Using Google Earth Engine with R

D G Rossiter

d.g.rossiter@cornell.edu

21-March-2023

Table of Contents

Setup.......1
- Install Python.......2
- Install the `rgee` package.......3
- Install required Python packages for GEE.......3
- Importing Python packages.......4
- Google Cloud.......4

Initializing the GEE interface.......4

Example: Imagery.......6

Importing GEE rasters into R.......9

Example: `ggplot2` graphics on GEE objects.......21
- A simple chloropleth map.......21
- Climate analysis.......24

Resources.......30

The `rgee` package provides an interface from R to Google Earth Engine (GEE). This tutorial leads you through its installation and basic use.

An extensive set of examples can be found `here`.

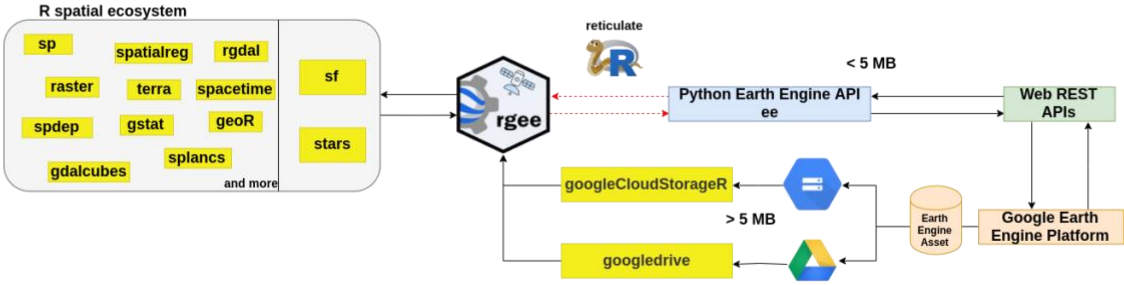
Setup

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 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."



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.

Also install the `reticulate` package that links R to Python.

```
## 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"

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

##             python3
## "/usr/bin/python3"

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

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.

```
# 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
```

```
## array([1, 2, 3, 4])
print(py_to_r(a))  # this should be an R array
## [1] 1 2 3 4
(sum <- a$cumsum())
## array([ 1,  3,  6, 10])
# 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.

Install the rgee package

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

```
# 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 has a useful explanation on the main page on how to get rgee to work, and its syntax.

You also must install the earthengine-api Python package. This depends on how you have set up your Python environment. Something like this:

```
# note the use of -H to set HOME variable to user's home dir
sudo -H python3 -m pip install earthengine-api
```

An explanation of the Python API for GEE is here.

Load the rgee library:

```
library(rgee)
```

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.

```
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`.

This simple use of `ee_install()` may fail if you have several Python installations. An alternative is to specify a Python 3 as follows, using the `ee_install_set_pyenv()` function (note: your path may be different):

```
rgee::ee_install_set_pyenv(py_path = "/usr/bin/python3", py_env="rgee")1
```

You might also be able to specify the Python environment with the `py_env` optional argument to `ee_install()`, see the help `?ee_install`.

Importing Python packages

If you want to work with other Python packages, these can be initialized in the Python environment with the `import` function of the `reticulate` package.

For example, the Scikit-Learn machine learning library for Python:

```
reticulate::import("sklearn")
```

Of course, these packages must be first installed in the Python environment, for example using the `pip3` shell command.

Google Cloud

To authenticate yourself to Google, you now need to use Google Cloud, with the `gcloud` command-line interface. Set this up here.

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.

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

## ◉  Python version
## ✔ [Ok] /usr/bin/python3 v3.9
## ◉  Python packages:
## ✔ [Ok] numpy
## ✔ [Ok] earthengine-api
```

```
## NOTE: The Earth Engine Python API version 0.1.341 is installed
## correctly in the system but rgee was tested using the version
## 0.1.345. To avoid possible issues, we recommend install the
## version used by rgee (0.1.345). You might use:
## * rgee::ee_install_upgrade()
## * reticulate::py_install('earthengine-api==0.1.345',
envname='PUT_HERE_YOUR_PYENV')
## * pip install earthengine-api==0.1.345 (Linux and MacOS)
## * conda install earthengine-api==0.1.345 (Linux, MacOS, and Windows)

# ee_clean_credentials() # Remove credentials if you want to change the user
ee_clean_pyenv() # Remove reticulate system variables
## do this interactively
# ee_Authenticate()
ee_Initialize()

## ── rgee 1.1.6.9999 ─────────────────────────────── earthengine-api
0.1.341 ──
##  ✔ user: not_defined
##  ✔ Initializing Google Earth Engine: ✔ Initializing Google Earth Engine:
DONE!
##  ✔ Earth Engine account: users/cyrus621
##
────────────────────────────────────────────────────────────────────────────
───

ee_user_info()

## ────────────────────────────────────────────────────── Earth Engine user
info ──
## Reticulate python version:
##  -  /usr/bin/python3
## Earth Engine Asset Home:
##  -  users/cyrus621
## Credentials Directory Path:
##  -  /Users/rossiter/.config/earthengine/
## Google Drive Credentials:
##  -  134f22af3ae3a9f0b7f0eb57dd61916f_cyrus621@gmail.com
## Google Cloud Storage Credentials:
##  -  NOT FOUND
##
────────────────────────────────────────────────────────────────────────────
───
```


```
## NOTE: The Earth Engine Python API version 0.1.341 is installed
## correctly in the system but rgee was tested using the version
## 0.1.345. To avoid possible issues, we recommend install the
## version used by rgee (0.1.345). You might use:
## * rgee::ee_install_upgrade()
## * reticulate::py_install('earthengine-api==0.1.345',
envname='PUT_HERE_YOUR_PYENV')
```

```
## * pip install earthengine-api==0.1.345 (Linux and MacOS)
## * conda install earthengine-api==0.1.345 (Linux, MacOS, and Windows)

## $asset_home
## [1] "users/cyrus621"
##
## $user_session
## [1] "/Users/rossiter/.config/earthengine/"
##
## $gd_id
## [1] "134f22af3ae3a9f0b7f0eb57dd61916f_cyrus621@gmail.com"
##
## $gcs_file
## [1] "NOT FOUND"
```

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.

Example: Imagery

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

```
dataset <-
ee$ImageCollection('LANDSAT/LC08/C01/T1_8DAY_EVI')$filterDate('2017-01-01',
'2017-12-31')
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       : 2
##  - 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.
```

```
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"
```

```
ee_print(landsat)
```

```
## ────────────────────────────────────────────────── Earth Engine
ImageCollection ──
## ImageCollection Metadata:
##   - Class                       : ee$ImageCollection
##   - Number of Images            : 46
##   - Number of Properties        : 2
##   - 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, so you should refer to the function descriptions in the GEE console, i.e., the “docs” tab of the code editor.

