Introduction to Geospatial Analysis in R

NASA Earthdata Webinar

ORNL DAAC https://daac.ornl.gov

January 25, 2024

Contents

Install and Load Packages

In addition to the built-in functionality of R, we will use four packages throughout this exercise. Packages are a collection of documentation, functions, and other items that someone has created and compiled for others to use in R. Install the packages, as well as their dependencies, using the function `install.packages()`.

```
install.packages("terra", dependencies = TRUE)
install.packages("sf", dependencies = TRUE)
install.packages("tigris", dependencies = TRUE)
install.packages("raster", dependencies = TRUE)
```

Most of the functions we will use are from the *terra* and *sf* packages. The *terra*, *sf*, and *stars* (not used here) packages are largely replacing older R packages, such as *raster*, for for managing and manipulating spatial data. However, we will use a *raster* function at the end of this tutorial to export the final object in KML/KMZ format.

```
library(terra)
library(sf)
library(tigris)  # provides access to TIGER shapefiles from US Census Bureau
library(raster)
```

For package details, try `help()` (e.g., `help("terra")`), and to view the necessary arguments of a function try `args()` (e.g., `args(cover)`).

Load Data

Functions featured in this section:
terra::rast()
The `rast()` function from the terra package creates a 'SpatRaster' (spatial raster) object from a file
tigris::states()
The `states()` function from the tigris package downloads a shapefile of the United States that will be loaded as a SpatialPolygonsDataFrame object

Two GeoTiff files are needed to complete this tutorial, both from the dataset titled "CMS: Forest Carbon Stocks, Emissions, and Net Flux for the Conterminous US: 2005-2010" and freely available through the ORNL DAAC at https://doi.org/10.3334/ORNLDAAC/1313.

The dataset provides maps of estimated carbon emissions in forests of the conterminous United States for the years 2006-2010. We will use the maps of carbon emissions caused by fire (*GrossEmissions_v101_USA_Fire.tif*) and insect damage (*GrossEmissions_v101_USA_Insect.tif*). These maps are provided at 100-meter spatial resolution in GeoTIFF format using Albers North America projection. Refer to the accompanying "README.md" for instructions on how to download the data.

To begin, be sure to set your working directory using `setwd()`. The code below assumes that the GeoTIFF files are saved in a folder called 'data' located under the working directory (e.g., "./data/GrossEmissions_v101_USA_Fire.tif").

With the `rast()` function, load *GrossEmissions_v101_USA_Fire.tif* and name it *fire*, then load *GrossEmissions_v101_USA_Insect.tif* and name it *insect*. The contents of these two files are stored as SpatRaster objects; *fire* and *insect* are the primary focus of our manipulations throughout this exercise.

The function `states()` downloads a shapefile of the United States from the United States Census Bureau. It will be stored as a simple feature data frame object named *myStates*.

```
fire <- rast("./data/GrossEmissions_v101_USA_Fire.tif")  # read GeoTIFF, store as SpatRaster object
insect <- rast("./data/GrossEmissions_v101_USA_Insect.tif")
myStates <- states(cb = TRUE, progress_bar=FALSE)  # will download a generalized (1:500k) file
```

Check the Coordinate Reference System and Plot a Raster

Functions featured in this section:
terra::crs()
gets the coordinate reference system of a raster object

Use `print()` to view details about the internal data structure of the raster object we named *fire*.

```
print(fire)
```

```
## class       : SpatRaster
## dimensions  : 32818, 59444, 1  (nrow, ncol, nlyr)
## resolution  : 100, 100  (x, y)
## extent      : -2972184, 2972216, 36233.75, 3318034  (xmin, xmax, ymin, ymax)
## coord. ref. : USA_Contiguous_Albers_Equal_Area_Conic_USGS_version
## source      : GrossEmissions_v101_USA_Fire.tif
## name        : GrossEmissions_v101_USA_Fire
## min value   :                            2
## max value   :                          373
```

The output lists important attributes of *fire*, like its dimensions, resolution, spatial extent, coordinate reference system, and the minimum and maximum values of the cells (i.e., carbon emissions).

The next two commands retrieve the coordinate reference system (CRS) of *fire* and display the CRS in 'WKT' (Well Known Text) and 'Proj4' formats.

```
fire_wkt <- crs(fire)  # CRS in WKT format
writeLines(strwrap(fire_wkt, width = 70, exdent = 5)) # needed to control text wrapping
```

```
## PROJCRS["USA_Contiguous_Albers_Equal_Area_Conic_USGS_version",
##      BASEGEOGCRS["NAD83", DATUM["North American Datum 1983",
##      ELLIPSOID["GRS 1980",6378137,298.257222101004,
##      LENGTHUNIT["metre",1]]], PRIMEM["Greenwich",0,
##      ANGLEUNIT["degree",0.0174532925199433]], ID["EPSG",4269]],
##      CONVERSION["Albers Equal Area", METHOD["Albers Equal Area",
##      ID["EPSG",9822]], PARAMETER["Latitude of false origin",23,
##      ANGLEUNIT["degree",0.0174532925199433], ID["EPSG",8821]],
##      PARAMETER["Longitude of false origin",-96,
##      ANGLEUNIT["degree",0.0174532925199433], ID["EPSG",8822]],
##      PARAMETER["Latitude of 1st standard parallel",29.5,
##      ANGLEUNIT["degree",0.0174532925199433], ID["EPSG",8823]],
##      PARAMETER["Latitude of 2nd standard parallel",45.5,
##      ANGLEUNIT["degree",0.0174532925199433], ID["EPSG",8824]],
##      PARAMETER["Easting at false origin",0, LENGTHUNIT["metre",1],
##      ID["EPSG",8826]], PARAMETER["Northing at false origin",0,
##      LENGTHUNIT["metre",1], ID["EPSG",8827]]], CS[Cartesian,2],
##      AXIS["easting",east, ORDER[1], LENGTHUNIT["metre",1,
##      ID["EPSG",9001]]], AXIS["northing",north, ORDER[2],
##      LENGTHUNIT["metre",1, ID["EPSG",9001]]]]
```

```
fire_prj <- crs(fire, proj=T)  # CRS as proj4 string
writeLines(strwrap(fire_prj, width = 70, exdent = 5))
```

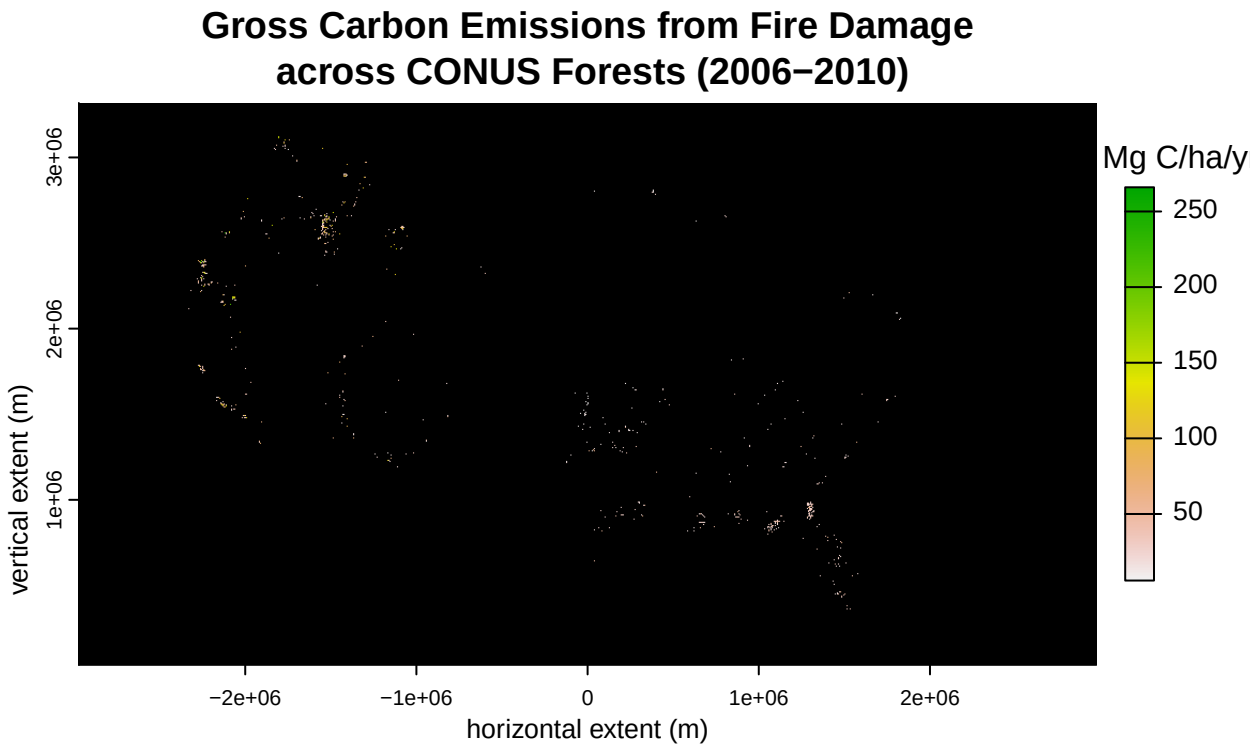
```
## +proj=aea +lat_0=23 +lon_0=-96 +lat_1=29.5 +lat_2=45.5 +x_0=0 +y_0=0
##      +datum=NAD83 +units=m +no_defs
```

