Using Google Earth Engine with R

CODE ▾

D G Rossiter

d.g.rossiter@cornell.edu (mailto:d.g.rossiter@cornell.edu)

13-June-2021

The `rgee` package (https://github.com/r-spatial/rgee) provides an interface from R to Google Earth Engine (GEE). This tutorial leads you through its installation and basic use.

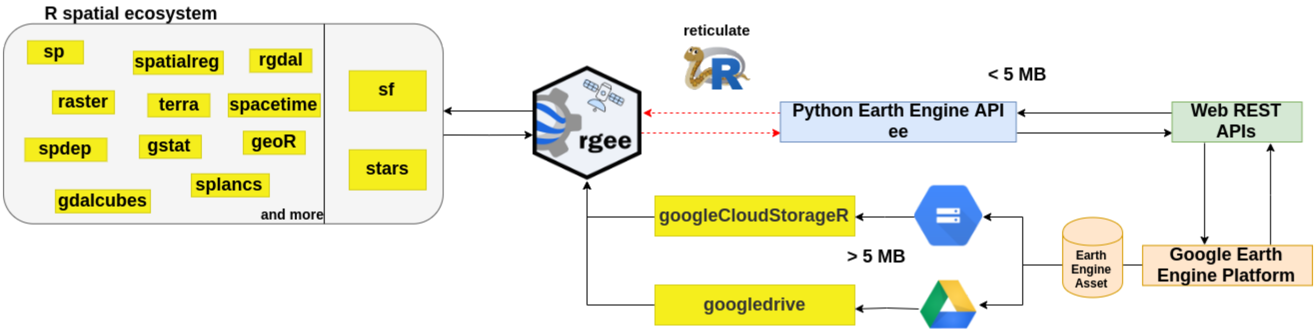
1 Setup

There is no native R interface to GEE. The `rgee` package uses on a link to the native Python interface and the R `reticulate` package which links R and Python.

A brief explanation of how the pieces fit together (R, Python, GEE) is at the bottom of the `rgee` package (https://github.com/r-spatial/rgee) page:

“How does rgee work?”

“ `rgee` is not a native Earth Engine API like the Javascript or Python client, to do this would be extremely hard, especially considering that the API is in active development. So, how is it possible to run Earth Engine using R? the answer is `reticulate`. `reticulate` is an R package designed to allow a seamless interoperability between R and Python. When an Earth Engine request is created in R, `reticulate` will transform this piece into Python. Once the Python code is obtained, the Earth Engine Python API transform the request to a JSON format. Finally, the request is received by the Google Earth Engine Platform thanks to a Web REST API. The response will follow the same path.”



1.1 Install Python

So the first step in using `rgee` is to install Python (version > 3.5) on your system.

Check your Python installation. There may be more versions on your system, if the default version is not > 3.5 you will have to manually specify the correct one.

HIDE

```
## install`reticulate`, only one time
if (!("reticulate" %in% installed.packages()[,1])) {
  print("Installing package `reticulate`...")
  install.packages("reticulate")
} else {
  print("Package `reticulate` already installed") }
```

```
## [1] "Package `reticulate` already installed"
```

HIDE

```
library(reticulate)
Sys.which("python")   # system default
```

```
##            python
## "/usr/bin/python"
```

HIDE

```
Sys.which("python3")  # is a V3 installed?
```

```
##                  python3
## "/usr/local/bin/python3"
```

HIDE

```
use_python(Sys.which("python3"))  # use it
```

Do some simple things in Python (via `reticulate`) to make sure it works. For example, import the `numpy` "numeric Python" Python library, define an array and show its cumulative sum. Here we also show the conversion to R objects using the `py_to_r` function.

HIDE

```
# use the standard Python numeric library
np <- reticulate::import("numpy", convert = FALSE)
# do some array manipulations with NumPy
a <- np$array(c(1:4))
print(a)  # this should be a Python array
```

```
## [1 2 3 4]
```

HIDE

```
print(py_to_r(a))  # this should be an R array
```

```
## [1] 1 2 3 4
```

HIDE

```
(sum <- a$cumsum())
```

```
## [ 1  3  6 10]
```

HIDE

```
# convert to R explicitly at the end
print(py_to_r(sum))
```

```
## [1]  1  3  6 10
```

If this works, your Python is set up correctly.

1.2 Install the `rgee` package

Second, install the package that contains the R functions to work with GEE. You only need to do this once.

HIDE

```
# install -- one time
## development version -- use with caution
# remotes::install_github("r-spatial/rgee")
## stable version on CRAN
if (!("rgee" %in% installed.packages()[,1])) {
  print("Installing package `rgee`...")
  install.packages("rgee")
} else
    { print("Package `rgee` already installed") }
```

The GitHub page for `rgee` (https://github.com/r-spatial/rgee) has a useful explanation on the main page on how to get `rgee` to work, and its syntax.

Load the `rgee` library:

HIDE

```
library(rgee)
```

1.3 Install required Python packages for GEE

Now that `rgee` is installed, it can be used to install the Python packages that interface to GEE; this only needs to be done once. Although this is an R function, the packages are installed in your Python installation, and can be used directly from Python if you wish.

HIDE

```
rgee::ee_install()
```

When you are asked if you want to store environment variables, answer `Y`.

At the end of this installation you should see something like:

> Well done! rgee was successfully set up in your system. You need restart R to see changes. After doing that, we recommend run ee_check() to perform a full check of all non-R rgee dependencies. Do you want restart your R session?

Answer `yes`.

2 Initializing the GEE interface

Each time you use GEE, you must establish a link with GEE.

This will ask you to authenticate with your GEE account, if you are not already logged in.

