Diversity Metrics Using Sparse Matrices

Cory

8/25/2020

Background

NOTE: this file is the development version; the currently stable vignette is fastDiversityLite.rmd

> Details on computations will be stored in text blocks like this. They can safely be ignored if you're just getting started.

Set up

Pacakges you'll need.

```
library(changeRangeR)
library(raster)
library(rasterVis)
library(rgdal)
library(Matrix.utils)
library(tidyverse)
if(Sys.info()["sysname"]== "Windows") library (parallelsugar)
mc.cores=6
```

Main Inputs

```
### determine where you want outputs
summaryBaseDir='/Volumes/cm2/changeRangerDemos/trees190_test6'
if(!file.exists(summaryBaseDir)) dir.create(summaryBaseDir)

# Indicate scenarios. these names will be used throughout to structure folders
allScen=c('present','8580')
#load environnent with the reprojected projection (typically the one you used for modeli
ng). You only need the raster grid that , not the actual layer values
envGrid=raster::stack(system.file("extdata/treeDemo/envGrid.tif",package='changeRangeR'
))

# folder of binary range rasters
myDir=system.file("extdata/treeDemo/BinaryMaps",package='changeRangeR')

# shapefiles for plotting. This one comes preinstallted
world.shp=readOGR(system.file(
  "extdata/treeDemo/TM_WORLD_BORDERS_SIMPL-0.3/TM_WORLD_BORDERS_SIMPL-0.3.shp",
  package='changeRangeR'),'TM_WORLD_BORDERS_SIMPL-0.3',verbose=F)
world.shp2=spTransform(world.shp,projection(envGrid))
```

Set up directory structure

Set up a standardized set of directory names and structure so that that `changeRanger` functions can access the right files internally. You can specify any optional subdirectories you like, although `changeRanger` will only write outputs there.

```
sumDirs=setupSummaryDirectories(summaryBaseDir, optionalSubDirs=c('funcDiv','phyloDiv'))
# its a good idea to save this in case you need to start an analysis in the middle
saveRDS(sumDirs,file=paste0(sumDirs$sumBaseDir,'/dirList.rds'))
str(sumDirs)
```

```
## List of 12
##  $ myBaseDir   : chr "/Volumes/cm2/changeRangerDemos/trees190_test6"
##  $ sumBaseDir  : chr "/Volumes/cm2/changeRangerDemos/trees190_test6"
##  $ figDir      : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/Figures"
##  $ attrDir     : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/AttributeTables"
##  $ sparseDir   : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/SparseMatrices"
##  $ cbsDir      : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/SparseMatrices/Ce
llBySpecies"
##  $ rangeSizeDir: chr "/Volumes/cm2/changeRangerDemos/trees190_test6/RangeSize"
##  $ richDir     : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/Richness/"
##  $ envDir      : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/Env/"
##  $ miscDir     : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/Misc/"
##  $ funcDivDir  : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/funcDiv"
##  $ phyloDivDir : chr "/Volumes/cm2/changeRangerDemos/trees190_test6/phyloDiv"
```

Make Sparse Matrices

Begin by converting the range rasters to a matrix with rows indexing cells and columns indexing species. With the `nCellChunks` argument, I'm specifying that the globe should be split up into 10 chunks; this is important when your cell x species matrix is large, because the whole thing can't be read into memory at once. 'Large' will

depend on the number of species, the resolution and extent of your maps, and your computer's capabilities.

```
# make a table of the raster file names and the associated species names
allSpeciesMaps=tibble(rasterFiles=list.files(paste0(myDir,'/present'),
                                             recursive=T, full.names=T)) %>%
  mutate(sp.names= rasterFiles %>% basename %>% file_path_sans_ext) %>%
  separate(sp.names,into=c('g','sp','t','s'),sep='_') %>% select(-t,-s) %>%
  unite(sp.names,g,sp)
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 31 rows [10,
## 11, 13, 14, 18, 23, 24, 32, 36, 66, 67, 74, 80, 85, 91, 112, 113, 129, 140,
## 144, ...].
```

```
# species index table. columns: species name, integer index
sp.ind=speciesIndexTable(allSpeciesMaps,sumDirs)
# cell index Table. columns: long, lat, cellid
cell.ind=cellIndexTable(envGrid,nCellChunks=10,sumDirs)
```

Two key outputs are:

- cellIndexTable.rds is a data frame where the row indices of the CBS matrices are stored along with the coordinates and cellIDs of your environmental grid.
- speciesIndexTable.rds is a data frame where the column indices of the CBS matrices are stored along with species names.

As we develop more complex analyses with auxilary data, we'll add 'cell attributes' and 'species attributes' to these data frames, respectively, that will allow us to generate a wide range of summaries.

```
cell.ind=readRDS(paste0(sumDirs$myBaseDir,'/cellIndexTable.rds'))
head(cell.ind)
```

