The `landscapemetrics` and `motif` Packages for Measuring Landscape Patterns and Processes

Jakub Nowosad and Maximilian H. K. Hesselbarth

Abstract

This book chapter emphasizes the significance of categorical raster data in ecological studies, specifically land use or land cover (LULC) data, and highlights the pivotal role of landscape metrics and pattern-based spatial analysis in comprehending environmental patterns and their dynamics. It explores the usage of R packages, particularly **landscapemetrics** and **motif**, for quantifying and analyzing landscape patterns using LULC data from three distinct European regions. It showcases the computation, visualization, and comparison of landscape metrics, while also addressing additional features such as patch value extraction, subregion sampling, and moving window computation. Furthermore, the chapter delves into the intricacies of pattern-based spatial analysis, explaining how spatial signatures are computed and how the **motif** package facilitates comparisons and clustering of landscape patterns. The chapter concludes by discussing the potential of customization and expansion of the presented tools.

Keywords

Categorical raster data · Land use · Land cover · Landscape metrics · Pattern-based spatial analysis · Spatial ecology

J. Nowosad (✉)
Institute of Geoecology and Geoinformation, Adam Mickiewicz University, Poznań, Poland

M. H. K. Hesselbarth
Statistics Austria, Directorate Spatial Statistics, Energy and Environment, Vienna, Austria

International Institute for Applied Systems Analysis, Biodiversity, Ecology, and Conservation Group, Laxenburg, Austria

D. Rocchini (ed.), *R Coding for Ecology*, Use R!,
https://doi.org/10.1007/978-3-031-99665-8_10

1 Introduction

Categorical raster data, such as ones representing land use or land cover (LULC), is often used in ecological studies (Fassnacht et al., 2006; Wulder et al., 2018; Chandra Pandey et al., 2021). This data is typically derived using remote sensing data, for example, satellite images in combination with statistical learning (Talukdar et al., 2020; Wang et al., 2022; Wang and Mountrakis, 2023). LULC describes the spatial distribution of different anthropogenic uses of land or the natural land cover in a landscape. Thus, it provides information about human activities (usage) and natural features (physical material at the surface) in a given area within a specific time frame (Fisher et al., 2005). This information can then be used to better understand the processes and changes taking place in the landscape, such as urbanization (Fu and Weng, 2016), deforestation (Floreano and De Moraes, 2021), or the spread of invasive species (Manzoor et al., 2021).

From a computational perspective, LULC data can be conceptualized as a collection of cells organized in a regular grid, with each cell assigned to a specific category (With, 2019). This framework enables us to analyze the data in terms of its two fundamental components: the composition, referring to the number of cells for each category, and the configuration, pertaining to the spatial arrangement of cells within each category (Riitters, 2019). These components jointly create spatial patterns of categorical raster data, or landscape patterns, in short, which are commonly used to characterize the structure of landscapes. Furthermore, additional characteristics of a categorical raster, for example, the diversity of categories, connectivity, or patch shape, can also be considered (Gustafson, 1998; Uuemaa et al., 2013).

The aim of this chapter is to provide an overview of the methods and tools that allow to quantify and analyze landscape patterns. We will focus on the methods that are mainly implemented in two R packages: **landscapemetrics** (Hesselbarth et al., 2019) and **motif** (Nowosad, 2021). The **landscapemetrics** package allows to compute a collection of landscape metrics, which quantify the composition and configuration of categorical raster data. The package also provides a set of additional functions, making it possible to visualize the results of the calculations, extract values of patches, sample metrics within subregions, or apply a moving window computation. Thus, its main purpose is to provide a set of tools to describe landscape patterns. The **motif** package, on the other hand, is focused on the analysis of these patterns. It allows to compare landscape patterns between different times, search for areas with resembling landscape patterns, or cluster areas with similar landscape patterns.

In this chapter, we will consistently utilize a set of example data. These data consist of spatial raster representing the LULC of three distinct regions in Europe: the Centre-Val de Loire region in France, the Noord-Brabant region in the Netherlands, and the Norra Mellansverige region in Sweden. We obtained the data from the Copernicus mission 2018 (European Environment Agency, 2023), available at https://land.copernicus.eu/), and cropped them to the respective regions.

Additionally, all original 45 LULC classes were reclassified into five general classes (i.e., "urban," "agriculture," "vegetation," "marshes," and "water") to simplify all examples. To reproduce the results presented in this chapter, you can download the corresponding data and code from the GitHub repository at https://github.com/Nowosad/landscapemetrics_motif_2024.

1.1 Landscape Metrics

Landscape metrics allow to quantify the composition (i.e., amount) of and configuration (i.e., arrangement) of spatial landscape characteristics within the context of categorical raster data (Gustafson, 1998, 2019; Uuemaa et al., 2013). The advantages of the landscape metrics approach include their easy application, interpretation, and communication, especially in landscapes with clear distinctions between classes (Lausch et al., 2015). Thus, they are a common tool in landscape and spatial ecology (Kupfer, 2012; Lausch et al., 2015; Frazier and Kedron, 2017). Even though the first conceptual developments date back to the 1980s (Turner, 1989; Gustafson, 1998, 2019), the application of landscape metrics gained popularity with the release of the FRAGSTATS software (McGarigal et al., 2012) in 1995, which allows to calculate an extensive collection of metrics (Kupfer, 2012; Gustafson, 2019).