HIDE

```
library(rgee)
ee_check() # Check non-R dependencies
```

```
## ◉  Python version
## ✓ [Ok] /Users/rossiter/.virtualenvs/rgee/bin/python v3.7
## ◉  Python packages:
## ✓ [Ok] numpy
## ✓ [Ok] earthengine-api
```

HIDE

```
# ee_clean_credentials() # Remove credentials if you want to change the user
ee_clean_pyenv() # Remove reticulate system variables
ee_Initialize()
```

```
## ── rgee 1.0.9 ─────────────────────────────────────── earthengine-api 0.1.262 ──
##  ✓ email: not_defined
##  ✓ Initializing Google Earth Engine:
 ✓ Initializing Google Earth Engine:  DONE!
##
 ✓ Earth Engine user: users/cyrus621
## ────────────────────────────────────────────────────────────────────────────────
```

You might receive a message explaining how to upgrade the Earth Engine interface. If so, you should follow those instructions and then continue.

Now we are ready to work.

3 Example: Imagery

Task: Specify an image collection, filter it for a date range, and select one product:

HIDE

```
dataset <- ee$ImageCollection('LANDSAT/LC08/C01/T1_8DAY_EVI')$filterDate('2017-01-01', '20
17-12-31')
ee_print(dataset)
```

```
## Registered S3 method overwritten by 'geojsonsf':
##   method        from
##   print.geojson geojson
## ──────────────────────────────────────────────── Earth Engine ImageCollection ──
## ImageCollection Metadata:
##  - Class                      : ee$ImageCollection
##  - Number of Images           : 46
##  - Number of Properties       : 21
##  - Number of Pixels*          : 2980800
##  - Approximate size*          : 9.10 MB
## Image Metadata (img_index = 0):
##  - ID                         : LANDSAT/LC08/C01/T1_8DAY_EVI/20170101
##  - system:time_start          : 2017-01-01
##  - system:time_end            : 2017-01-09
##  - Number of Bands            : 1
##  - Bands names                : EVI
##  - Number of Properties       : 3
##  - Number of Pixels*          : 64800
##  - Approximate size*          : 202.50 KB
## Band Metadata (img_band = 'EVI'):
##  - EPSG (SRID)                : WGS 84 (EPSG:4326)
##  - proj4string                : +proj=longlat +datum=WGS84 +no_defs
##  - Geotransform               : 1 0 0 0 1 0
##  - Nominal scale (meters)     : 111319.5
##  - Dimensions                 : 360 180
##  - Number of Pixels           : 64800
##  - Data type                  : DOUBLE
##  - Approximate size           : 202.50 KB
##  ────────────────────────────────────────────────────────────────────────────────
##  NOTE: (*) Properties calculated considering a constant  geotransform and data type.
```

HIDE

```
landsat <- dataset$select('EVI')
class(landsat)
```

```
## [1] "ee.imagecollection.ImageCollection" "ee.collection.Collection"
## [3] "ee.element.Element"                 "ee.computedobject.ComputedObject"
## [5] "ee.encodable.Encodable"             "python.builtin.object"
```

HIDE

```
ee_print(landsat)
```

```
## ──────────────────────────────────────────────── Earth Engine ImageCollection ──
## ImageCollection Metadata:
##  - Class                      : ee$ImageCollection
##  - Number of Images           : 46
##  - Number of Properties       : 21
##  - Number of Pixels*          : 2980800
##  - Approximate size*          : 9.10 MB
## Image Metadata (img_index = 0):
##  - ID                         : LANDSAT/LC08/C01/T1_8DAY_EVI/20170101
##  - system:time_start          : 2017-01-01
##  - system:time_end            : 2017-01-09
##  - Number of Bands            : 1
##  - Bands names                : EVI
##  - Number of Properties       : 3
##  - Number of Pixels*          : 64800
##  - Approximate size*          : 202.50 KB
## Band Metadata (img_band = 'EVI'):
##  - EPSG (SRID)                : WGS 84 (EPSG:4326)
##  - proj4string                : +proj=longlat +datum=WGS84 +no_defs
##  - Geotransform               : 1 0 0 0 1 0
##  - Nominal scale (meters)     : 111319.5
##  - Dimensions                 : 360 180
##  - Number of Pixels           : 64800
##  - Data type                  : DOUBLE
##  - Approximate size           : 202.50 KB
##  ────────────────────────────────────────────────────────────────────────────────
##  NOTE: (*) Properties calculated considering a constant  geotransform and data type.
```

The syntax of `rgee` is based on the Javascript of GEE, but using R conventions. So for example the command to filter an `ImageCollection` by date has these syntaxes:

```
ee.ImageCollection().filterDate()  # Javascript
ee$ImageCollection()$filterDate()  # R
```

The function arguments are exactly as in Javascript.

Also, there is no need for the `var` declaration as in Javascript; any GEE objects defined by `<- ee$...` are references to objects on the GEE server, as you can see from the results of the `class()` function, which shows class names beginning with `ee.`, e.g., ee.imagecollection.ImageCollection.

The `ee_print()` function gives a nicely formatted summary of the object. You can see the classes of these R data types match those from GEE.

Task: Convert this `ImageCollection` to a multi-band `Image` and display the band names.

HIDE

```
evi <- landsat$select('EVI')$toBands()
class(evi)
```

```
## [1] "ee.image.Image"                   "ee.element.Element"
## [3] "ee.computedobject.ComputedObject" "ee.encodable.Encodable"
## [5] "python.builtin.object"
```