In the PROJ.4 representation, the first argument is "+proj=" and defines the projection. "aea" refers to the `NAD83 / Albers NorthAm` projection (EPSG 42303), and "+units=m" tells us that the resolution of the raster object is in meters.

Refer to the attributes of *fire* provided by `print()` above. The resolution of the raster is "100, 100 (x, y)" meaning that each cell is 100 meters by 100 meters, or one hectare (ha).

Use the `plot()` function to make a simple image of *fire* and visualize the carbon emissions from fire damage across the forests of the conterminous United States between 2006 and 2010. According to the documentation for the dataset, gross carbon emissions were measured in megagrams of carbon per year (Mg C/y) per cell.

```
plot(fire,
     main = "Gross Carbon Emissions from Fire Damage\n across CONUS Forests (2006-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg=list( title="\n      Mg C/ha/yr", size=c(0.7,1) ),
     colNA = "black",
     box = FALSE)
```



The spatial extent of the raster object is displayed on the x- and y-axes. All NA cells (i.e., cells that have no values) are colored black for better visualization of fire damage. The legend offers the range of cell values and represents them using a default color theme. Because the extent of this object covers the conterminous US, we cannot see much spatial detail in this figure.

Let's examine the raster object we named *insect*. The function `crs()` retrieves the CRS information for each raster object. Then, we use `identical()` to determine if *fire* and *insect* have the same CRS.

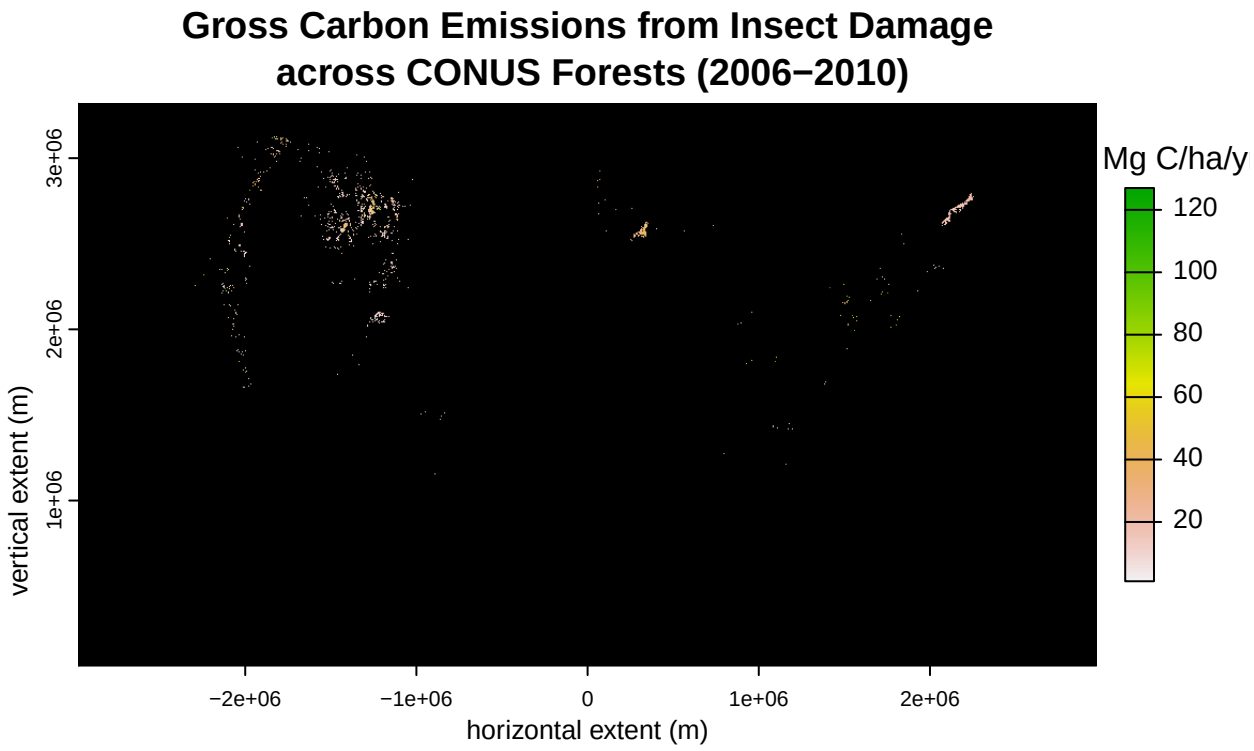
```
identical(crs(fire), crs(insect))
```

```
## [1] TRUE
```

"TRUE" indicates that the CRS for the two raster objects have the same CRS.

Plot *insect* but change the content for the argument "main =", which defines the main title of the plot.

```
plot(insect,
     main = "Gross Carbon Emissions from Insect Damage\n across CONUS Forests (2006-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg=list( title="\n      Mg C/ha/yr", size=c(0.7,1) ),
     colNA = "black",
     box = FALSE)
```



You can likely imagine an outline of the United States given the distribution of spatial data in the two raster objects.

Select Data Within a Region of Interest

Functions featured in this section:
terra::crs()
retrieves or sets the CRS for a spatial object
sf::st_transform()
provides re-projection given a CRS
sf::st_bbox()
retrieves the bounding box of a simple feature (sf) spatial object terra::crop()
returns a geographic subset of an object as specified by an Extent object
terra::mask()

creates a new raster object with the same values as the input object, except for the cells that are NA in the second object (the 'mask')

Next, we reduce the spatial extent of *fire* and *insect* to focus on three states in the western US. We can select the states from the *myStates* to create a new feature then use it to crop the *fire* and *insect* rasters.

First, use `print()` to view details about the internal data structure of the simple feature we named *myStates*.

```
print(myStates)
```

```
## Simple feature collection with 56 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -179.1489 ymin: -14.5487 xmax: 179.7785 ymax: 71.36516
## Geodetic CRS:  NAD83
## First 10 features:
##    STATEFP  STATENS    AFFGEOID GEOID STUSPS           NAME LSAD        ALAND
## 1       56 01779807 0400000US56    56     WY        Wyoming   00 2.514587e+11
## 2       02 01785533 0400000US02    02     AK         Alaska   00 1.478943e+12
## 3       24 01714934 0400000US24    24     MD       Maryland   00 2.515199e+10
## 4       60 01802701 0400000US60    60     AS American Samoa   00 1.977591e+08
## 5       05 00068085 0400000US05    05     AR       Arkansas   00 1.346608e+11
## 6       38 01779797 0400000US38    38     ND   North Dakota   00 1.786943e+11
## 7       10 01779781 0400000US10    10     DE       Delaware   00 5.046732e+09
## 8       66 01802705 0400000US66    66     GU           Guam   00 5.435558e+08
## 9       35 00897535 0400000US35    35     NM     New Mexico   00 3.141986e+11
## 10      49 01455989 0400000US49    49     UT           Utah   00 2.133551e+11
##          AWATER                       geometry
## 1    1867503716 MULTIPOLYGON (((-111.0546 4...
## 2  245378425142 MULTIPOLYGON (((179.4825 51...
## 3    6979074857 MULTIPOLYGON (((-76.05015 3...
## 4    1307243751 MULTIPOLYGON (((-168.1458 -...
## 5    3121950081 MULTIPOLYGON (((-94.61792 3...
## 6    4414779956 MULTIPOLYGON (((-104.0487 4...
## 7    1399179670 MULTIPOLYGON (((-75.56555 3...
## 8     934337453 MULTIPOLYGON (((144.6454 13...
## 9     726482113 MULTIPOLYGON (((-109.0502 3...
## 10   6529973239 MULTIPOLYGON (((-114.053 37...
```

myStates has 56 features (i.e., polygons) each with 9 attribute fields. This *sf* object can be thought of a dataframe with 56 rows and ten columns (variables or features). The columns include the 9 attribute fields plus a "geometry" column.

