Preprocessing spatial and non-spatial data

Introduction

This tutorial is designed to guide participants through geo-processing techniques relevant for making raster and tabular data ready for feed balance modelling. Throughout the tutorial, we will introduce participants to a variety of datasets and techniques for handling data.

Learning outcomes

By the end of this course, participants will be able to:

- Import raster and tabular data into R
- Perform several types of spatial analyses in R and make them ready for feed balance modelling

R and RStudio

If you plan to follow along with the R coding during the workshop, please ensure that you have the latest versions of R and RStudio installed on your computer.

First, you will need to download and install from https://cran.r-project.org.

Next you will need to download and install RStudio from https://rstudio.com/products/rstudio/download/#download.

Setting the working directory

We map our working directory to the `Feed-balance-modeling-training` folder we created earlier. We assign the folder the variable name `root`.

For linux:

```
# linux systems
root <- "/home/Feed-balance-modeling-training"
```

For Windows system:

```
# for windows systems
root <- "c:/Documents/Feed-balance-modeling-training"
```

Load R packages

To begin, load the “raster R package along with a couple of others.

```
library(ncdf4)  # package for netcdf manipulation
library(raster)  # package for raster manipulation
library(dplyr)
library(rgdal)
library(sf)
library(tidyterra)
library(ggplot2)
library(terra)
library(readr)
```

We then create outputs folder to store the results.

```
# output folder
outdir <- paste0(root, "/Day_2/SpatialData/inputs/Feed_DrySeason/DMP")
dir.create(outdir, F, T)
```

Dry matter productivity

Next we read in the a layer `gadm40_BFA_0.shp` to define the area of interest and list all `.nc` files downloaded in the previous tutorial.

```
# read AOI
aoi <- read_sf(paste0(root,
 ↪  "/Day_1/SpatialData/inputs/AdminBound/gadm40_BFA_0.shp"))

nc_files <- list.files(paste0(root, "/Day_1/SpatialData/inputs/Feed/DMP"),
    pattern = ".nc$", full.names = TRUE, recursive = TRUE)
nc_files
```

```
[1] "/home/s2255815/rdrive/AU_IBAR/Feed-balance-modeling-training/Day_1/SpatialData/inputs/Fe
[2] "/home/s2255815/rdrive/AU_IBAR/Feed-balance-modeling-training/Day_1/SpatialData/inputs/Fe
[3] "/home/s2255815/rdrive/AU_IBAR/Feed-balance-modeling-training/Day_1/SpatialData/inputs/Fe
```