HIDE

```
ee_print(evi)
```

```
## Warning in ee_utils_py_to_r(.): restarting interrupted promise evaluation

## Warning in ee_utils_py_to_r(.): restarting interrupted promise evaluation

## Warning in ee_utils_py_to_r(.): restarting interrupted promise evaluation

## Warning in ee_utils_py_to_r(.): restarting interrupted promise evaluation
```

```
## ──────────────────────────────────────────────────────────── Earth Engine Image ──
## Image Metadata:
##  - Class                      : ee$Image
##  - ID                         : no_id
##  - Number of Bands            : 46
##  - Bands names                : 20170101_EVI 20170109_EVI 20170117_EVI 20170125_EVI 201
70202_EVI 20170210_EVI 20170218_EVI 20170226_EVI 20170306_EVI 20170314_EVI 20170322_EVI 20
170330_EVI 20170407_EVI 20170415_EVI 20170423_EVI 20170501_EVI 20170509_EVI 20170517_EVI 2
0170525_EVI 20170602_EVI 20170610_EVI 20170618_EVI 20170626_EVI 20170704_EVI 20170712_EVI
20170720_EVI 20170728_EVI 20170805_EVI 20170813_EVI 20170821_EVI 20170829_EVI 20170906_EVI
20170914_EVI 20170922_EVI 20170930_EVI 20171008_EVI 20171016_EVI 20171024_EVI 20171101_EVI
20171109_EVI 20171117_EVI 20171125_EVI 20171203_EVI 20171211_EVI 20171219_EVI 20171227_EVI
##  - Number of Properties       : 0
##  - Number of Pixels*          : 2980800
##  - Approximate size*          : 9.10 MB
## Band Metadata (img_band = 20170101_EVI):
##  - EPSG (SRID)                : WGS 84 (EPSG:4326)
##  - proj4string                : +proj=longlat +datum=WGS84 +no_defs
##  - Geotransform               : 1 0 0 0 1 0
##  - Nominal scale (meters)     : 111319.5
##  - Dimensions                 : 360 180
##  - Number of Pixels           : 64800
##  - Data type                  : DOUBLE
##  - Approximate size           : 202.50 KB
##  ────────────────────────────────────────────────────────────────────────────────
##  NOTE: (*) Properties calculated considering a constant geotransform and data type.
```

Task: Set a region of interest and centre the map display on it:

HIDE

```
region <- ee$Geometry$BBox(-76.7, 42.2, -76.2, 42.7)
Map$centerObject(region, 11);
```

Task: Set up visualization parameters for EVI images:

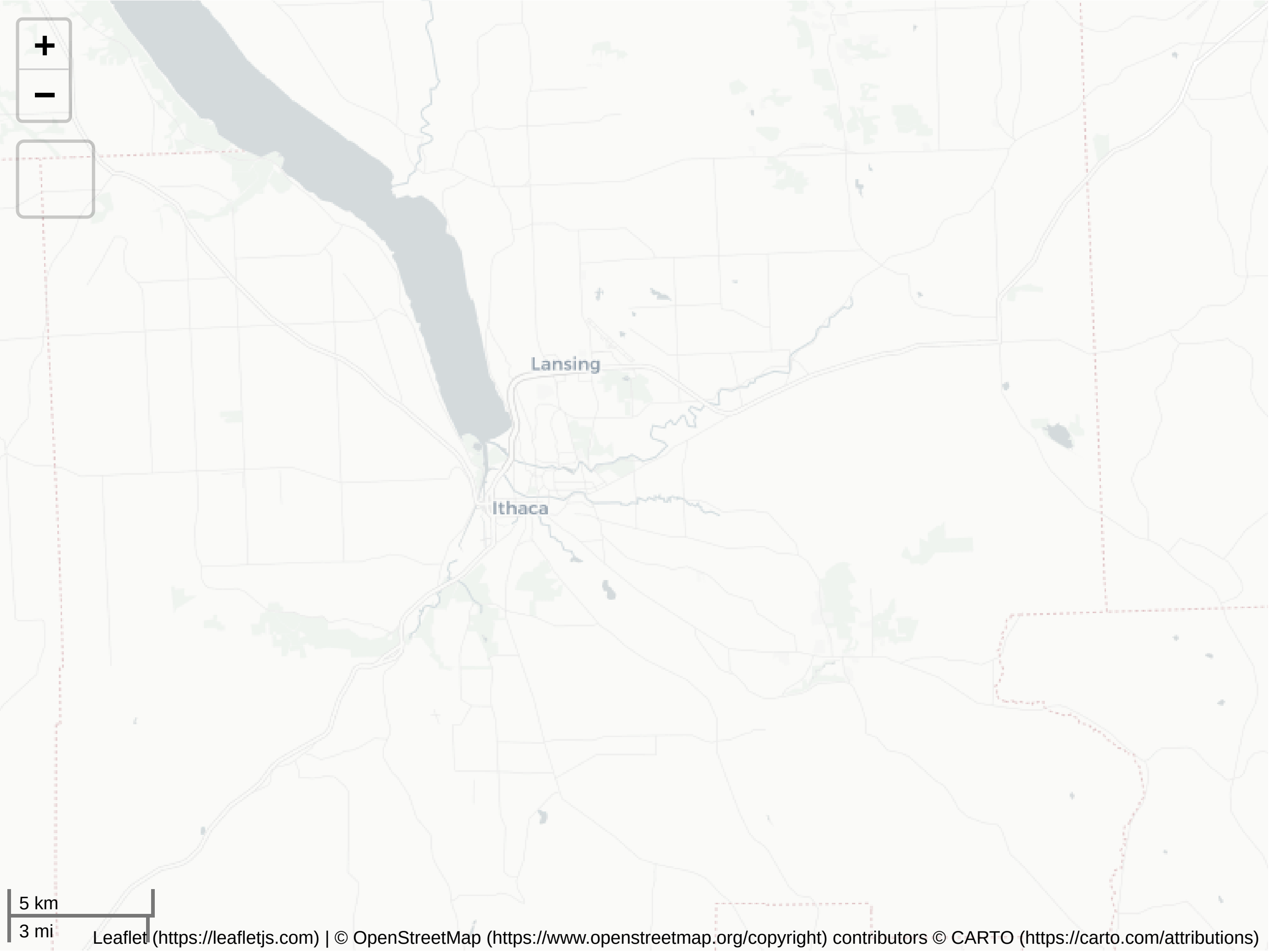
HIDE

```
colorizedVis <- list(
  min=0.0,
  max=1.0,
  palette=c(
    'FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718', '74A901',
    '66A000', '529400', '3E8601', '207401', '056201', '004C00', '023B01',
    '012E01', '011D01', '011301'
  )
)
```

Task: Select one date and display its EVI in a map window with this visualization:

HIDE

```
evi02jul <- evi$select("20170704_EVI")
Map$addLayer(evi02jul, colorizedVis, 'Landsat 8 EVI 02-July-2017')
```


+
−
Lansing
Ithaca
5 km
3 mi
Leaflet (https://leafletjs.com) | © OpenStreetMap (https://www.openstreetmap.org/copyright) contributors © CARTO (https://carto.com/attributions)

As in Javascript, `Map.addLayer` displays a map, here in the R graphics output window.