For this exercise, we will focus on carbon emissions for Idaho, Montana, and Wyoming. We can use column referencing and indexing to select all column information contained in *myStates*, but for only three rows (polygons). This code selects the polygons for which the 'NAME' is either Idaho, Montana, or Wyoming. These three polygons are saved in the resultant simple feature *threeStates*.

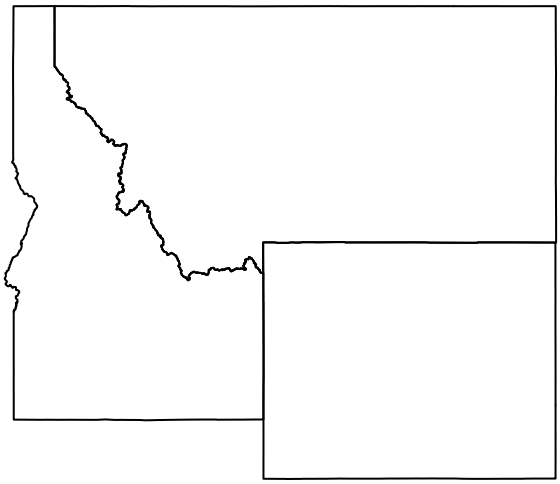
```
threeStates <- myStates[myStates$NAME == "Idaho" |
                        myStates$NAME == "Montana" |
                        myStates$NAME == "Wyoming", ]
threeStates <- threeStates[order(threeStates$NAME),]  # sort by name: ID, MT, WY
print(threeStates)
```

```
## Simple feature collection with 3 features and 9 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -117.243 ymin: 40.99475 xmax: -104.0396 ymax: 49.00139
## Geodetic CRS:  NAD83
##    STATEFP  STATENS    AFFGEOID GEOID STUSPS    NAME LSAD        ALAND
## 18      16 01779783 0400000US16    16     ID   Idaho   00 214049931578
## 27      30 00767982 0400000US30    30     MT Montana   00 376973729130
## 1       56 01779807 0400000US56    56     WY Wyoming   00 251458712294
##        AWATER                       geometry
## 18 2391569647 MULTIPOLYGON (((-117.2427 4...
## 27 3866634365 MULTIPOLYGON (((-116.0491 4...
## 1  1867503716 MULTIPOLYGON (((-111.0546 4...
```

threeStates has only three rows, but the same number of columns as *myStates*.

What does *threeStates* look like plotted? We'll plot the geometry (i.e., the 10th column) of *threeStates* so we only see the outline.

```
plot(threeStates$geometry)
```



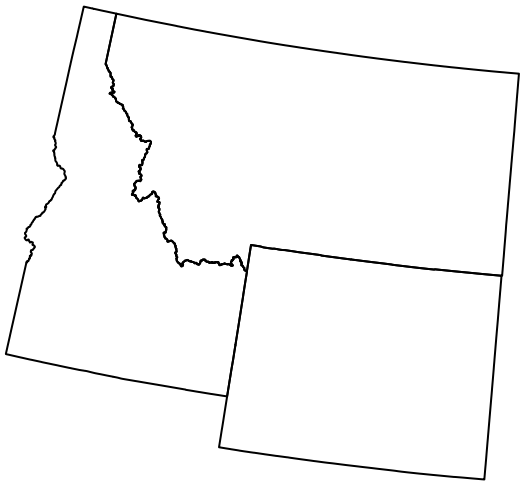
We can get the *fire* and *insect* data that occurs "within" *threeStates*. First, we must confirm that the three objects share the same CRS before we can overlay or otherwise combine information from them. Some GIS software, such as QGIS or ArcGIS, can automatically combine data layers having different CRS, but that is not a default capability with R.

```
identical(crs(fire), crs(threeStates))
```

```
## [1] FALSE
```

"FALSE" indicates that *threeStates* does not have the same CRS as *fire*, so we will make a new version of these state polygons that has *fire*'s CRS. The `st_transform()` from the *sf* package will work. This function uses `crs=crs(fire)` to set the CRS of the new simple feature object to the same projection as *fire*.

```
transStates <- st_transform(threeStates, crs=crs(fire) )
plot(transStates$geometry)  # draw outline of the state polygons
```



Plotting the geometry of the new object *transStates* shows that the projection has changed. Notice how the orientation of the polygons has shifted to match the *NAD83 / Albers NorthAm projection*.

Now that our objects share a CRS, we will compare the extent of *fire* and *transStates*. Use the `st_bbox()` function to view the bounding (i.e., bounding box) of *transStates* and *fire*.

```
cat("fire extent\n"); st_bbox(fire)
```

```
## fire extent
```

```
##        xmin        ymin        xmax        ymax
## -2972184.50    36233.75  2972215.50  3318033.75
```

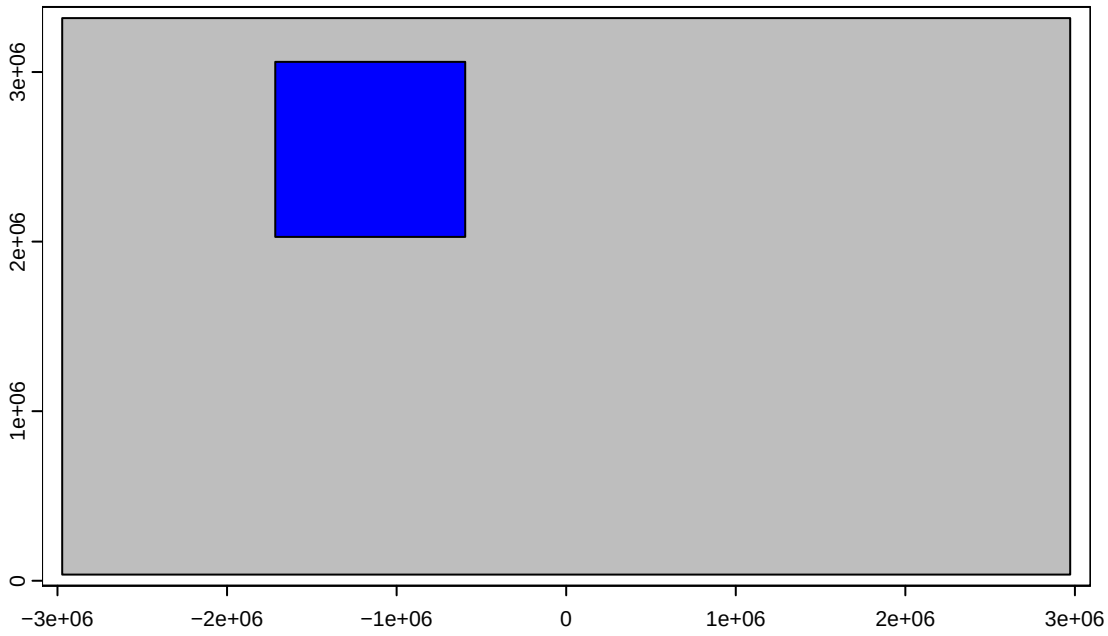
```
cat("\ntransStates extent\n"); st_bbox(transStates)
```

```
##
## transStates extent
```

```
##       xmin       ymin       xmax       ymax
## -1715671.1  2027603.7  -595594.4  3059862.0
```

To visualize the bounding boxes, use the `ext()` (i.e., extent) function with `plot()`.

```
plot(ext(fire), col='grey')  # bounding box of fire (grey)
plot(ext(transStates), col='blue', add=T)  # bounding box of tranStates (blue)
```



