Landscape Metrics Extraction

Daniel Maurenza (Conceptualization, Validation, and Writing). Contact - dmaurenza@gmail.com
Marina Schmoeller (Conceptualization, Programming, and Review)

November 1, 2022

Context

We have a database with information on biological communities of 1825 sampling sites, each representing a forest fragment in the Brazilian Atlantic Forest. Our goal is to estimate landscape metrics for landscapes around all sampling sites. In our design, a single landscape is represented by a buffer area surrounding the coordinates of sampling sites. As we intend to detect the best scale of effect of landscape metrics on biological communities, we will extract metrics at five different buffer sizes (500m, 1km, 2km, 4km, and 8 km) for each landscape.

As the surveys were conducted in different years (from 1997 to 2017), we will estimate the metrics from land cover maps corresponding to the median year of data collection, thereby ensuring that information from maps and community data refer to the same time period.

Strategy

We will conduct the following steps:

1. Create a polygon (the landscape area) for each sampling site, for each of five different buffer sizes
2. Download raster maps from the project *Mapbiomas*, collection 7.0. See session "Downloads" for more information on how to download the maps.
3. Randomly select 10 sampling sites from the entire database. To reduce processing time, since this document aims to show the code and not to provide the final result, subsequent procedures will be based on this data subset.
4. Create and run a function for the dataset, and for each buffer size
5. Organize results into a dataframe where rows represent landscapes and columns represent the extracted metrics

Step 1 - Creating landscapes at different sizes

We cannot provide the original data due to copyright limitations. Users may add their own databases, which should be a simple feature (.gpkg) with Coordinate Reference System (CRS) defined as WGS84.

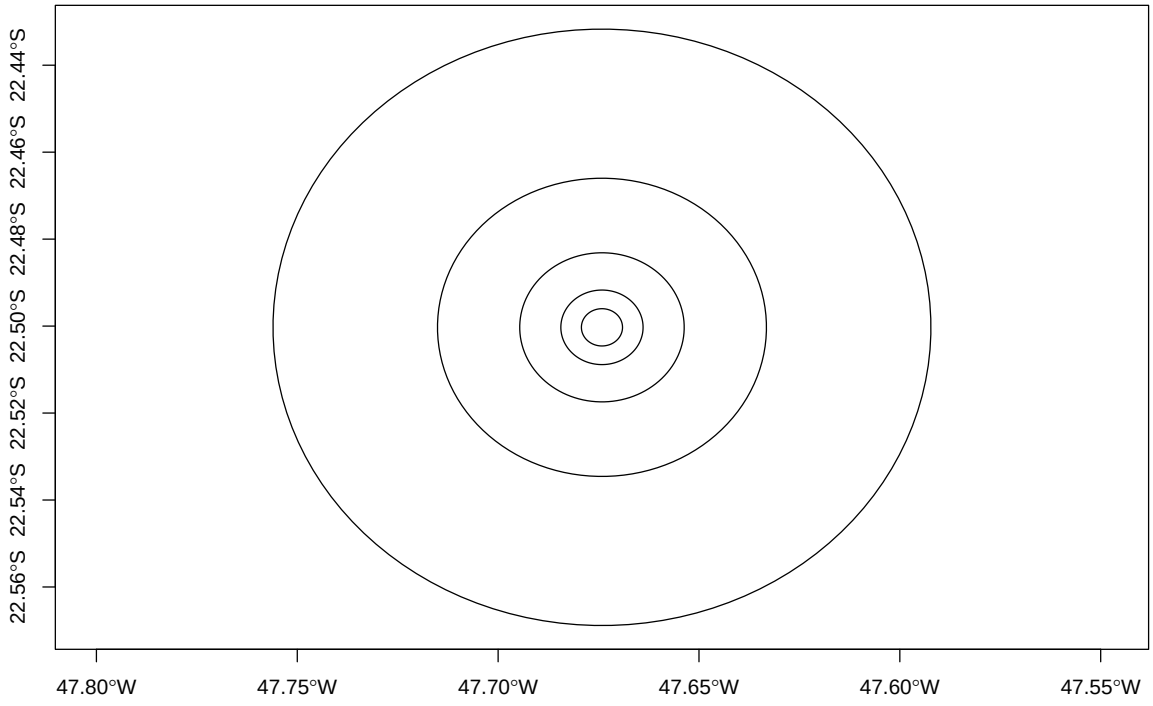
For operations involving distances like buffering, we have to project spatial data to a suitable CRS, with meters as units. For our study area, Albers equal-area conic is applied.

```
# Loading Packages ----
library(sf) # version 1.0-8
library(tidyverse) # version 1.3.2

# Reading datafull
datafull <- st_read("Data/datafull.gpkg")
```

```
Sites <- datafull %>%
  distinct(study_id, id_site, .keep_all = T) %>% # defining each line as a site
  dplyr::select(study_id, id_site, year_median) # Discarding unuseful information minimize the Simple F

# CRS
Albers <- "+proj=aea +lat_1=-5 +lat_2=-42 +lat_0=-32 +lon_0=-60 +x_0=0 +y_0=0 +ellps=aust_SA +units=m +
wgs84 <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs "

# Creating 5 different buffer sizes for sampling sites
buffers8000 <- Sites %>%
  st_transform(crs = Albers) %>%
  st_buffer(dist = 8000) %>%
  st_transform(crs = wgs84)


buffers4000 <- Sites %>%
  st_transform(crs = Albers) %>%
  st_buffer(dist = 4000) %>%
  st_transform(crs = wgs84)

buffers2000 <- Sites %>%
  st_transform(crs = Albers) %>%
  st_buffer(dist = 2000) %>%
  st_transform(crs = wgs84)

buffers1000 <- Sites %>%
  st_transform(crs = Albers) %>%
  st_buffer(dist = 1000) %>%
  st_transform(crs = wgs84)

buffers500 <- Sites %>%
  st_transform(crs = Albers) %>%
  st_buffer(dist = 500) %>%
  st_transform(crs = wgs84)

# Puting the results in a list

buffer.list <- list("500" = buffers500, "1k" = buffers1000, "2k" = buffers2000,
  "4k" = buffers4000, "8k" = buffers8000)
```