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. 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.

```
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"

ee_print(evi)

## ──────────────────────────────────────────────────────────── 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 20170202_EVI 20170210_EVI 20170218_EVI 20170226_EVI 20170306_EVI
20170314_EVI 20170322_EVI 20170330_EVI 20170407_EVI 20170415_EVI 20170423_EVI
20170501_EVI 20170509_EVI 20170517_EVI 20170525_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:

```
region.ithaca <- ee$Geometry$Rectangle(
  coords = c(-76.7, 42.2, -76.2, 42.7),
  proj = "EPSG:4326",
  geodesic = FALSE
)
Map$centerObject(region.ithaca, 11);
```

Task: Set up visualization parameters for EVI images:

```
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:

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

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

Importing GEE rasters into R

You may want to do some work with GEE rasters in R, using the `raster` or newer `terra` packages, or other spatial representations of rasters. For this the `ee_to_raster` function is used. This requires the R packages `googledrive` and `googleCloudStorageR`, which should have been installed with `rgee`.

We also need to load the R packages dealing with rasters, first `raster` and then `terra`. We will work in `terra` but the import is to the older `raster` package.

```
require(raster)
```

```
## Loading required package: raster

## Loading required package: sp

require(terra)

## Loading required package: terra

## terra 1.7.18
```

For large rasters this function by default uses your Google Drive storage to export the GEE raster and then import to R, so you will be asked allow `tidyverse` packages to use your Google Drive.

For example, we will import the EVI stack and process it in R.

We need to bound it with the `region` argument, a `ee$Geometry$Rectangle`). We defined a region above, for the map display, and export a downscaled EVI (1 km nominal pixel):

```
class(evi)

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

evi.r <- ee_as_raster(evi,
                      region=region.ithaca,
                      via = "drive",
                      scale = 1000)

## Warning: ee_as_raster will be removed in version 1.2.0. Please note that
you
## can use the ee_as_rast instead, which is compatible with terra packages.

##
## NOTE: Google Drive credentials were not loaded. Running ee_Initialize(user
= 'ndef', drive = TRUE) to fix.

## - region parameters
##  sfg      : POLYGON ((-76.7 42.2, -76.2 .... .7, -76.7 42.7, -76.7 42.2))
##  CRS      : GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
##         ELLIPSOID["WGS 84",6378137,298.257223563, .....
##  geodesic : FALSE
##  evenOdd  : TRUE
##
## - download parameters (Google Drive)
##  Image ID    : noid_image
##  Google user : ndef
##  Folder name : rgee_backup
##  Date        : 2023_03_21_18_26_14
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 0s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 5s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 10s>.
```

```
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 15s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 20s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 25s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 30s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 35s>.
## Polling for task <id: J3G4NE3KSBYMXMJ5PLGWWCDG, time: 40s>.
## State: COMPLETED
## Moving image from Google Drive to Local ... Please wait

## Auto-refreshing stale OAuth token.
```

```
class(evi.r)
```

```
## [1] "RasterStack"
## attr(,"package")
## [1] "raster"
```

Notice how the function polls the export task from GEE to your Google Drive, and waits until this is done. There is no guarantee about how long this will take.

Convert to a `terra` raster:

```
evi.r <- rast(evi.r)  # convert to terra
class(evi.r)
```

```
## [1] "SpatRaster"
## attr(,"package")
## [1] "terra"
```

```
dim(evi.r)
```