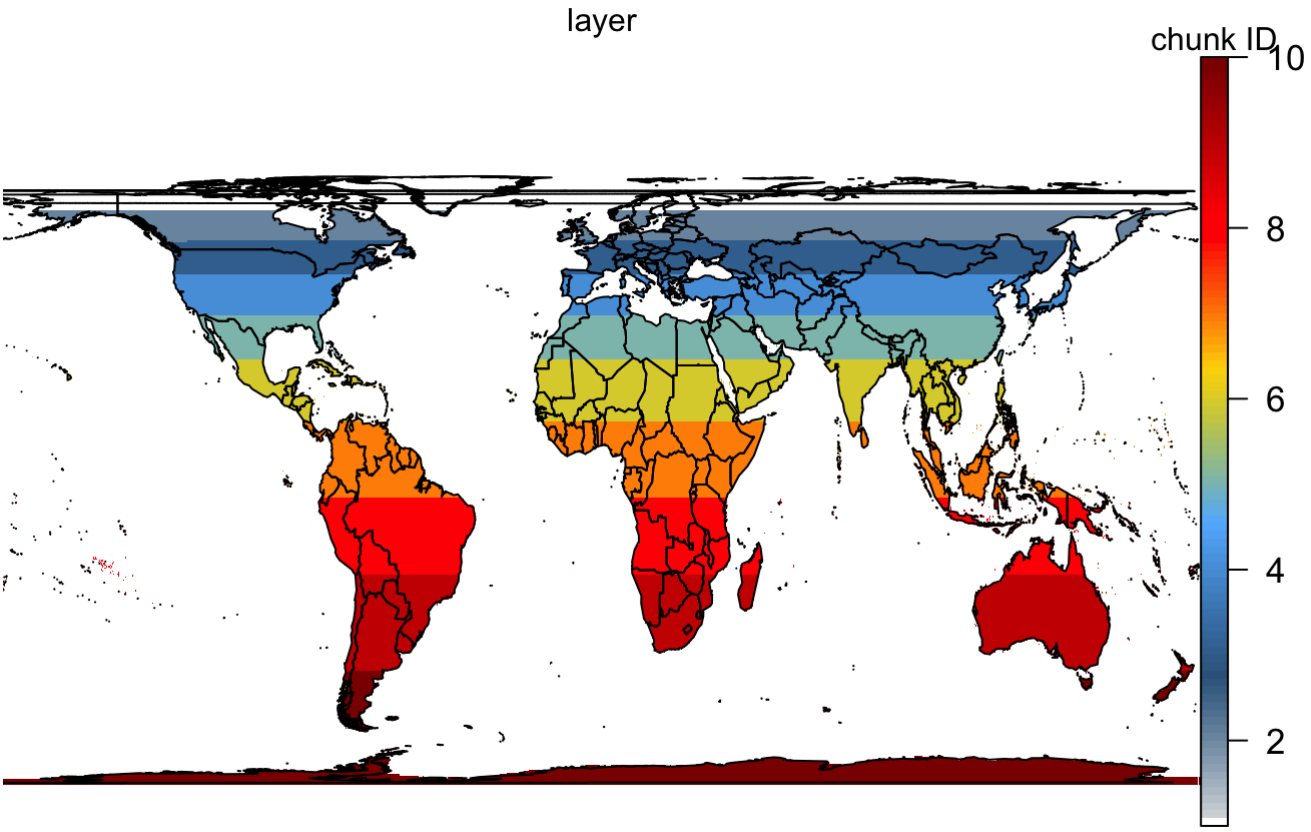
```
##          x       y cellID chunkID
## 1 -3454004 7961463  22972       1
## 2 -3446634 7961463  22973       1
## 3 -3439264 7961463  22974       1
## 4 -3431894 7961463  22975       1
## 5 -3424524 7961463  22976       1
## 6 -3417154 7961463  22977       1
```

```
sp.ind=readRDS(paste0(sumDirs$myBaseDir,'/speciesIndexTable.rds'))
head(sp.ind)
```

```
##                   species index
## 1  Coccoloba_argentinensis     1
## 2     Coccoloba_belizensis     2
## 3     Coccoloba_caracasana     3
## 4       Coccoloba_coriacea     4
## 5        Coccoloba_gentryi     5
## 6 Coccoloba_guanacastensis     6
```

You can see how the globe is broken up into chunks. `sparseToRaster` is also useful to convert anything stored as a sparse matrix back to a raster for spatial operations.

```
cell.ind=readRDS(paste0(sumDirs$myBaseDir,'/cellIndexTable.rds'))
# a convenient function to convert between sparse matrices and rasters, where plotting i
s easier
chunks.r=cellIDToRaster(cell.ind,envGrid,'chunkID')
# a convient plotting function in changeRanger
fdMapPlot(chunks.r,shp=world.shp2,legend.args=list(text='chunk ID'))
```



Now we're ready to make the sparse matrices. This step can take a little time, but makes all the downstream steps fast. It converted all the ranges into sparse matrices and stored them in the folder '…/SparseMatrices/CellBySpecies'. We refer to these as 'cell by species' matrices, or CBS matrices for short. Depending on file size, or your settings, CBS matrices can be stored in chunks to reduce the need to load in huge files in later operations where you may only need a subset of the chunks. Chunking is done based on rows; that is each contains a subset of cells, but all species.

```
for (scn in allScen){
  allSpeciesMaps=tibble(rasterFiles=list.files(paste0(myDir,'/',scn),
                                               recursive=T, full.names=T)) %>%
    mutate(spNames= rasterFiles %>% basename %>% file_path_sans_ext) %>%
    separate(spNames,into=c('g','sp','t','s'),sep='_') %>% select(-t,-s) %>%
    unite(spNames,g,sp)
  cellBySpeciesMatrices(sumDirs$cbsDir,
                        rasterFiles=allSpeciesMaps$rasterFiles,
                       spNames=allSpeciesMaps$spNames,
                        scenario=scn,
                        envGrid=envGrid,
                        sp.ind=sp.ind,
                       cell.ind=cell.ind,
                        nCellChunks=10, # number of chunks to split the envGrid into
                        mc.cores=mc.cores,
                        overwrite=T)
}
```

```
## Warning: Expected 4 pieces. Missing pieces filled with `NA` in 31 rows [10,
## 11, 13, 14, 18, 23, 24, 32, 36, 66, 67, 74, 80, 85, 91, 112, 113, 129, 140,
## 144, ...].
```

Metrics based only on ranges

In this section we only require that the user provide ranges for each species; auxilary information, such as other environmental layers, traits, or phylogeny is handled in subsequent sections.