Let's plot the five different landscape sizes, for which metrics will be estimated

```
poly_1 <- buffers8000[1, ]
poly_2 <- buffers4000[1, ]
poly_3 <- buffers2000[1, ]
poly_4 <- buffers1000[1, ]
poly_5 <- buffers500[1, ]


polys <- rbind(poly_1, poly_2, poly_3, poly_4, poly_5)
plot(polys$geom, axes = T)
```




```
# Saving buffer sites ----

rm(buffers1000, buffers2000, buffers4000, buffers500, buffers8000)
rm(Sites, datafull)

st_write(buffer.list[["500"]], "Buffer/site.buffers500_wgs84.gpkg", delete_dsn = T)
st_write(buffer.list[["1k"]], "Buffer/site.buffers1k_wgs84.gpkg", delete_dsn = T)
st_write(buffer.list[["2k"]], "Buffer/site.buffers2k_wgs84.gpkg", delete_dsn = T)
st_write(buffer.list[["4k"]], "Buffer/site.buffers4k_wgs84.gpkg", delete_dsn = T)
st_write(buffer.list[["8k"]], "Buffer/site.buffers8k_wgs84.gpkg", delete_dsn = T)
```

Step 2 - Download raster maps

We previously downloaded maps from Project Mapbiomas and stored files in a folder called TIFF. We will list all the rasters and name each one as its year of reference. Since we are interested in extracting information only for the class "forest", we will reclassify the maps, attributing 1 to all forest formations, and 0 to all "non-forest" formation classes. We are using Mapbiomas collection 7.0. Legend for classes can be found *here*.

```
library(terra) # version 1.6-17

# Loading and naming mapbiomas rasters
mapbiomas <- list.files("TIFF/", full.names = T, pattern = "brasil")
mapbiomas.final <- lapply(mapbiomas, rast)
names(mapbiomas.final) <- 1997:2017
```

```
# matrix for reclassification

is <- c(1, 3, 4, 5, 49, 10, 11, 12, 32, 29, 50, 13, 14, 15, 18, 19, 39, 20, 40, 62, 41, 36, 46, 47, 48,
becomes <- c(1, 1, 1, 1, 1, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0,  0, 0, 0, 0, 0,  0, 0, 0, 0, 0,  0, 0, 0, 0,

reclass_matrix <- matrix(c(is, becomes), ncol = 2)
```

Step 3 - Testing for 10 sites

Firstly, we will test the commands for a set of 10 polygons, in order to create a working function that can be applied to the entire database.