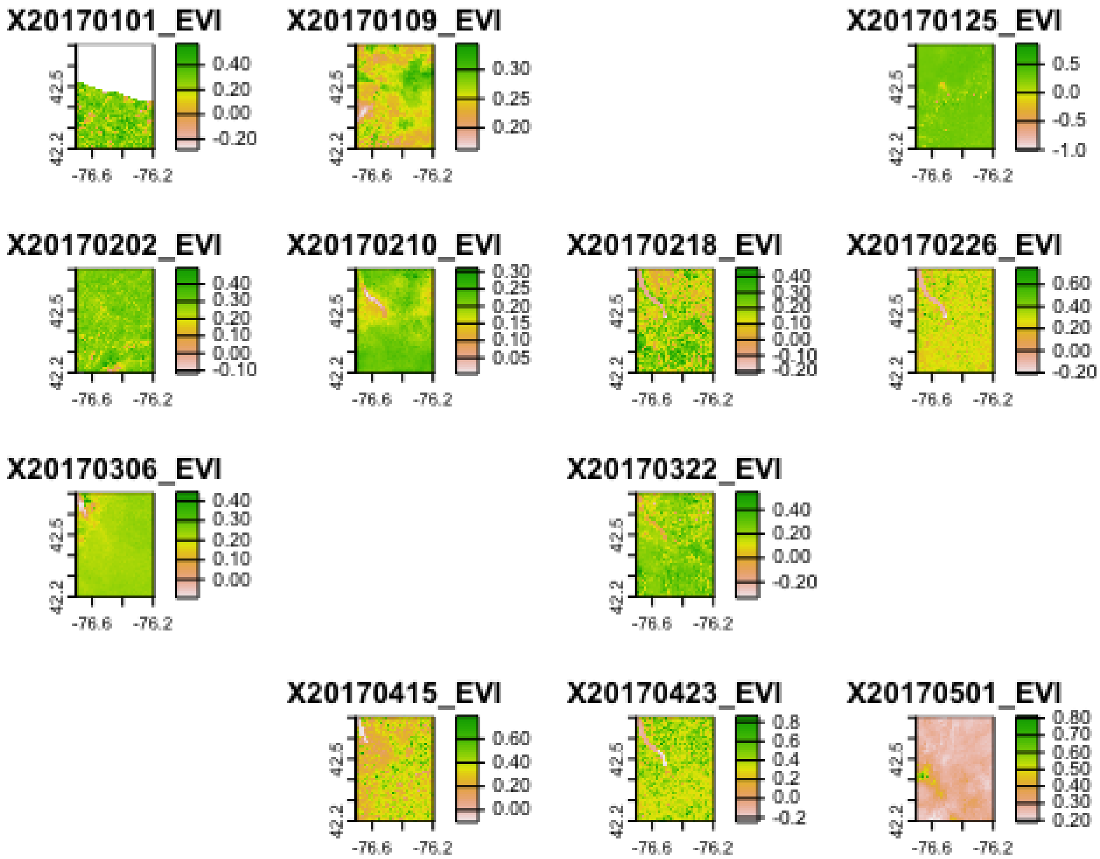
```
## [1] 57 57 46
```

```
summary(evi.r)
```

```
##  X20170101_EVI     X20170109_EVI    X20170117_EVI  X20170125_EVI
##  Min.   :-0.2868   Min.   :0.1621   Min.   : NA    Min.   :-1.0000
##  1st Qu.: 0.2252   1st Qu.:0.2373   1st Qu.: NA    1st Qu.: 0.2386
##  Median : 0.2767   Median :0.2514   Median : NA    Median : 0.2672
##  Mean   : 0.2668   Mean   :0.2532   Mean   :NaN    Mean   : 0.2564
##  3rd Qu.: 0.3319   3rd Qu.:0.2678   3rd Qu.: NA    3rd Qu.: 0.2898
##  Max.   : 0.5720   Max.   :0.3446   Max.   : NA    Max.   : 0.8578
##  NA's   :1493                       NA's   :3249
##  X20170202_EVI     X20170210_EVI      X20170218_EVI      X20170226_EVI
##  Min.   :-0.1191   Min.   :0.003603   Min.   :-0.21649   Min.   :-0.2107
##  1st Qu.: 0.2625   1st Qu.:0.183419   1st Qu.: 0.09432   1st Qu.: 0.2465
##  Median : 0.2851   Median :0.205097   Median : 0.16483   Median : 0.2791
##  Mean   : 0.2796   Mean   :0.198383   Mean   : 0.16793   Mean   : 0.2736
##  3rd Qu.: 0.3038   3rd Qu.:0.220094   3rd Qu.: 0.24100   3rd Qu.: 0.3141
##  Max.   : 0.4932   Max.   :0.310568   Max.   : 0.45424   Max.   : 0.7501
##  NA's   :5
##  X20170306_EVI      X20170314_EVI  X20170322_EVI     X20170330_EVI
##  Min.   :-0.09351   Min.   : NA    Min.   :-0.3319   Min.   : NA
```

```
##  1st Qu.: 0.21556   1st Qu.: NA    1st Qu.: 0.1772   1st Qu.: NA
##  Median : 0.22643   Median : NA    Median : 0.2263   Median : NA
##  Mean   : 0.21874   Mean   :NaN    Mean   : 0.2179   Mean   :NaN
##  3rd Qu.: 0.23472   3rd Qu.: NA    3rd Qu.: 0.2654   3rd Qu.: NA
##  Max.   : 0.44390   Max.   : NA    Max.   : 0.5462   Max.   : NA
##                     NA's   :3249                     NA's   :3249
##  X20170407_EVI  X20170415_EVI     X20170423_EVI     X20170501_EVI
##  Min.   : NA    Min.   :-0.1012   Min.   :-0.2488   Min.   :0.1890
##  1st Qu.: NA    1st Qu.: 0.2465   1st Qu.: 0.3137   1st Qu.:0.2554
##  Median : NA    Median : 0.3130   Median : 0.3609   Median :0.2705
##  Mean   :NaN    Mean   : 0.3109   Mean   : 0.3716   Mean   :0.2821
##  3rd Qu.: NA    3rd Qu.: 0.3729   3rd Qu.: 0.4347   3rd Qu.:0.2936
##  Max.   : NA    Max.   : 0.7902   Max.   : 0.8603   Max.   :0.8108
##  NA's   :3249
##  X20170509_EVI     X20170517_EVI       X20170525_EVI  X20170602_EVI
##  Min.   :-0.2006   Min.   :-0.001267   Min.   : NA    Min.   :-0.08114
##  1st Qu.: 0.4202   1st Qu.: 0.447062   1st Qu.: NA    1st Qu.: 0.49282
##  Median : 0.4712   Median : 0.573556   Median : NA    Median : 0.73784
##  Mean   : 0.4647   Mean   : 0.570256   Mean   :NaN    Mean   : 0.68293
##  3rd Qu.: 0.5166   3rd Qu.: 0.697907   3rd Qu.: NA    3rd Qu.: 0.88607
##  Max.   : 0.9161   Max.   : 1.000000   Max.   : NA    Max.   : 1.00000
##                                        NA's   :3249
##  X20170610_EVI      X20170618_EVI     X20170626_EVI      X20170704_EVI
##  Min.   :-0.04758   Min.   :-0.0161   Min.   :-0.06577   Min.   :-0.1061
##  1st Qu.: 0.52654   1st Qu.: 0.1158   1st Qu.: 0.39871   1st Qu.: 0.7496
##  Median : 0.68652   Median : 0.1775   Median : 0.58975   Median : 0.8456
##  Mean   : 0.67620   Mean   : 0.1768   Mean   : 0.60768   Mean   : 0.7899
##  3rd Qu.: 0.84991   3rd Qu.: 0.2373   3rd Qu.: 0.84701   3rd Qu.: 0.8962
##  Max.   : 1.00000   Max.   : 0.3772   Max.   : 1.00000   Max.   : 1.0000
##
##  X20170712_EVI     X20170720_EVI      X20170728_EVI    X20170805_EVI
##  Min.   :-0.2545   Min.   :-0.07964   Min.   :0.3387   Min.   :0.1435
##  1st Qu.: 0.1487   1st Qu.: 0.74898   1st Qu.:0.5261   1st Qu.:0.7193
##  Median : 0.2429   Median : 0.82593   Median :0.5652   Median :0.7944
##  Mean   : 0.2317   Mean   : 0.78349   Mean   :0.5663   Mean   :0.7741
##  3rd Qu.: 0.3270   3rd Qu.: 0.87934   3rd Qu.:0.6058   3rd Qu.:0.8579
##  Max.   : 0.6991   Max.   : 1.00000   Max.   :0.8399   Max.   :1.0000
##  NA's   :11
##  X20170813_EVI      X20170821_EVI    X20170829_EVI     X20170906_EVI
##  Min.   :-0.08419   Min.   :0.2346   Min.   :0.01794   Min.   :-0.08604
##  1st Qu.: 0.46810   1st Qu.:0.5625   1st Qu.:0.36793   1st Qu.: 0.46287
##  Median : 0.72585   Median :0.6686   Median :0.41506   Median : 0.58717
##  Mean   : 0.65070   Mean   :0.6886   Mean   :0.45309   Mean   : 0.60258
##  3rd Qu.: 0.84241   3rd Qu.:0.8359   3rd Qu.:0.49671   3rd Qu.: 0.75779
##  Max.   : 1.00000   Max.   :1.0000   Max.   :1.00000   Max.   : 1.00000
##
##  X20170914_EVI     X20170922_EVI      X20170930_EVI     X20171008_EVI
##  Min.   :-0.1090   Min.   :-0.06977   Min.   :0.09507   Min.   : NA
##  1st Qu.: 0.2822   1st Qu.: 0.60024   1st Qu.:0.37898   1st Qu.: NA
##  Median : 0.3265   Median : 0.64770   Median :0.47002   Median : NA
##  Mean   : 0.3167   Mean   : 0.63140   Mean   :0.48722   Mean   :NaN
```

```
##  3rd Qu.: 0.3700   3rd Qu.: 0.69875   3rd Qu.:0.57860   3rd Qu.: NA
##  Max.   : 0.6640   Max.   : 1.00000   Max.   :0.99677   Max.   : NA
##  NA's   :10                           NA's   :5         NA's   :3249
##  X20171016_EVI    X20171024_EVI      X20171101_EVI  X20171109_EVI
##  Min.   :0.1040   Min.   :-0.08065   Min.   : NA    Min.   :-0.1664
##  1st Qu.:0.2623   1st Qu.: 0.36562   1st Qu.: NA    1st Qu.: 0.2145
##  Median :0.3258   Median : 0.42125   Median : NA    Median : 0.2848
##  Mean   :0.3538   Mean   : 0.42906   Mean   :NaN    Mean   : 0.2921
##  3rd Qu.:0.4202   3rd Qu.: 0.48726   3rd Qu.: NA    3rd Qu.: 0.3653
##  Max.   :0.9836   Max.   : 1.00000   Max.   : NA    Max.   : 1.0000
##  NA's   :1504                        NA's   :3249
##  X20171117_EVI     X20171125_EVI     X20171203_EVI     X20171211_EVI
##  Min.   :-0.1272   Min.   :-0.2631   Min.   :-0.1425   Min.   : NA
##  1st Qu.: 0.2587   1st Qu.: 0.2249   1st Qu.: 0.2348   1st Qu.: NA
##  Median : 0.3244   Median : 0.2361   Median : 0.2897   Median : NA
##  Mean   : 0.3220   Mean   : 0.2315   Mean   : 0.2845   Mean   :NaN
##  3rd Qu.: 0.3924   3rd Qu.: 0.2450   3rd Qu.: 0.3388   3rd Qu.: NA
##  Max.   : 0.8242   Max.   : 0.2839   Max.   : 0.7928   Max.   : NA
##                                                        NA's   :3249
##  X20171219_EVI     X20171227_EVI
##  Min.   :-0.1414   Min.   :-1.0000
##  1st Qu.: 0.2293   1st Qu.: 0.2370
##  Median : 0.3020   Median : 0.2710
##  Mean   : 0.2901   Mean   : 0.2582
##  3rd Qu.: 0.3695   3rd Qu.: 0.2992
##  Max.   : 0.6870   Max.   : 0.5214
##

plot(evi.r)
```




```
names(evi.r)
```

```
##  [1] "X20170101_EVI" "X20170109_EVI" "X20170117_EVI" "X20170125_EVI"
##  [5] "X20170202_EVI" "X20170210_EVI" "X20170218_EVI" "X20170226_EVI"
##  [9] "X20170306_EVI" "X20170314_EVI" "X20170322_EVI" "X20170330_EVI"
## [13] "X20170407_EVI" "X20170415_EVI" "X20170423_EVI" "X20170501_EVI"
## [17] "X20170509_EVI" "X20170517_EVI" "X20170525_EVI" "X20170602_EVI"
## [21] "X20170610_EVI" "X20170618_EVI" "X20170626_EVI" "X20170704_EVI"
## [25] "X20170712_EVI" "X20170720_EVI" "X20170728_EVI" "X20170805_EVI"
## [29] "X20170813_EVI" "X20170821_EVI" "X20170829_EVI" "X20170906_EVI"
## [33] "X20170914_EVI" "X20170922_EVI" "X20170930_EVI" "X20171008_EVI"
## [37] "X20171016_EVI" "X20171024_EVI" "X20171101_EVI" "X20171109_EVI"
## [41] "X20171117_EVI" "X20171125_EVI" "X20171203_EVI" "X20171211_EVI"
## [45] "X20171219_EVI" "X20171227_EVI"
```