4 Example: `ggplot2` graphics on GEE objects

4.1 A simple chloropleth map

This is from the `rgee` examples (https://github.com/r-spatial/rgee). It shows how the results of GEE computation can easily be integrated with R functions, in this case a nice visualization of a time series.

Load the `tidyverse` data manipulation packages and the `sf` "Simple Features" spatial geometry package:

HIDE

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.6
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

HIDE

```
# library(rgee)
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.1.4, PROJ 6.3.1
```

Task: Read the `nc` shapefile of North Carolina counties. This is a built-in example in the `sf` package, accessed with the `system.file` function. It is described in the Help, `?nc`, which links to a long description (https://r-spatial.github.io/spdep/articles/sids.html) on the R-Spatial website.

> … the 100 counties of North Carolina, and includes counts of numbers of live births (also non-white live births) and numbers of sudden infant deaths, for the July 1, 1974 to June 30, 1978 and July 1, 1979 to June 30, 1984 periods.

Plot the number of births in 1974, by county:

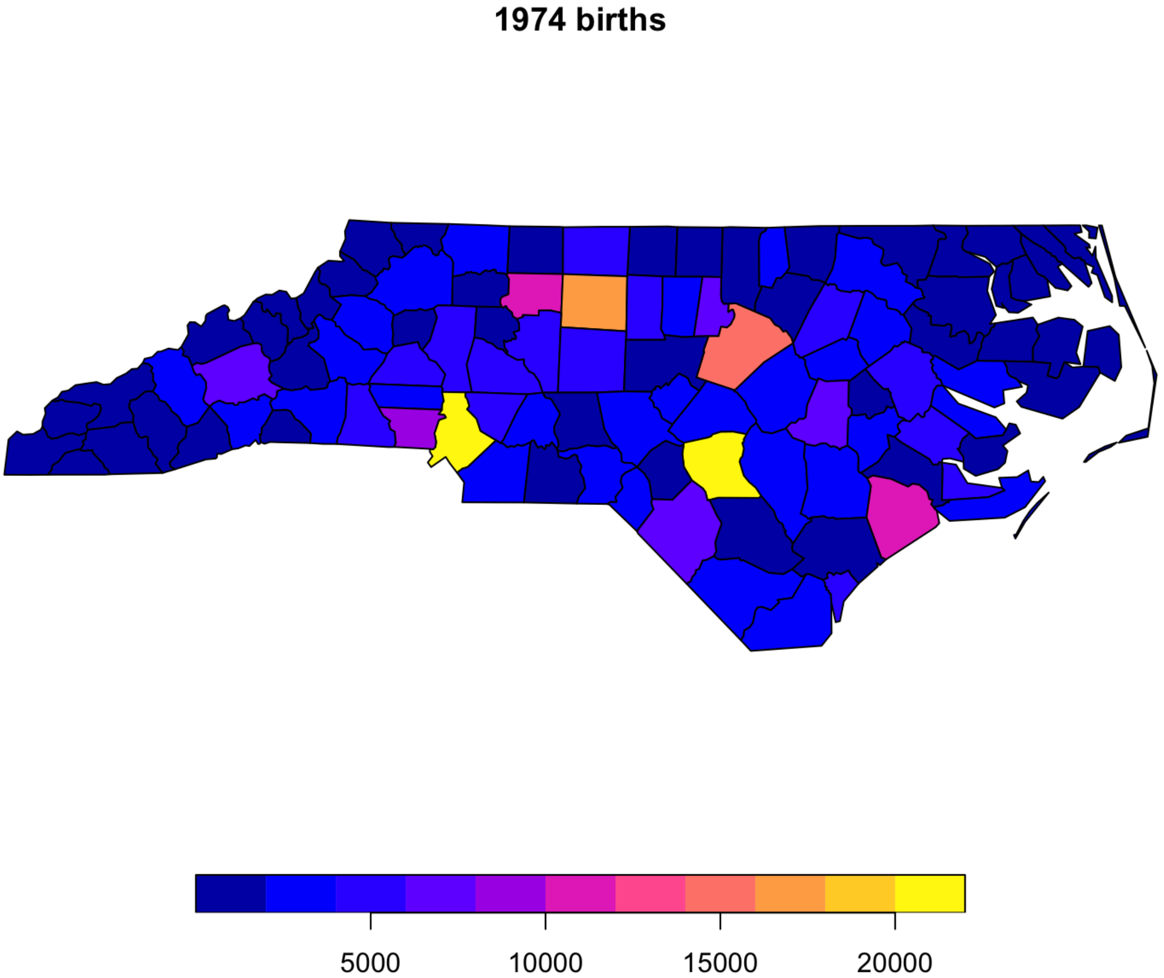
HIDE

```
nc <- st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
summary(nc)
```

```
##       AREA          PERIMETER         CNTY_          CNTY_ID    
##  Min.   :0.0420   Min.   :0.999   Min.   :1825   Min.   :1825  
##  1st Qu.:0.0910   1st Qu.:1.324   1st Qu.:1902   1st Qu.:1902  
##  Median :0.1205   Median :1.609   Median :1982   Median :1982  
##  Mean   :0.1263   Mean   :1.673   Mean   :1986   Mean   :1986  
##  3rd Qu.:0.1542   3rd Qu.:1.859   3rd Qu.:2067   3rd Qu.:2067  
##  Max.   :0.2410   Max.   :3.640   Max.   :2241   Max.   :2241  
##      NAME               FIPS               FIPSNO         CRESS_ID     
##  Length:100         Length:100         Min.   :37001   Min.   :  1.00  
##  Class :character   Class :character   1st Qu.:37050   1st Qu.: 25.75  
##  Mode  :character   Mode  :character   Median :37100   Median : 50.50  
##                                        Mean   :37100   Mean   : 50.50  
##                                        3rd Qu.:37150   3rd Qu.: 75.25  
##                                        Max.   :37199   Max.   :100.00  
##      BIR74           SID74          NWBIR74           BIR79      
##  Min.   :  248   Min.   : 0.00   Min.   :   1.0   Min.   :  319  
##  1st Qu.: 1077   1st Qu.: 2.00   1st Qu.: 190.0   1st Qu.: 1336  
##  Median : 2180   Median : 4.00   Median : 697.5   Median : 2636  
##  Mean   : 3300   Mean   : 6.67   Mean   :1050.8   Mean   : 4224  
##  3rd Qu.: 3936   3rd Qu.: 8.25   3rd Qu.:1168.5   3rd Qu.: 4889  
##  Max.   :21588   Max.   :44.00   Max.   :8027.0   Max.   :30757  
##      SID79          NWBIR79                 geometry  
##  Min.   : 0.00   Min.   :    3.0   MULTIPOLYGON :100  
##  1st Qu.: 2.00   1st Qu.:  250.5   epsg:4267    :  0  
##  Median : 5.00   Median :  874.5   +proj=long...:  0  
##  Mean   : 8.36   Mean   : 1352.8                      
##  3rd Qu.:10.25   3rd Qu.: 1406.8                      
##  Max.   :57.00   Max.   :11631.0                      
```