Use `nc_open` function to read and explore one of the datasets.

```
nc_file <- nc_open(paste0(root,
 ↪  "/Day_1/SpatialData/inputs/Feed/DMP/c_gls_DMP300-RT6_202001100000_GLOBE_OLCI_V1.1.2.nc"))
nc_file
```

```
File /home/s2255815/rdrive/AU_IBAR/Feed-balance-modeling-training/Day_1/SpatialData/inputs/Fe

     3 variables (excluding dimension variables):
        char crs[]   (Contiguous storage)
            long_name: coordinate reference system
            semi_major_axis: 6378137
            spatial_ref: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.25722
            grid_mapping_name: latitude_longitude
            _CoordinateTransformType: Projection
            _CoordinateAxisTypes: GeoX GeoY
            inverse_flattening: 298.257223563
            GeoTransform: -180.0000000000 0.0029761905 0.0 80.0000000000 0.0 -0.0029761905
            longitude_of_prime_meridian: 0
        short DMP[lon,lat,time]   (Chunking: [2283,888,1])  (Compression: shuffle,level 4)
            _FillValue: -1
            grid_mapping: crs
            scale_factor: 0.00999999977648258
            missing_value: -1
            add_offset: 0
            long_name: Dry matter productivity 333M
            units: kg / hectare / day
            valid_range: 0
             valid_range: 32767
            flag_meanings: Missing Sea
            flag_values: -1
             flag_values: -2
        unsigned byte QFLAG[lon,lat,time]   (Chunking: [3270,1272,1])  (Compression: shuffle
            _FillValue: 255
            grid_mapping: crs
            flag_meanings: Fapar_invalid Meteo_invalid EBF Previous_filled Gap_filled
            flag_masks: 1
             flag_masks: 2
             flag_masks: 8
             flag_masks: 64
             flag_masks: 128
            valid_range: 0
```

```
        valid_range: 254
       long_name: Quality flag on DMP 333M
       units:
       missing_value: 255

 3 dimensions:
   lon  Size:120960
       standard_name: longitude
       long_name: longitude
       DIMENSION_LABELS: lon
       units: degrees_east
       _CoordinateAxisType: Lon
       axis: X
   lat  Size:47040
       standard_name: latitude
       long_name: latitude
       DIMENSION_LABELS: lat
       units: degrees_north
       _CoordinateAxisType: Lat
       axis: Y
   time  Size:1   *** is unlimited ***
       units: days since 1970-01-01 00:00:00
       long_name: Time
       calendar: standard
       axis: T

19 global attributes:
   Conventions: CF-1.6
   parent_identifier: urn:cgls:global:dmp300_v1_333m
   platform: Proba-V
   copyright: Copernicus Service information 2020
   time_coverage_end: 2020-01-10T23:59:59Z
   product_version: V1.0.1
   long_name: Dry matter productivity
   time_coverage_start: 2020-01-01T00:00:00Z
   source: Derived from EO satellite imagery
   processing_mode: Consolidated
   references: https://land.copernicus.eu/global/products/dmp
   orbit_type: LEO
   title: 10-daily Dry Matter Productivity 333M: GLOBE 2020-01-10T00:00:00Z
   archive_facility: VITO
   identifier: urn:cgls:global:dmp300_v1_333m:DMP300-RT5_202001100000_GLOBE_PROBAV_V1.0
   sensor: VEGETATION
```

```
        institution: VITO NV
        processing_level: L3
        history: 2020-03-04 Processing line DMP2 333M
```

As shown, the variable is `Dry matter productivity`, provided at a resolution of `333` meters, with units in 'kg/hectare/day.

All done exploring the data. We can close the `netCDF` file.

```
nc_close(nc_file)
```

Luckily, the `raster` R package allows us to read `.nc` files. We will loop through the three files, crop them to the area of interest, and save the new files in the results folder.

```
lapply(nc_files, function(nc_file) {

    nc_name <- gsub(".{3}$", "", basename(nc_file))

    iDMP <- raster::raster(nc_file, varname = "DMP", ncdf = TRUE)
    iDMP <- crop(iDMP, extent(aoi))
    iDMP <- mask(iDMP, aoi)

    # save as GeoTIFF
    raster::writeRaster(iDMP, filename = paste0(outdir, "/",
        nc_name, ".tif"), overwrite = TRUE)
})
```

```
[[1]]
class       : RasterLayer
dimensions  : 3233, 4035, 13045155  (nrow, ncol, ncell)
resolution  : 0.00297619, 0.00297619  (x, y)
extent      : 2.668155, 14.67708, 4.269345, 13.89137  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
source      : c_gls_DMP300-RT6_202001100000_GLOBE_OLCI_V1.1.2.tif
names       : c_gls_DMP300.RT6_202001100000_GLOBE_OLCI_V1.1.2
values      : -0.02, 157.19  (min, max)


[[2]]
class       : RasterLayer
dimensions  : 3233, 4035, 13045155  (nrow, ncol, ncell)
```