While FRAGSTATS pioneered the field and made landscape metrics available to many scientists using a graphical user interface, more recently, the **landscapemetrics** R package (Hesselbarth et al., 2019) allows to calculate a large collection of metrics using the R programming environment, which is popular among ecologists (Lai et al., 2019; Hesselbarth et al., 2021). The **landscapemetrics** package is based on the **terra** R package (Hijmans, 2021), but it additionally supports raster objects from the **stars** (Pebesma, 2019) and **raster** (Hijmans, 2019) packages. Furthermore, some functionality also supports vector data (e.g., sample points) using the **sf** package (Pebesma, 2018).

Following (McGarigal et al., 2012), landscape metrics can be classified depending on the characteristics they describe, namely, area and edge, shape, core area, contrast, aggregation, and diversity metrics. Furthermore, landscape metrics can be calculated on patch-, class-, and landscape-scale. While patch-level metrics describe each patch, i.e., contiguous cells belonging to the same class as identified by a connected component labeling algorithm, class-level metrics describe all patches belonging to the same class, and finally, landscape-level metrics describe the entire landscape, Fig. 1. In this chapter, we will demonstrate how to calculate all these types of landscape metrics.

In order to use all functionality of the **landscapemetrics** package and to pre- and post-process data, we need to load several packages related to spatial analysis and data wrangling. This includes the **terra** and **sf** packages for methods related to predominantly raster and vector data, respectively. Furthermore, the **dplyr** (Wickham, 2019) and **tidyr** (Wickham et al., 2023) packages provide a grammar for data manipulation and cleaning, while **ggplot2** (Wickham, 2016) is a data

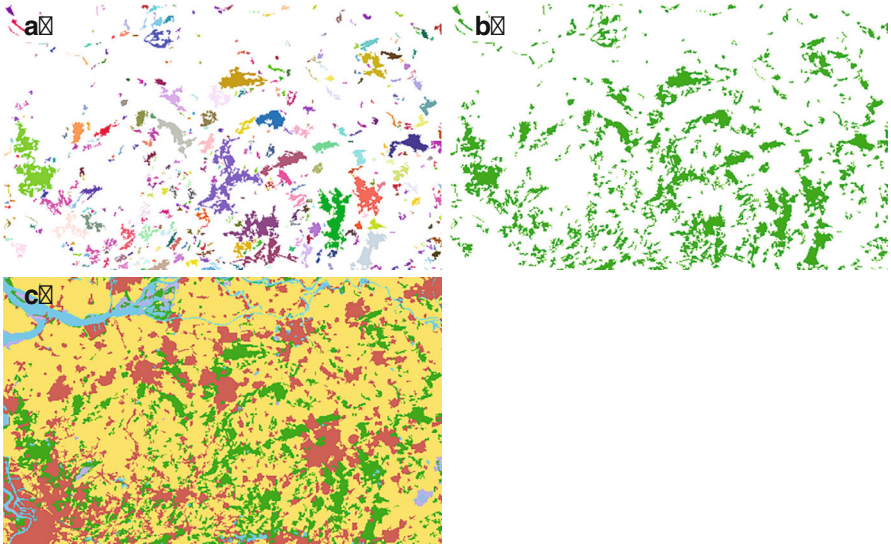


Fig. 1 (**a**) Patch level: patches of an exemplary class indicated by color, i.e., all connected cells belonging to the same LULC class. (**b**) Class level: exemplary class indicated by color, i.e., all patches belonging to the same LULC class. (**c**) Landscape level: exemplary landscape including all LULC classes

visualization package. Additionally, we create a color scheme for all maps we are going to create in later examples.

```
library(landscapemetrics)
library(terra)
library(sf)
library(dplyr)
library(tidyr)
library(ggplot2)
color_scale <- c(urban="#C86058", agriculture="#FCE569",
                 vegetation="#44A321", marshes="#A3A6FF",
                 water="#00CFFD", nodata="#666666")
```

Next, we import the three raster objects using the **terra** package representing the LULC data of the three exemplary regions as `SpatRaster` objects.

```
france <- rast("data/raster_france.tif")
netherlands <- rast("data/raster_netherlands.tif")
sweden <- rast("data/raster_sweden.tif")
```

Before calculating metrics, we need to check if the input raster fulfills certain requirements of the **landscapemetrics** package. For this, we use the `check_landscape()` function. The function checks if the coordinate reference

system (CRS) is projected (i.e., using Cartesian coordinates on a planar surface), the units of the map, and the number of classes. If some of the checks fail, we are still able to calculate all metrics; however, especially metrics relying on distances might not be correct or hard to interpret, e.g., due to the decimal degree units of most geographic CRS (on the contrary, metric distance units of most projected CRS). In this case, the function will return a corresponding warning.

```
check_landscape(france)
```

```
  layer       crs units   class n_classes OK
1     1 projected     m integer         5  v
```

The input raster, `france`, has a projected CRS with units in meters and five discrete classes in total. Because the raster does not violate any of the checks, the function returns an OK check mark.