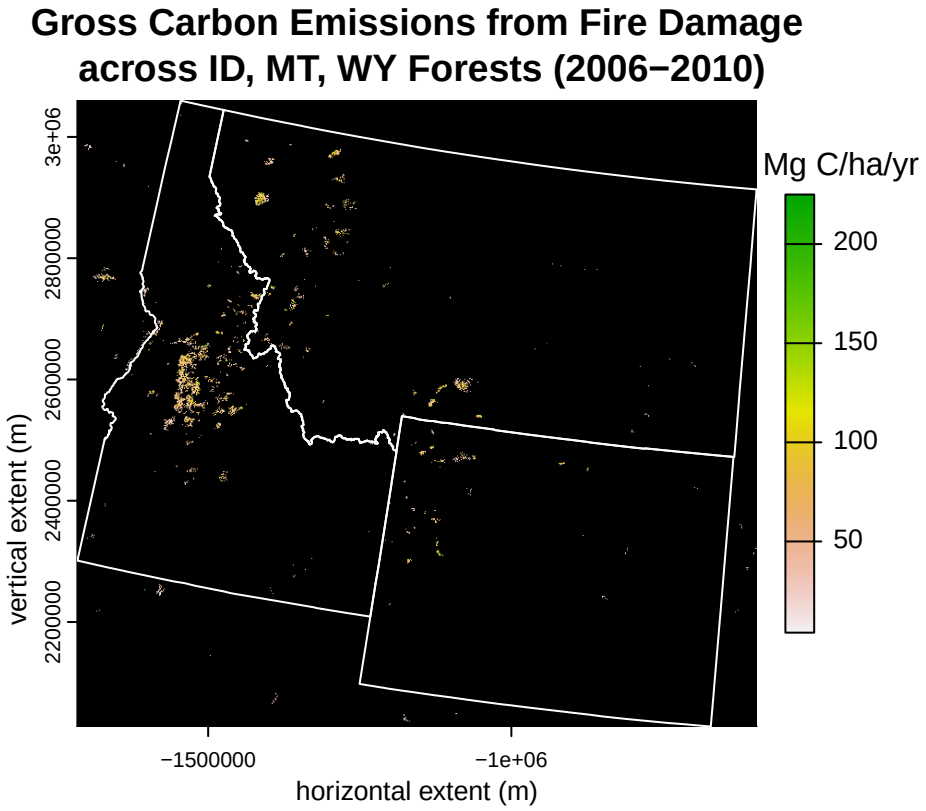
The raster *fire* has a much larger extent than *transStates* because *fire* covers the entire conterminous US.

Reducing the extent of the *fire* and *insect* rasters will greatly speed up processing and reduce the size of saved files. The `crop()` function will serve this purpose. Cropping will create a geographic subset of *fire* and *insect* as specified by the extent of *transStates*. We prepend "crop" to the names of the new raster objects to reflect this manipulation.

```
# this will take a minute to run
cropFire <- crop(fire, transStates)  # crop(raster object, extent object)
cropInsect <- crop(insect, transStates)
```

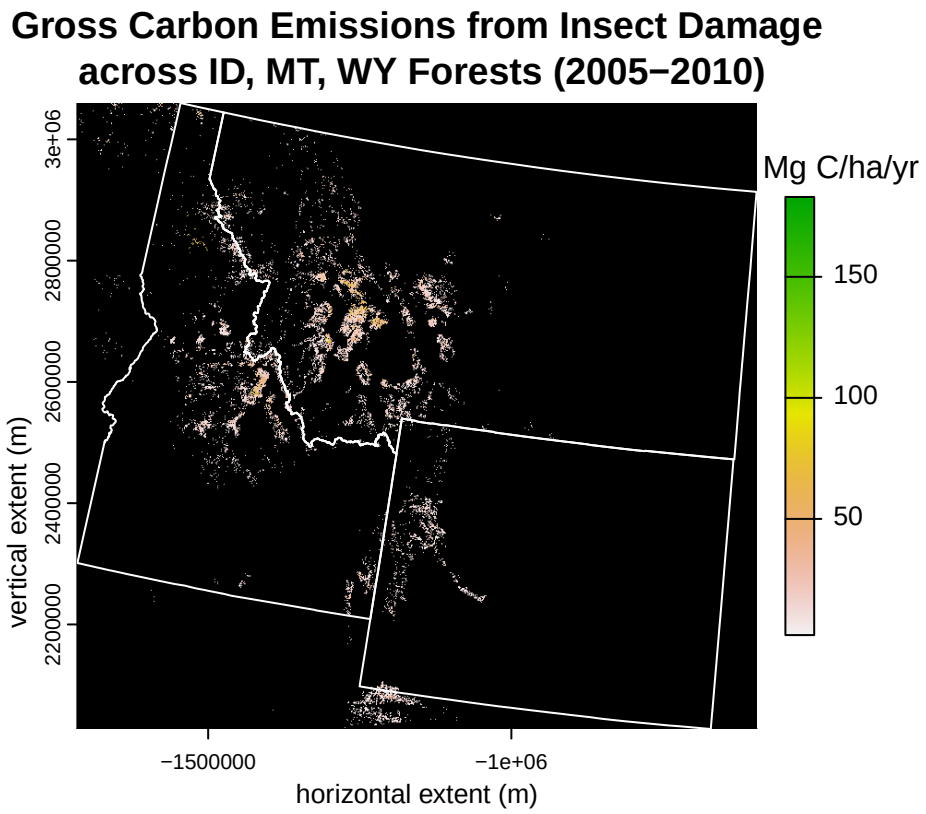
Now when we plot *cropFire* and *cropInsect*, we will also plot *transStates* "on top" to envision how carbon emissions are distributed across the three states.

```
par(mfrow=c(1,1))
mar.val <- c(3.1, 3.1, 3.1, 3.1)  # set plot margin values for maps: bot,lft,top,rgt
    plot(cropFire,
     main = "Gross Carbon Emissions from Fire Damage\n across ID, MT, WY Forests (2006-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg=list( title="\n      Mg C/ha/yr", size=c(0.7,1) ),
     colNA = "black",
     mar=mar.val,
     box = FALSE)
plot(transStates$geometry,
     border = "white",
     add = TRUE)
```




```
plot(cropInsect,
     main = "Gross Carbon Emissions from Insect Damage\n across ID, MT, WY Forests (2005-2010)",
     # cex.main=0.85,
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg=list( title="\n      Mg C/ha/yr", size=c(0.7,1) ),
     colNA = "black",
     mar=mar.val,
     box = FALSE)
plot(transStates$geometry,
     border = "white",
```

```
     add = TRUE)
```



If you look closely at the cells "outside" the boundary of the *transStates* polygons, you can still see cell values. That's because `crop()` changed the extent of the two raster objects to match that of the simple feature object, but `crop()` did not change cells outside of the polygon boundaries.

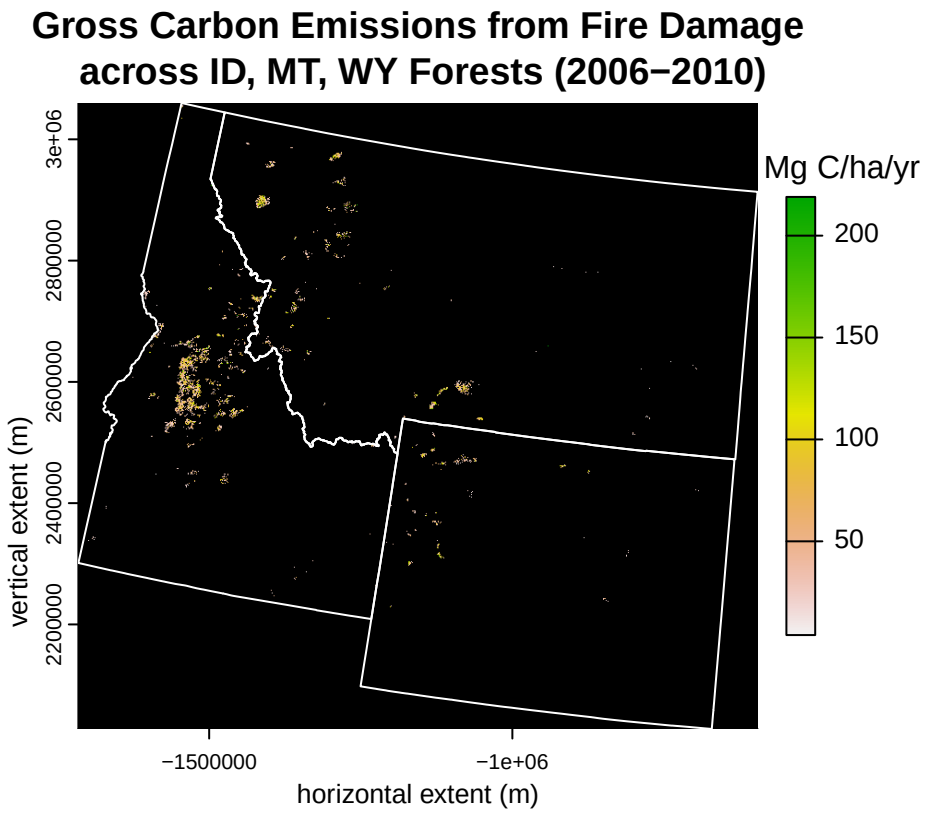
To remove those extraneous cell values, use the `mask()` function to create two new rasters, one for fire damage and one for insect damage. The *terra* function `mask()` will convert raster cells outside of the state polygons to NA. **Note:** You can use `mask()` or `crop()` in either order.

```
# this will take a couple of minutes to run
maskFire <- mask(cropFire, transStates)  # mask(raster object, mask object)
maskInsect <- mask(cropInsect, transStates)
```