Total richness

```
rich=lapply(allScen,function(scn){
  r=richnessFromCBS(cbsDir=sumDirs$cbsDir,
                    scenario=scn,env=envGrid,
                    mc.cores=mc.cores, outDir=sumDirs$richDir)
  # I like to store all the plots as I go
  fdMapPlot(stack(r),paste0(sumDirs$figDir,'/Richness_',scn,'.pdf'),shp=world.shp2,
            legend.args=list(text='Species Richness'))
  r
}) %>% stack
```

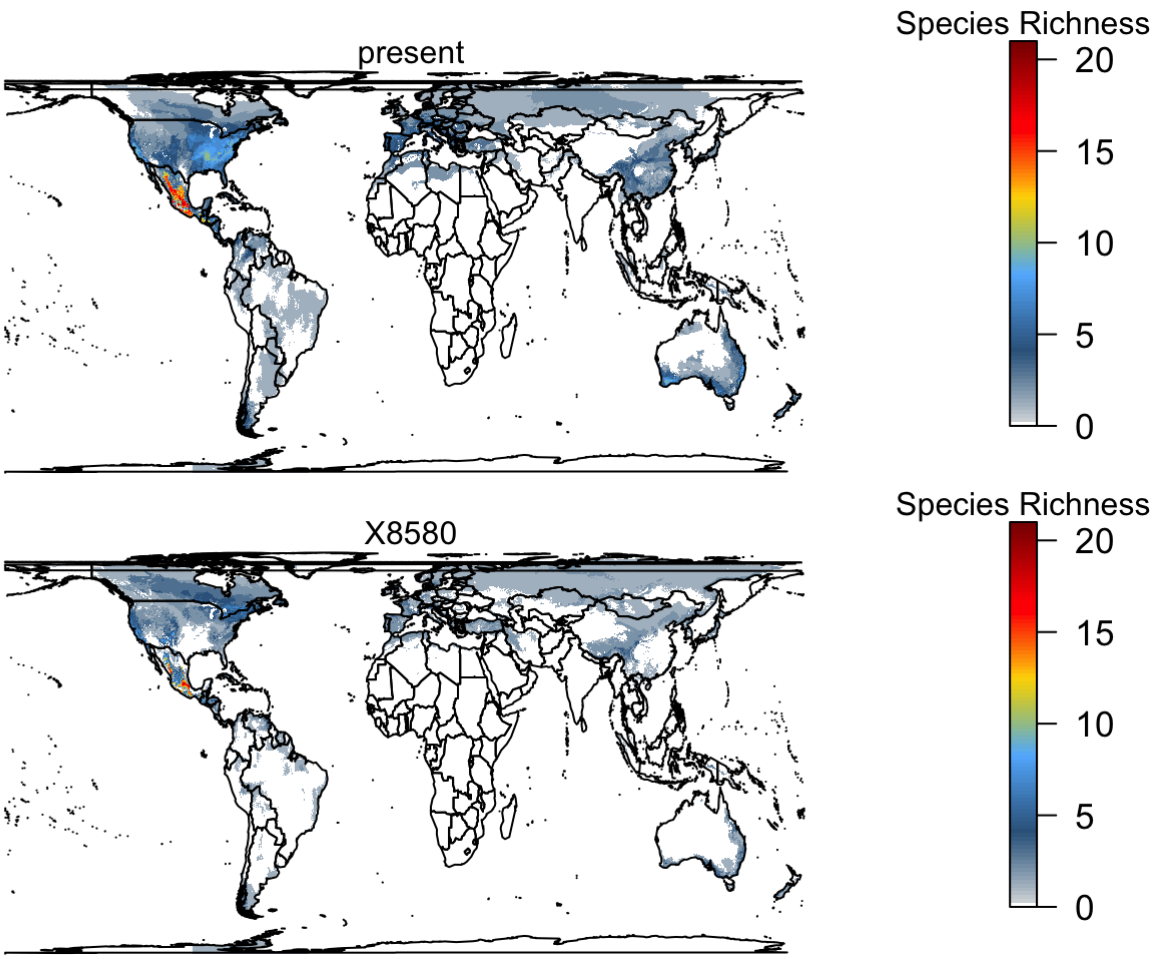
```
## starting present
```

```
## 2.73 s
```

```
## starting 8580
```

```
## 2.61 s
```

```
fdMapPlot(rich,shp=world.shp2,legend.args=list(text='Species Richness'))
```



Calculation: This is calculated based on column sums of the CBS matrix

Range Size

Range size can be defined in a variety of ways; we define it here as area of occurrence at the native resolution of the SDMS (here ~10km)

```
sp.ind=readRDS(paste0(sumDirs$sumBaseDir,'/speciesIndexTable.rds'))
ra=lapply(allScen,function(scn){
        rangeArea(cbsDir=sumDirs$cbsDir,scenario=scn,
                  sp.ind=sp.ind,outDir=sumDirs$rangeSizeDir,mc.cores=mc.cores)
})
```

```
## starting present
```

```
## 0.55 s
```