HIDE

```
plot(nc["BIR74"], main="1974 births")
```



HIDE

```
nc[order(nc$BIR74, decreasing=TRUE), c("NAME", "BIR74")]
```

```
## Simple feature collection with 100 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS:  NAD27
## First 10 features:
##          NAME BIR74                       geometry
## 68 Mecklenburg 21588 MULTIPOLYGON (((-81.0493 35...
## 82  Cumberland 20366 MULTIPOLYGON (((-78.49929 3...
## 26    Guilford 16184 MULTIPOLYGON (((-79.53782 3...
## 37        Wake 14484 MULTIPOLYGON (((-78.92107 3...
## 25     Forsyth 11858 MULTIPOLYGON (((-80.0381 36...
## 93      Onslow 11158 MULTIPOLYGON (((-77.53864 3...
## 76      Gaston  9014 MULTIPOLYGON (((-81.32282 3...
## 30      Durham  7970 MULTIPOLYGON (((-79.01814 3...
## 94     Robeson  7889 MULTIPOLYGON (((-78.86451 3...
## 53    Buncombe  7515 MULTIPOLYGON (((-82.2581 35...
```

Most births were in the heavily-populated Mecklenburg county (Charlotte), but also the less-populated Cumberland county (Fayetteville), which contains large military bases.

4.2 Climate analysis

Task: Make a reference to the TerraClimate (https://developers.google.com/earth-engine/datasets/catalog/IDAHO_EPSCOR_TERRACLIMATE/) “Monthly Climate and Climatic Water Balance for Global Terrestrial Surfaces” dataset.

HIDE

```
terraclimate <- ee$ImageCollection("IDAHO_EPSCOR/TERRACLIMATE")
print(terraclimate)
```

```
## EarthEngine Object: ImageCollection
```

Task: Filter this to the 2001 records, select only the precipitation bands, cast to an `ee.Image` , and rename the bands.

We do this with the `%>%` pipe operator of the `dplyr` library (loaded above). This chains a series of operations. In Javascript this is symbolized by `.` .

HIDE

```
terraclimate <- ee$ImageCollection("IDAHO_EPSCOR/TERRACLIMATE") %>% # dataset
  ee$ImageCollection$filterDate("2001-01-01", "2002-01-01") %>%   # data range
  ee$ImageCollection$map(function(x) x$select("pr")) %>% # Select only precipitation bands
  ee$ImageCollection$toBands() %>% # from ImageCollection to Image
  ee$Image$rename(sprintf("%02d",1:12)) # rename the bands of an image
print(terraclimate)
```

```
## EarthEngine Object: Image
```

The most interesting function here is `ee$ImageCollection$map()` . This `map` has nothing to do with the `Map` “display map” set of GEE functions. Here “map” is a mathematics term that means to apply some function *in parallel* over a set of objects. At this point in the pipe sequence the GEE object is an `ee.ImageCollection` , which has many `ee.Image` members. The `map` will apply the function defined with the R `function()` . Here this function is defined by:

```
function(x) x$select("pr")
```

The dummy argument `x` will be replaced by each `ee.Image` in the `ee.ImageCollection` , i.e., those images in the `TerraClimate` collection filtered by date. Then the `ee$Image$select` function will be run, the selection criterion being images named `"pr"` , i.e., the precipitation images.

How do we know this code? From the description of the bands (https://developers.google.com/earth-engine/datasets/catalog/IDAHO_EPSCOR_TERRACLIMATE/#bands) at the GEE Datasets Catalog.

After the set of precipitation images is selected, these are re-formatted to a set of bands in one `ee.Image` . This is possible because they all cover the same area and have the same data format.

Finally, the bands are renamed.