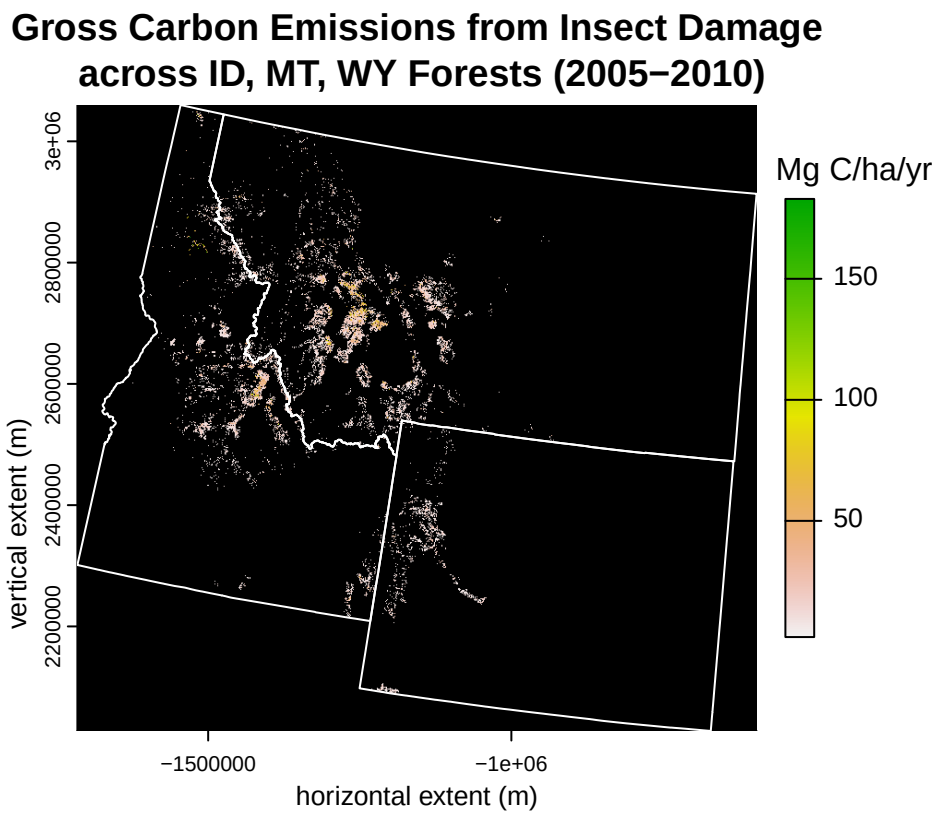
Plot *maskFire* and *maskInsect*.

```
plot(maskFire,
     main = "Gross Carbon Emissions from Fire Damage\n across ID, MT, WY Forests (2006-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg=list( title="\n      Mg C/ha/yr", size=c(0.7,1) ),
     colNA = "black",
     mar=mar.val,
     box = FALSE)
plot(transStates$geometry,
```

```
     border = "white",
     add = TRUE)
```




```
plot(maskInsect,
     main = "Gross Carbon Emissions from Insect Damage\n across ID, MT, WY Forests (2005-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg=list( title="\\n      Mg C/ha/yr", size=c(0.7,1) ),
     colNA = "black",
     mar=mar.val,
     box = FALSE)
plot(transStates$geometry,
     border = "white",
     add = TRUE)
```



These plots demonstrate that all cell values outside of the *transStates* polygons are now NA.

Examine Summaries of Raster Values

> *Functions featured in this section:*
> terra::extract()
> extracts values from a raster object at the locations of other spatial data

In this section, we will compare the three states by their carbon emissions from fire damage.

We will use *terra*'s `extract()` function to collect the cell values of *maskFire* where the *transStates* simple feature object overlaps the raster object. We will use `summary()` to examine the distribution of cell values that we collect.

```
# this can take up to an hour to run, so load a saved copy for the demonstration
if(file.exists("./data/val_fireStates.Rds")) {
  val_fireStates <- readRDS("./data/val_fireStates.rds")
  summary(val_fireStates)
  }else{
    val_fireStates <- extract(maskFire, transStates, df = TRUE)  # extract(raster object, extent object,
    summary(val_fireStates)
    # saveRDS(val_fireStates, "./data/val_fireStates.Rds") # uncomment this line to save to file
  }
```

```
##        ID        GrossEmissions_v101_USA_Fire
```

```
##  Min.   :1.000   Min.   :  2
##  1st Qu.:1.000   1st Qu.: 27
##  Median :2.000   Median : 47
##  Mean   :1.807   Mean   : 56
##  3rd Qu.:3.000   3rd Qu.: 76
##  Max.   :3.000   Max.   :333
##                  NA's   :84454561
```

There are two columns in this summary of *val_fireStates*. One is ID, which corresponds with the three states; 1 = Idaho, 2 = Montana, and 3 = Wyoming. The ID value corresponds to the row order of the polygons. That is why it was important to sort *threeStates* object by 'NAME' in the code above. Still, it is a good idea to verify this order.

```
cbind("ID"=1:3, "state"=transStates$NAME)
```

```
##      ID  state
## [1,] "1" "Idaho"
## [2,] "2" "Montana"
## [3,] "3" "Wyoming"
```

The second column is a summary of all cell values across those three states. On average (mean), 56 megagrams of carbon per ha per year are a result of forest destruction by fire damage for all states combined.

To look at the summary for cell values by state, we will use **subset()** to split *val_fireStates* into three data frames, one for each state. In the code below, we subset *val_fireStates* so that only the rows with ID == "1" (representing Idaho) will be returned. We name the new object with the prefix "temp" and suffix "id" (Idaho).

```
temp_val_id <- subset(val_fireStates, subset = ID %in% 1)  # Idaho values
summary(temp_val_id)
```

```
##        ID    GrossEmissions_v101_USA_Fire
##  Min.   :1   Min.   :  3
##  1st Qu.:1   1st Qu.: 27
##  Median :1   Median : 49
##  Mean   :1   Mean   : 58
##  3rd Qu.:1   3rd Qu.: 79
##  Max.   :1   Max.   :254
##              NA's   :37907760
```

The summary demonstrates that there is now only a single value in the ID column, and that the distribution of cell values has changed. This resultant data frame object is quite large and has more information than we need. We need only the second column and don't care for the large number of NA's.

To create a new vector of cells values from *temp_val_id*, use the **is.na()** function to eliminate the rows with NA cells and select the second column holding emissions values.

```
val_id <- temp_val_id[!is.na(temp_val_id$GrossEmissions_v101_USA_Fire),2]
summary(val_id)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   27.00   49.00   58.06   79.00  254.00
```

The resultant object, *val_id*, is a vector object (a single column of numbers) with no NA's.