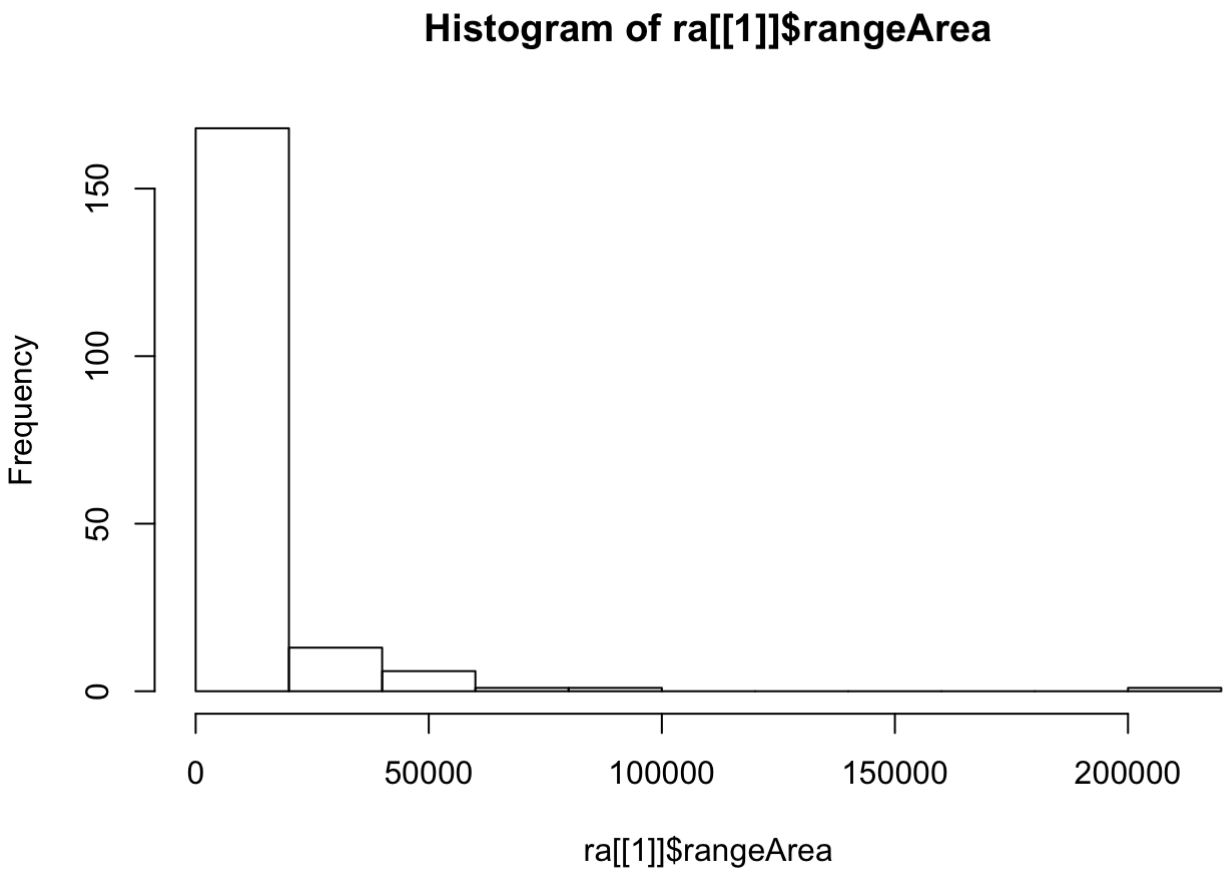
```
## starting 8580
```

```
## 0.6 s
```

```
str(ra)
```

```
## List of 2
##  $ :'data.frame':    190 obs. of  3 variables:
##   ..$ species  : Factor w/ 190 levels "Coccoloba_argentinensis",..: 1 2 3 4 5 6 7 8 9
10 ...
##   ..$ index    : int [1:190] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ rangeArea: num [1:190] 16397 2454 14068 174 75 ...
##  $ :'data.frame':    190 obs. of  3 variables:
##   ..$ species  : Factor w/ 190 levels "Coccoloba_argentinensis",..: 1 2 3 4 5 6 7 8 9
10 ...
##   ..$ index    : int [1:190] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ rangeArea: num [1:190] 2131 325 23388 0 0 ...
```

```
hist(ra[[1]]$rangeArea)
```



> **Calculation**: row sums over CBS matrices.

Now let's get in the habit of storing species attributes. Throughout this workflow, we'll repeatedly calculate something about a species, like it's range size, and it's helpful to store it in the `speciesIndexTable` for later use. Of course this is entirely optional, but just how I like to do it, so that I keep everything about a species in one place. The plan is to read in the speciesIndexTable, join the range areas just calculated to it, and then overwrite the old speciesIndexTable (you might also choose not to rewrite, and store this as a separate attribute table. Whatever.)

```
newSpInd= sp.ind %>% full_join(ra[[1]],by=c('species','index')) %>%
  rename(rangeAreaPresent=rangeArea) %>%
  full_join(ra[[2]],by=c('species','index')) %>%
  rename(rangeArea8580=rangeArea)
str(newSpInd)
```

```
## 'data.frame':    190 obs. of  4 variables:
##  $ species         : Factor w/ 190 levels "Coccoloba_argentinensis",..: 1 2 3 4 5 6 7
8 9 10 ...
##  $ index           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ rangeAreaPresent: num  16397 2454 14068 174 75 ...
##  $ rangeArea8580   : num  2131 325 23388 0 0 ...
```

Rarity

Here's an example using a 'species attribute', which is a common application. Here we use species attributes to refer to any property of an individual species. Common operations are to (1) summarize species attributes within a spatial unit, or (2) group species based on different values of an attribute and calculate a summary statistic. As an example of (1) we map species rarity (average value of 1/range area over species within a cell). Here's an example where we add a new place to store outputs, since rarity probably isn't common enough to include as a default.

```
sumDirs$rarityDir=file.path(sumDirs$rangeSizeDir,'Rarity')
if(!file.exists(sumDirs$rarityDir)) dir.create(sumDirs$rarityDir)

rar=lapply(allScen,function(scn){
    # generate the value of 1/range size for each species in an attribute table
    raritySpAttr=sumDirs$rangeSizeDir %>%
      list.files(full.names=T,pattern=scn) %>%
      readRDS %>%
      mutate(rarity=1/rangeArea) %>%
      select(-rangeArea)
    r=speciesAttributeByCell(cbsDir=sumDirs$cbsDir,scenario=scn,
                             attrTable=raritySpAttr, method='mean',
                             env=envGrid, outDir=sumDirs$rarityDir)
    fdMapPlot(log(r),plotFile=paste0(sumDirs$figDir,'/Rarity_',scn,'.pdf'),
              shp=world.shp2,legend.args=list(text='log(rarity)',line=2,side=4))
    r
}) %>% stack
```