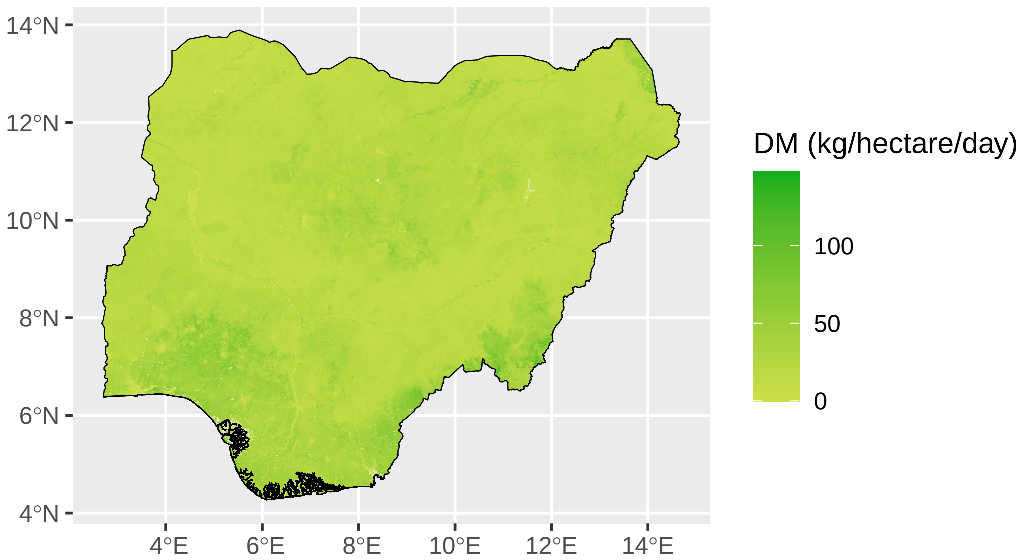
```
resolution : 0.00297619, 0.00297619  (x, y)
extent     : 2.668155, 14.67708, 4.269345, 13.89137  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : c_gls_DMP300-RT6_202001200000_GLOBE_OLCI_V1.1.2.tif
names      : c_gls_DMP300.RT6_202001200000_GLOBE_OLCI_V1.1.2
values     : -0.02, 152.52  (min, max)


[[3]]
class      : RasterLayer
dimensions : 3233, 4035, 13045155  (nrow, ncol, ncell)
resolution : 0.00297619, 0.00297619  (x, y)
extent     : 2.668155, 14.67708, 4.269345, 13.89137  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : c_gls_DMP300-RT6_202001310000_GLOBE_OLCI_V1.1.2.tif
names      : c_gls_DMP300.RT6_202001310000_GLOBE_OLCI_V1.1.2
values     : -0.02, 149.52  (min, max)
```

Here is how the layers look like

```
iDMP <- rast(paste0(root,
 ↪  "/Day_2/SpatialData/inputs/Feed_DrySeason/DMP/c_gls_DMP300-RT6_202001310000_GLOBE_OLCI_V1

ggplot() + geom_sf(data = aoi, colour = "black", show.legend = F) +
    geom_spatraster(data = iDMP) + geom_sf(data = aoi, colour = "black",
    fill = NA, show.legend = F) + scale_fill_gradient(low = "#CDDF4A",
    high = "#0BAE1C", na.value = NA, name = "DM (kg/hectare/day)")
```



Crop type and distribution

Similarly, we crop the `Crop type and distribution` data to the area of interest and save the new files in the results folder.

We create an outputs folder to store the results.

```
# output folder
outdir <- paste0(root, "/Day_2/SpatialData/inputs/SPAM2020")
dir.create(outdir, F, T)
```

Then read the raster properties of the `Dry matter productivity` files to make all other raster files the same in `extent` and `spatial resolution`.

```
dmpTemp <- rast(paste0(root,
↪  "/Day_2/SpatialData/inputs/Feed_DrySeason/DMP/c_gls_DMP300-RT6_202001100000_GLOBE_OLCI_V
```