Let's take 10 random samples of each buffer size. Note that we have created a new column "id_unique" for each site from the database. This column will also be used to organize the final table in Step 5.

```
# Loading polygons at different buffer sizes

buffer.500 <- st_read("Buffer/site.buffers500_wgs84.gpkg")
buffer.1k <- st_read("Buffer/site.buffers1k_wgs84.gpkg")
buffer.2k <- st_read("Buffer/site.buffers2k_wgs84.gpkg")
buffer.4k <- st_read("Buffer/site.buffers4k_wgs84.gpkg")
buffer.8k <- st_read("Buffer/site.buffers8k_wgs84.gpkg")
buffers.list <- list("0.5k" = buffer.500, "1k" = buffer.1k, "2k" = buffer.2k, "4k" = buffer.4k, "8k" = b
rm(buffer.500, buffer.1k, buffer.2k, buffer.4k, buffer.8k)
gc()

## creating id_unique for buffers. A id_unique represent a sampling site and buffer area

for (i in 1:length(buffers.list)) {
  # i = 1
  buffers.list[[i]] <- buffers.list[[i]] %>%
    mutate(id_unique = 1:length(buffers.list[[i]]$id_site))
}

# Taking the a sample of 10 sites

sample10 <- sample(buffers.list[[1]]$id_unique, size = 10, replace = F)
buffers.list.10 <- list()

for (i in 1:length(buffers.list)) {
  # i = 1
  buffers.list.10[[i]] <- buffers.list[[i]] %>%
    filter(id_unique %in% sample10)
}

names(buffers.list.10) <- names(buffers.list)
```

Now, let's create some lists to be used during the estimates, and a vector containing the metrics that we are interested in.

```
final.8k <- list()
proj8 <- list()
metrics.8k <- list()
metrics <- c("lsm_c_ed", "lsm_c_np", "lsm_c_pland")
```

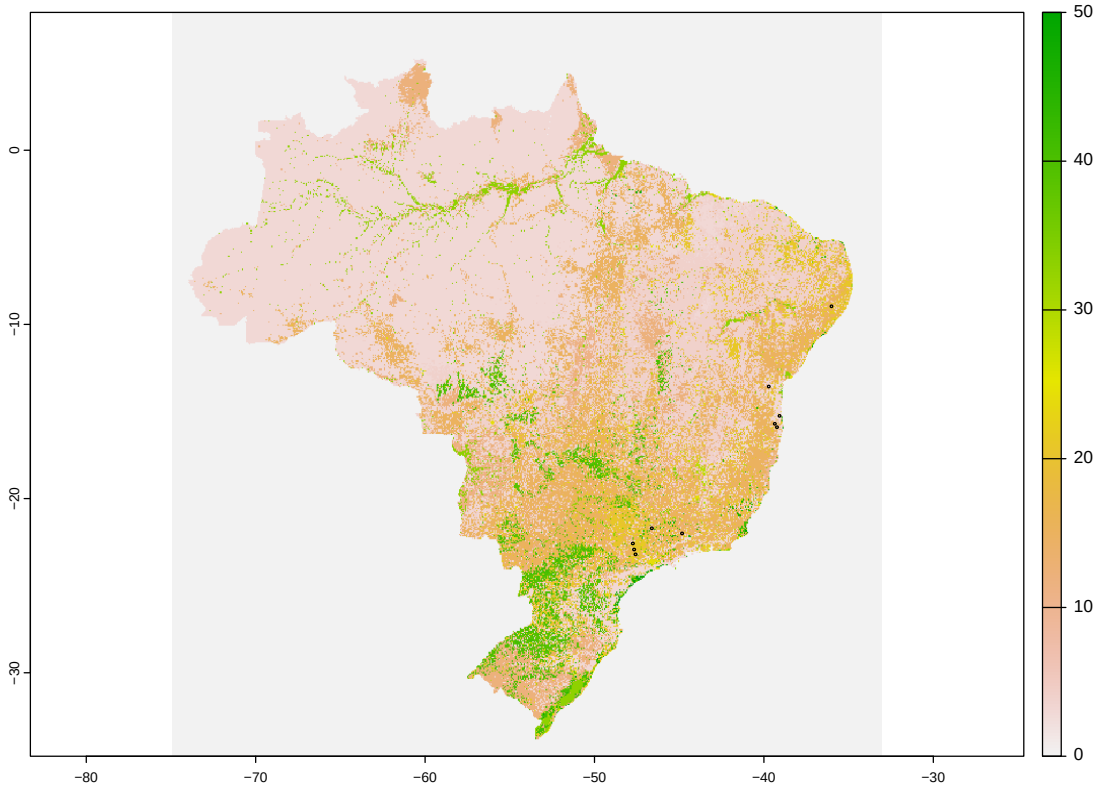
Let's go!

```
library(landscapemetrics) # The package that estimates landscape metrics.  version 1.5.4

for (i in 1:nrow(buffers.list.10[[5]])) {
  # i = 1

  message(i)

  year.i <- (as.data.frame(buffers.list.10[[5]])[i, "year_median"]) # filter the year of raster to be u

  final.8k[[i]] <- terra::crop(x =  mapbiomas.final[[which(names(mapbiomas.final) == year.i)]], y = buf
    terra::mask(mask = buffers.list.10[[5]][i, ]) # crop and mask polygon from the raster

  proj8[[i]] <- final.8k[[i]] %>%
    terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
    terra::classify(rcl = reclass_matrix)


  ### compute landscape metrics (for buffer 8k)
  ### save landscape metrics to a table

  metrics.8k[[i]] <- calculate_lsm(landscape = proj8[[i]], what = metrics)
  names(metrics.8k) <- buffers.list.10[[5]][i, ]$id_unique
}
# End test.
```

