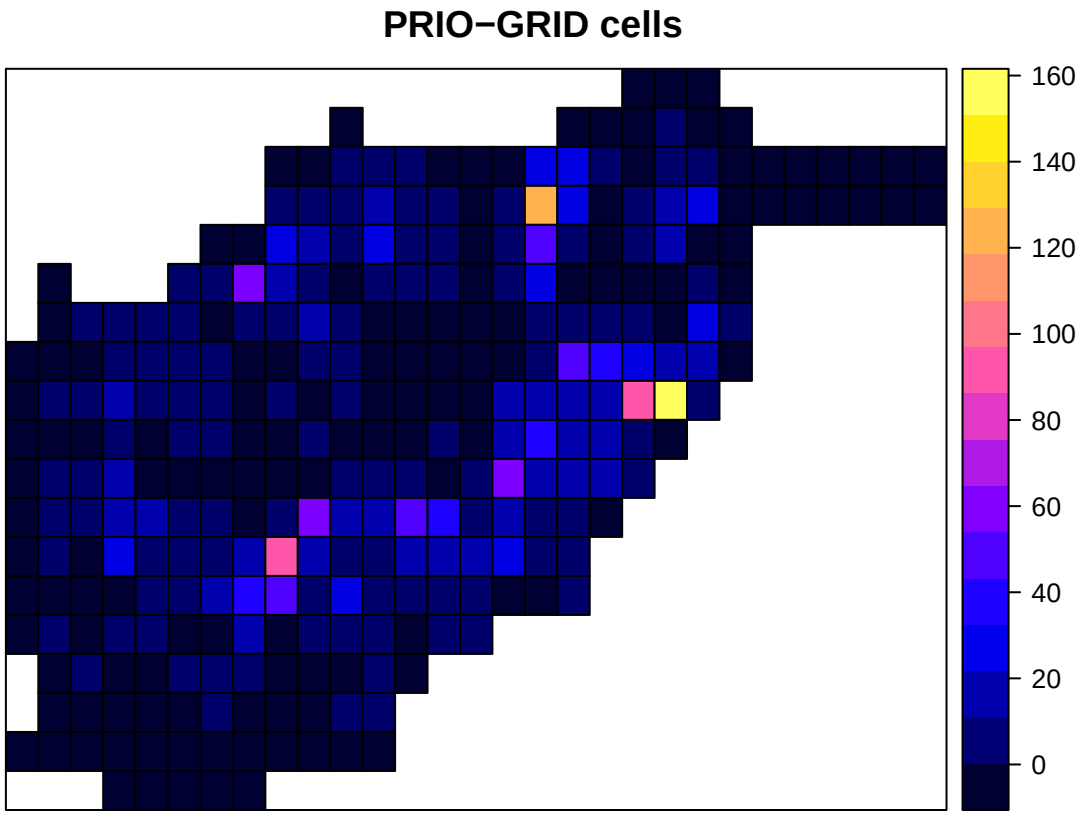
```
X_2015 <- X_2015[,c("PRIO_GID",setdiff(names(X_2015),names(map)))]
```

Merge by common ID variable:

```
map_2015 <- merge(map,X_2015,by="PRIO_GID")
```

And plot it to make sure everything makes sense:

```
spplot(map_2015,zcol="ACTION_ANY", main="PRIO-GRID cells")
```



And we're done!

Questions, comments, suggestions: xsub-project@umich.edu