We list the files downloaded in the previous session, read them in a loop, crop them to the area of interest, and save the processed files in the results folder. According to the `SPAM2020` documentation, there are three technologies available: `irrigated (R)`, `rainfed (R)`, and `all technologies together (A)`. For this tutorial, we will focus on the `A` technology.

```
sPamfiles <- list.files(paste0(root,
↪  "/Day_1/SpatialData/inputs/Feed/CropType/spam2020V0r1_global_physical_area"),
   pattern = "_A.tif$", full.names = TRUE, recursive = TRUE)
```

```r
lapply(sPamfiles, function(sPamfile) {
    sPamfile_name <- tolower(gsub(".{4}$", "", basename(sPamfile)))

    isPamFile <- rast(sPamfile)
    isPamFile <- crop(isPamFile, ext(dmpTemp))
    isPamFile <- resample(isPamFile, dmpTemp, method = "bilinear")
    isPamFile <- mask(isPamFile, mask = dmpTemp)

    isPamFile[is.nan(values(isPamFile))] <- NA

    names(isPamFile) <- sPamfile_name
    varnames(isPamFile) <- sPamfile_name

    # save as GeoTIFF
    writeRaster(isPamFile, filename = paste0(outdir, "/", sPamfile_name,
        ".tif"), overwrite = TRUE)
})
```

```
[[1]]
class       : SpatRaster
dimensions  : 3233, 4035, 1  (nrow, ncol, nlyr)
resolution  : 0.00297619, 0.00297619  (x, y)
extent      : 2.668155, 14.67708, 4.269345, 13.89137  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : spam2020_v0r1_global_a_bean_a.tif
name        : spam2020_v0r1_global_a_bean_a
min value   :                  1.042974e-04
max value   :                  1.483067e+03

[[2]]
class       : SpatRaster
dimensions  : 3233, 4035, 1  (nrow, ncol, nlyr)
resolution  : 0.00297619, 0.00297619  (x, y)
extent      : 2.668155, 14.67708, 4.269345, 13.89137  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : spam2020_v0r1_global_a_maiz_a.tif
name        : spam2020_v0r1_global_a_maiz_a
min value   :                  2.806621e-02
max value   :                  8.220979e+03
```

Deriving Crop residue fraction

We will derive crop residue fraction layers by combining data on Crop type and distribution' with harvest indices obtained from published sources.

Since the next process is memory intensive, we will set specific parameters to optimize performance.

```
rasterOptions(maxmemory = 1e+60)
rasterOptions(todisk = TRUE)
```

Read in crop harvest index data.

```
cropHI <- read_csv(paste0(root, "/Day_2/Tables/crop_harvest index.csv"))
```

Set path to the `SPAM2020` data

```
pathSPAM <- paste0(root, "/Day_2/SpatialData/inputs/SPAM2020")
pathSPAMInter <- paste0(root,
 ↪  "/Day_2/SpatialData/inputs/SPAM2020/intermediate")
dir.create(pathSPAMInter, F, T)
```

List the files clipped in the previous session, and stack the together.

```
filesSPAM <- list.files(path = pathSPAM, pattern = "_a.tif$",
    full.names = T)
stSPAM <- rast(filesSPAM)
```

Extract the names of crops listed above using a `regex` function.

```
crops <- sub(".*_a_(.*?)_a\\.tif$", "\\1", filesSPAM)
```

Calculate total crop area

```
iSPAMcropArea <- app(stSPAM, fun = sum, na.rm = TRUE)
```

Loop through the crop list, and create residue layer considering harvest index values.

```r
for (i in 1:length(crops)) {

    # Extract the relevant SPAM data for the crop
    tmpCropIndex <- grep(pattern = paste(crops[i], collapse = "|"),
        names(stSPAM))
    iSPAMtmpArea <- stSPAM[[tmpCropIndex]]

    # Adjust crop data based on the harvest index
    icrop <- stSPAM[[tmpCropIndex]]
    icrop[icrop > 0] <- (1 - cropHI$harvest_index[cropHI$codeSPAM ==
        crops[i]])

    # Write the adjusted crop data
    writeRaster(icrop, paste0(pathSPAMInter, "/", crops[i], "_res_frac.tif"),
        overwrite = T)

    # Calculate crop proportion and write the output
    stSPAMcropProp <- iSPAMtmpArea/iSPAMcropArea
    writeRaster(icrop, paste0(pathSPAMInter, "/", crops[i], "_prop.tif"),
        overwrite = T)
}
```