OK, it works well.

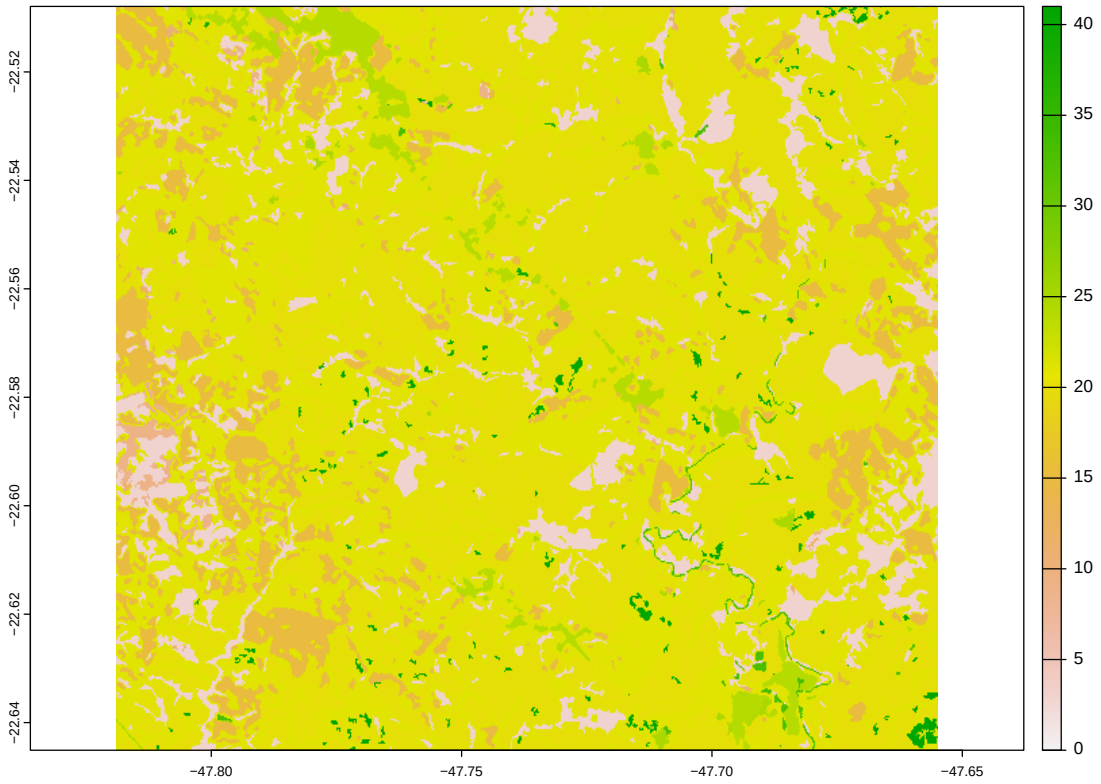
To clarify what we did, let's illustrate this step. We have a map of Brazil and 10 polygons representing our landscapes (in this case using an 8 km buffer size).

```
plot(mapbiomas.final$`1997`)
plot(buffers.list.10[[5]]$geom, add = T)
```