Task: Get some information about the `ee$Image` :

HIDE

```
bandNames <- terraclimate$bandNames()
cat("Band names: ",paste(bandNames$getInfo(),collapse=","))
```

```
## Band names:  01,02,03,04,05,06,07,08,09,10,11,12
```

HIDE

```
b0proj <- terraclimate$select('01')$projection()
cat("Band 01 projection: ", paste(b0proj$getInfo(),"\n", collapse = " "))
```

```
## Band 01 projection:  Projection
##  GEOGCS["unknown",
##   DATUM["unknown",
##     SPHEROID["Spheroid", 6378137.0, 298.257223563]],
##   PRIMEM["Greenwich", 0.0],
##   UNIT["degree", 0.017453292519943295],
##   AXIS["Longitude", EAST],
##   AXIS["Latitude", NORTH]]
##  list(0.0416666666666667, 0, -180, 0, -0.0416666666666667, 90)
```

HIDE

```
b0scale <- terraclimate$select('01')$projection()$nominalScale()
cat("Band 01 Scale: ", paste(b0scale$getInfo(),"\n", collapse = " "))
```

```
## Band 01 Scale:  4638.3121163864
```

HIDE

```
metadata <- terraclimate$propertyNames()
cat("Metadata: ", paste(metadata$getInfo(),"\n",collapse = " "))
```

```
## Metadata:  system:bands
##  system:band_names
```

Now we can see the link to R objects.

Task: Extract monthly precipitation values from the `Terraclimate ee$Image` Collection`.

This uses the `ee_extract` function, which requires the GEE object (`x`), the geometry (`y`), and a function to summarize the values (`fun`). Here the geometry has been defined as an `sf` “Simple Features” object.

HIDE

```
ee_nc_rain <- ee_extract(x = terraclimate, y = nc["NAME"], sf = FALSE)
```

```
## The image scale is set to 1000.
```

HIDE

```
str(ee_nc_rain)
```

```
## 'data.frame':    100 obs. of  13 variables:
##  $ NAME: chr  "Ashe" "Alleghany" "Surry" "Currituck" ...
##  $ X01 : num  62.3 53.5 56.7 42 37.5 ...
##  $ X02 : num  68.1 59.2 56.4 58.5 69.7 ...
##  $ X03 : num  139 135 130 94 119 ...
##  $ X04 : num  38.8 34.8 37.9 40.5 50.9 ...
##  $ X05 : num  114.4 127.2 124.5 68.6 61.5 ...
##  $ X06 : num  121 107 103 171 160 ...
##  $ X07 : num  219 189 159 118 139 ...
##  $ X08 : num  74.6 73.4 79.7 142 135.7 ...
##  $ X09 : num  105.9 91.4 78.3 48.9 44 ...
##  $ X10 : num  32.9 30.3 23.3 18.8 18.9 ...
##  $ X11 : num  26.4 24.2 16.3 16.4 18.6 ...
##  $ X12 : num  74.1 73.7 77.2 36.6 39.6 ...
```

A key point here is whether the returned object should be a spatial object (`sf = TRUE`) or a `data.frame` (`sf = FALSE`). Here we just want the data values, we already have the spatial information from the county map loaded above.

Notice the default scale for `ee_extract` is 1000 m. This can be changed with the `scale` optional argument.

Reformat the `data.frame` to make it easier to plot, using some functions from the `tidyverse` packages:

HIDE

```
ee_nc_rain_long <- ee_nc_rain %>%
  pivot_longer(-NAME, names_to = "month", values_to = "pr") %>%
  mutate(month, month=gsub("X", "", month)) # reformat the month name
str(ee_nc_rain_long)
```

```
## tibble [1,200 × 3] (S3: tbl_df/tbl/data.frame)
##  $ NAME : chr [1:1200] "Ashe" "Ashe" "Ashe" "Ashe" ...
##  $ month: chr [1:1200] "01" "02" "03" "04" ...
##  $ pr   : num [1:1200] 62.3 68.1 138.7 38.8 114.4 ...
```

These can now be plotted in various ways.

Task: Plot the Edgecombe county time series as a bar chart.

HIDE

```
dim(ee_nc_rain_long)  # 100 counties, county name + 12 months are the attributes
```

```
## [1] 1200    3
```

