NICAR 2021: Using data to report on climate change

Peter Aldhous

Learning to love gridded climate data

The slides for this session introduce the concept of gridded climate data and the widely used Network Common Data Form or netCDF data format. Gridded data means that a geographical area has been divided into a regular grid, and then values have been assigned to each cell in that grid. Often the data has a time-series element, with a separate grid layer for each of multiple periods of time.

In the session I'll also attempt a live demo, working with NASA's **Panoply** software to view a netCDF file and to show how to make customized maps and animations from data like this.

An introduction to the R raster package for working with gridded climate data

If you are willing to learn some simple code, the R **raster** package is very powerful, allowing you to work with gridded data in flexible ways. Below are some code examples to give you a flavor of the possibilities. You can download the data used from here.

Processing NASA's GISTEMP global temperature analysis

NASA's GISTEMP surface temperature analysis contains monthly temperature records from 1880 onward for a global grid with cells of 2 degrees latitude by 2 degrees longitude. The values in each cell are the difference between average temperature for that time period compared to the average for a reference period from 1951 to 1980.

The code below processes this netCDF data to the data files used to make this interactive:

An earlier version of this map first appeared in this Apr. 22, 2019 BuzzFeed News post.

Setting up

```
# load required packages
library(raster)
library(rgdal)

# load data from netCDF file as raster brick object
```

```
temperature_monthly <-
brick("data/gistemp1200_GHCNv4_ERSSTv5.nc")
```

In addition to **raster**, I loaded the **rdgal** package, which connects to the Geospatial Data Abstraction Library. GDAL allows you to read, write, and process geodata. You may already be familiar with it if you have worked with QGIS.

NetCDF files often contain multiple gridded data layers, one for each time interval – in this case a layer for each month. Using the **brick** function, such netCDF data files can be loaded as RasterBrick objects, which hold multiple data layers.

Making new layers for years, rather than months

The following code uses the **stackApply** function to run some calculations on the data to create one layer for each year, rather than each month:

```
# create vector to serve as index for calculating annual values
# we need an index with 12 repeats of an index number for each of
the 141 years in the data
num_years <- rep(1:141, each = 12)

# calculate annual averages, giving 141 layers one for each year
1880-2020 in the data
temperature_annual <- stackApply(temperature_monthly, indices =
num_years, fun = mean)
```

Convert to a spatial polygons data frame, then write to GeoJSON

```
# convert to spatial polygons data frame
temperature_annual_df <-
as(temperature_annual,"SpatialPolygonsDataFrame")
# name the variables in the data frame as years
names(temperature_annual_df@data) <- c(as.character(1880:2020))

# write to GeoJSON
writeOGR(temperature_annual_df,
"data/temperature_annual.geojson",
layer = "temperature", driver = "GeoJSON")
```

Make the map overlay

The code below first subsets the annual data to isolate the most recent decade, and then uses the **calc** function to calculate the average temperature differences for this decade from the 1951-1980 reference period.

```
temperature_diff <- subset(temperature_annual, 132:141)
temperature_diff <- calc(temperature_diff, mean, na.rm = TRUE)
```

Interpolate the data to a higher resolution and write to a GeoTIFF

We saw that the Panoply software can apply an interpolation to smooth the data so is doesn’t display as an obvious grid. We can apply a similar effect using the

resample function. Be cautious with such interpolations: I was comfortable doing this to provide a less jarring appearance for the map overlay, but I would not have done this for the data displayed on the charts.

```
# create a raster object with the same extent but higher
resolution
r <- raster(nrow = 1800, ncol = 3600, extent(c(-180, 180, -90,
90)))
# resample the data using this raster
temperature_diff <- resample(temperature_diff, r, method =
"bilinear")

# write to GeoTIFF
writeRaster(temperature_diff,
filename="data/temperature_diff.tif", format = "GTiff", overwrite
= TRUE)
```

I styled the GeoTIFF in QGIS and imported to Mapbox as a raster tileset. The GeoJSON was imported to Mapbox as a vector tileset.

The interactive map and chart was made with Mapbox GL and Highcharts. The vector tileset is queried on a map click or location search to yield the data displayed in the chart.

Making gridded data on burn frequency from CAL FIRE's historical fire footprints data.

For a student project at UC Berkeley J-School, we wanted to look at the distribution of rare plant species in relation to historical burn frequencies. The California Department of Forestry and Fire Protection, or CAL FIRE, maintains a geodatabase of historical fire footprints, currently complete until 2019. The following code takes a shapefile processed from this geodatabase for a century of fires from 1920 to 2019 and uses the **rasterize** function to calculate the number of overlapping polygons at the center of each cell for a grid of 0.005 degrees latitude by 0.005 degrees longitude drawn across the entire state. Using "count" to summarize any variable in the data simply counts the number of polygons; other summary functions are available.

```
# load data from shapefile
fire_perimeters <-
shapefile("data/fire_perimeters/fire_perimeters.shp")

# review variables in the data
dplyr::glimpse(fire_perimeters@data)

## Rows: 19,336
## Columns: 18
## $ YEAR_    <chr> "2007", "2007", "2007", "2007", "2007",
"2007", "2007", "2007…
## $ STATE    <chr> "CA", "CA", "CA", "CA", "CA", "CA", "CA",
"CA", "CA", "CA", "…
## $ AGENCY   <chr> "CCO", "CCO", "USF", "CCO", "CCO", "CCO",
```