1.1.1 Deriving Landscape Metrics

All functions to calculate metrics start with the prefix `lsm_`, followed by an abbreviation for the level (either `p` = patch, `c` = class, and `l` = landscape), and finally, an abbreviation of the metric. We can get an overview of all metrics that are currently provided by the **landscapemetrics** package using the `list_lsm()` function. Additionally, we can use the function to display only a certain subset of metrics based on type or level.

```
list_lsm()
```

```
# A tibble: 133 x 5
  metric name                          type                level function_name
  <chr>  <chr>                         <chr>               <chr> <chr>
1 area   patch area                    area and edge metric patch lsm_p_area
2 cai    core area index               core area metric     patch lsm_p_cai
3 circle related circumscribing circle shape metric        patch lsm_p_circle
# i 130 more rows
```

```
list_lsm(type="area and edge metric", level="class")
```

```
# A tibble: 11 x 5
  metric  name       type                level function_name
  <chr>   <chr>      <chr>               <chr> <chr>
1 area_cv patch area area and edge metric class lsm_c_area_cv
2 area_mn patch area area and edge metric class lsm_c_area_mn
3 area_sd patch area area and edge metric class lsm_c_area_sd
# i 8 more rows
```

To calculate a singular metric on patch-, class-, or landscape-level, we simply use the function name and provide the input raster.

```
df_p_shape <- lsm_p_shape(france)
df_c_area <- lsm_c_area_mn(france)
df_l_lpi <- lsm_l_lpi(france)
```

Thus, to calculate the shape index of each patch, the mean patch area of each class *i*, or the largest patch index of the landscape, we use the three functions shown above. The returned data frames always include the same columns regardless of the specified metric.

```
df_l_lpi
```

```
# A tibble: 1 x 6
  layer level     class    id metric value
  <int> <chr>     <int> <int> <chr>  <dbl>
1     1 landscape    NA    NA lpi     65.1
```

These are an identifier of the raster layer, the level of the calculated metric, the class identifier, the patch identifier, the name of the metric, and finally, the metric value. If any of the columns are not applicable (e.g., the patch identifier on the landscape level), the column will contain `NA` values. This allows us to combine different data frames or apply the same post-processing workflow, even for different metrics and/or levels. For example, we can use the `rbind()` function to combine all three results into a single data frame and use functions from the **dplyr** package to calculate the minimum and maximum value for each metric.

```
result_combined <- rbind(df_p_shape, df_c_area, df_l_lpi)
result_range <- group_by(result_combined, metric) |>
  summarise(min=min(value), max=max(value))
result_range
```

```
# A tibble: 3 x 3
  metric    min    max
  <chr>   <dbl>  <dbl>
1 area_mn  64.4 3049.
2 lpi      65.1   65.1
3 shape     1     84.5
```

If we want to derive more than a few metrics, we can also use the `calculate_lsm()` function, which allows to calculate several metrics with only one function call. However, as pointed out by other authors (Gustafson, 2019), "metric fishing" is an issue, i.e., computing as many metrics as possible and searching for any signal. Thus, we strongly advise not to calculate all metrics, but rather select metrics based on research questions and hypotheses.

The calculate_lsm() function has many options for how to specify metrics, e.g., a vector of function names or the types and levels of metrics that can be specified.

```
df_area <- calculate_lsm(france, what=c("lsm_p_area",
                                        "lsm_c_area_mn",
                                        "lsm_l_ta"))
df_aggr_lsm <- calculate_lsm(france, level="landscape",
                                     type="aggregation metric")
```

For many metrics, we can specify further arguments, such as the direction of the connected labeling algorithm, which cells are considered core or edge cells, or the potential number of maximum classes. To see what arguments are available, we can have a look at the help page of a function, e.g., ?lsm_p_core, but all arguments of a specific metric function can also be used with calculate_lsm(). For example, we can change the directions of the connected labeling algorithm and calculate the number of patches on the class level. Comparing the results using a workflow including the **dplyr** package, we can see that the number of patches is larger when using "rook's" (4-neighborhood) instead of "queen's" rule (8-neighborhood; the default).

```
df_np_queen <- lsm_c_np(france, directions=8)
df_np_rook <- lsm_c_np(france, directions=4)
df_comparison <- full_join(x=df_np_queen, y=df_np_rook,
                           by=c("layer", "level", "class", "id",
                                "metric"),
                           suffix=c(".queen", ".rook")) |>
  mutate(value.diff=abs(value.queen - value.rook))
df_comparison
```

```
# A tibble: 5 x 8
  layer level class    id metric value.queen value.rook value.diff
  <int> <chr> <int> <int> <chr>        <dbl>      <dbl>      <dbl>
1     1 class     1    NA np            3231       4589       1358
2     1 class     2    NA np            1541       3592       2051
3     1 class     3    NA np            5512       9242       3730
# i 2 more rows
```

We can also use the calculate_lsm() function (or any other lsm_ function) to calculate metrics of several raster layers or objects simultaneously. For this, we need to use a SpatRaster with several layers or a list of SpatRaster's as input. We are using the later example to calculate the perimeter and area of each patch for two regions in Europe. For this example, we only want to keep the vegetation class, i.e., class 3. Additionally, because some of the patches within each landscape are disproportional large, we are going to use the 95% quantiles of each region and metric to remove some larger values. Next, we are relabeling the unique layer identification column using the country names. Last, we are reshaping the data frame from a long to a wider format to produce a **ggplot2** figure in the final step, Fig. 2.