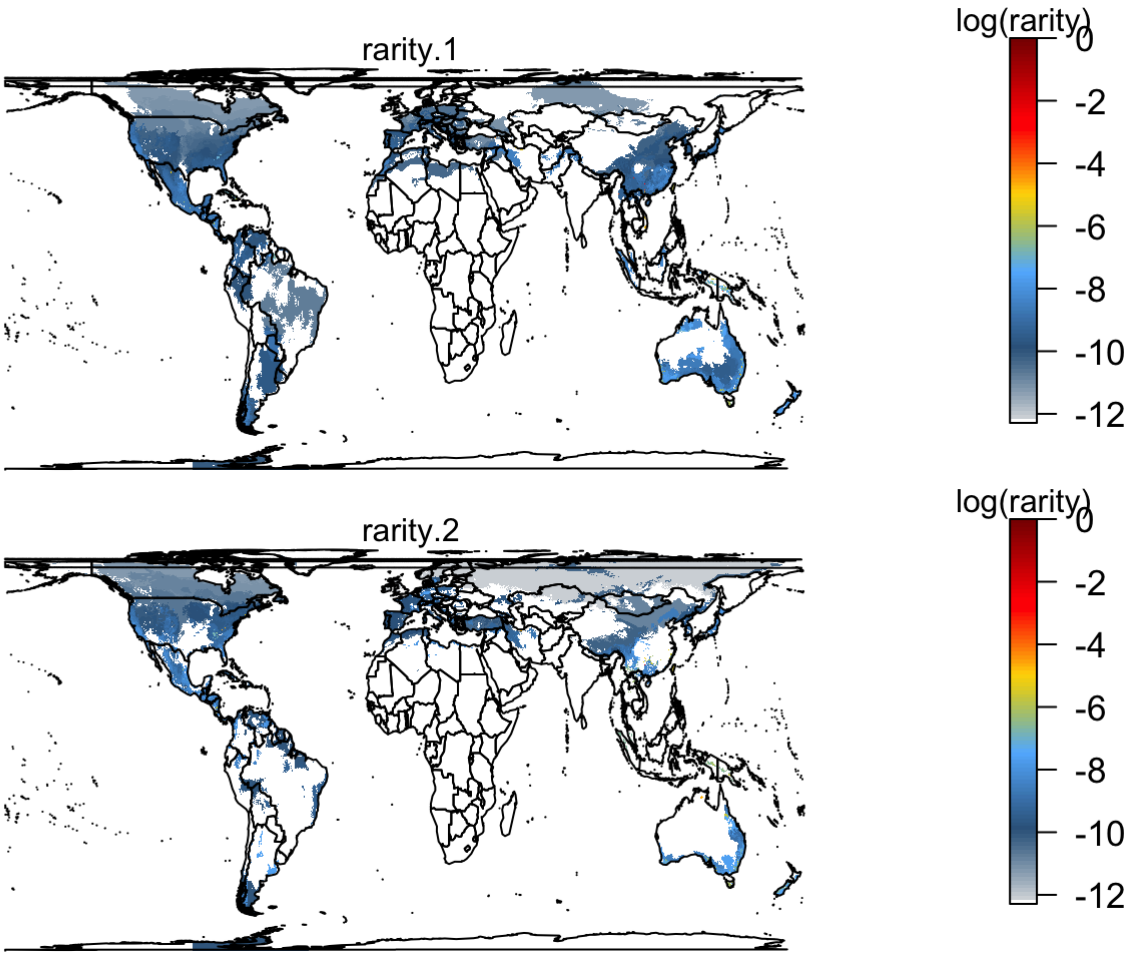
```
## starting present
```

```
## 2.66 s
```

```
## starting 8580
```

```
## 2.65 s
```

```
fdMapPlot(log(rar),shp=world.shp2,legend.args=list(text='log(rarity)'))
```



Calculation: Define a 1/range size as a species attribute. Use matrix multiplication of the CBS matrix and species attribute vector.

Other species and cell summaries

At present, we support three main types of operations to build summaries from CBS matrices over taxa or spatial units which can be combined in a variety of ways: (1) row/column sums, (2) collapsing rows/columns, and (3) moments over rows/columns.

The first type, **row/column sums** have already been demonstrated above. Column sums of the CBS matrices correspond to

The second type, **collapsing** operations, involves combining related rows or columns of the CBS matrix. When a summary statistic only requires that we know whether a given taxon (e.g., species) occurs in a given spatial unit (e.g., cell), we can make binary sparse matrices that collapse the CBS matrices over cells (rows) or species (columns). For example, if we want to study diversity patterns of genera, we can collapse the columns of the CBS matrices so that each represents a genus (rather than a species). Similarly if one wants to study the richness patterns in different countries, we can collapse the rows of the CBS matrices so that each row represents a country (rather than a cell). We define **species attribute tables** below that specify how species should be collapsed and **cell attribute tables** that specify how cells should be collapsed.

The third type, **moments** over rows/columns, currently supports estimating the mean value (eventually first four moments) of an attribute over rows or columns. For example, this could be the mean attribute of species that occur in the same cell; in fact we just saw this when calculating the cell-level rarity above by taking the mean of 1/range_size over species. Relatedly, as we'll demonstrate below, one might also be interested in the mean values of cells where a species occurs.

Richness by species attributes

Species might have different attributes, like growth form, clade, etc., such that it is useful to split up analyses by group. Here, we'll make a separate richness map for each genus. This corresponds to **row sum** operations, as described above.

```
sumDirs$richByGenusDir=file.path(sumDirs$richDir,'byGenus')
if(!file.exists(sumDirs$richByGenusDir)) dir.create(sumDirs$richByGenusDir)
```