We will do the same with *val_fire* for the states Montana and Wyoming.

```
temp_val_mt <- subset(val_fireStates, subset = ID %in% 2)  # Montana
val_mt <- temp_val_mt[!is.na(temp_val_mt$GrossEmissions_v101_USA_Fire), 2]
temp_val_wy <- subset(val_fireStates, subset = ID %in% 3)  # Wyoming
val_wy <- temp_val_wy[!is.na(temp_val_wy$GrossEmissions_v101_USA_Fire), 2]
rm(temp_val_id, temp_val_mt, temp_val_wy)  # clean up
```

What's the total carbon emissions from fire for each state and the range of values within each state for the period 2006 to 2010?

```
cat("Number of cells burned (ha)\n")  # divide totals by 1000 to improve ability to compare
```

```
## Number of cells burned (ha)
```

```
cat("Idaho: ", length(val_id), "  Montana: ", length(val_mt),
        " Wyoming: ", length(val_wy),"\n")
```

```
## Idaho:  176544   Montana:  49758  Wyoming:  379739
```

```
cat("\nTotal emissions (/1000)\n")  # divide totals by 1000 to improve ability to compare
```

```
##
## Total emissions (/1000)
```

```
cat("Idaho: ", sum(val_id)/1000, "  Montana: ", sum(val_mt)/1000,
        " Wyoming: ", sum(val_wy)/1000,"\n")
```

```
## Idaho:  10250.22   Montana:  2660.297  Wyoming:  20961.93
```

```
cat("\nDistribution of cell values\n")
```

```
##
## Distribution of cell values
```

```
cat("Idaho\n"); summary(val_id); cat("Montana\n"); summary(val_mt); cat("Wyoming\n"); summary(val_wy)
```

```
## Idaho
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   27.00   49.00   58.06   79.00  254.00
```

```
## Montana
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.00   24.00   43.00   53.46   70.00  230.00
```

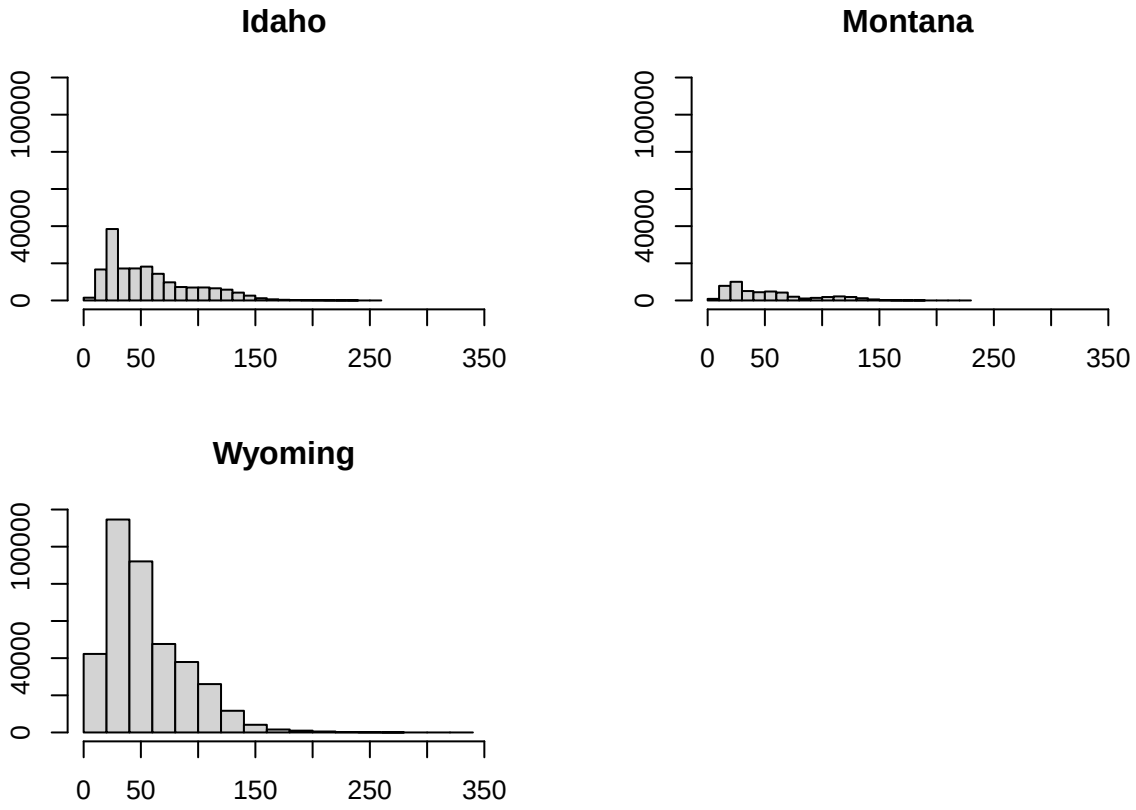
```
## Wyoming
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     2.0    27.0    47.0    55.2    75.0   333.0
```

Idaho has the greatest number of cells burned, the highest total carbon emissions, and the maximum gross carbon emissions from a single cell. In contrast, the largest mean emissions from cells occurred in Montana. Wyoming had the least amount of carbon emissions due to fire.

In addition to using `summary()`, we can create graphs to visualize carbon emissions from fire damage within each of the three states. The function `hist()` plots the frequency of cell values. We will set some arguments of the plot so that we can compare carbon emissions across all three states.

```
par(mfrow=c(2,2), mar=c(3,3,3,3))
hist(val_id,
     main = "Idaho",
     ylab = "number of cells",
     xlab = "megagrams of carbon per ha per year (Mg C/ha/yr)",
     ylim = c(0, 120000),  # same y-axis limit for all three states
     xlim = c(0, 350))  # same x-axis limit for all three states
hist(val_mt,
     main = "Montana",
     ylab = "number of cells",
     xlab = "megagrams of carbon per ha per year (Mg C/ha/yr)",
     ylim = c(0, 120000),
     xlim = c(0, 350))
hist(val_wy,
     main = "Wyoming",
     ylab = "number of cells",
     xlab = "megagrams of carbon per ha per year (Mg C/ha/yr)",
     ylim = c(0, 120000),
     xlim = c(0, 350))
```



The histograms show the number of times (on the y-axis) each unique cell value (on the x-axis) occurs in each state. In other words, these plots illustrate the variation in carbon emissions from fire damage within the three different states and provide a visual display of the numerical summaries above.

Reclassify Raster Values

Functions featured in this section:
reclassify()
reclassifies groups of values of a raster object to other values
calc()
calculates values for a new raster object from another raster object using a formula

Now we are going to change the values of our two raster objects using different methods. The goal is to generate a single map that shows the cells where fire and insect disturbances occurred. Fire and insect maps are reclassified individually then combined.

Beginning with *maskFire*, we convert all cells with values >0 to the value 2 and save the new raster object to *reclassFire* .

```
reclassFire <- maskFire
reclassFire[reclassFire >0] <- 2
```

Check that our reclassification of *maskFire* worked as expected using `summary()`. The pair of square brackets ("[]") after the raster name tells the summary command to read all values of the raster.

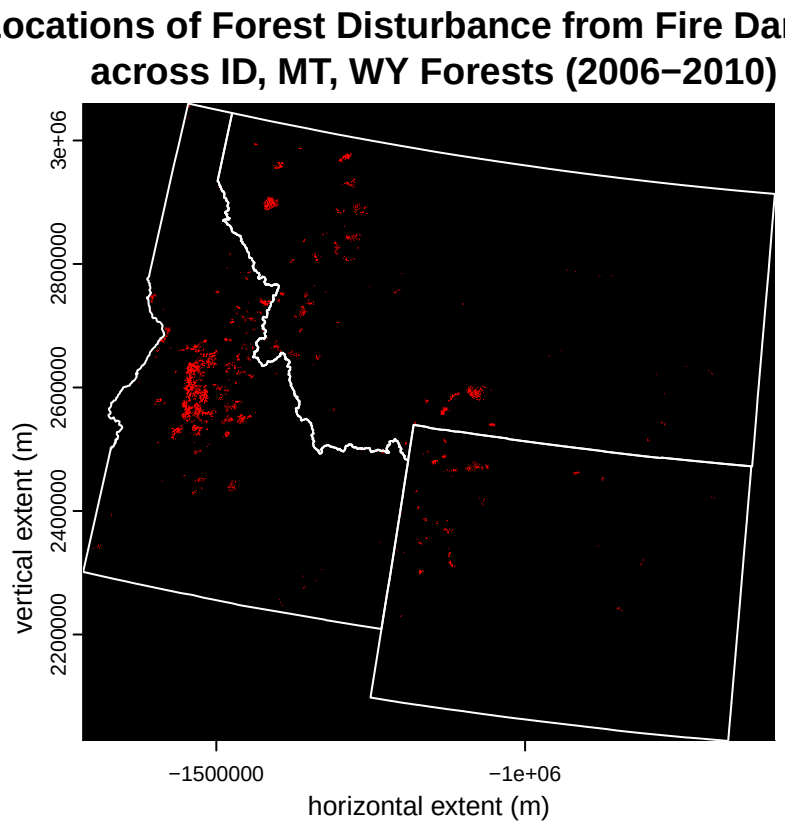
```
summary(reclassFire[])
```

```
##  GrossEmissions_v101_USA_Fire
##  Min.   :2
##  1st Qu.:2
##  Median :2
##  Mean   :2
##  3rd Qu.:2
##  Max.   :2
##  NA's   :115010680
```

Yes, all values are either 2 or NA.