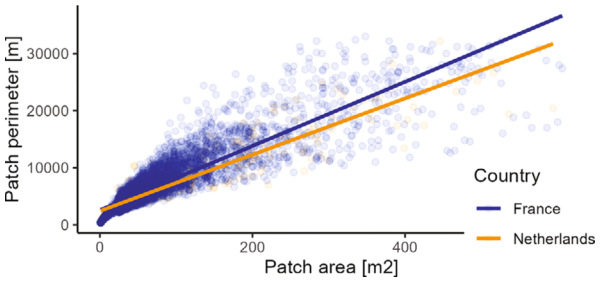


Fig. 2 Relationship between the patch area and patch perimeter in the Centre-Val de Loire region in France (blue) and Noord-Brabant region in the Netherlands (orange)

```
df_perim_core <- list(nl=netherlands, fr=france) |>
  calculate_lsm(what=c("lsm_p_perim", "lsm_p_area")) |>
  filter(class %in% 3) |>
  mutate(layer=case_when(layer==1~"Netherlands",
                         layer==2~"France")) |>
  pivot_wider(names_from=metric, values_from=value) |>
  filter(area<=quantile(area, probs=0.95) &
         perim<=quantile(perim, probs=0.95))
ggplot(data=df_perim_core, aes(x=area, y=perim, color=layer)) +
  geom_point(alpha=0.1) +
  geom_smooth(se=FALSE, method="lm", formula="y ~ x") +
  scale_color_manual(name="Country",
                     values=c(Netherlands="#F79400",
                              France="#001E96")) +
  labs(x="Patch area [m2]", y="Patch perimeter [m]") +
  theme_classic() + theme(legend.position = c(0.9, 0.1))
```

This example also demonstrates how easily the **landscapemetrics** package can be incorporated into larger workflows. As expected, there is a strong relationship between the patch area and perimeter. Logically, as the patch area increases, also the perimeter increases, and this trend is comparable between the two regions.

1.1.2 Visualization of Landscape Metrics

We can also use built-in functions to visualize the landscapes or calculated metrics. All visualization functions start with the `show_` prefix and return a list storing **ggplot2** objects. This is done to be type-stable (i.e., always returning the same data type) regardless of the number of raster objects/layers or selected metrics. First, we can visualize all patches, i.e., all connected cells as specified by the `directions` argument of the connected components labeling algorithm. We can show the patches of the entire landscape (using the argument `class="global"`) or only patches of specific classes (e.g., using `class=3`). Here, we show all patches of the vegetation class in the French region, Fig. 3.

```
show_patches(france, class=3)
```

Fig. 3 Patches of the natural vegetation class in the Centre-Val de Loire region in France

Furthermore, showcasing a more advanced approach, we have the capability to fill each cell of a patch accordingly to a specific landscape metric. For instance, in the following example, the fill color of each patch in the vegetation class corresponds to its area. Since we utilize a list to store two landscapes, the resulting list contains two elements, each representing a landscape and storing the associated `ggplot2` objects. It becomes apparent that the landscape in France is dominated by a single large patch situated at the center of the region, Fig. 4.

```
list_gg_area <- list(fr=france, nl=netherlands) |>
  show_lsm(class=3, what="lsm_p_area")
plot_grid(plotlist=list_gg_area, ncol=2, labels=c("NL", "FR"))
```

Correlation between several metrics can be an issue (Cushman et al., 2008; Schindler et al., 2008; Nowosad and Stepinski, 2018), especially in combination with the previously discussed practice of "metric fishing." Thus, we can use the `show_correlation()` function to check correlations between metrics that were previously calculated using either `calculate_lsm()` or singular metric functions, Fig. 5.

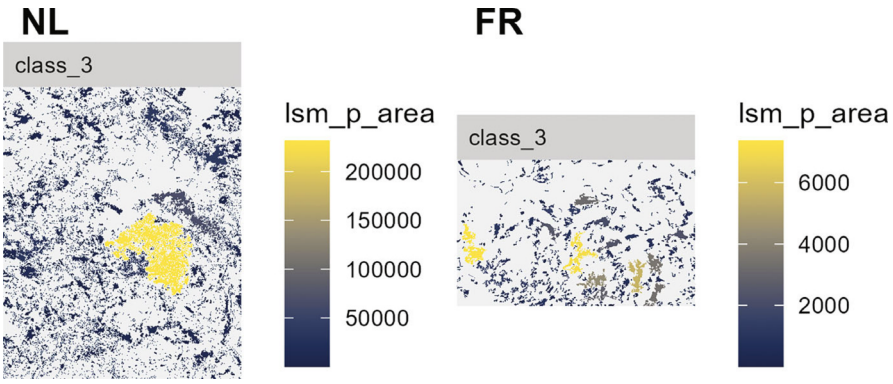
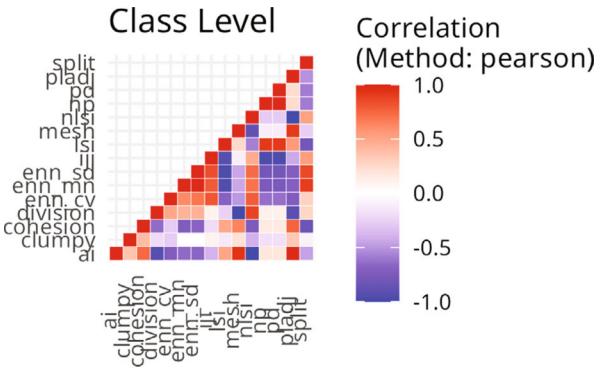


Fig. 4 Visualization of the patch area for the natural vegetation LULC class in the Centre-Val de Loire region in France (left) and the Noord-Brabant region in the Netherlands (right)

Fig. 5 Correlation matrix of all class-level metrics for the Noord-Brabant region in the Netherlands




```
class_metrics <- calculate_lsm(netherlands, level="class",
                               type="aggregation metric")
show_correlation(class_metrics)
```

1.1.3 Additional Features

The **landscapemetrics** also provides some more advanced functionality. This includes extracting values of patches, sampling of metrics within subregions, or applying a moving window approach. In terms of metrics selection or arguments passed on to the metric functions, all these functions behave similarly to the `calculate_lsm()` function.

