Shapefiles to Raster Cell Indices

Matthew Leonawicz

December 24, 2014

1 Introduction

R code is provided showing how I convert polygon shapefiles to lists of raster cell indices.

1.1 Motivation

At SNAP I often find myself performing large numbers of data extractions on raster layers by way of shapefiles. This can be time consuming with respect to our high-resolution downscaled geotiffs. Large raster layers in combination with large (or large numbers of) shapefiles can make for the perfect storm of wasteful and painfully slow processing time. In light of the additional fact that such processing is commonplace at SNAP and that almost any data extraction done once is bound to recur at a later date in some overlapping and redundant sense, I have moved toward an a priori establishment of the more preliminary and repetitive aspects of common spatial data extraction tasks at SNAP.

1.2 Details

I compile the following:

* Lists of commonly used groups of polygon shapefiles * Key raster/geotiff format data sets from which we commonly extract data * Settings pertaining to the methods and circumstances under which cell indexing can occur in anticipated, subsequent data extraction exercises

1.2.1 Capabilities

The most straightforward purpose here is to obtain a nested list of groups of related geographic regions based on input shapefiles, each such lower list element storing a vector of cell numbers, or indices, pertaining to a given shapefile's spatial location over a given rasterized data set. The nested list is named. The top level elements are themselves lists, named based on related shapefile groupings. Each of these lists is originally a list of shapefiles but is transformed into a list of vectors of raster layer cell indices.

Sometimes data extraction is performed on SNAP data sets with explicit a priori removal of `NA` values. Anticipating this context of usage, cell indices corresponding to shapefiles are also obtained with respect to an NA-omitted version of a raster layer.

Additionally, the context of data extraction procedures sometimes includes sampling methods where not all raster cells are used. Sampling may be applied to rasterized objects in **R** prior to doing any other work with them. As with `NA` removal, this may be done to improve computational efficiencies. Under these circumstances, obtaining cell indices by shapefile in advance would not work. However, the code below also provides an example of one a priori sampling schemes, the five percent random sample. In anticipation of commonly performing such sampling on raster layers, here this is done in advance by obtaining a representative sample for each shapefile region.

1.2.2 Limitations

Obviously, the vector of cell indices for a shapefile differs for different rasterized data sets. At this time, cell index nested lists are produced for two common rasterized data products at SNAP:

* Alaska-Canada 2-km downscaled climate data * Alaska-Canada 1-km Alfresco simulation data

Although the extents are identical, resolutions differ, and hence so do cell numbers pertaining to any given polygon shapefile. Furthermore, taking the climate and Alfresco datasets as an example again, even if they were the same resolution, the Alfresco map has spatial regions of `NA` where data values exist in the climate layers. This would have an independent effect on the NA-omitted versions of the nested lists. It would also affect sampled versions where `NA` cells are removed.

These are not really limitations, but trivial facts. The thing to remain aware of is simply that any version of these polygon shapefile-specific nested lists of cell numbers are of course indexed with respect to a specific rasterized dataset.

2 Related items

2.1 Files and Data

Input files include polygon shapefiles commonly used at SNAP and two of SNAP's current geotiff data products, Alaska-Canada 2-km downscaled climate data and 1-km Alfresco simulation outputs. Output files are **R** workspaces, .RData files. There is one workspace storing each version of a nested list for each type of rasterized data set.

2.2 Code flow

The code flow is simple for this task, only involving a single **R** script. However, a diagram is provided here to show it in the context of both input and output data.