There are 3249 pixels in each image.

Now we can do some typical raster functions from within R.

For example, k-means clustering to find more-or-less homogeneous zones within one date (10-February, in the winter):

```
# one image
k5 <- kmeans(as.vector(evi.r["20170210_EVI"]), centers = 5, nstart = 5)
print(k5)
```

```
## K-means clustering with 5 clusters of sizes 673, 1203, 855, 464, 54
##
## Cluster means:
##         [,1]
## 1 0.23273212
## 2 0.21186044
## 3 0.18682551
## 4 0.15239892
## 5 0.04814491
##
## Clustering vector:
##    [1] 3 2 2 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 2 2 2 2 2 3 3 2 2 2 1 1 1 2 1 1
1 2 2
##   [38] 1 1 1 1 1 1 1 2 1 1 2 1 2 2 1 3 2 1 1 1 3 3 3 1 1 1 1 1 1 1 1 2 2 1
1 2 2
##   [75] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 1 1 2 1 1 1 1 1 2 2 1 2 2 2 2 1
1 3 3
##  [112] 2 2 2 3 4 3 3 1 1 1 1 1 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2
2 2 2
##  [149] 2 2 2 1 1 1 1 1 2 1 1 1 1 1 1 1 2 3 2 3 2 3 3 3 3 3 3 2 1 1 1 1 1 1
2 2 2
##  [186] 1 2 2 2 2 2 1 1 2 1 2 2 2 2 1 1 2 2 3 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2
2 2 2
##  [223] 2 3 3 3 1 3 2 3 3 4 2 1 1 1 1 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 3
3 3 3
##  [260] 3 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 3 3 3 1 2 3 4 4 1 1 1
1 1 1
##  [297] 2 2 1 1 1 1 1 1 1 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1 1 1 1 1 1 1
1 2 2
##  [334] 3 3 3 3 3 3 4 4 4 1 2 3 3 3 3 1 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 2 2 3
2 3 2
##  [371] 3 3 3 3 3 3 2 3 3 3 3 1 1 1 1 2 1 2 1 2 2 3 3 4 3 4 4 4 4 2 2 3 4 4
3 1 1
##  [408] 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 2 2
2 2 2
##  [445] 2 3 2 3 3 3 3 3 4 4 4 4 2 2 4 4 4 4 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2
2 2 3
##  [482] 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 1 2 2 2 3 3 2 3 3 3 3 4 4 4 4 4 2 2
4 4 4
##  [519] 4 4 3 1 2 2 3 2 2 3 2 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 3 3 3 3 3 3 2
1 1 2
##  [556] 2 2 2 3 3 3 2 3 3 4 3 4 3 4 4 2 3 2 4 4 5 5 4 2 3 3 2 3 2 2 2 1 3 3
2 2 2
##  [593] 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 2 2 3 2 2 3 3 3 3 3 4 4 4 4 4 4 4
4 3 3
##  [630] 3 3 4 5 5 5 3 3 3 3 3 2 2 3 3 3 2 2 2 3 3 3 3 3 3 3 3 3 2 2 3 3 3 2
3 3 2
##  [667] 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4 4 4 3 3 3 3 4 5 5 5 5 4 4 4 3 3 3 3
3 3 2
##  [704] 3 3 2 3 2 2 3 3 3 3 3 2 2 3 3 3 2 2 2 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4
4 4 4
```

```
##  [741] 4 3 3 3 3 4 4 5 5 5 5 4 4 4 4 4 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 1 2 2
3 2 3
##  [778] 1 2 2 1 2 2 2 3 2 3 3 3 3 4 4 4 4 4 4 4 4 3 3 3 3 3 4 4 5 5 5 5 4 3
4 4 4
##  [815] 4 3 3 3 3 3 3 2 2 3 3 3 3 3 2 2 3 3 3 2 3 2 2 1 2 2 2 2 3 3 3 3 4 4
4 4 4
##  [852] 4 4 4 4 3 3 3 3 4 4 4 4 5 5 5 5 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 2
2 1 1
##  [889] 1 1 1 2 1 2 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4 4 3 3 4 4 4 4 4 4 4 5
5 5 5
##  [926] 5 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 1 1 2 1 1 1 2 2 2 2 3 3 3
4 4 4
##  [963] 3 4 4 4 3 4 4 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 4 4 4 4 4 4 4 3 3 3 3
2 2 2
## [1000] 2 2 2 2 2 2 1 1 2 2 2 2 3 3 3 2 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4
## [1037] 4 4 4 5 5 5 5 5 4 4 4 4 4 3 3 3 3 3 2 2 2 2 2 2 2 1 2 1 2 1 1 1 3 3
2 2 3
## [1074] 3 4 3 4 4 4 4 4 4 4 3 4 4 3 3 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 4 4 4 3
3 3 2
## [1111] 3 3 3 2 2 2 1 2 2 2 1 2 2 2 3 3 3 3 3 3 3 4 3 3 4 4 4 4 3 3 3 4 4 4
4 4 4
## [1148] 4 4 4 4 4 4 4 4 4 4 5 5 5 4 4 4 4 3 3 3 3 3 3 3 2 2 2 1 1 1 1 2 1 2
3 3 3
## [1185] 3 3 3 3 4 3 3 3 4 3 4 3 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5
4 4 4
## [1222] 3 3 3 2 3 3 3 2 3 2 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 3 4 4 4 3 3 4 4
4 4 4
## [1259] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 4 4 4 4 3 3 2 3 3 3 2 3 2 2 2 2 3
3 2 2
## [1296] 2 4 4 3 4 4 4 4 4 3 3 4 4 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 5
## [1333] 5 4 4 3 3 3 3 3 3 3 2 2 2 2 2 1 3 3 2 3 3 4 4 4 4 4 4 4 4 4 4 4 4 3
3 3 4
## [1370] 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 4 4 4 4 3 4 4 3 3 3 3 3 2
3 3 3
## [1407] 3 3 3 3 4 4 3 4 4 4 4 4 4 4 4 3 3 3 3 3 4 3 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4
## [1444] 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 2 2 2 2 3 3 2 3 3 3 4 3 3 4 4 4 4 3
3 3 2
## [1481] 2 2 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3
3 3 3
## [1518] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4
4 4 4
## [1555] 4 4 4 3 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 2 3 3
3 3 2
## [1592] 2 2 2 2 2 3 3 4 3 3 4 3 3 4 3 4 4 4 3 4 4 3 3 4 3 3 3 3 3 3 3 3 3 3
3 3 3
## [1629] 3 2 3 2 3 3 2 2 2 3 3 3 1 1 2 3 3 3 3 3 2 2 2 2 1 3 3 3 3 4 4 4 4 4
3 3 4
## [1666] 3 3 3 3 3 3 3 3 3 3 4 4 4 3 3 3 3 3 3 3 2 2 3 3 3 2 2 2 3 1 2 3 2 2
```

```
2 2 3
## [1703] 3 3 3 3 2 2 2 3 3 3 4 3 4 4 4 3 4 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3
3 3 3
## [1740] 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 2 2 2 3 3 3 4 4 3
4 3 3
## [1777] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 2 2 2 1 1 2 2 3 3 1 1
2 2 2
## [1814] 2 2 2 2 3 3 3 3 3 2 2 3 3 4 3 4 3 4 4 3 3 2 2 2 2 2 2 2 3 3 3 2 2 3
3 2 2
## [1851] 2 3 3 3 3 3 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 2 4 3 3
3 3 4
## [1888] 3 3 3 3 3 2 2 2 2 2 2 2 2 2 3 2 3 3 2 2 2 3 2 3 3 2 2 2 2 2 2 2 2 2
3 3 2
## [1925] 2 2 3 2 3 3 3 2 3 3 3 3 2 2 3 3 3 2 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
2 2 2
## [1962] 2 2 2 2 2 3 2 3 3 2 2 2 2 2 2 1 3 3 3 3 3 2 3 3 2 3 3 2 2 2 2 3 3 2
4 3 3
## [1999] 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3
2 3 3
## [2036] 3 3 3 3 3 3 2 2 3 2 3 3 3 3 3 3 2 2 2 2 2 2 2 3 2 3 2 2 3 2 2 2 2 2
2 2 2
## [2073] 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 2 2 3 2 3 3 3 3 3 3 3 3 3 2 2 2 3 3
2 2 2
## [2110] 2 2 2 3 2 2 2 3 3 3 2 2 2 3 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2
## [2147] 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2 2 3 2 3 2 3 2 2 3 3 3 2 2 3 3
2 2 2
## [2184] 2 2 2 1 2 2 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3
2 2 2
## [2221] 3 2 3 3 2 2 3 2 3 2 2 2 2 2 3 3 2 2 2 2 2 2 2 1 2 2 2 1 1 1 2 1 2 1
2 2 2
## [2258] 2 2 2 2 3 3 3 3 3 3 3 3 3 3 2 2 2 3 2 2 2 2 2 2 3 3 2 3 3 2 2 2 2 3
3 3 3
## [2295] 2 2 2 2 2 2 2 1 2 2 2 1 1 2 1 2 2 2 1 1 2 2 2 1 2 2 3 2 2 3 2 3 2 3
3 2 2
## [2332] 2 2 2 3 2 2 2 3 2 2 1 1 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2
2 2 2
## [2369] 2 2 1 1 2 2 2 2 2 2 3 2 3 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2
2 2 1
## [2406] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 1 1 2 1
2 2 2
## [2443] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 1 1 1 1
1 1 2
## [2480] 2 2 1 2 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 1
1 1 1
## [2517] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1
2 2 2
## [2554] 2 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 2 2 2 2 2 2 1
1 1 1
## [2591] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 2 1 1 1
1 1 1
```