We begin by looking at richness within each genus separately.

```
sp.ind=readRDS(paste0(sumDirs$sumBaseDir, '/speciesIndexTable.rds'))
genusSpAttr=sp.ind %>% select(species) %>%
  separate(species,c("genus",NA),sep='_')
sp.ind = sp.ind %>% bind_cols(genusSpAttr)
saveRDS(sp.ind,file=paste0(sumDirs$sumBaseDir,'/speciesIndexTable.rds'))
head(sp.ind)
```

```
##                    species index     genus
## 1  Coccoloba_argentinensis     1 Coccoloba
## 2     Coccoloba_belizensis     2 Coccoloba
## 3     Coccoloba_caracasana     3 Coccoloba
## 4       Coccoloba_coriacea     4 Coccoloba
## 5        Coccoloba_gentryi     5 Coccoloba
## 6 Coccoloba_guanacastensis     6 Coccoloba
```

```
richWithinGen=lapply(allScen,function(scn){
    r=richnessFromCBS(cbsDir=sumDirs$cbsDir,scenario=scn,
                      envGrid=envGrid,mc.cores=mc.cores,
                      attrTable=sp.ind,attrName='genus',outDir=sumDirs$richByGenusDir)
    names(r)=paste0(names(r),'_',scn)
    fdMapPlot(r,plotFile=paste0(sumDirs$figDir,'/RichnessByGenus_',scn,'.pdf'),
              shp=world.shp2, legend.args=list(text='# species',line=2,side=4))
    r
})
```

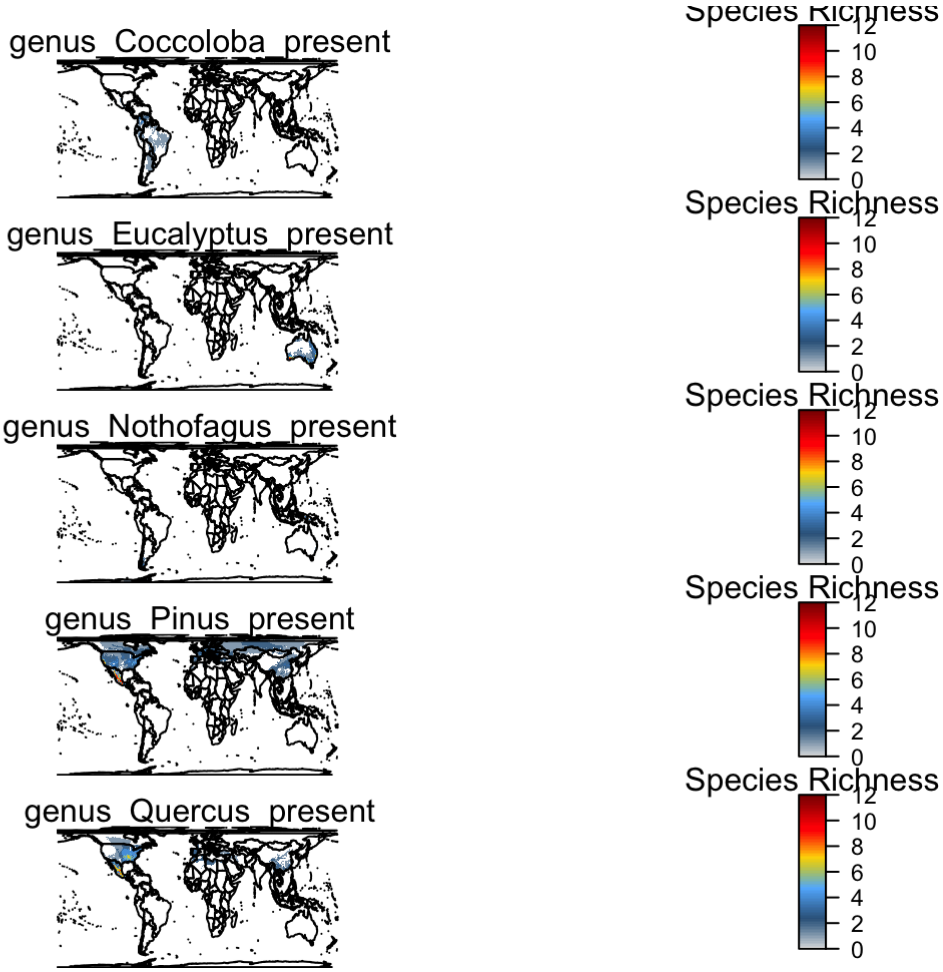
```
## starting present
```

```
## 12.24 s
```

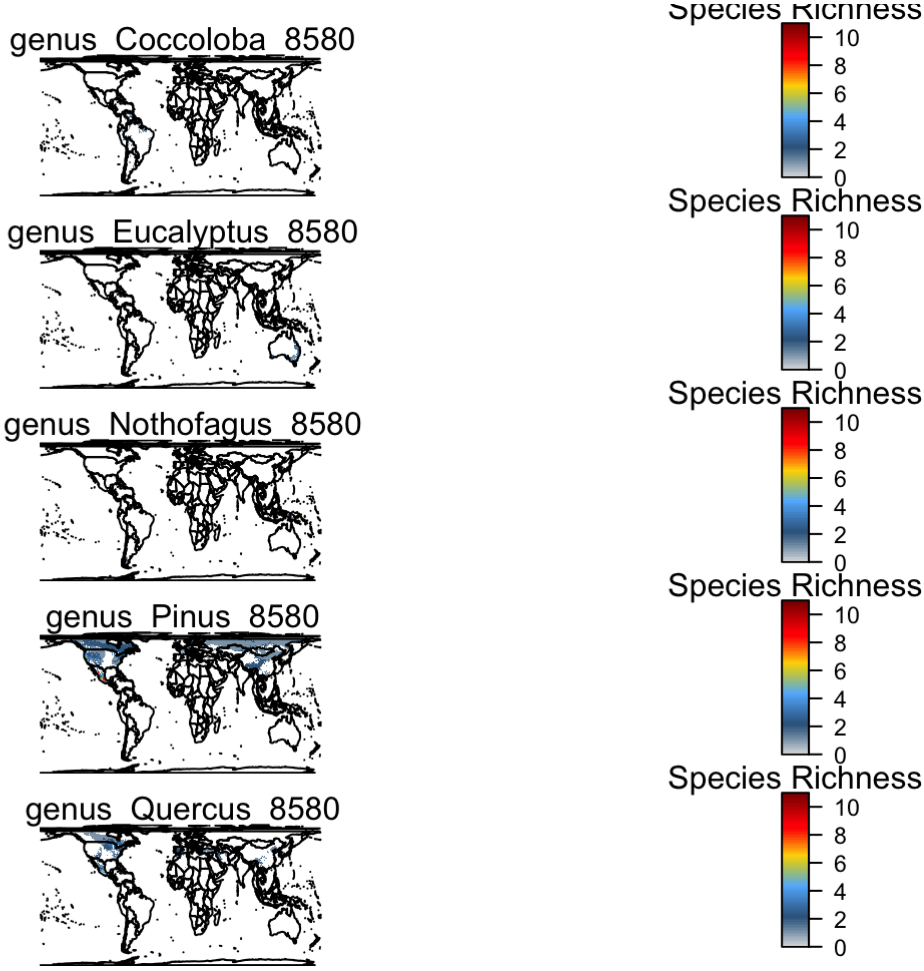
```
## starting 8580
```

```
## 13.1 s
```

```
fdMapPlot(richWithinGen[[1]],shp=world.shp2,legend.args=list(text='Species Richness'))
```




```
fdMapPlot(richWithinGen[[2]],shp=world.shp2,legend.args=list(text='Species Richness'))
```