ADD LATER

3 R code

3.1 Initial setup

Load required packages, define output directory, and load shapefiles. Shapefiles are organized into related groups. I ensure certain idiosyncrasies are addressed, such as reprojection of shapefiles with differing coordinate reference systems. Some shapefiles also contain single polygon regions whereas others contain multiple. Care must be taken to ensure all object manipulation is as intended.

```r
library(raster)
library(maptools)
library(parallel)

outDir <- "/workspace/UA/mfleonawicz/Leonawicz/Projects/2014/DataExtraction/workspaces"

# Political boundaries
Alaska_shp <- shapefile("/workspace/Shared/Users/mfleonawicz/shapefiles/Political/Alaska")
Alberta_shp <- shapefile("/workspace/Shared/Users/mfleonawicz/shapefiles/Political/alberta_albers")
BC_shp <- shapefile("/workspace/Shared/Users/mfleonawicz/shapefiles/Political/BC_albers")

# Alaska ecoregions
eco32_shp <- shapefile("/workspace/Shared/Users/mfleonawicz/shapefiles/AK_ecoregions/akecoregions")
eco32_shp <- spTransform(eco32_shp, CRS(projection(Alaska_shp)))
eco9_shp <- unionSpatialPolygons(eco32_shp, eco32_shp@data$LEVEL_2)
eco3_shp <- unionSpatialPolygons(eco32_shp, eco32_shp@data$LEVEL_1)

eco32_IDs <- gsub("\\.", "", as.data.frame(eco32_shp)[, 1])
eco9_IDs <- sapply(slot(eco9_shp, "polygons"), function(x) slot(x, "ID"))
eco3_IDs <- sapply(slot(eco3_shp, "polygons"), function(x) slot(x, "ID"))
```

```
# LCC regions
LCC_shp <- shapefile("/workspace/Shared/Users/mfleonawicz/shapefiles/LCC/LCC_summarization_units_singlep
LCC_IDs <- gsub(" LCC", "", gsub("South", "S", gsub("western", "W", gsub("Western",
    "W", gsub("North", "N", gsub("  ", " ", gsub("\\.", "", as.data.frame(LCC_shp)[,
        1])))))))
```

3.2 Organization and metadata

Lists of names and IDs must be created to prepare for cell index extraction by shapefile.

```
# organize shapefile lists and associated metadata
shp.names <- c("Political 0", "Political 1", "Political 2", "Political 3", "Alaska L3 Ecoregions",
    "Alaska L2 Ecoregions", "Alaska L1 Ecoregions", "LCC Regions")
shp.list <- list(Alaska_shp, Alberta_shp, BC_shp, eco32_shp, eco9_shp, eco3_shp,
    LCC_shp)
shp.IDs.list <- list("Alaska", "Alberta", "British Columbia", eco32_IDs, eco9_IDs,
    eco3_IDs, LCC_IDs)
region.names.out <- c(list(c("AK-CAN", unlist(shp.IDs.list[1:3]))), shp.IDs.list[4:length(shp.IDs.list)]
names(region.names.out) <- c("Political", shp.names[5:length(shp.names)])
```

3.3 Alfresco example

A representative map layer is loaded with the `raster` package. A nested list of cell numbers is obtained efficiently for several shapefiles by using `mclapply` from the `parallel` package. This is further processed with a call to `rapply` and then a full extent region is appended to the list (no shapefile was used here of course).