All cell values of *reclassFire* should be at the same locations as *maskFire* but with a single value.

```
plot(reclassFire,
     main = "Locations of Forest Disturbance from Fire Damage\n across ID, MT, WY Forests (2006-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     legend = FALSE,
     col = "red", colNA = "black",
     mar=c(3.1, 1.1, 2.8, 1.1),  # bot,lft,top,rgt
     box = FALSE)
plot(transStates$geometry,
     border = "white",
     add = TRUE)
```



The plot of *reclassFire* now illustrates locations where there were carbon emissions due to forest fire. Notice that we chose a single color to represent the presence of values using the argument "col ="red" ".

Now we will reclassify all values of *maskInsect* that are greater than zero to be 1. Let's check the range of values for this raster.

```
range(maskInsect[], na.rm=TRUE)
```

This time, we will use the `classify()` function in the *terra* package. This function uses a matrix to identify the target cell values and to what value those cells will change.

```
reclassInsect <- classify(maskInsect,
                          rcl = matrix(data = c(0, 285, 1),  # c(from value, to value, becomes)
                                       nrow = 1, ncol = 3))
```

The argument following "rcl =" tells R that values from 1 to 285 should be reclassified as one. Essentially, we are making the presence of insect damage equal one.

Check the reclassification of *maskInsect* using `summary()`.

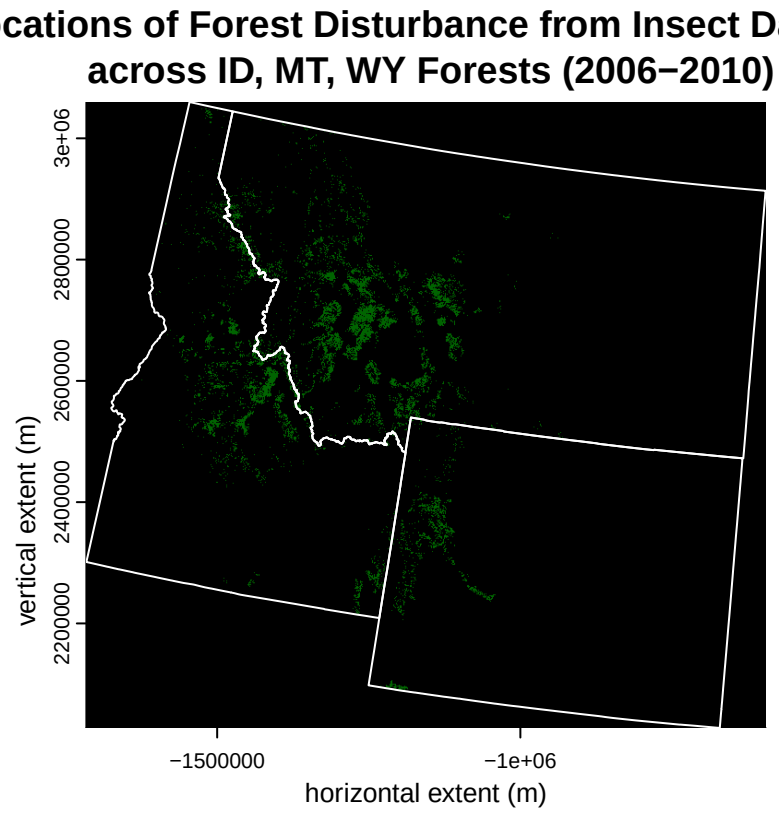
```
summary(reclassInsect[])
```

```
##  GrossEmissions_v101_USA_Insect
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
##  NA's   :113254371
```

All values are 1 or NA.

Plot *reclassInsect*. All the cell values should be at the same locations as *maskInsect* but will all be the value one.

```
plot(reclassInsect,
     main = "Locations of Forest Disturbance from Insect Damage\n across ID, MT, WY Forests (2006-2010)
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     legend = FALSE,
     col = "dark green",
     colNA = "black",
     mar=c(3.1, 1.1, 3.1, 1.1),
     box = FALSE)
plot(transStates$geometry,
     border = "white",
     add = TRUE)
```



The plot illustrates locations where there were carbon emissions due to insect damage in forests, so now the information conveyed by the *maskInsect* raster object is presence or absence of insect damage.

Combine Two Rasters

Functions featured in this section:
cover()
replaces NA values in the first raster object with the values of the second

Next, we will join *reclassFire* and *reclassInsect* to form a single raster object. According to the documentation for this dataset, there are no overlapping, non-NA cells between the two raster objects. That is, if you were to combine the two rasters object, a cell could take only the value provided by *reclassFire* (i.e., 2) or *reclassInsect* (i.e., 1), or be NA. This allows us to use the `cover()` function to combine objects. `cover()` will replace NA values of *reclassFire* with non-NA values of *reclassInsect*.

```
# this could take a couple of minutes to run
fireInsect <- cover(reclassFire, reclassInsect)
```

Check the combination of *reclassFire* and *reclassInsect* using `summary()`.

```
summary(fireInsect[])
```

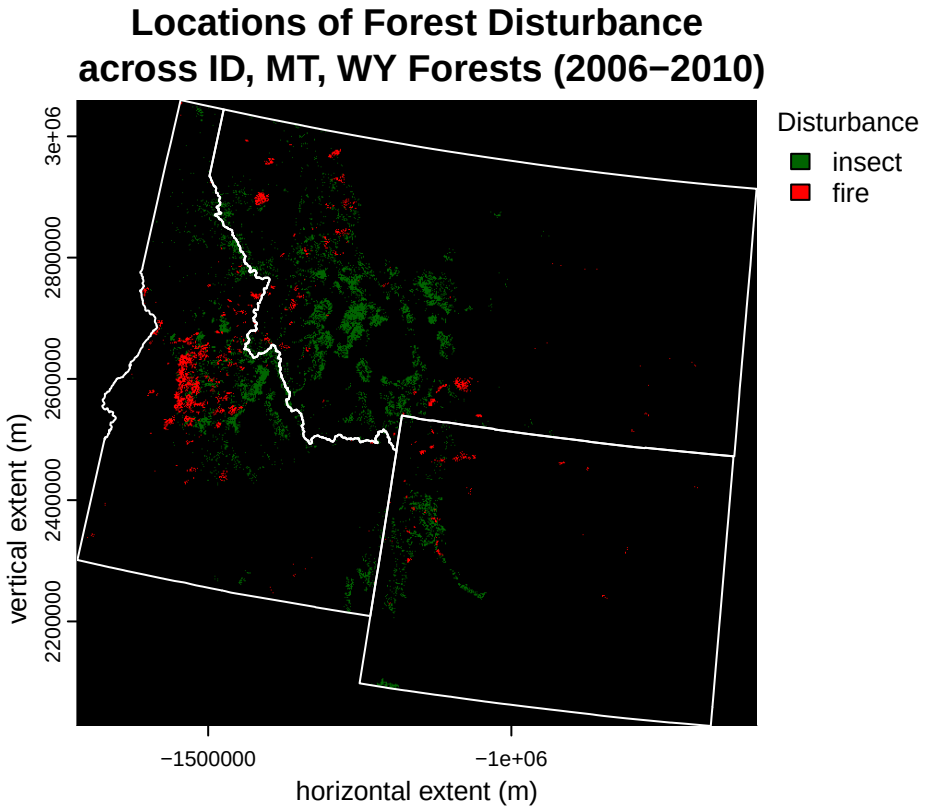
```
##  GrossEmissions_v101_USA_Fire
```

```
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :2
##  NA's   :112648329
```

The data distribution of the new raster object shows that the minimum value is now 1 (i.e., the insect damage value we specified during reclassification) and the maximum value is 2 (i.e., the fire damage value).