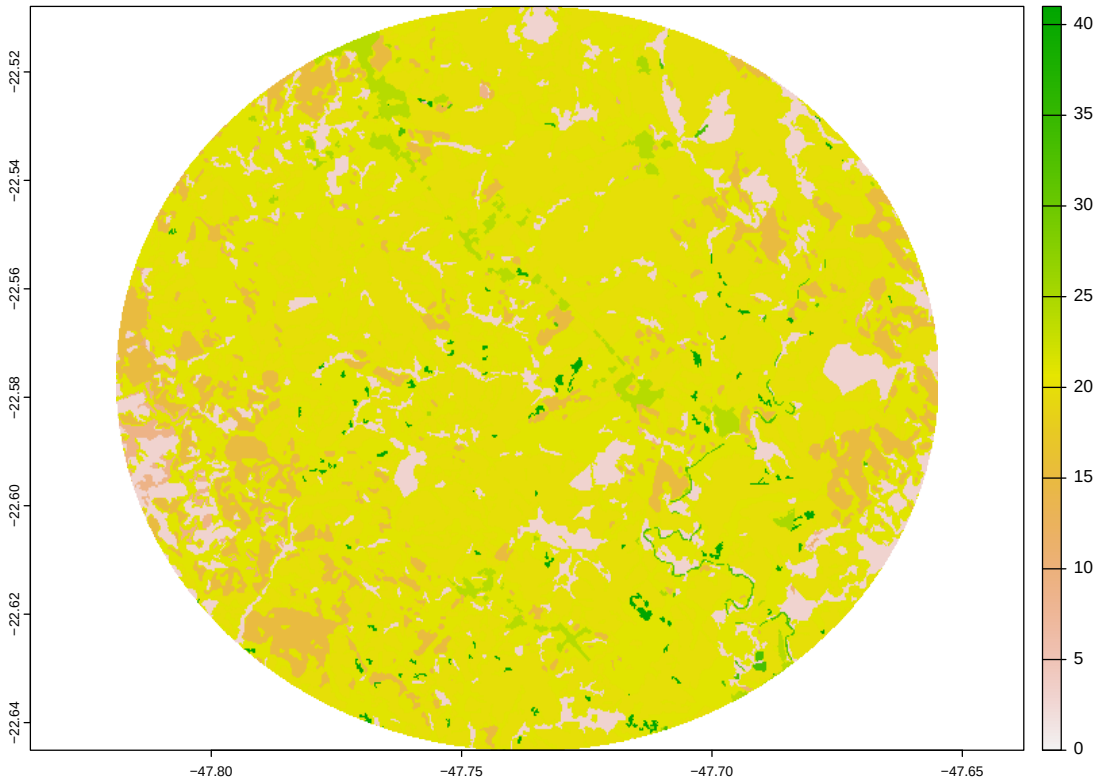


For each landscape, we first select the rectangular area where the polygon intersects with the map of Brazil. Second, we mask the area of the polygon. These two first steps intend to discard unusable information, reducing the amount of data processed and, consequently, memory usage and processing time. Third, we transform the geodata, projecting it to the CRS Albers. Fourth, we reclassify the pixels within the polygon. Then, we estimate the landscape metrics only for the area of interest (the landscape).