At this point, subsampling and/or `NA` removal is done, resulting, in this example, in four total versions of nested lists which can be used in conjunction with SNAP's Alfresco output geotiffs under various conditions of data extraction for any and all of the input spatial regions.

```
# For AK-CAN 1-km ALfresco extractions
dirs <- list.files("/big_scratch/apbennett/Calibration/FinalCalib", pattern = ".*.sres.*.",
    full = T)
r <- readAll(raster(list.files(file.path(dirs[1], "Maps"), pattern = "^Age_0_.*.tif$",
    full = T)[1]))  # template done
data.ind <- Which(!is.na(r), cells = T)
cells_shp_list <- mclapply(1:length(shp.list), function(x, shp, r) extract(r,
    shp[[x]], cellnumbers = T), shp = shp.list, r = r, mc.cores = 32)
cells_shp_list <- rapply(cells_shp_list, f = function(x, d.ind) intersect(x[,
    1], d.ind), classes = "matrix", how = "replace", d.ind = data.ind)
cells_shp_list <- c(list(c(list(data.ind), cells_shp_list[[1]], cells_shp_list[[2]],
    cells_shp_list[[3]])), cells_shp_list[-c(1:3)])  # Combine full domain and other political boundarie

n.shp <- sum(unlist(lapply(cells_shp_list, length)))
names(cells_shp_list) <- names(region.names.out)
for (i in 1:length(cells_shp_list)) names(cells_shp_list[[i]]) <- region.names.out[[i]]

cells_shp_list_5pct <- rapply(cells_shp_list, f = function(x, pct) sort(sample(x,
    size = pct * length(x), replace = FALSE)), classes = "integer", how = "replace",
    pct = 0.05)
cells_shp_list_rmNA <- rapply(cells_shp_list, f = function(x, n.cells, d.ind) which(c(1:n.cells %in%
    x)[d.ind]), classes = "integer", how = "replace", n.cells = ncell(r), d.ind = data.ind)
```

```
cells_shp_list_rmNA_5pct <- rapply(cells_shp_list_5pct, f = function(x, n.cells,
    d.ind) which(c(1:n.cells %in% x)[d.ind]), classes = "integer", how = "replace",
    n.cells = ncell(r), d.ind = data.ind)

save(cells_shp_list, region.names.out, n.shp, file = file.path(outDir, "shapes2cells_AKCAN1km.RData"))
save(cells_shp_list_5pct, region.names.out, n.shp, file = file.path(outDir,
    "shapes2cells_AKCAN1km_5pct.RData"))
save(cells_shp_list_rmNA, region.names.out, n.shp, file = file.path(outDir,
    "shapes2cells_AKCAN1km_rmNA.RData"))
save(cells_shp_list_rmNA_5pct, region.names.out, n.shp, file = file.path(outDir,
    "shapes2cells_AKCAN1km_rmNA_5pct.RData"))
```

3.4 Climate example

THe process for this data set is the same as above.

```
# For AK-CAN 2-km extractions
r <- readAll(raster("/Data/Base_Data/Climate/AK_CAN_2km/projected/AR5_CMIP5_models/rcp60/5modelAvg/pr/pr
data.ind <- Which(!is.na(r), cells = T)
cells_shp_list <- mclapply(1:length(shp.list), function(x, shp, r) extract(r,
    shp[[x]], cellnumbers = T), shp = shp.list, r = r, mc.cores = 32)
cells_shp_list <- rapply(cells_shp_list, f = function(x, d.ind) intersect(x[,
    1], d.ind), classes = "matrix", how = "replace", d.ind = data.ind)
cells_shp_list <- c(list(c(list(data.ind), cells_shp_list[[1]], cells_shp_list[[2]],
    cells_shp_list[[3]])), cells_shp_list[-c(1:3)])  # Combine full domain and other political boundarie

n.shp <- sum(unlist(lapply(cells_shp_list, length)))
names(cells_shp_list) <- names(region.names.out)
for (i in 1:length(cells_shp_list)) names(cells_shp_list[[i]]) <- region.names.out[[i]]

cells_shp_list_5pct <- rapply(cells_shp_list, f = function(x, pct) sort(sample(x,
    size = pct * length(x), replace = FALSE)), classes = "integer", how = "replace",
    pct = 0.05)
cells_shp_list_rmNA <- rapply(cells_shp_list, f = function(x, n.cells, d.ind) which(c(1:n.cells %in%
    x)[d.ind]), classes = "integer", how = "replace", n.cells = ncell(r), d.ind = data.ind)
cells_shp_list_rmNA_5pct <- rapply(cells_shp_list_5pct, f = function(x, n.cells,
    d.ind) which(c(1:n.cells %in% x)[d.ind]), classes = "integer", how = "replace",
    n.cells = ncell(r), d.ind = data.ind)

save(cells_shp_list, region.names.out, n.shp, file = file.path(outDir, "shapes2cells_AKCAN2km.RData"))
save(cells_shp_list_5pct, region.names.out, n.shp, file = file.path(outDir,
    "shapes2cells_AKCAN2km_5pct.RData"))
save(cells_shp_list_rmNA, region.names.out, n.shp, file = file.path(outDir,
    "shapes2cells_AKCAN2km_rmNA.RData"))
save(cells_shp_list_rmNA_5pct, region.names.out, n.shp, file = file.path(outDir,
    "shapes2cells_AKCAN2km_rmNA_5pct.RData"))
```