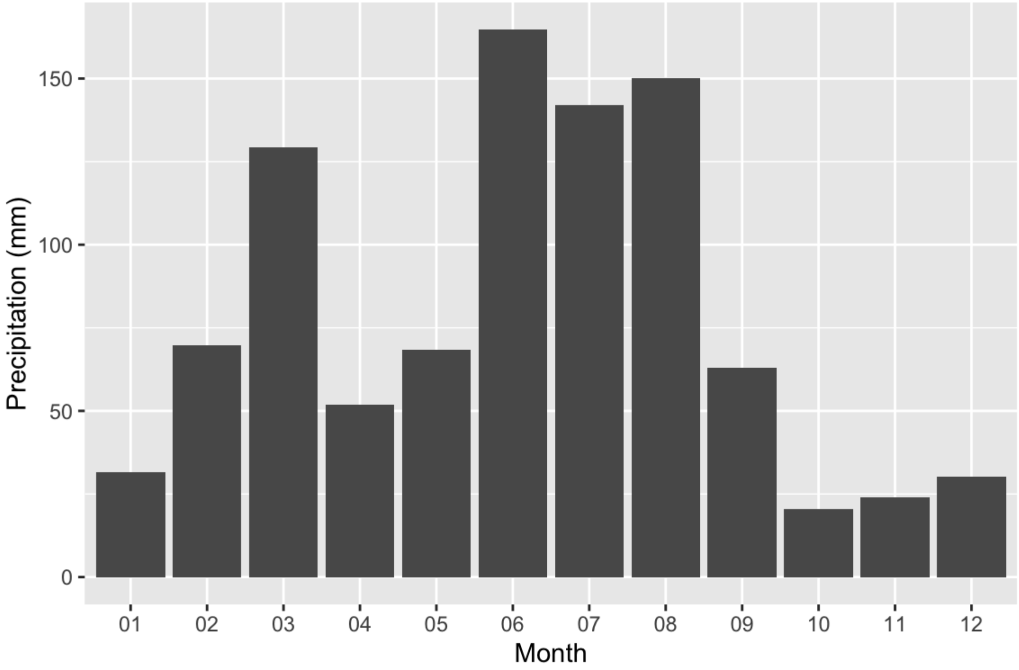
HIDE

```
sort(unique(ee_nc_rain_long$NAME)) # county names
```

```
##   [1] "Alamance"     "Alexander"    "Alleghany"    "Anson"        "Ashe"
##   [6] "Avery"        "Beaufort"     "Bertie"       "Bladen"       "Brunswick"
##  [11] "Buncombe"     "Burke"        "Cabarrus"     "Caldwell"     "Camden"
##  [16] "Carteret"     "Caswell"      "Catawba"      "Chatham"      "Cherokee"
##  [21] "Chowan"       "Clay"         "Cleveland"    "Columbus"     "Craven"
##  [26] "Cumberland"   "Currituck"    "Dare"         "Davidson"     "Davie"
##  [31] "Duplin"       "Durham"       "Edgecombe"    "Forsyth"      "Franklin"
##  [36] "Gaston"       "Gates"        "Graham"       "Granville"    "Greene"
##  [41] "Guilford"     "Halifax"      "Harnett"      "Haywood"      "Henderson"
##  [46] "Hertford"     "Hoke"         "Hyde"         "Iredell"      "Jackson"
##  [51] "Johnston"     "Jones"        "Lee"          "Lenoir"       "Lincoln"
##  [56] "Macon"        "Madison"      "Martin"       "McDowell"     "Mecklenburg"
##  [61] "Mitchell"     "Montgomery"   "Moore"        "Nash"         "New Hanover"
##  [66] "Northampton"  "Onslow"       "Orange"       "Pamlico"      "Pasquotank"
##  [71] "Pender"       "Perquimans"   "Person"       "Pitt"         "Polk"
##  [76] "Randolph"     "Richmond"     "Robeson"      "Rockingham"   "Rowan"
##  [81] "Rutherford"   "Sampson"      "Scotland"     "Stanly"       "Stokes"
##  [86] "Surry"        "Swain"        "Transylvania" "Tyrrell"      "Union"
##  [91] "Vance"        "Wake"         "Warren"       "Washington"   "Watauga"
##  [96] "Wayne"        "Wilkes"       "Wilson"       "Yadkin"       "Yancey"
```

HIDE

```
ee_nc_rain_long %>%
  filter(NAME=="Edgecombe") %>%
  ggplot() +
    geom_col(aes(x=month, y=pr)) +
    xlab("Month") + ylab("Precipitation (mm)")
```

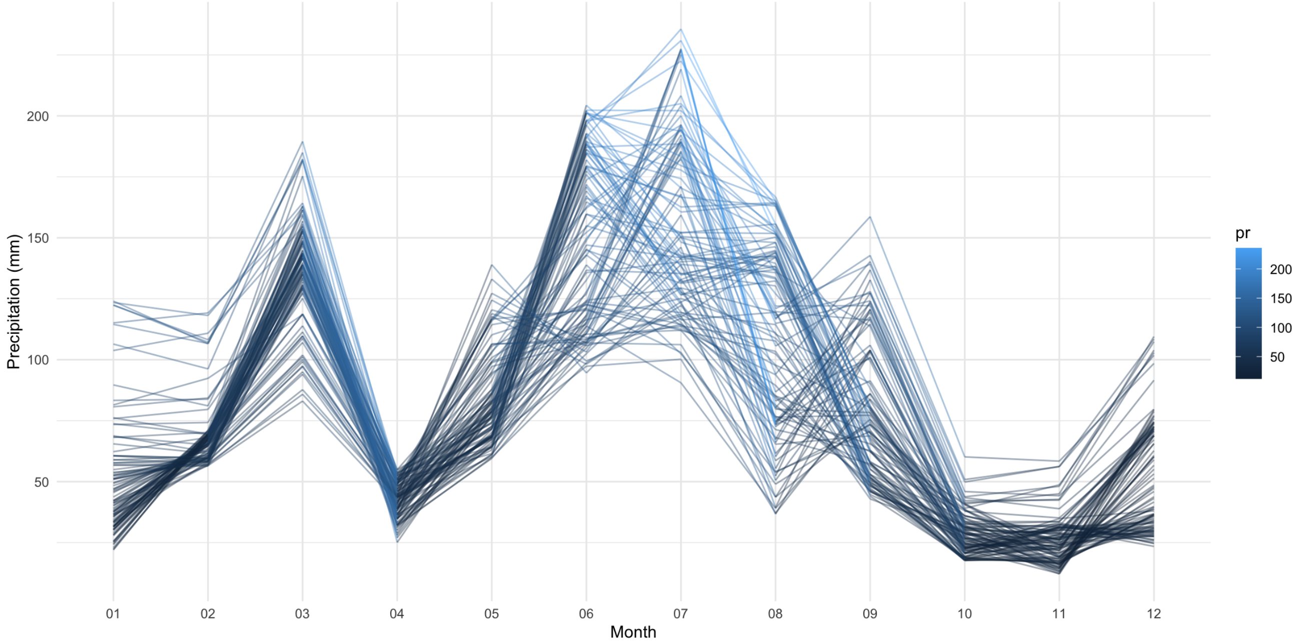


The early Spring and all Summer are the rainiest seasons.

Task: show all the counties as lines on one chart, each line segment coloured by the precipitation amoiunt in the previous month:

HIDE

```
ee_nc_rain_long %>%
  ggplot(aes(x = month, y = pr, group = NAME, color = pr)) +
   geom_line(alpha = 0.4) +
   xlab("Month") +
   ylab("Precipitation (mm)") +
   theme_minimal()
```



Most of the State has a similar precipitation pattern.

Task Make a chloropleth map of the January precipitation for the counties in the State.