In order to demonstrate the extraction and sampling of metrics, we first need to create sample points. For this, we are going to randomly create ten points within a region, Fig. 6.

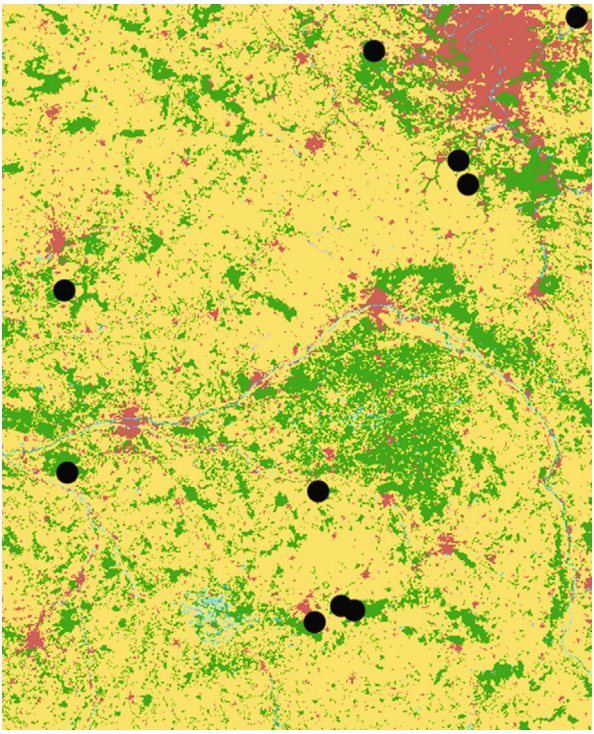


Fig. 6 Location of exemplary sample points used to extract and sample landscape metrics in the Centre-Val de Loire region in France

```
samplepoints <- spatSample(france, size=10, as.points=TRUE) |>
  st_as_sf()
```

Now, we can extract patch-level metrics of all patches that contain a sample point. For this, we use the `extract_lsm()` function. The resulting data frame has 20 rows, one for each metric and sample point. It is worth noting that compared to previous result data frames, there is an additional column named `extract_id`, which facilitates matching the metric results with the respective sample points.

```
df_extract <- extract_lsm(france, samplepoints,
                          what=c("lsm_p_area", "lsm_p_perim"))
df_extract
```

```
# A tibble: 20 x 7
  layer level class    id metric  value extract_id
  <int> <chr> <int> <int> <chr>   <dbl>      <int>
1     1 patch     3  5207 area    15202          1
2     1 patch     3  5207 perim  297200          1
3     1 patch     3  7899 area     7652          2
# i 17 more rows
```

Similarly, we can also sample metrics within a buffer around each sample point. For this, we need to additionally provide a size of the sampling buffer and its shape (either a square or a circle), but it is also possible to directly use polygon objects as sampling buffers.

```
df_sample <- sample_lsm(france, samplepoints,
                        what=c("lsm_c_pland", "lsm_l_ta"),
                        size=10000, shape="circle")
filter(df_sample, percentage_inside>=75)
```

```
# A tibble: 47 x 8
  layer level class    id metric value plot_id percentage_inside
  <int> <chr> <int> <int> <chr>  <dbl>   <int>              <dbl>
1     1 class     1    NA pland   3.54       1               101.
2     1 class     2    NA pland  48.6        1               101.
3     1 class     3    NA pland  46.8        1               101.
# i 44 more rows
```

The resulting data frame has two additional columns. First, similar to the metrics extraction, an additional column identifies the sampling buffer. Second, an additional column stores information about the actual clipped sampling buffer area. Theoretically, this value should be equal to 100%. However, it includes all of the raster cells whose centroids are inside the selector polygon (Lovelace et al., 2019), and thus the actual sampling buffer area might be slightly larger or smaller than the specified area resulting in a deviation from the theoretical value. Furthermore, sample points at the edge of the landscape might include a smaller area than specified by the size argument. In these cases, a warning is returned, and these sample plots can potentially be excluded from further analyses.

Similar to the previously used `show_lsm()` function, patch-level metrics can be returned as a raster object in which each cell stores the metric value of the patch it belongs to. Thus, each cell in the raster object stores its corresponding metrics value. For this, the function `spatialize_lsm()` can be used. However, in order to be type-stable, this function always returns a nested list for each layer and metric (even if only one layer and/or metric is present). The first level of the list corresponds to the number of layers, while the second level of the list corresponds to the number of selected metrics. So we need to use some indexing in order to get only one raster, e.g., the fractal dimension index raster, which we now can use for further analysis, such as calculating kernel density estimates (results not shown).

```
list_shape <- spatialize_lsm(netherlands,
                             what=c("lsm_p_shape", "lsm_p_frac"))
class(list_shape)
str(list_shape)
values(list_shape$layer_1$lsm_p_frac, mat=FALSE) |> density()
```

