Landscape metrics are algorithms that quantify physical characteristics of landscape mosaics (aka categorical raster) in order to connect them to some ecological processes.  
Many different landscape metrics exist and they can provide three main levels of information: (i) landscape level, (ii) class level, and (iii) patch level.  
A landscape level metric gives just one value describing a certain property of a landscape mosaic, such as its diversity.  
A class level metric returns one value for each class (category) present in a landscape. A patch level metric separates neighboring cells belonging to the same class, and calculates a value for each patch.[1](https://nowosad.github.io/post/lsm-bp1/#fn:For-a-more-detai)

Landscape metrics are often applied to better understand a landscape structure around some sampling points.  
In this post, I show how to prepare input data and how to calculate selected landscape metrics based on different buffer sizes and shapes.  
All calculations were performed in R, using [**gdalUtils**](https://cran.r-project.org/web/packages/gdalUtils/index.html), [**sf**](https://github.com/r-spatial/sf), and [**raster**](https://github.com/rspatial/raster) packages for spatial data handling, [**purrr**](https://github.com/tidyverse/purrr) to ease some calculations, and most importantly the [**landscapemetrics**](https://nowosad.github.io/post/lsm-bp1/) package was used to calculate landscape metrics.

**Data preparation**

Every software for calculations of landscape metrics, such as FRAGSTATS or **landscapemetrics**, expects the input data in a projected coordinate reference system instead of a geographic coordinate reference system.  
This is due to a fact that geographic coordinate reference systems are expressed in degrees, and one degree on the equator has a different length in meters than one degree on a middle latitude.  
Projected coordinate reference systems, on the other hand, have a linear unit of measurement such as meters. This allows for proper calculations of distances or areas.[2](https://nowosad.github.io/post/lsm-bp1/#fn:More-information)

In this case, the input datasets (cci\_lc\_australia.tif and sample\_points.csv) are in a geographic coordinate reference system called WGS84.  
Therefore, it is important to reproject the data before calculating any landscape metrics. Reprojection of raster objects is possible with an R package called **gdalUtils**:

library(gdalUtils)

dir.create("data")

# downloading the land cover data

download.file("https://github.com/Nowosad/lsm-bp1/raw/master/data/cci\_lc\_australia.zip",

destfile = "data/cci\_lc\_australia.zip")

# unziping the file

unzip("data/cci\_lc\_australia.zip", "data/cci\_lc\_australia.tif")

# reprojecting the raster data to a local projected coordinate reference system

# read more at http://spatialreference.org/ref/epsg/gda94-australian-albers/

gdalwarp(srcfile = "data/cci\_lc\_australia.tif",

dstfile = "data/cci\_lc\_australia\_aea.vrt",

of = "VRT",

t\_srs = "+proj=aea +lat\_1=-18 +lat\_2=-36 +lat\_0=0 +lon\_0=132 +x\_0=0 +y\_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no\_defs")

# compressing the output file

gdal\_translate(src\_dataset = "data/cci\_lc\_australia\_aea.vrt",

dst\_dataset = "data/cci\_lc\_australia\_aea.tif",

co = "COMPRESS=LZW")

Reading and reprojecting vector (points) objects can be done with the **sf** package:

library(sf)

# downloading the points

download.file("https://raw.githubusercontent.com/Nowosad/lsm-bp1/master/data/sample\_points.csv",

destfile = "data/sample\_points.csv")

# reading the point data and reprojecting it

points = read.csv("data/sample\_points.csv") %>%

st\_as\_sf(coords = c("longitude", "latitude")) %>%

st\_set\_crs(4326) %>%

st\_transform("+proj=aea +lat\_1=-18 +lat\_2=-36 +lat\_0=0 +lon\_0=132 +x\_0=0 +y\_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no\_defs ")

# saving the new object to a spatial data format GPKG

st\_write(points, "data/sample\_points.gpkg")

Location of three sample points on a land cover map of Australia. Source of the input map:

**Calculations of landscape metrics in a buffer**

Now, we can read the new files – the new raster with the raster() function and the new points file with the st\_read() function:

# attaching the packages used in the next examples

library(raster)

library(sf)

library(purrr)

library(landscapemetrics)

# reading the raster and vector datasets

lc = raster("data/cci\_lc\_australia\_aea.tif")

sample\_points = st\_read("data/sample\_points.gpkg", quiet = TRUE)

The **landscapemetrics** package has a function designed for calculations of landscape metrics for a given buffer.  
It is called sample\_lsm() and it expects, at least three input arguments – a raster, a vector (points), and a buffer size. As a default, it calculates all of the available metrics in a square buffer, where buffer size is the side-length in map units (e.g. meters).

square\_all = sample\_lsm(lc,

points = sample\_points,

size = 3000)

square\_all

## # A tibble: 1,282 x 8

## layer level class id metric value plot\_id percentage\_inside

##

## 1 1 class 11 NA ai 92.1 1 109.

## 2 1 class 100 NA ai 100 1 109.

## 3 1 class 122 NA ai NA 1 109.

## 4 1 class 130 NA ai 50 1 109.

## 5 1 class 150 NA ai 71.1 1 109.

## 6 1 class 11 NA area\_cv NA 1 109.

## 7 1 class 100 NA area\_cv NA 1 109.

## 8 1 class 122 NA area\_cv NA 1 109.

## 9 1 class 130 NA area\_cv 33.3 1 109.

## 10 1 class 150 NA area\_cv 119. 1 109.

## # … with 1,272 more rows

The output, square\_all, is a data frame containing values of all of the landscape metrics available in the **landscapemetrics** package.  
The structure of the output is stable and contains the following columns:

1. layer – id of the input raster (landscape). It is possible to calculate landscape metrics on several rasters at the same time.
2. level – a landscape metrics level (“landscape”, “class”, or “patch”).
3. class – a number of a class for class and patch level metrics.
4. id – a number of a patch for patch level metrics.
5. metric – a metric name.
6. value – an output value of the metric.
7. plot\_id – id of the input vector (point).
8. percentage\_inside – the size of the actual sampled landscape in %.  
   It is not always 100% due to two reasons: (i) clipping raster cells using a buffer not directly intersecting a raster cell center lead to inaccuracies and (ii) sample plots can exceed the landscape boundary.

Square buffers with the side-length of 3000 meters around the three sample points.

The sample\_lsm() function also allows for selecting metrics using a level name ("landscape", "class", or "patch"), a metric abbreviation, full name of a metric, or a metric type.  
For example, SHDI (Shannon’s diversity index) on a landscape level ("lsm\_l\_shdi") can be calculated using the code below.

square\_shdi = sample\_lsm(lc,

points = sample\_points,

size = 3000,

what = "lsm\_l\_shdi")

square\_shdi

## # A tibble: 3 x 8

## layer level class id metric value plot\_id percentage\_inside

##

## 1 1 landscape NA NA shdi 0.870 1 109.

## 2 1 landscape NA NA shdi 1.49 2 98.9

## 3 1 landscape NA NA shdi 0.665 3 98.9

The result shows that the area around the second sample point is the most diverse (SHDI value of 1.49), and the area around the third sample point is the least diverse (SHDI value of 0.665).

As the default, sample\_lsm() uses a square buffer, but it could be changed to a circle one with the shape argument set to "circle".  
In this case, the size argument represents the radius in map units.

circle\_shdi = sample\_lsm(lc,

points = sample\_points,

size = 3000,

what = "lsm\_l\_shdi",

shape = "circle")

circle\_shdi

## # A tibble: 3 x 8

## layer level class id metric value plot\_id percentage\_inside

##

## 1 1 landscape NA NA shdi 1.03 1 99.6

## 2 1 landscape NA NA shdi 1.65 2 100.

## 3 1 landscape NA NA shdi 0.689 3 99.9

Circle buffers with the radius of 3000 meters around the three sample points.

**Calculations of landscape metrics for many buffers**

Finally, it is also possible to calculate landscape metrics on many different buffer sizes at the same time.  
You just need to select what sizes you are interested in and use the sample\_lsm() function inside a map\_dfr() function.

# creating buffers of 3000 and 6000 meters

sizes = c(3000, 6000)

# calculating shdi (landscape metric) for each buffer

two\_sizes\_output = sizes %>%

set\_names() %>%

map\_dfr(~sample\_lsm(lc, points = sample\_points, what = "lsm\_l\_shdi", size = .),

.id = "buffer")

It calculates all of the selected metrics for all of the buffers.

Square buffers with the side-length of 3000 and 6000 meters around the three sample points.

The result is a data frame, where the first column describes the buffer size, and the rest is a standard **landscapemetrics** output.

two\_sizes\_output

## # A tibble: 6 x 9

## buffer layer level class id metric value plot\_id percentage\_inside

##

## 1 3000 1 landscape NA NA shdi 0.870 1 109.

## 2 3000 1 landscape NA NA shdi 1.49 2 98.9

## 3 3000 1 landscape NA NA shdi 0.665 3 98.9

## 4 6000 1 landscape NA NA shdi 1.02 1 99.1

## 5 6000 1 landscape NA NA shdi 1.62 2 99.1

## 6 6000 1 landscape NA NA shdi 0.731 3 99.1

**Summary**

This post gave a few examples of how to calculate selected landscape metrics for buffers around sampling points using the **landscapemetrics** package and its function sample\_lsm().  
In the same time, the **landscapemetrics** package has much more to offer, including [a lot of implemented landscape metrics](https://r-spatialecology.github.io/landscapemetrics/reference/index.html), a number of utility functions (e.g. extract\_lsm() or get\_patches()), and several C++ functions designed for the development of new metrics.

The **landscapemetrics** is also an open-source collaborative project.  
It enables [community contributions](https://r-spatialecology.github.io/landscapemetrics/CONTRIBUTING.html), including documentation improvements, bug reports, and ideas to add new metrics and functions.  
To learn more you can visit the package website at <https://r-spatialecology.github.io/landscapemetrics/>.