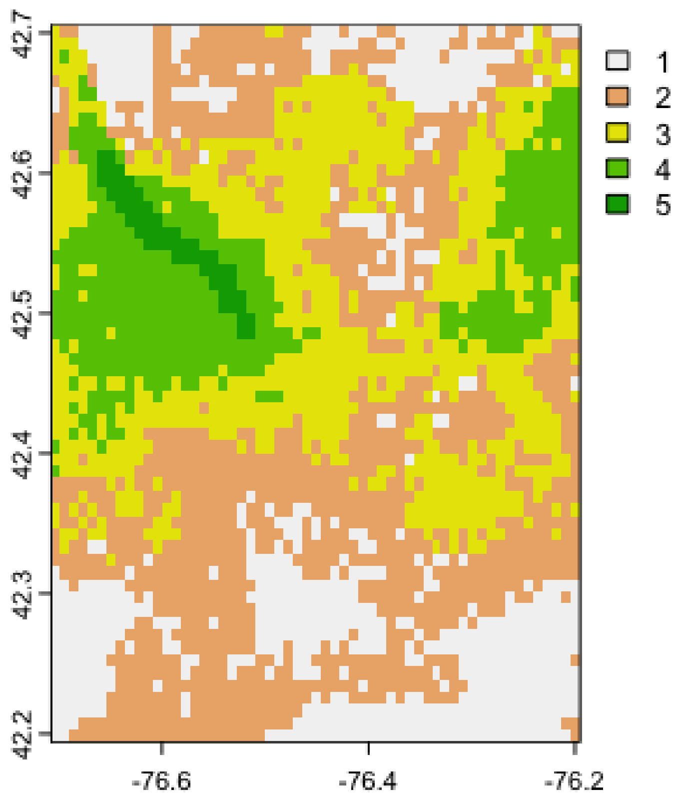
```
## [2628] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2 2
## [2665] 2 2 2 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 2 2 1 2
2 2 2
## [2702] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1
1 1 1
## [2739] 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1
2 2 2
## [2776] 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 1 1 2 2
2 2 2
## [2813] 2 2 2 1 1 1 2 1 1 1 2 1 2 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [2850] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 1 2 2 2 1 1 1 2 2 1 2 1 1 2 2 2 2 2
2 2 2
## [2887] 2 2 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 2 2 2 2
2 1 2
## [2924] 2 2 2 1 2 2 2 1 2 2 1 1 1 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 1 1 1
1 1 1
## [2961] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2
## [2998] 1 2 2 2 2 2 2 2 2 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2 2
## [3035] 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2 2 1 2 1 1 2 2 2 2 1 1 1
1 1 1
## [3072] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1
## [3109] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
2 2 2
## [3146] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [3183] 1 1 1 1 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 1 1
## [3220] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## 
## Within cluster sum of squares by cluster:
## [1] 0.07393101 0.05268721 0.06198373 0.08604367 0.04022348
##  (between_SS / total_SS =  91.4 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
print(k5$centers)
```

```
##         [,1]
## 1 0.23273212
## 2 0.21186044
## 3 0.18682551
## 4 0.15239892
## 5 0.04814491
```

Convert back to a `terra::SpatRaster` and display:

```
k5.rast <- rast(evi.r[[1]])
values(k5.rast) <- k5$cluster
plot(k5.rast)
```



The lowest EVI (see the table of cluster means) are Cayuga Lake; the next-lowest may represent different amounts of snow.

This likely shows areas with more snow as one class 1; Cayuga Lake is class 5.

Use five images from 20-July through 29-August. This period usually has some drought, and is when field crops are ripening but forest continues well-vegetated.

```
# several images
names(evi.r)

##  [1] "X20170101_EVI" "X20170109_EVI" "X20170117_EVI" "X20170125_EVI"
##  [5] "X20170202_EVI" "X20170210_EVI" "X20170218_EVI" "X20170226_EVI"
##  [9] "X20170306_EVI" "X20170314_EVI" "X20170322_EVI" "X20170330_EVI"
## [13] "X20170407_EVI" "X20170415_EVI" "X20170423_EVI" "X20170501_EVI"
## [17] "X20170509_EVI" "X20170517_EVI" "X20170525_EVI" "X20170602_EVI"
## [21] "X20170610_EVI" "X20170618_EVI" "X20170626_EVI" "X20170704_EVI"
## [25] "X20170712_EVI" "X20170720_EVI" "X20170728_EVI" "X20170805_EVI"
## [29] "X20170813_EVI" "X20170821_EVI" "X20170829_EVI" "X20170906_EVI"
```

```
## [33] "X20170914_EVI" "X20170922_EVI" "X20170930_EVI" "X20171008_EVI"
## [37] "X20171016_EVI" "X20171024_EVI" "X20171101_EVI" "X20171109_EVI"
## [41] "X20171117_EVI" "X20171125_EVI" "X20171203_EVI" "X20171211_EVI"
## [45] "X20171219_EVI" "X20171227_EVI"
```

```
evi.r.3 <- evi.r[[26:31]]
summary(evi.r.3)
```

```
##  X20170720_EVI      X20170728_EVI    X20170805_EVI    X20170813_EVI
##  Min.   :-0.07964   Min.   :0.3387   Min.   :0.1435   Min.   :-0.08419
##  1st Qu.: 0.74898   1st Qu.:0.5261   1st Qu.:0.7193   1st Qu.: 0.46810
##  Median : 0.82593   Median :0.5652   Median :0.7944   Median : 0.72585
##  Mean   : 0.78349   Mean   :0.5663   Mean   :0.7741   Mean   : 0.65070
##  3rd Qu.: 0.87934   3rd Qu.:0.6058   3rd Qu.:0.8579   3rd Qu.: 0.84241
##  Max.   : 1.00000   Max.   :0.8399   Max.   :1.0000   Max.   : 1.00000
##  X20170821_EVI    X20170829_EVI
##  Min.   :0.2346   Min.   :0.01794
##  1st Qu.:0.5625   1st Qu.:0.36793
##  Median :0.6686   Median :0.41506
##  Mean   :0.6886   Mean   :0.45309
##  3rd Qu.:0.8359   3rd Qu.:0.49671
##  Max.   :1.0000   Max.   :1.00000
```

```
k5 <- kmeans(as.matrix(evi.r.3), centers = 5, nstart = 5)
print(k5$centers)
```

```
##   X20170720_EVI X20170728_EVI X20170805_EVI X20170813_EVI X20170821_EVI
## 1    0.81643550     0.5883449     0.8346142    0.76161281     0.6068144
## 2   -0.00281735     0.3869958     0.2684122   -0.01026876     0.3860842
## 3    0.80069152     0.5717717     0.7765784    0.39867402     0.7060635
## 4    0.83582899     0.5839544     0.8002237    0.80761701     0.8576777
## 5    0.79124462     0.5583151     0.7844018    0.79262717     0.5763050
##   X20170829_EVI
## 1     0.8011278
## 2     0.2738336
## 3     0.4482133
## 4     0.4249979
## 5     0.4161970
```

```
k5$size
```

```
## [1]  247  101  979  887 1035
```

```
k5$betweenss
```

```
## [1] 333.6228
```

```
k5$tot.withinss
```

```
## [1] 189.4621
```