Last, **landscapemetrics** provides a moving window approach, i.e., metrics can be calculated for the local neighborhood of each focal cell. For this, the `window_lsm()` function can be used. Similar to the `focal()` function of the **terra** package, a local neighborhood must be specific using a matrix object. As with the previously discussed functions, metrics can be selected using the type of metric or specific metrics using the `what` argument. However, currently only landscape-level metrics are implemented. Depending on the size of the landscape and the local neighborhood matrix, this can be computationally demanding with a long run time.

```
mat_window <- matrix(1, nrow=501, ncol=501)
window_lsm(netherlands, window=mat_window, what="lsm_l_pr")
```

1.1.4 Utility Functions

Many of the functions internally used by the **landscapemetrics** package might also be useful for users during pre- or post-processing of raster data or to develop new methods to quantify landscape characteristics. All of these functions start with the `get_` suffix and usually return a (nested) list to be type-stable for different inputs.

One of the most fundamental functions of the package is the `get_patches()` function, which returns the patches of each class using the connected component labeling algorithm. Yet, the result is a nested list in which the first level refers to the number of raster layers/objects and the second level stores all patches separated by class *i*. All other cells not belonging to the current class *i* are set to `NA`. In the following example, we first get all patches of the region in French separated by class. Next, we use all wetland and water-related classes and combine them into one `SpatRaster` object as layers. Last, we use raster algebra to sum all cells which returns a singular layer in which all cells are labeled by their cell identifier and all other cells are `NA`. This results in a single raster layer combining all patches of one of the two water-related classes.

```
list_patches <- get_patches(france)
ras_water <- list_patches$layer_1[4:5] |>
  rast() |>
  sum(na.rm=TRUE)
ras_water
```

```
class       : SpatRaster
dimensions  : 2860, 2333, 1  (nrow, ncol, nlyr)
resolution  : 100, 100  (x, y)
extent      : 3568800, 3802100, 2619700, 2905700  (xmin, xmax, ymin, ymax)
coord. ref. : ETRS89-extended / LAEA Europe (EPSG:3035)
source(s)   : memory
name        :   sum
min value   : 10285
max value   : 10844
```

The `get_boundaries()` function returns a raster in which all boundary/edge cells of a patch are labeled 1 and all core cells are labeled 0. We define an edge cell as a cell that shares a neighboring cell with a different LULC value than itself. Usually, we apply the function to previously connected component labeled landscapes and receive a list with one element for each layer/object. This list can be used for further analysis, such as counting the number of edge and core cells.

```
ras_boundaries_urban <- get_boundaries(list_patches$layer_1$class_1)
lapply(ras_boundaries_urban, freq, wide=TRUE)
```

```
[[1]]
  layer      0      1
1     1 304595 164282
```

We can use the two functions `get_unique_values()` and `get_adjacencies()` to receive a vector with all unique class values and their adjacency matrix, respectively. Because both return a list in order to be type-stable, we can directly index the first element if we only provide one raster layer/object. Of course, the dimensions of the adjacency matrix must be identical to the length of the unique class values given.

```
vec_unique <- get_unique_values(sweden)[[1]]
mat_adjacencies <- get_adjacencies(sweden)[[1]]
dim(mat_adjacencies)
```

```
[1] 5 5
```

```
length(vec_unique)
```

```
[1] 5
```

Relatedly, we can use the `get_nearestneighbour()` function on a previously connected component labeled landscape, which returns a data frame with the minimum Euclidean distances to the nearest patch of the same class. In the following example, we calculate the minimum distances between all patches of class 1, i.e., urban areas. There are 1,175 patches in total and the distance, as well as the identifier of the nearest neighbor, is included. We see that the distance between patches 626 and 650 of the urban class is the largest in the LULC data of Sweden (results are not shown).

```
sweden_cl1 <- get_patches(sweden, class=1)
df_neighbour <- sweden_cl1$layer_1$class_1 |>
  get_nearestneighbour(return_id=TRUE) |>
  arrange(-dist)
```

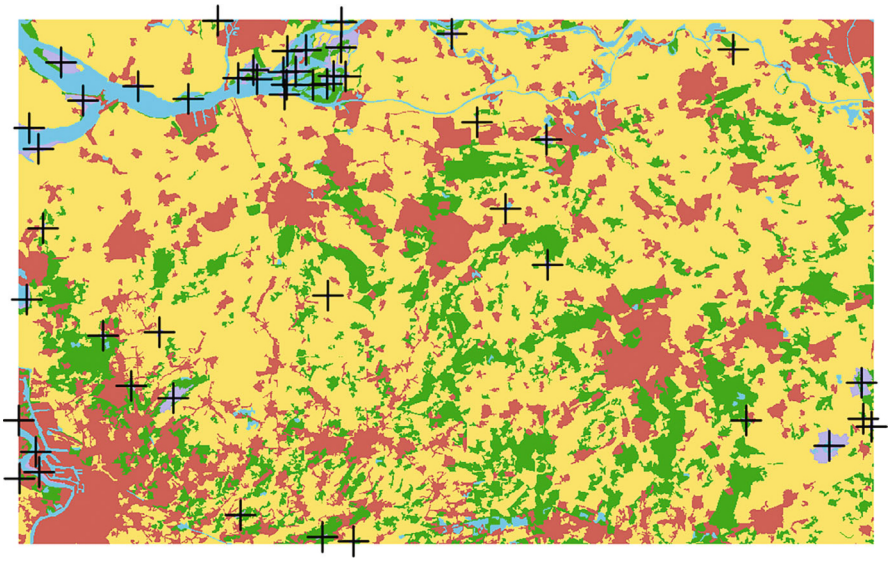


Fig. 7 Patch centroids of all marsh patches in the Noord-Brabant region in the Netherlands

Last, there are two utility functions related to the shape of the patches. First, the function `get_circumscribingcircle()` returns a data frame with the diameter of the smallest circumscribing circle around each patch and the x- and y-coordinate of it. Similar to this, the `get_centroids()` function returns a data frame with the coordinates of the centroids of each patch. For this example, we will filter the data frame to include only the centroids of the marshes class and create a map plotting the patches and all the centroids, Fig. 7.

```
df_circle <- get_circumscribingcircle(netherlands)
df_centroids <- get_centroids(netherlands) |>
  dplyr::filter(class==4)
as.data.frame(netherlands, xy=TRUE) |>
  ggplot(aes(x=x, y=y)) +
  geom_raster(aes(fill=as.factor(cover))) +
  geom_point(data=df_centroids, aes(x=x, y=y), shape=3, size=2.5) +
  scale_fill_manual(values=color_scale) +
  coord_equal() + theme_void() + theme(legend.position="none")
```

1.2 Pattern-Based Spatial Analysis

Pattern-based spatial analysis is a set of methods allowing for performing various tasks on landscape patterns, such as comparing landscape patterns or searching for areas with similar landscape patterns to the query one (Wickham and Norton,

1994; Jasiewicz et al., 2015; Netzel et al., 2018; Nowosad, 2021). Instead of treating each cell independently, this approach focuses on considering local composition and configuration. Firstly, each area, a group of adjacent cells encompassing a landscape pattern, is described numerically as spatial signatures. Next, the similarity between spatial signatures for two areas can be measured using various dissimilarity measures. Low dissimilarity values suggest that the two areas have similar composition and configuration. These ideas allow, for example, to compare spatial pattern change between an area in time, search for similarities between areas of interest, or group areas with similar patterns together. Spatial signatures may also be used directly as new variables, for example, in machine learning models.

The pattern-based spatial analysis methods are available in the R package **motif** (Nowosad, 2021), which allows to describe spatial patterns of one or more categorical raster data for any defined regular and irregular regions. It accepts spatial raster objects from the **terra** (Hijmans, 2021) and **stars** (Pebesma, 2019) R packages as inputs. All **motif** functions use the `lsp_` (*local spatial pattern*) prefix consistently, allowing users to find the desired tool quickly. Most functions in the package are based on a computationally fast C++ code, and the software is designed to work on larger-than-RAM raster datasets.

We need to start by attaching the **motif** package for pattern-based spatial analysis and two main packages for handling spatial data, **terra** and **sf** (Pebesma, 2018).

```
library(motif)
library(terra)
library(sf)
```

Our main dataset for the following examples will be the spatial raster representing land cover classes in the Centre-Val de Loire region in France for the year 2018.

```
france <- rast("data/raster_france.tif")
plot(france) # result not shown
```

1.2.1 Spatial Signatures

The backbone of pattern-based spatial analysis is the concept of spatial signatures. A spatial signature is a short numerical descriptor of landscape patterns featured in a particular area and represented by a categorical raster. Landscape patterns can be described in various ways, and thus many possible spatial signatures exist. That being said, they usually express the composition and/or configuration (arrangement) of categorical raster cells (Table 1). It is also worth noting that spatial signatures can be calculated for a large area or many smaller subareas divided by regular or irregular zones (*windows*).

Spatial signatures are derived using the `lsp_signature()` function. We need to provide two arguments to calculate a single spatial signature: our input categorical raster data (`x`) and the spatial signature type (`type`). The most basic spatial signature is `"composition"`: It is a share of cells of each category in a given area.

Table 1 Spatial signatures available in the motif package

Type	Name	Description
Composition	Composition	A representation of the share of cells for each category within a local landscape. The length of the composition vector corresponds to the number of unique categories in a raster. Input: one categorical raster
Cove	Co-occurrence vector	A count of all of the pairs of the adjacent cells for each category in a local landscape. By default, its length equals $(n * n - n)/2 + n$, where n is a number of unique categories. Input: one categorical raster
Wecove	Weighted co-occurrence vector	A modification of a co-occurrence vector, in which each adjacency contributes to the output based on the values from the weight raster. By default, its length equals $(n * n - n)/2 + n$, where n is a number of unique categories. Input: one categorical and one numerical raster
Incove	Integrated co-occurrence vector	A count of all of the pairs of the adjacent cells for each category in a local landscape for all of the input rasters. Input: two or more categorical rasters
-	A user-defined function	Any user-defined function that can summarize categorical spatial raster data

```
france_comp <- lsp_signature(france, type="composition")
france_comp
```

```
# A tibble: 1 x 3
     id na_prop signature
* <int>   <dbl> <list>
1     1       0 <dbl [1 x 5]>
```

The `lsp_signature()` function always returns a data frame with three columns: a unique identifier (`id`), the proportion of cells in an area with missing values (`na_prop`), and the calculated signature (`signature`). This last column is a list containing a single signature per area of interest. We can see the obtained spatial signature by accessing the first element of the `signature` list.

```
france_comp$signature[[1]]
```

```
              1         2         3            4         5
[1,] 0.07027133 0.7041287 0.2184272 0.0004055524 0.0067673
```

It shows that ~70% of the area is covered by agriculture (2), ~22% by vegetation (3), and ~7% by the urban class (1).