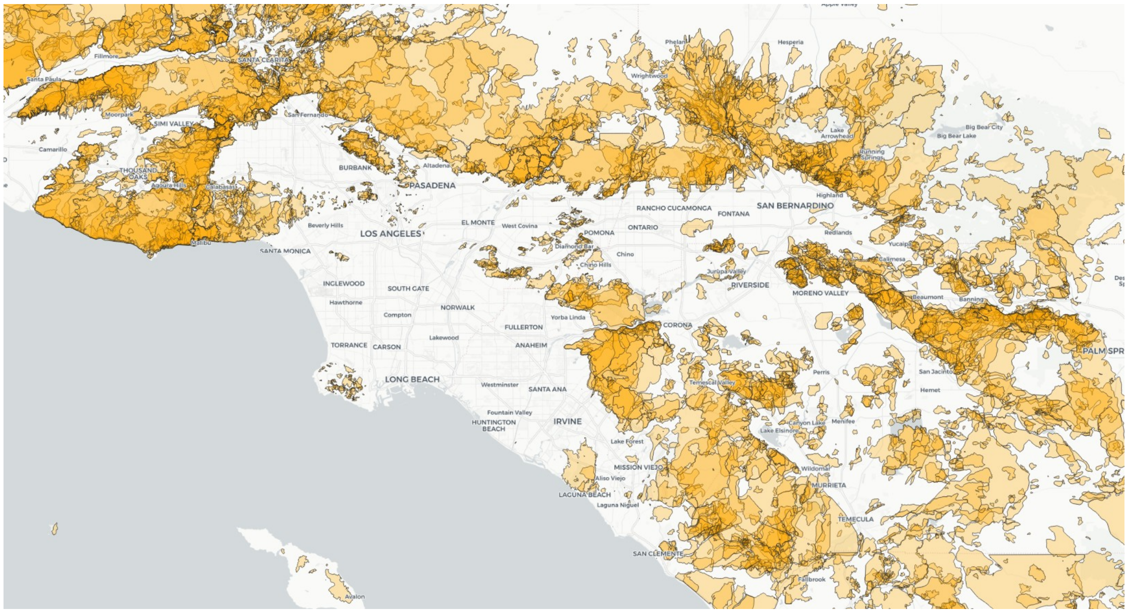
```
"CCO", "CCO", "CCO"…
## $ UNIT_ID <chr> "LAC", "LAC", "ANF", "LAC", "LAC", "LAC",
"LAC", "LAC", "LAC"…
## $ FIRE_NA <chr> "OCTOBER", "MAGIC", "RANCH", "EMMA", "CORRAL",
"GORMAN", "WES…
## $ INC_NUM <chr> "00246393", "00233077", "00000166",
"00201384", "00259483", "…
## $ ALARM_D <chr> "2007/10/21", "2007/10/22", "2007/10/20",
"2007/09/11", "2007…
## $ CONT_DA <chr> "2007/10/23", "2007/10/25", "2007/11/15",
"2007/09/11", "2007…
## $ CAUSE   <int> 14, 14, 2, 14, 14, 14, 14, 14, 14, 14, 14, 14,
4, 14, 7, 2, 1…
## $ COMMENT <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, N…
## $ REPORT_ <dbl> NA, NA, 54716.00, NA, NA, NA, NA, NA, NA, NA,
NA, NA, 837.00,…
## $ GIS_ACR <dbl> 25.73671, 2824.87720, 58410.33594, 172.21495,
4707.99707, 237…
## $ C_METHO <int> 8, 8, 7, 8, 8, 8, 8, 8, 8, 8, 8, NA, 7, 7, 2,
2, 1, 1, 1, 7, …
## $ OBJECTI <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1…
## $ FIRE_NU <chr> "00233414", "00233077", "00000166",
"00201384", "00259483", "…
## $ Shp_Lng <dbl> 1902.439, 20407.966, 169150.716, 6117.777,
22907.182, 19693.2…
## $ Shap_Ar <dbl> 104152.78, 11431872.51, 236378245.17,
696929.17, 19052588.51,…
## $ year    <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007,
2007, 2007, 2007, 2…

# raster template of grid 0.005 degrees lat and lon for bounding
box around CA
r <- raster(ncol = 2400, nrow = 2000, extent(c(-126, -113, 32,
42)))

# create raster layer with number of overlapping fire footprints
the center of each cell in that grid
burn_freq <- rasterize(fire_perimeters, r, field = "Shp_Lng", fun
= "count")

# write to GeoTIFF
writeRaster(burn_freq, "data/burn_frequency.geotiff", format =
"GTiff", overwrite = TRUE)
```

Here is the original data displayed in QGIS showing 100 years of fire footprints around Los Angeles:



Here is the gridded data layer showing relative burn frequencies over the past 100 years: