Test case: Alaska wildfires

Brooke Anderson (BA), Sheryl Magzamen (SM), Ryan Gan (RG), Miranda Fix (MF), Ryan Hicks (RH), Steven J. Brey (SJB), Joshua Ferreri (JF), Molly Gutilla (MG)

Main task

You are trying to get observational data related to wildfires from throughout the state of Alaska for the years 2000-2016. Note, a NOAA API key is required to run the R code in this markdown document.

Relevant NOAA weather products

In later sections, you will have specific goals to try to achieve, in terms of what data to get. However, as you work on this project, please keep a running list here of any of the NOAA data products that you think might have data that could be used to explore the storm you're working on. Include the name of the dataset, a short description of what you might be able to get from it, and a link to any websites that give more information.

- SJB: Here is data from the NOAA Hazard Mapping System. This data shows smoke plumes outlined by a human analysist using visible satelite imagery (mostly GEOS). This dataset has no readme but the product has been described in several publications: ftp://satepsanone.nesdis.noaa.gov/volcano/FIRE/HMS_smoke/

BA: Note: For many of the `rnoaa` functions, you will need an API key from NOAA. Here's how you should save that on your computer so that you can get this document to compile without including your API key (these instructions are adapted from here:

1. Ask NOAA for an API key: http://www.ncdc.noaa.gov/cdo-web/token
2. Open a new text file. Put the following in it (note: this might not show up here, but there should be a second blank line):

```
noaakey == [your NOAA API key]
```

3. Save this text file as `.Renviron` in your home directory. Your computer might be upset that the file name starts with a dot. That's okay– it should; ignore the warning and save like this.
4. Restart R.
5. Now you can pull your key using `Sys.getenv("noaakey")`

Now you can gear up to use functions that require the API key:

```
options("noaakey" = Sys.getenv("noaakey"))
```

Relevant other data sources

As you work on this project, also keep a running list here of any data from sources other than NOAA that you think might have data that could be used to explore the storm you're working on. Include the name of the dataset, a short description of what you might be able to get from it, and a link to any websites that give more information and / or the data itself.

- SM: There's some data from the Alaska Interagency Coordination Center on lightning strikes and wildfires: http://wildfiretoday.com/2015/06/23/alaska-46000-lightning-strikes-and-many-fires/
- SM: Vegetation is usually USDA and NDVI is a NOAA product (http://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring_vegetation_2.php)

Specific tasks

As you work on these, include the code that you try. Include things that worked and things that you thought would work but that did not. Also write down what parts were easy to figure out how to do, and which you had to search around a lot to figure out what to do.

Historic fire location

- RG: The Bureau of Land Management has maps of historic fire records that can be downloaded in dataframe format. Web address: http://afsmaps.blm.gov/imf_firehistory/imf.jsp?site=firehistory I couldn't figure out a way to scrap the website itself as it was a interactive map. However, there is an option to download excel file. I've downloaded xls file, deleted first two rows that were useless, and saved as a csv. I've put in rnoaa data file.

The following code creates a map of fire locations for each year.

```
# infile fire csv
fire_dir <- paste0('/Users/ryangan/Google Drive/CSU/wild_fire/rnoaa/alaska/',
                   'rnoaa/data/alaska_historic_fire_location_2000_2015.csv')

ak_fire <- read.csv(fire_dir)

# base map
alaska <- c(-180, 52, -130, 72) # set bounding box by lat long min and max

ak_fire_map <- get_map(location=alaska, # use bounding box for location
                       source="stamen", # source stamen
                       maptype= "toner", crop=FALSE) # toner style
# for loop
for(i in 2000:2015){
ak_fire_year <- filter(ak_fire, Fire.Year == i)
# adding station locations to the map
print(
ggmap(ak_fire_map) + geom_point(aes(x = Longitude, y = Latitude),
                                data = ak_fire_year , alpha = .5, color="red",
                                size = 1, pch = 17) +
                     ggtitle(paste("Alaska Fire Locations for", i))
      )
      } # end loop
```

Lightning strikes

- RG: rnoaa swdi does not appear to be able to pull lightning strike data using nldn command. I get the following error message 'Error in UseMethod("xpathApply")'

- RG Note 4/21/16: I found asource of lightning strikes (same AK BLM website). http://afsmaps.blm.gov/imf_firehistory/imf.jsp?site=firehistory. Downloaded historic lightning strike shapefile. There are two shapefiles with historic lightning data; 1986 to 2012, and 2012 to 2015. The projection is off a bit. The issue here is that the coordinate system of the lightning strikes (lat long) are in an Albers projection, with a rather odd orgin.

Note on projection of lightning strikes for Alaska

Display Projection Information Vector source data for layers in all Alaska Fire Service MapServices are in North American Datum 1983 (NAD83) Geographic Coordinate System. Raster layers (generally these are background layers) are in the Albers Equal Area Projection for Alaska. For display purposes the layers are reprojected on the fly to NAD83 Albers. Coordinates displayed for the current location of the mouse are in NAD83 geographic. All coordinates displayed in the dataframe in response to a query or identify request are in NAD83 geographic. All data exported from this site will default to NAD83 geographic. NAD 83 Alaska Albers may be requested on an individual export. Additional Information for Alaska Albers: Central Meridian = -154 Latitude of Origin = 50 Standard Parallel 1 = 55 Standard Parallel 2 = 65 False Easting = 0 False Northing = 0 Units = Meters

Datum = North American 1983

Source: http://afsmaps.blm.gov/imf_firehistory/sites/help/AFSProjectionStory.jsp

```
# The goal for this chunk of code is to read in lighting data shapefile for 2012
# to 2015 and plot the strikes

# Look at whats in the shapefile
# saved shapefile in rnoaa data
# library used
library(rgdal) # package for reading shapefiles
library(sp) # package for redefining coord ref (CRS)

light_dir <- paste0('/Users/ryangan/Google Drive/CSU/wild_fire/rnoaa/alaska/',
                    'rnoaa/data/CONDUCT')
lightning_shp <- readOGR(dsn = light_dir, layer = 'TOALightning2012Thru2015')
str(lightning_shp)
plot(lightning_shp) # you can see the weird projection where data is missing in the middle

# I need to redefine the CRS
# lightning strikes are in an albers projeciton
# creating a custom proj4 string with relevant info
custom_crs <- paste0("+proj=laea +lat_0=50 +lon_0=-154 +lat_1=55 +lat_2=65",
                     "+datum=NAD83 +no_defs +x_0=0 +y_0=0 +a=6370997 +b=6370997",
                     " +units=m")
# create new shapefile with new projection
lightning_shp_albers <- spTransform(lightning_shp, CRS(custom_crs))
summary(lightning_shp_albers)
head(lightning_shp_albers$LONGITUDE)
# plot looks better, but the lat and longitude are still the same
# I will need to find a way to convert to values of more use
plot(lightning_shp_albers)
```

-RG: I found a conversion forumla to convert albers projeciton lat longs to a more useable lat long. However it's in Java. Perhaps someone wants to write the r code converstion? Source: http://stackoverflow.com/questions/26474475/convert-lat-longs-to-x-y-coordinates-on-albers-projection-map

Get a dataset of lightning strikes within Alaska for 2000-2015, with the following columns:

- Date / time: The date and, if available, time of the lightning strike
- Latitude: The latitude of the strike, in decimal degrees North
- Longitude: The longitude of the strike, in decimal degrees West

Are there any other variables available for each strike? If so, please describe what any other variables measure and include them in your dataframe.

How many lightning strikes did you record in Alaska over this time period? Create a map by year of lightning strike locations. If you can, also create a heatmap, where you break the state up into either grid boxes or counties and show with color the number of lightning strikes per box / county over the time period.

- BA: `rnoaa`'s `swdi` series of functions looks like it might have data on lightning strikes. The vignette for that series is here. The NCDC webpage for the severe weather data inventory is here. From the documentation, it sounds like for non-military users, you may only be able to get number of strikes per day per region, not exact time and location. UPDATE: It does not appear any data is openly available to the public.
- BA: Here is some more information from NOAA on lightning strikes. I'm not clear yet on how much of this overlaps the SWDI resource.
- RG: Test comment
- SJB: There is a website that maps lighting. Not sure how they generate these maps but the response time and spatial coverage is excellent. It looks like they use geostationary satellites to make the detections?: http://www.lightningmaps.org/. It looks like the raw data are only avialable to 'contributors' (http://en.blitzortung.org/archive_data.php).

```
# The goal for this chunk of code is to read in lighting data and plot it
devtools::load_all()
library('rnoaa')
library('plyr')
library('XML')

# Stock example getting Tornado Vortex Signatures.
df <- swdi(dataset='nx3tvs',
           startdate='20060505',
           enddate='20060506',
           limit=20
           )

# NOTE: When I up my limit (number of detections to retrieve) I get curl
# NOTE: timeout issues when I go as high as 100. Bummer. I can see that
# NOTE: is probably why there is an index option so you have a way to get
# NOTE: all the detections for day.
# TODO: Change the source code on swdi to return numeric.
dfl <- swdi(dataset='nx3hail',
           startdate='20060505',
           enddate='20060506',
           limit=20
           )
```

Almost lightning data

So it looks like lightning data is largely not for free (bummer!). But it is easy to find MODIS detected hotspots GIS data. Fires that are detected by MODIS out in the middle of nowhere Alaska are probably

started by lightning, so maybe this could be a proxy data source?

- Continental US http://activefiremaps.fs.fed.us/gisdata.php
- Alaska http://activefiremaps.fs.fed.us/gisdata.php?sensor=modis&extent=alaska

Relevant weather measures

Get observed measurements for the following variables:

- Air temperature
- A measure of air moisture (dew point temperature or relative humidity)
- Wind (speed and direction)

First, get these values at a daily resolution. Get a separate time series for each weather monitor in Alaska (although limit to just weather monitors that have non-missing observations for at least 95% of days over 2000-2016). Also aggregate these monitor time series by county to get county-level time series.

For the first week of July 2015, get values for each of the weather variables, at as fine a time resolution as possible, for each lightning strike in Alaska. In other words, for a given lightning strike, get weather data from the closest available weather monitor. If you can get the exact time for a lightning strike, try to get the matching weather variables at an hourly or minute resolution at the time of the lightning strike. If you can only get daily counts for a region, get a daily aggregated value of the weather variables for all monitors in the region.

- BA: For daily data, you should be able to use `meteo_pull_monitors` from `rnoaa` to collect this data. However, you'll need to find all the weather stations IDs that you want to pull first. For that, you may want to try out `ncdc_stations`, using the `locationid` argument to get a list of stations either by county or by state. If you are pulling and aggregating by county, you can also use the `weather_fips` function from the `countyweather` package, which lets you pull data by county FIPS code.

```
ak_monitors <- ncdc_stations(datasetid = "GHCND", locationid = "FIPS:02",
                             limit = 1000)
ak_station_2016 <- data_frame(ak_monitors$data$mindate, ak_monitors$data$maxdate,
                              ak_monitors$data$name, ak_monitors$data$id,
                              ak_monitors$data$datacoverage)
colnames(ak_station_2016) <- c("mindate", "maxdate", "name", "id", "coverage")
ak_station_2016$id <- gsub("GHCND:", "", ak_station_2016$id)#Must remove "GHCND:" to get id alone
alaska_weather <- meteo_pull_monitors(ak_station_2016$id[1:3],
                                      date_min = "2000-01-01",
                                      date_max = "2016-01-01") ##THIS WILL TAKE
                                                               ##FOREVER
#startDates <- as.POSIXct(ak_station_2016$mindate, tz="GMT")
#endDates <- as.POSIXct(ak_station_2016$maxdate, tz="GMT")
#startRangeMask <- startDates <= as.POSIXct("2000-01-01", tz="GMT")
#endRangeMask   <- endDates >= as.POSIXct("2015-12-31", tz="GMT")
#inRange <- startRangeMask & endRangeMask
test <- ak_station_2016
test$mindate <- as.Date(test$mindate, format = "%Y-%m-%d")
test$maxdate <- as.Date(test$maxdate, format = "%Y-%m-%d")
##limit number of monitors to those with data that covers desired range, as well
##as have greater than .75 data coverage
test <- subset(test, mindate <= as.Date("2000-01-01") & maxdate >= as.Date("2015-12-31") & coverage > .
```

```
alaska_weather2 <- meteo_pull_monitors(test$id[1:3],
                                       date_min = "2000-01-01",
                                       date_max = "2016-01-01")#for 168 monitors
```

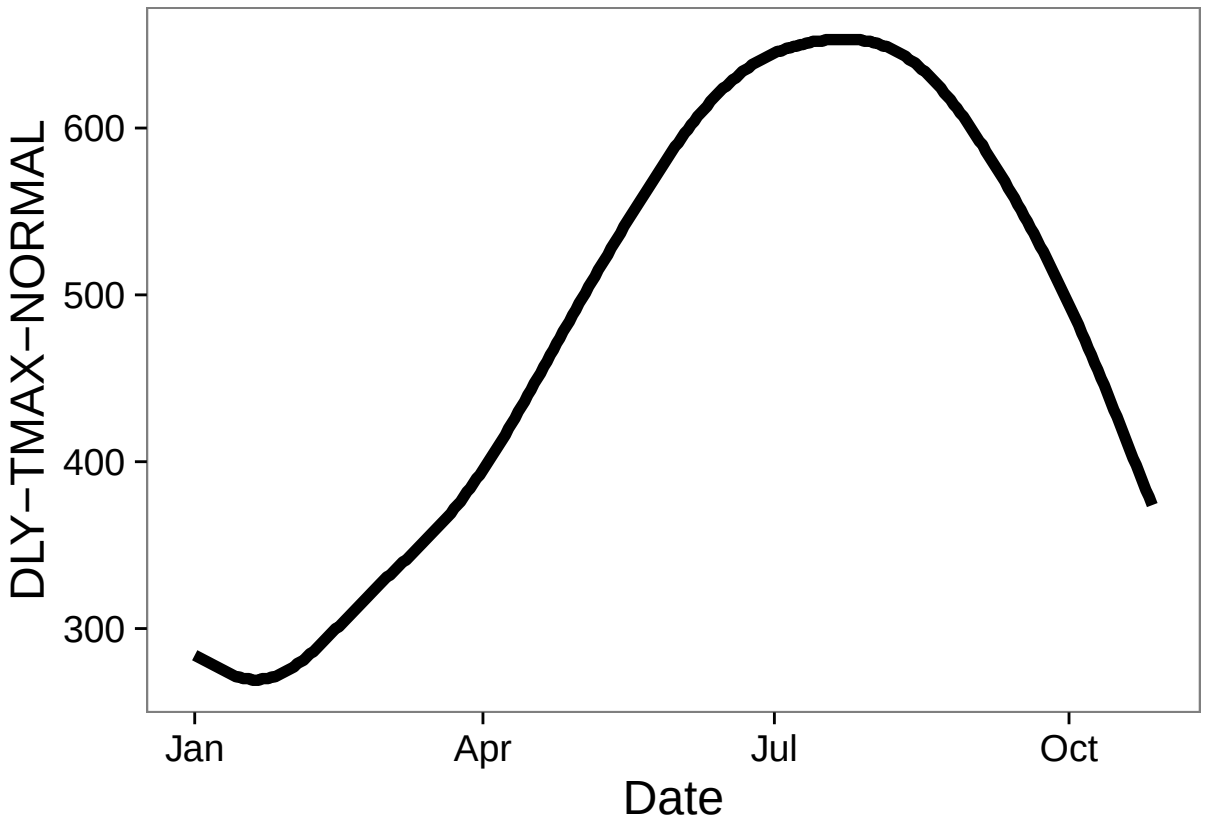
-JF: Trying to pull weather data for all stations in Alaska. Pulling for all stations will take a very long time, therefore a test trial using the first 3 stations listed for Alaska were used (`alaska_weather`). Stations were then limited by a variety of parameters to create a new data frame, `test`, to limit the number of monitors used to create a data frame with weather variables (`alaska_weather2`).

- MF: (Very preliminary, looks like JF was able to develop this further above!) We used `ncdc_stations` to get a list of stations in Alaska. First we had to find the FIPS code for Alaska, which is "02". It took some experimenting to figure out the correct way to input the `locationid` argument. The `limit` defaults to 25 so we needed to change it in order to see all 831 stations in Alaska. At this point we don't actually know if `GHCND` is the dataset we want, but it was one of the examples in the documentation for `ncdc_stations`. The station list may be different depending on the dataset you're using.
- RG: Adding a map of weather stations in Alaska, using MFs station dataframe

```
# options("noaakey" = Sys.getenv("noaakey"))
# library(devtools)
# install_github("ropenscilabs/rnoaa") # if you need to install the package

out <- ncdc_stations(datasetid='GHCND', locationid='FIPS:02', limit=1000)
out$meta # totalCount indicates the total number of stations
```

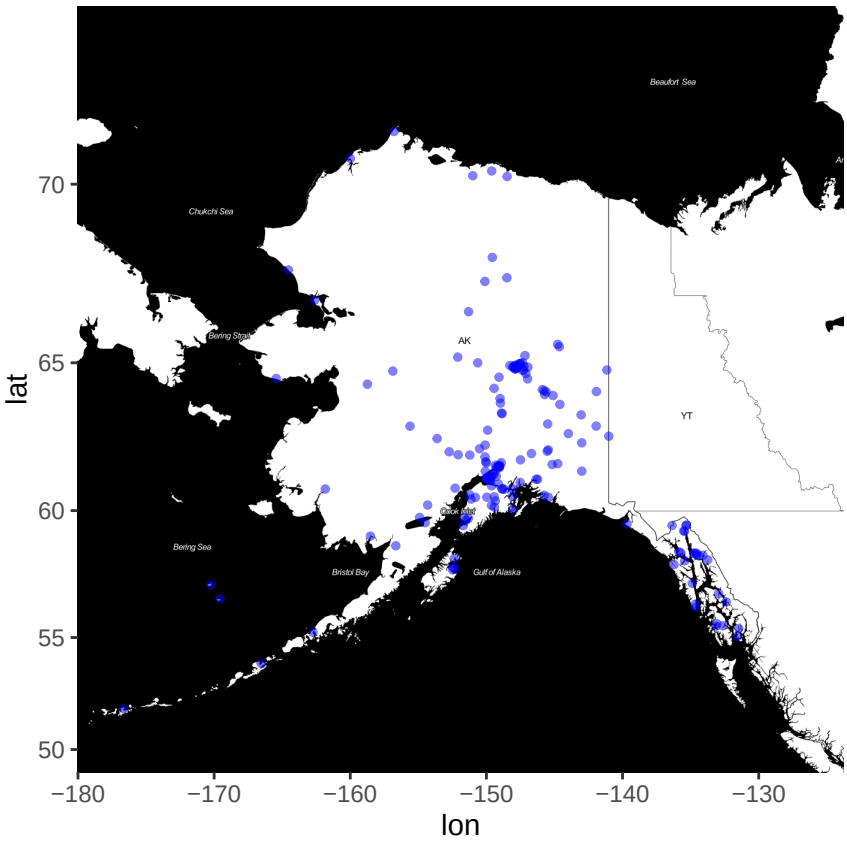
- MF: **THIS IS NOT DATA!** Following an example in a `rnoaa` tutorial, below I am using the `NORMAL_DLY` dataset (I'm not sure exactly what "dly-tmax-normal" is, but I'm assuming it must be some kind of climate normal, i.e. an average over a long time period).

```
# NORMAL_DLY has a different set of stations in Alaska (only 171 of them)
out <- ncdc_stations(datasetid='NORMAL_DLY', locationid='FIPS:02', limit=1000)
mydata <- data.frame(id=out[[2]]$id, lat=out[[2]]$latitude, lon=out[[2]]$longitude)

# I selected the first station id to pull some data from
temp <- ncdc(datasetid='NORMAL_DLY', datatypeid='dly-tmax-normal',
             stationid='GHCND:USC00500243',
             startdate = '2010-01-01', enddate = '2010-12-10', limit = 300)
ncdc_plot(temp)
```




```
# Plotting Weather Stations in Alaska
alaska <- c(-180, 52, -130, 72) # set bounding box by lat long min and max

station_map <- get_map(location= alaska, # use bounding box for location
                       source="stamen", # source stamen
                       maptype= "toner", crop=FALSE) # toner style

# adding station locations to the map
ggmap(station_map) + geom_point(aes(x = lon, y = lat),
                                data = mydata, alpha = .5, color="blue",
                                size = 1)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Taking a look at what the wunderground API has to offer. The following is exploratory and not yet operational. What is shown below is what looks like a promising source for hourly resolution weather data from high quality stations all over the world. Station metadata can be found http://weather.rap.ucar.edu/surface/stations.txt.

```
library(weatherData)

# Lets first check out what data is available, because I am a nerd (SJB) I
# Know the name of the station at Sea-tac
try(checkSummarizedDataAvailability("KSEA",
                                    start_date="2006-01-01",
                                    end_date = "2006-01-10",
                                    station_type = "airportCode"))



# Hmmmmmm looks like this package is broken. There is data at the url but
# the function is not pulling it. Maybe I should contact the author on
# CRAN.

# If you go to the followung URL the data clearly exists
#http://www.wunderground.com/history/airport/KSEA/2006/1/1/CustomHistory.html?dayend=10&monthend=1&year

# OK soo here is the deal, there are about a billion surface weather
# stations around the world.
# http://weather.rap.ucar.edu/surface/stations.txt

library(RCurl)
```

```
stationURLText <- "http://weather.rap.ucar.edu/surface/stations.txt"
lines <- readLines("http://weather.rap.ucar.edu/surface/stations.txt")
noHeader <- lines[44:length(lines)]
# Gives a billion locations of stations

# Below is some more broken code form this package.
# getDetailedWeather("KSEA", date="2006-01-01")

# But if you build a url that looks like this
deatiledDataURL <- "https://www.wunderground.com/history/airport/KSEA/2006/1/1/DailyHistory.html?format=

# Beautiful data with hourly weather observations
KSEAData <- read.csv(text = getURL(deatiledDataURL))

# This request could be made for any data and any station!
t <- as.POSIXct(KSEAData$DateUTC,tz="PST")
plot(t, KSEAData$TemperatureF, type="p",
     ylab="T [f]", xlab="Local Time (PST)",
     main="Example of hourly data avialable from Wunderground (KSEA)")
```

RNCEP package

This work is explore using the RNCEP package to obtain global weather and climate data.

-MG: explore some examples using this package.

[BA: For right now, I'm having some errors running this on my version of R, so I'm going to set to not evaluate until I can figure out why I'm having a problem.]

```
# install.packages('RNCEP')
library(RNCEP)

# retrieve the temperature from a particular pressure level for a specified spatial and temporal event
# in this example, gather temperature data from a spatial extent for August and September from 2000-200

wx.extent <- NCEP.gather(variable='air', level=850,
                         months.minmax=c(8,9), years.minmax=c(2000,2001),
                         lat.southnorth=c(50,55), lon.westeast=c(0,5),
                         reanalysis2 = FALSE, return.units=TRUE)
wx.extent

# calculate the average temperature per month and year
wx.ag <- NCEP.aggregate(wx.data=wx.extent, YEARS = TRUE,
                        MONTHS = TRUE, DAYS = FALSE, HOURS = FALSE, fxn='mean')
wx.ag
```

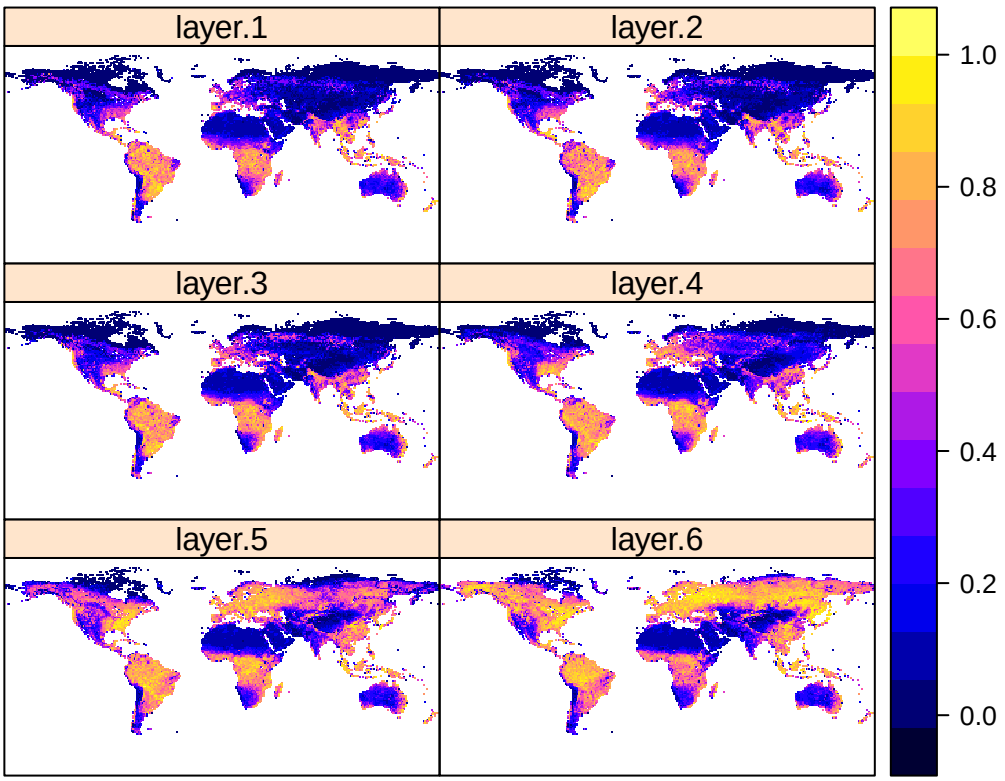
Vegetation / NDVI

Gather the data on the vegetation / NDVI throughout Alaska. Do so at as fine of a temporal and spatial resolution as possible. If you can gather this at a very fine spatial and temporal resolution, start by only collecting for the first week of July 2015. If you can only get yearly values, get values for each year from 2000-2016.

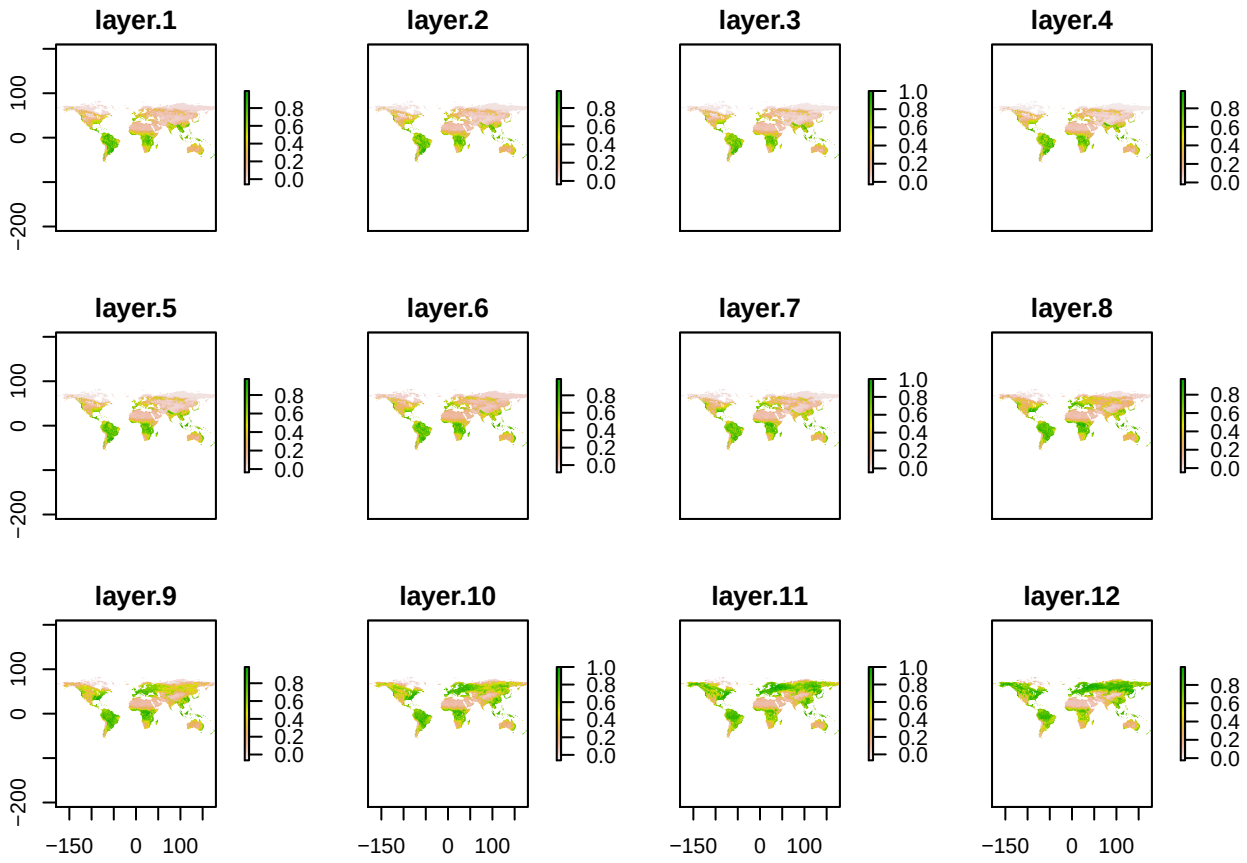
Tentative leads:

CRAN has an R package called "gimms", that purports to be able to read in GIMMS NDVI3g data. It seems the package was explicitly designed to work with data available at http://ecocast.arc.nasa.gov/data/pub/gimms/3g.v0/. The data runs from 1981 - 2013 in monthly increments.

```
library(gimms)

# Download the (.envi files I believe) by specifying the dates
gimms_files_date <- downloadGimms(x = as.Date("2013-01-01"), #Takes a while to run
                                  y = as.Date("2013-06-30"))

# .envi files are not useable as is, we need to rasterize them
gimms_raster <- rasterizeGimms(x = gimms_files_date, remove_header = TRUE)
# The first six months of 2013 took 24 seconds to rasterize. (2.7 Ghz i5, 8 GB RAM)
# Define some indices
indices <- monthlyIndices(gimms_files_date)
# Compute the monthly maximum value composites
gimms_raster_mvc <- monthlyComposite(gimms_raster, indices = indices) # 16 seconds

# Now, let's visualize the data
library(sp) # package for spatial data
spplot(gimms_raster_mvc) # This still works. 10 seconds
```

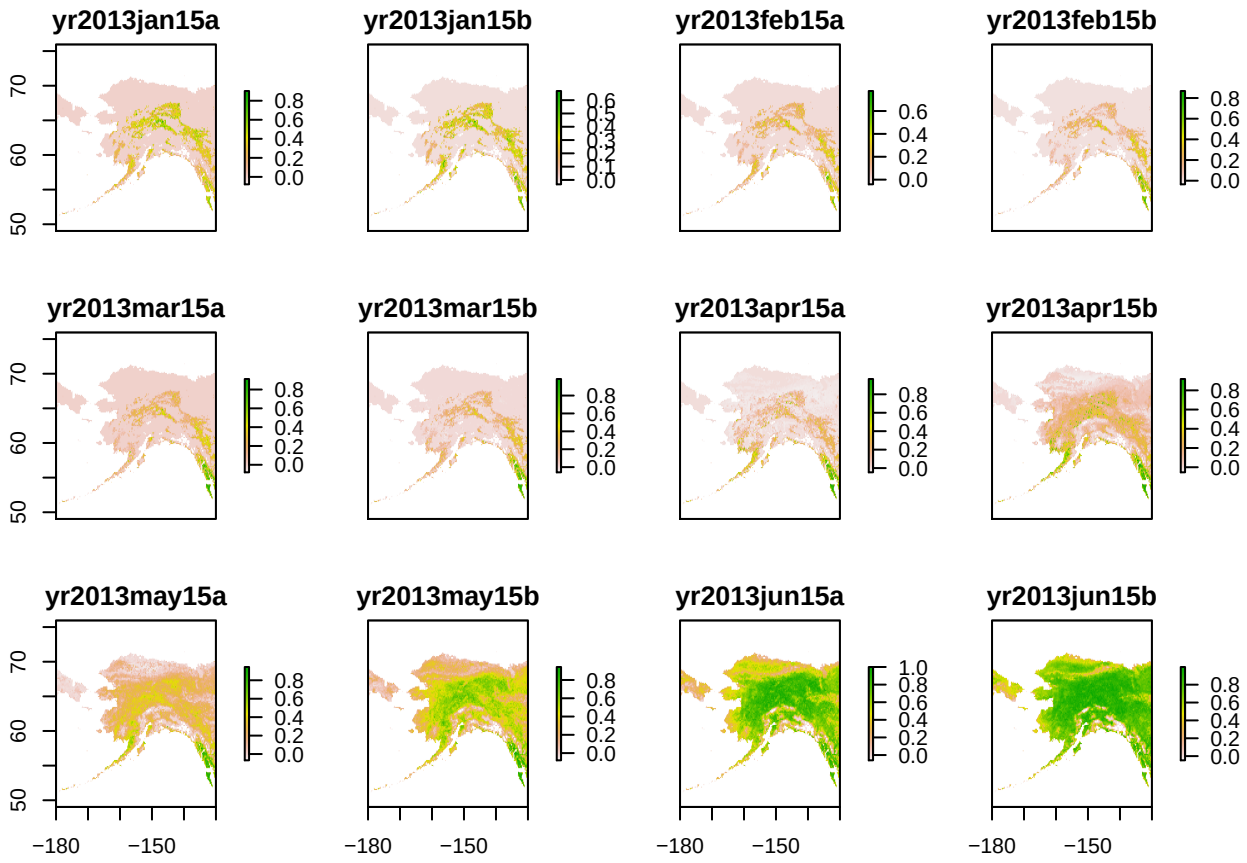



```
# This is an aesthetic plot, but what form should this data be in?
# We could also ignore the monthly maximum value composites.
plot(gimms_raster)
```



- MF: Continuing this example and bounding the extent to Alaska. We can see that the vegetation index increases from January 2013 to June 2013.

```
# Change the layer titles to which time period they correspond to
names(gimms_raster) <- gsub("\\.+.*", "", gsub(".+/geo", "yr20", gimms_files_date))

# Select a bounding box (extent)
ext <- extent(-180, -130, 50, 75)
  # ext <- drawExtent(show=TRUE) # Or you could use this to draw the bounding box manually (click on op

# Crop using ext
gimms_raster_crop <- crop(gimms_raster, ext)

# Now visualize the new cropped object
plot(gimms_raster_crop)
```




```
plot(gimms_raster_crop, 11) # to view just one of the layers (layer 11 in this case)
```