The plotting arguments below now reflect the "breaks" in the values we would like to see illustrated on the plot. Insect damage is displayed as green cells and fire damage as red.

```
plot(fireInsect,
     main = "Locations of Forest Disturbance\n across ID, MT, WY Forests (2006-2010)",
     xlab = "horizontal extent (m)",
     ylab = "vertical extent (m)",
     plg = list(legend=c("insect","fire"), title="  Disturbance"),
     col = c("dark green", "red"),
     colNA = "black",
     mar=c(3.1, 1.1, 3.1, 1.1),
     box = FALSE)
plot(transStates$geometry,
     border = "white",
     add = TRUE)
```



Reproject and Write a Raster

Functions featured in this section:
projectRaster {raster}
projects the values of a raster object to a new one with a different projection
writeRaster {raster}
writes an entire raster object to a file

Reprojecting a raster in R is different than transforming the CRS as we did with the simple feature earlier in the exercise. To reproject a raster, we use the `project()` function and the `crs()` function to provide the CRS information. The nearest neighbor `method` ('near') is used to maintain cell values of 1 and 2. Other reprojection methods can produce cell values that are the average of nearby cells, a situation we want to avoid here.

```
# this may take several minutes to run
prjFireInsect <- project(fireInsect,
                         crs("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"),
                         method="near")
```

Now, check the properties of this new raster object using `print()`.

```
print(prjFireInsect)
```

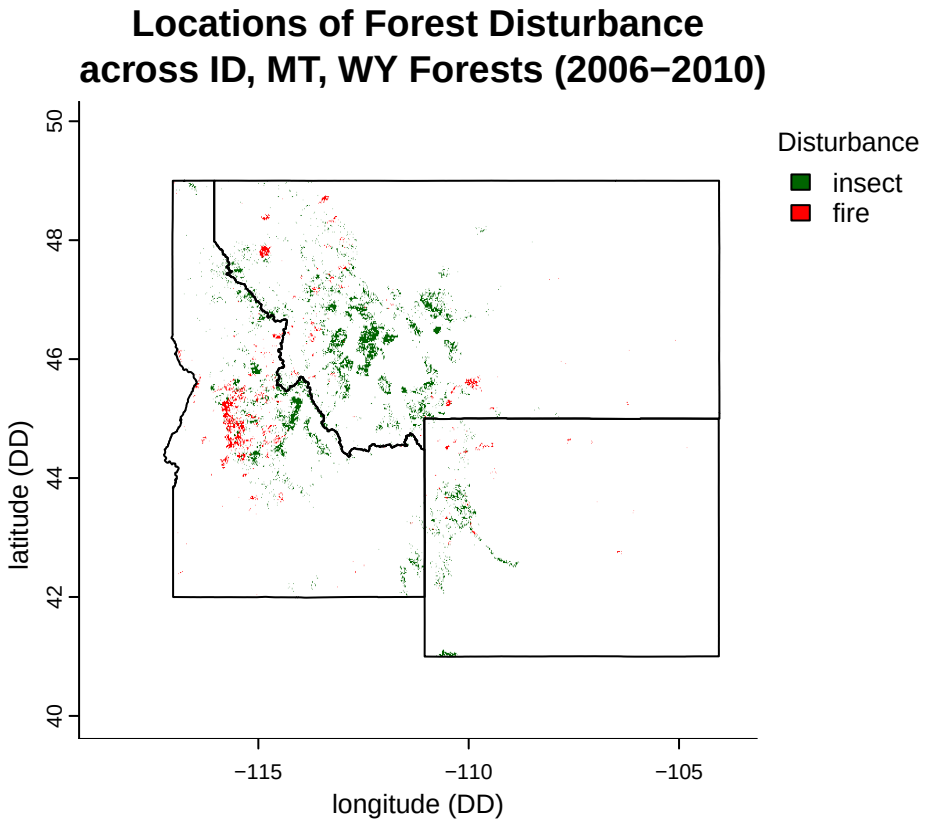
```
## class       : SpatRaster
## dimensions  : 9169, 13781, 1  (nrow, ncol, nlyr)
## resolution  : 0.001169079, 0.001169079  (x, y)
## extent      : -119.2604, -103.1493, 39.61248, 50.33177  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source      : spat_4a2c689a290b_18988.tif
## name        : GrossEmissions_v101_USA_Fire
## min value   :                           1
## max value   :                           2
```

It's a new raster object named *prjFireInsect* that has the standard Geographic projection with latitude and longitude expressed in decimal degrees (DD) as its CRS.

We will plot *prjFireInsect* with slightly different arguments than *fireInsect* to "zoom in" to the center of the plot. Also, we will use *threeStates* instead of *transStates* because *threeStates* also uses the Geographic projection.

```
plot(prjFireInsect,
     main = "Locations of Forest Disturbance\n across ID, MT, WY Forests (2006-2010)",
     xlab = "longitude (DD)",
     ylab = "latitude (DD)",
     plg = list(legend=c("insect","fire"), title="\n  Disturbance"),
     col = c("dark green", "red"),
     mar=c(3.1, 1.1, 2.8, 1.1),
     ext = st_bbox(prjFireInsect)/1.25,
     box = FALSE)

plot(threeStates$geometry,
     border = "black",
     add = TRUE)
```



Let's use the `writeRaster()` function to save *prjFireInsect* to the data directory. We will save the file in GeoTIFF (*.tif) format so that the geographic information of the raster object is retrievable outside of R.

```
writeRaster(prjFireInsect, filename = "./data/prjFireInsect.tif", overwrite=TRUE)
```

Use the function `file.exists()`, which tests for the existence of a given file, to ensure that *prjFireInsect* was successfully saved to our working directory.

```
file.exists("./data/prjFireInsect.tif")
```

```
## [1] TRUE
```

Now we are able to share the raster with others or open it in another program.

Export a Plot as PNG and Raster as KML

Functions featured in this section:
KML()
exports raster object data to a KML file

To save an imaage of the final plot, we use `png()` and repeat the plot() command. The png() function will open a graphics device that will save the plot we run in *.png format. We will use the function `dev.off()` to tell R when we are finished plotting and want to close the graphics device.

```
png("prjFireInsect.png", width=650, res=80)
plot(prjFireInsect,
     main = "Locations of Forest Disturbance\n across ID, MT, WY Forests (2006-2010)",
     xlab = "longitude (DD)",
     ylab = "latitude (DD)",
     plg = list(legend=c("insect","fire"), title="\n  Disturbance"),
     col = c("dark green", "red"),
     mar=c(3.1, 1.1, 2.8, 1.1),
     ext = st_bbox(prjFireInsect)/1.25,
     box = FALSE)
plot(threeStates$geometry,
     border = "black",
     add = TRUE)
dev.off()
```

It might be useful to save *prjFireInsect* in *.kmz format. KML stands for Keyhole Markup Language, and KMZ is the compressed version of KML format. These formats were developed for geographic visualization in Google Earth.

At present, the *terra* package does not include an option to export a SpatRaster object as a KML/KMZ; it is necessary to use the *raster* package. First, *prjFireInsect* must be converted to a Raster object then saved using the raster package's KML() function.

```
prjFireInsect.r <- as(prjFireInsect, "Raster")  # convert to a 'Raster' object
KML(prjFireInsect.r, "./data/prjFireInsect.kmz", col = c("dark green", "red"), overwrite=TRUE)
```

We successfully saved the raster object as a KMZ file.

This is the end to the tutorial. If you liked this tutorial, please tell us on EarthData Forum. If you would like to make a suggestion for a new tutorial, please email uso@ornl.gov.

There is a supplemental document included on GitHub that offers two additional sections, *Perform a Focal Analysis* and *Get Cell Coordinates*.