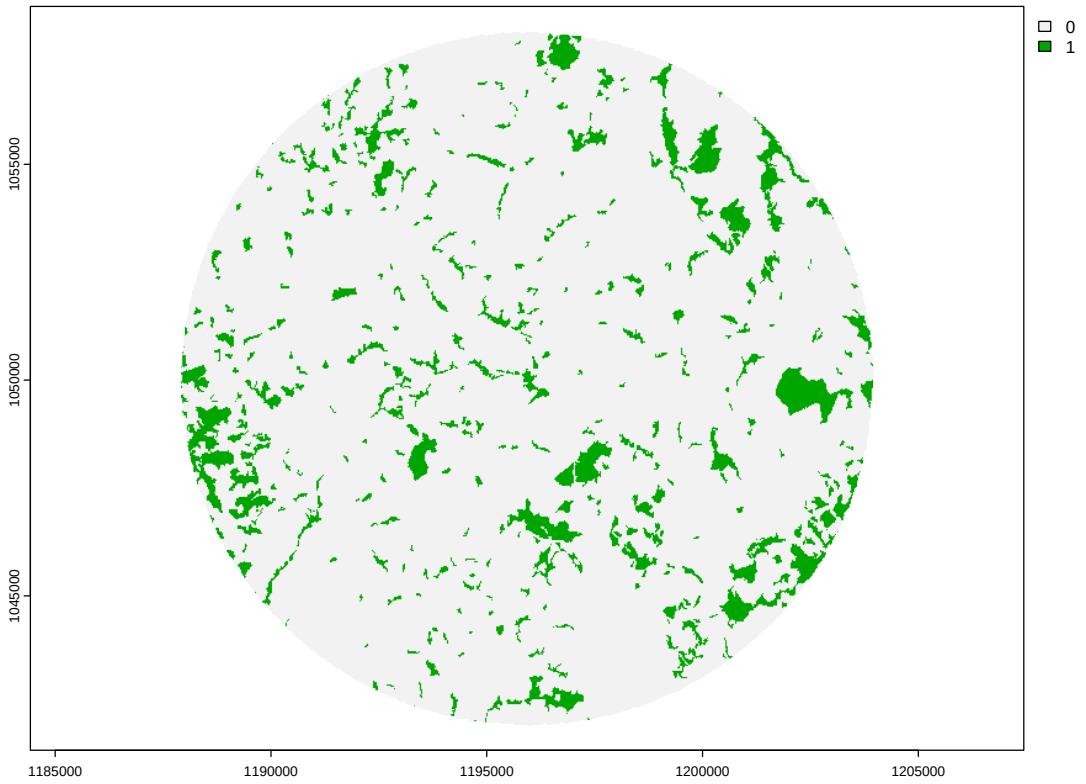
```
crop_poly <- final.8k[[i]] <- terra::crop(x = mapbiomas.final[[which(names(mapbiomas.final) == year.i)]]

mask_poly <-  terra::mask(crop_poly, mask = buffers.list.10[[5]][i, ])

proj_poly <- mask_poly %>%
  terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
  terra::classify(rcl = reclass_matrix)

metric_poly <- calculate_lsm(landscape = proj_poly, what = metrics)

plot(crop_poly)
```




```
plot(mask_poly)
```




```
plot(proj_poly)
```




```
metric_poly
```

```
## # A tibble: 6 x 6
##   layer level class    id metric  value
##   <int> <chr> <int> <int> <chr>   <dbl>
## 1     1 class     0    NA ed      26.4
## 2     1 class     1    NA ed      26.4
## 3     1 class     0    NA np      14
## 4     1 class     1    NA np     410
## 5     1 class     0    NA pland   90.7
## 6     1 class     1    NA pland    9.28
```

Step 4 - Run the function

Once we've tested the commands for a sample of 10 sites across five different buffer sizes, we'll create a function to run the entire database, for all buffer sizes.

The function needs 4 arguments:

buflist - a list of buffers containing all polygons of all buffer sizes

rclmatrix - a reclassification matrix for land cover classes (in this case, from Mapbiomas)

rstack - a list with the raster layers for all years

metrics - a vector containing the desired landscape metrics (lsm) functions

Raster layers are usually memory-heavy files, so a good practice is to eliminate unused information first. As we are extracting estimates from the same point but from different buffer sizes, we will first extract

information from the landscapes with the largest buffer size and downscale from there, extracting for the second largest, third largest, and so on. That is, we will use 8k results for 4k estimates, 4k for 2k estimates, and so on.