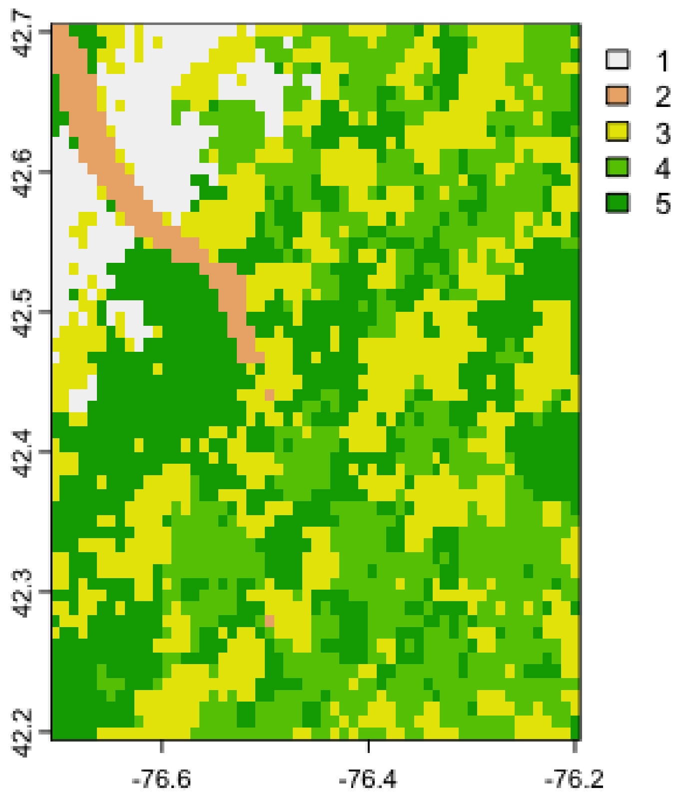
```
k5$betweenss/k5$totss
```

```
## [1] 0.6377986
```

The proportion of variance explained is 0.6378.

Convert back to a `terra::SpatRaster` and display:

```
k5.rast <- rast(evi.r[[1]])
values(k5.rast) <- k5$cluster
plot(k5.rast)
```



Again Cayuga Lake is the cluster with the lowest mean cluster EVI across the six dates.

The point of this section was just to show that it is possible to get GEE objects into R and then work with them locally. GEE is used to find datasets, select and filter them, and do the image processing. R is used for data analysis.

Example: `ggplot2` graphics on GEE objects

A simple chloropleth map

This is from the `rgee` examples. 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:

```r
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.0     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ tidyr::extract() masks terra::extract(), raster::extract()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ✖ dplyr::select()  masks raster::select()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force
all conflicts to become errors
```

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

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

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 on the R-Spatial website.

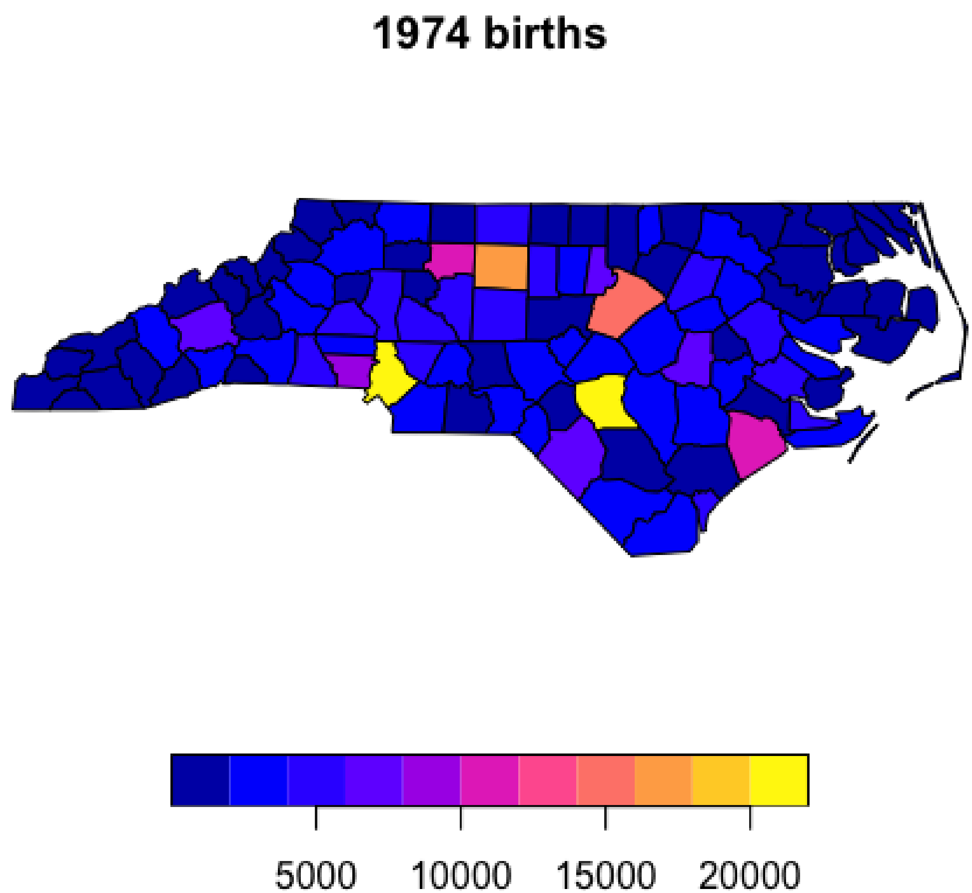
> ... the 100 counties of North Carolina, and includes counts of numbers of live births (also non-white live births) and numbers of sudden infant deaths, for the July 1, 1974 to June 30, 1978 and July 1, 1979 to June 30, 1984 periods.

Plot the number of births in 1974, by county:

```r
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

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




```
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.

Climate analysis

Task: Make a reference to the TerraClimate "Monthly Climate and Climatic Water Balance for Global Terrestrial Surfaces" dataset.

```
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 `.`.

```
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 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`:

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

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

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)

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

## Band 01 Scale:  4638.3121163864

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$ImageCollection``.

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.

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

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

```
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:

```
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.