Next we'll look at the richness of genera - that is, how many genera are in each pixel? This is our first demonstration of **collapsing column** operations, as described above, because we combine all the columns related to a particular genus together. We'll store a new set of matrices - **cell by genus (CBG)** matrices that are analogous to the CBS matrices, except that he columns now correspond to genera. Then we can calculate richness of genera using the same `richnessFromCBS` used above.

```
# make a directory to put the new cell by genus matrices in
sumDirs$cbgDir=paste0(sumDirs$sparseDir,'/CellByGenus')
if(!file.exists(sumDirs$cbgDir)) dir.create(sumDirs$cbgDir)
# overwrite existing directory settings to save this.
saveRDS(sumDirs,file=paste0(sumDirs$sumBaseDir,'/dirList.rds'))

sp.ind=readRDS(paste0(sumDirs$sumBaseDir, '/speciesIndexTable.rds'))  %>% select(specie
s) %>%
  separate(species,c("genus",NA),sep='_')

# collapse columns of the cbs matrices to make cell by genus matrices
lapply(allScen,function(scn){
  outDirScn=paste0(sumDirs$cbgDir,'/',scn)
  if(!file.exists(outDirScn)) dir.create(outDirScn)
  collapseCols(inDir=sumDirs$cbsDir,outDir=outDirScn,scenario=scn,
               spAttrTable=sp.ind,attrName='genus',
               type='binary',keepChunks=T,mc.cores=mc.cores)
})
```

```
## starting present
```

```
## 5 groups to aggregate over
```

```
## 0.11 s
```

```
## starting 8580
```

```
## 5 groups to aggregate over
```

```
## 0.11 s
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

```
# make genus index table (analogous to species index table) so we know which CBG columns
correspond  to which species
gen.ind=data.frame(genus=unique(sp.ind[,'genus']) %>% na.omit)
gen.ind$index=1:nrow(gen.ind)
saveRDS(gen.ind,file=paste0(sumDirs$sumBaseDir,'/generaIndexTable.rds'))

# calculate richness (row sums) from cell by genus matrices
richGen=lapply(allScen,function(scn){
    r=richnessFromCBS(cbsDir=sumDirs$cbgDir,scenario=scn,
                      env=envGrid,mc.cores=mc.cores,outDir=sumDirs$richByGenusDir)
    names(r)=paste0(names(r),'_',scn)
    fdMapPlot(r,plotFile=paste0(sumDirs$figDir,'/RichnessByGenus_',scn,'.pdf'),
              shp=world.shp2, legend.args=list(text='# species',line=2,side=4))
    r
}) %>% stack
```

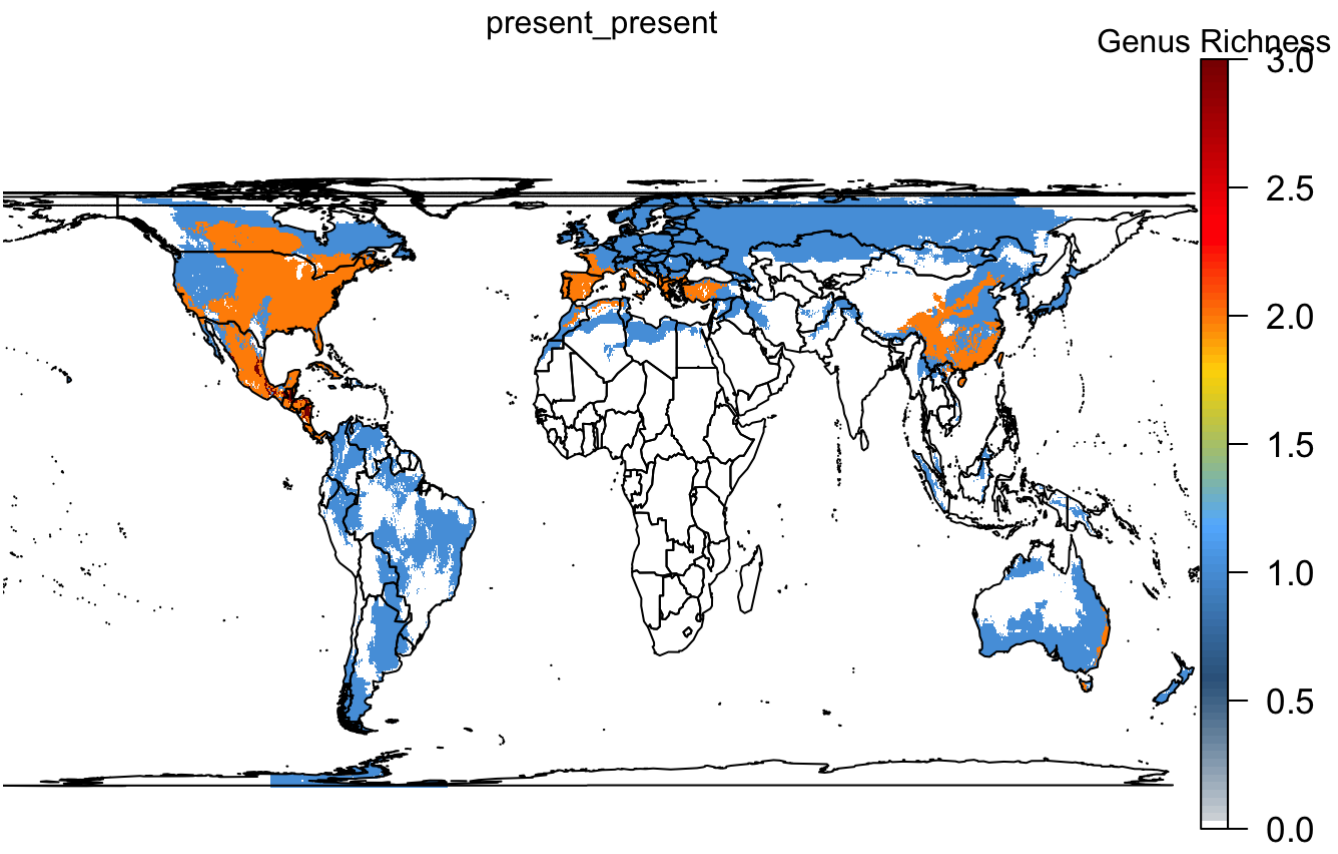
```
## starting present
```