Add the January precipitation to the NC counties geometry:

HIDE

```
ee_nc_rain_jan <- ee_nc_rain_long %>%
  filter(month=="01")
dim(ee_nc_rain_jan)
```

```
## [1] 100   3
```

HIDE

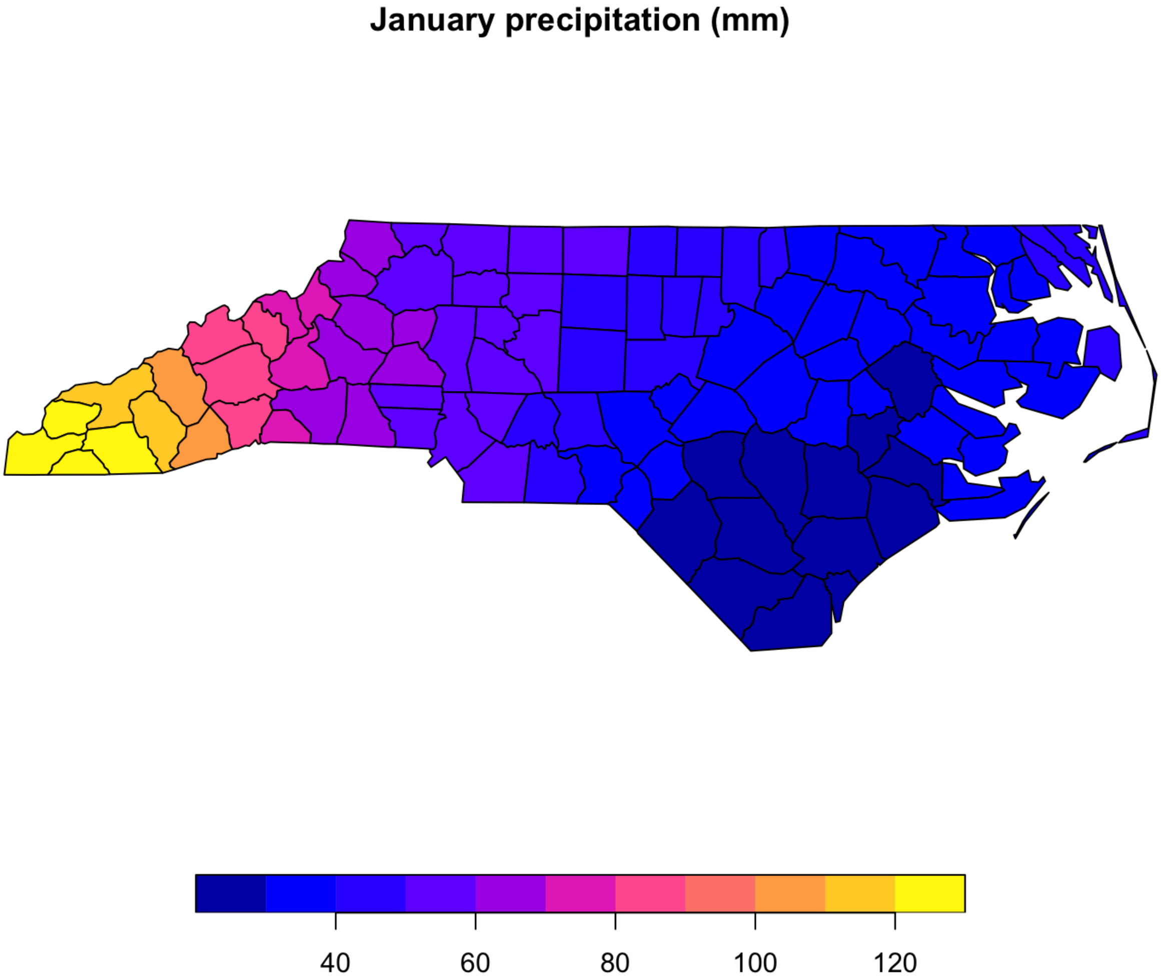
```
nc$JAN_PPT <- pull(ee_nc_rain_jan, pr)  # `pull` converts to a vector
str(nc)
```

```
## Classes 'sf' and 'data.frame':   100 obs. of  16 variables:
##  $ AREA     : num  0.114 0.061 0.143 0.07 0.153 0.097 0.062 0.091 0.118 0.124 ...
##  $ PERIMETER: num  1.44 1.23 1.63 2.97 2.21 ...
##  $ CNTY_    : num  1825 1827 1828 1831 1832 ...
##  $ CNTY_ID  : num  1825 1827 1828 1831 1832 ...
##  $ NAME     : chr  "Ashe" "Alleghany" "Surry" "Currituck" ...
##  $ FIPS     : chr  "37009" "37005" "37171" "37053" ...
##  $ FIPSNO   : num  37009 37005 37171 37053 37131 ...
##  $ CRESS_ID : int  5 3 86 27 66 46 15 37 93 85 ...
##  $ BIR74    : num  1091 487 3188 508 1421 ...
##  $ SID74    : num  1 0 5 1 9 7 0 0 4 1 ...
##  $ NWBIR74  : num  10 10 208 123 1066 ...
##  $ BIR79    : num  1364 542 3616 830 1606 ...
##  $ SID79    : num  0 3 6 2 3 5 2 2 2 5 ...
##  $ NWBIR79  : num  19 12 260 145 1197 ...
##  $ geometry :sfc_MULTIPOLYGON of length 100; first list element: List of 1
##   ..$ :List of 1
##   .. ..$ : num [1:27, 1:2] -81.5 -81.5 -81.6 -81.6 -81.7 ...
##   ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
##  $ JAN_PPT  : num  62.3 53.5 56.7 42 37.5 ...
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA N
A NA NA ...
##   ..- attr(*, "names")= chr [1:15] "AREA" "PERIMETER" "CNTY_" "CNTY_ID" ...
```

Plot it:

HIDE

```
plot(nc["JAN_PPT"], main="January precipitation (mm)")
```



Obviously the western counties (mountainous) get most of the winter precipitation.