Let's create the function!

```
extract_lsm <- function(buflist, rclmatrix, rstack, metrics) {
  # list of buffers
  final.8k <- list()
  final.4k <- list()
  final.2k <- list()
  final.1k <- list()
  final.500 <- list()

  # list of results projected in albers
  proj8 <- list()
  proj4 <- list()
  proj2 <- list()
  proj1 <- list()
  proj05 <- list()

  # List of metrics
  metrics.8k <- list()
  metrics.4k <- list()
  metrics.2k <- list()
  metrics.1k <- list()
  metrics.500 <- list()

  # The body of the function

  for (i in 1:nrow(buflist[[5]])) {
    # i = 1

    message(i)

    year.i <- (as.data.frame(buflist[[5]])[i, "year_median"]) # filter the year of raster to be used

    final.8k[[i]] <- terra::crop(x =  mapbiomas.final[[which(names(mapbiomas.final) == year.i)]], y = bu
      terra::mask(mask = buflist[[5]][i, ]) # crop and mask polygon from the raster

    proj8[[i]] <- final.8k[[i]] %>%
      terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
      terra::classify(rcl = reclass_matrix)

    ### compute landscape metrics (for buffer 8k)
    ### save landscape metrics to a table

    metrics.8k[[i]] <- calculate_lsm(landscape = proj8[[i]], what = metrics)
    metrics.8k[[i]] <- metrics.8k[[i]] %>%
      mutate(id_unique = buflist[[1]][i, ]$id_unique)

    names(metrics.8k) <- buflist[[5]][i, ]$id_unique

    ## crop and mask object created from buffer 8k to buffer 4k (attribute to an object)
    final.4k[[i]] <- terra::crop(x = final.8k[[i]], y = buflist[[4]][i, ]) %>%
```

```
  terra::mask(mask = buflist[[4]][i, ])

proj4[[i]] <- final.4k[[i]] %>%
  terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
  terra::classify(rcl = reclass_matrix)

### compute landscape metrics (for buffer 4k)
### save landscape metrics to a table
metrics.4k[[i]] <- calculate_lsm(landscape = proj4[[i]], what = metrics)
metrics.4k[[i]] <- metrics.4k[[i]] %>%
  mutate(id_unique = buflist[[1]][i, ]$id_unique)

names(metrics.4k) <- buflist[[4]][i, ]$id_unique

## crop and mask object created from buffer 4k to buffer 2k (attribute to an object)
final.2k[[i]] <- terra::crop(x = final.4k[[i]], y = buflist[[3]][i, ]) %>%
  terra::mask(mask = buflist[[3]][i, ])

proj2[[i]] <- final.2k[[i]] %>%
  terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
  terra::classify(rcl = reclass_matrix)

### compute landscape metrics (for buffer 4k)
### save landscape metrics to a table
metrics.2k[[i]] <- calculate_lsm(landscape = proj2[[i]], what = metrics)
metrics.2k[[i]] <- metrics.2k[[i]] %>%
  mutate(id_unique = buflist[[1]][i, ]$id_unique)

names(metrics.2k) <- buflist[[3]][i, ]$id_unique

## crop and mask object created from buffer 2k to buffer 1k (attribute to an object)

final.1k[[i]] <- terra::crop(x = final.2k[[i]], y = buflist[[2]][i, ]) %>%
  terra::mask(mask = buflist[[2]][i, ])

proj1[[i]] <- final.1k[[i]] %>%
  terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
  terra::classify(rcl = reclass_matrix)

### compute landscape metrics (for buffer 4k)
### save landscape metrics to a table
metrics.1k[[i]] <- calculate_lsm(landscape = proj1[[i]], what = metrics)
metrics.1k[[i]] <- metrics.1k[[i]] %>%
  mutate(id_unique = buflist[[1]][i, ]$id_unique)
names(metrics.1k) <- buflist[[2]][i, ]$id_unique

## crop and mask object created from buffer 1k to buffer 0.5k (attribute to an object)

final.500[[i]] <- terra::crop(x = final.1k[[i]], y = buflist[[1]][i, ]) %>%
  terra::mask(mask = buflist[[1]][i, ])

proj05[[i]] <- final.500[[i]] %>%
  terra::project(Albers, method = "near") %>% ## project to albers, and reclassify
```

```
      terra::classify(rcl = reclass_matrix)

    ### compute landscape metrics (for buffer 4k)
    ### save landscape metrics to a table
    metrics.500[[i]] <- calculate_lsm(landscape = proj05[[i]], what = metrics)
    metrics.500[[i]] <- metrics.500[[i]] %>%
      mutate(id_unique = buflist[[1]][i, ]$id_unique)
    names(metrics.500) <- buflist[[1]][i, ]$id_unique
    gc() # to clean our memory
  }

  metrics.list <- list(metrics.500, metrics.1k, metrics.2k, metrics.4k, metrics.8k)
  names(metrics.list) <- names(buflist)


  return(metrics.list)
}
```

OK, the function has been created. Finally, let's extract the estimates for the entire database. Depending on the size of the database and on the computational power, this step can take a few hours.

```
sites_lsm <- extract_lsm(buflist = buffers.list.10, rclmatrix = reclass_matrix, rstack = mapbiomas.final
sites_lsm[[1]][[1]]
```

```
## # A tibble: 6 x 7
##   layer level class    id metric value id_unique
##   <int> <chr> <int> <int> <chr>  <dbl>     <int>
## 1     1 class     0    NA ed      37.6        40
## 2     1 class     1    NA ed      37.6        40
## 3     1 class     0    NA np       1          40
## 4     1 class     1    NA np       1          40
## 5     1 class     0    NA pland   49.1        40
## 6     1 class     1    NA pland   50.9        40
```

OK, works well.