```
## 2.12 s
```

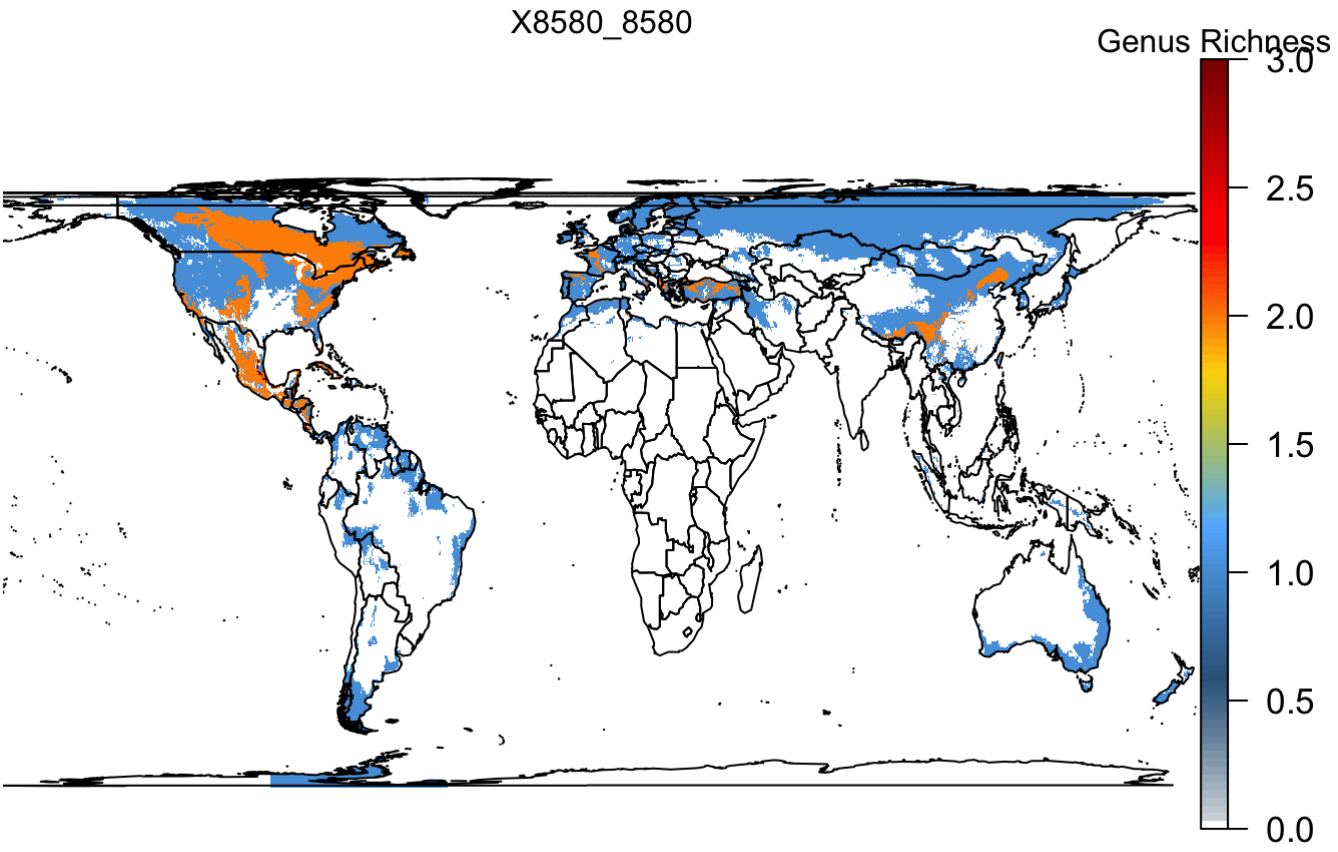
```
## starting 8580
```

```
## 1.92 s
```

```
fdMapPlot(richGen[[1]],shp=world.shp2,legend.args=list(text='Genus Richness'))
```




```
fdMapPlot(richGen[[2]],shp=world.shp2,legend.args=list(text='Genus Richness'))
```


X8580_8580
Genus Richness
3.0
2.5
2.0
1.5
1.0
0.5
0.0