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

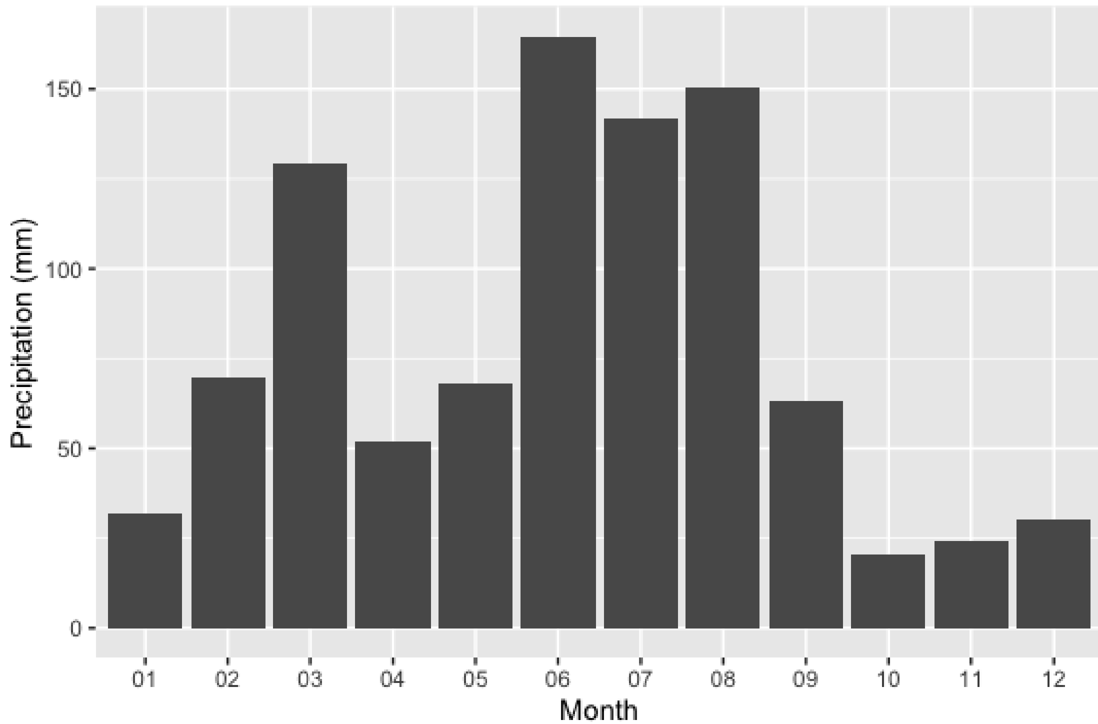
```
## [1] 1200    3
```

```
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"
```

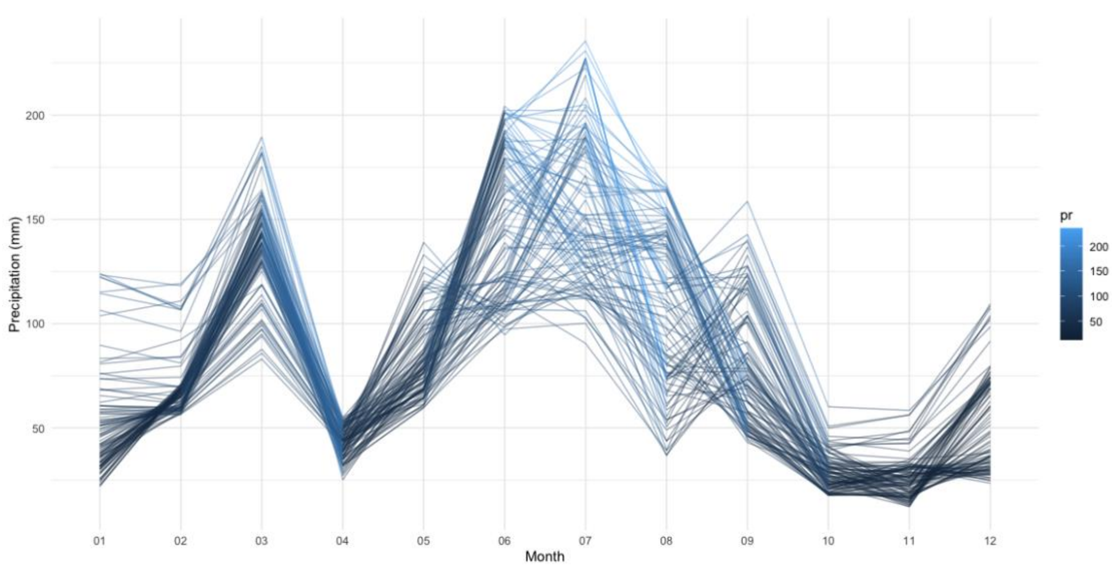
```
ee_nc_rain_long %>%
  filter(NAME=="Edgecombe") %>%
  ggplot() +
    geom_col(aes(x=month, y=pr)) +
    xlab("Month") + ylab("Precipitation (mm)")
```



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:

```
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.

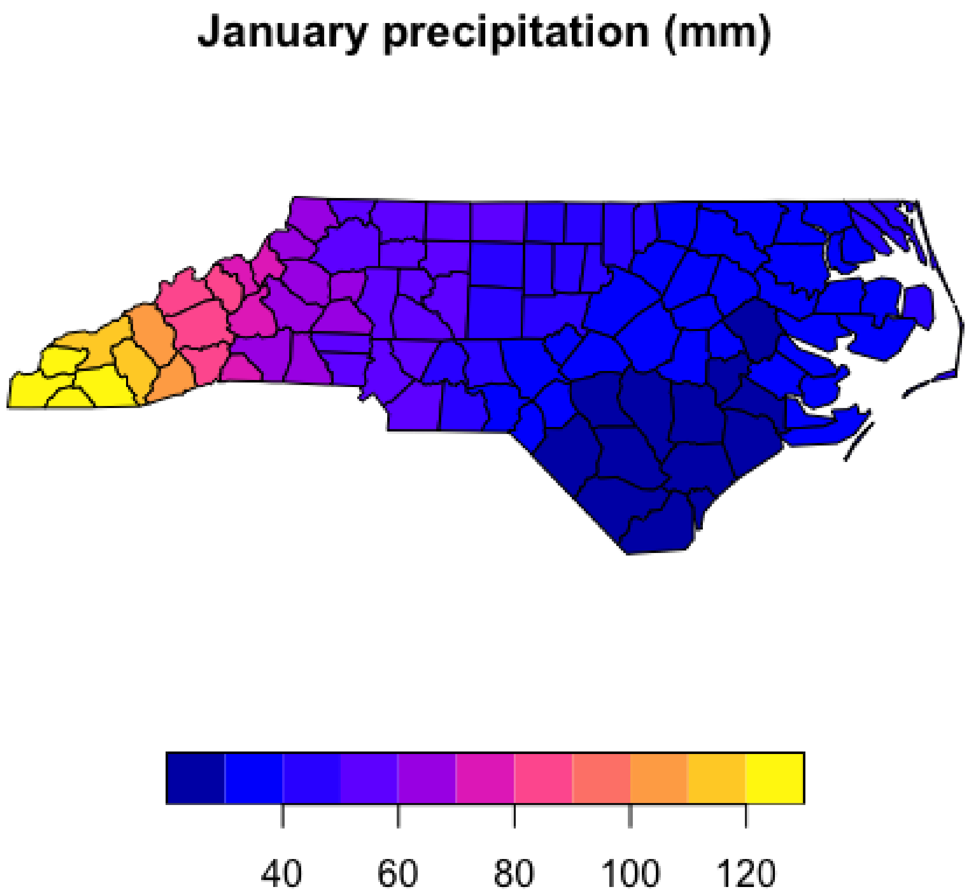
Task Make a chloropleth map of the January precipitation for the counties in the State.

Add the January precipitation to the NC counties geometry:

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

## [1] 100   3

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 NA NA NA ...
##   ..- attr(*, "names")= chr [1:15] "AREA" "PERIMETER" "CNTY_" "CNTY_ID"
...
```

Plot it:

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



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

Resources

- 250+ `rgee` examples: https://csaybar.github.io/rgee-examples/