Step 5 - Organizing the final table

We now have a list for each buffer size, where each landscape has n estimated metrics. For a better usability of this information, we will order all metrics in columns and landscapes (sampling points) in rows.

If you check at the end of the function, we include the column "id_unique" in the final result. This information will be used to organize the final table

```
### fix tables --------------
lsm05 <- sites_lsm[["0.5k"]]
lsm1k <- sites_lsm[["1k"]]
lsm2k <- sites_lsm[["2k"]]
lsm4k <- sites_lsm[["4k"]]
lsm8k <- sites_lsm[["8k"]]

library(data.table)
## join all data in a single table
lsm05_bind <- rbindlist(lsm05)
lsm1k_bind <- rbindlist(lsm1k)
lsm2k_bind <- rbindlist(lsm2k)
```

```
lsm4k_bind <- rbindlist(lsm4k)
lsm8k_bind <- rbindlist(lsm8k)

rm(lsm05, lsm1k, lsm2k, lsm4k, lsm8k)

lsm05 <- spread(lsm05_bind, key = metric, value = value)
lsm1k <- spread(lsm1k_bind, key = metric, value = value)
lsm2k <- spread(lsm2k_bind, key = metric, value = value)
lsm4k <- spread(lsm4k_bind, key = metric, value = value)
lsm8k <- spread(lsm8k_bind, key = metric, value = value)

lsm05 <- lsm05 %>%
  select(id_unique, class, ed, np, pland)
lsm1k <- lsm1k %>%
  select(id_unique, class, ed, np, pland)
lsm2k <- lsm2k %>%
  select(id_unique, class, ed, np, pland)
lsm4k <- lsm4k %>%
  select(id_unique, class, ed, np, pland)
lsm8k <- lsm8k %>%
  select(id_unique, class, ed, np, pland)

colnames(lsm05)[3:5] <- paste0("0.5k_", colnames(lsm05)[3:5])
colnames(lsm1k)[3:5] <- paste0("1k_", colnames(lsm1k)[3:5])
colnames(lsm2k)[3:5] <- paste0("2k_", colnames(lsm2k)[3:5])
colnames(lsm4k)[3:5] <- paste0("4k_", colnames(lsm4k)[3:5])
colnames(lsm8k)[3:5] <- paste0("8k_", colnames(lsm8k)[3:5])

final_table <- full_join(lsm05, lsm1k) %>%
  full_join(lsm2k) %>%
  full_join(lsm4k) %>%
  full_join(lsm8k)

head(final_table)
```

```
##    id_unique class  0.5k_ed 0.5k_np 0.5k_pland    1k_ed 1k_np  1k_pland
## 1:        40     0 37.58369       1   49.11765 18.08483     3 70.308968
## 2:        97     0 63.74329       5   60.74583 60.08462     5 70.446474
## 3:       425     0 30.26380       1   48.86831 36.58666     2 42.479514
## 4:       602     0 50.81991       1   15.97510 29.36250     4  8.562283
## 5:       710     0 36.30395       1   85.36585 44.50952     4 78.985689
## 6:       851     0 44.75483       2   61.91983 25.74670     2 80.526173
##       2k_ed 2k_np 2k_pland    4k_ed 4k_np 4k_pland    8k_ed 8k_np 8k_pland
## 1: 18.10449     3 83.23898 18.63868     3 91.51686 28.07195    11 90.11330
## 2: 49.36179     3 76.90629 50.01090    14 80.44034 49.66259    67 80.06378
## 3: 39.94344    12 34.63366 59.44276    51 49.16010 57.23130   132 55.22355
## 4: 44.54571    33 19.36205 50.35600   134 20.47015 59.07510   388 29.30593
## 5: 45.74170     3 80.94362 38.34965     4 86.44601 30.06849     8 90.67276
## 6: 23.74194     2 84.93160 28.25581    16 76.16262 22.06572    31 83.83123
```

```
write_csv(final_table, "Result/lsm.csv")
```

We then have a table where each row represents a landscape of a class (forest or non-forest) and the columns represent landscape metrics for different buffer sizes.