While the "composition" signature is straightforward to understand, it does not encompass the spatial arrangement of the categories: It does not tell if a given category represents one large continuous area or if it is distributed as many small patches. Thus, more complex and informative signatures should be used in most cases. For instance, when dealing with a single categorical raster, a co-occurrence vector ("cove") serves as an appropriate choice for such a signature.

The co-occurrence vector is a spatial signature that encapsulates both the composition and configuration of raster classes in a given area (Haralick et al., 1973; Jasiewicz et al., 2015; Nowosad and Stepinski, 2021). This is achieved by counting not how many times each class occurs but how often a pixel with a given class is adjacent to another pixel of some class. Thus, the result is a long vector (15 elements in this example) that counts how many times pairs of pixels with given classes are adjacent, e.g., urban cells are adjacent to other urban cells, and how many times urban cells are adjacent to, for example, vegetation cells. In general, you can think of spatial signatures as a way to compress information: Our original raster has 6,672,380 values, which we now compressed into a vector of only 15 elements.

```
france_cove <- lsp_signature(france, type="cove")
france_cove
```

```
# A tibble: 1 x 3
     id na_prop signature
* <int>   <dbl> <list>
1     1       0 <dbl [1 x 15]>
```

As shown above, spatial signatures may be used as multi-value descriptors of landscape patterns of a given area. This allows us to calculate spatial signatures of two or more areas and then compare them using a dissimilarity measure.

By default, the `window` argument is set to `NULL`, and **motif** performs calculations for the entire area. However, the package also offers the flexibility to compute numerous spatial signatures for a single large area by dividing it into multiple subareas referred to as "local landscapes." This partitioning is achieved by specifying either a numeric value or an `sf` polygon data as the `window` argument.

When a numeric value is provided, the input raster is subdivided into square-shaped subareas, with each side length equal to the specified numeric value, measured in cells. In the following example, we demonstrate the calculation of co-occurrence vectors for square subareas measuring 50 by 50 cells, which is equivalent to 5 by 5 km in this example.

```
france_cove2 <- lsp_signature(france, type="cove", window=50)
france_cove2
```

```
# A tibble: 2,726 x 3
     id na_prop signature
* <int>   <dbl> <list>
1     1       0 <dbl [1 x 15]>
2     2       0 <dbl [1 x 15]>
3     3       0 <dbl [1 x 15]>
# i 2,723 more rows
```

Now, notice that the output here is still a data frame with three columns, but instead of having only one row, we now have 2,726 rows—one row per subarea. We may visualize these subareas by creating a new `sf` object with `lsp_add_sf()` and then using the `plot()` function (Fig. 8).

```
france_cove2_sf <- lsp_add_sf(france_cove2)
plot(france, ext=ext(france_cove2_sf))
plot(st_geometry(france_cove2_sf), add=TRUE)
```

The `window` argument can also accept an `sf` polygon, allowing for the independent derivation of spatial signatures within each polygon-defined area. In the provided example, we begin by randomly selecting 100 sample points within the study area. For each point, we create a buffer of 300 m, resulting in a corresponding buffer polygon. By setting this buffer polygon as the `window` argument, we can obtain a spatial signature for the 300 m buffer surrounding each input point.

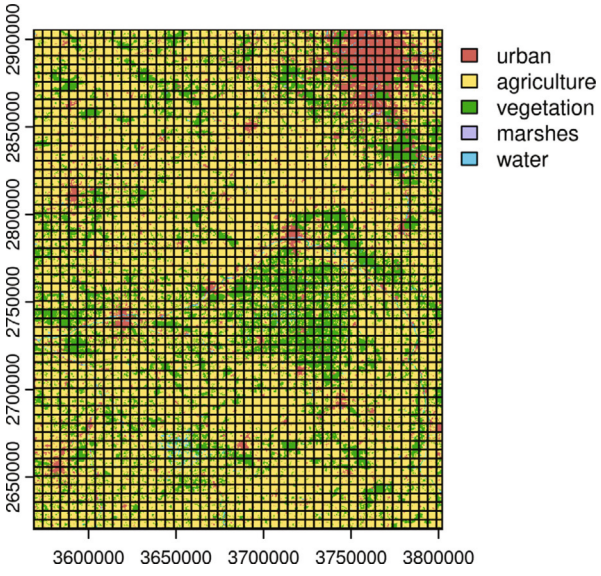


Fig. 8 Land cover of the study area overlaid with a grid with each subarea of 50 by 50 input raster cells

```
sample_points <- read_sf("data/france_sample_points.gpkg")
sample_polys <- st_buffer(sample_points, dist=300)
france_cove3 <- lsp_signature(france, type="cove",
↪  window=sample_polys)
france_cove3 # result not shown
```

1.2.2 Landscape Patterns' Comparison

Landscape patterns change through time, and we are able to evaluate and measure such a change. Here, we will use a second dataset, land cover classes for the Centre-Val de Loire region in France for the year 2000.

```
france2000 <- rast("data/raster_france2000.tif")
```

Now that we have two datasets, for the years 2000 and 2018, we can proceed with their comparison. In pattern-based spatial analysis, the comparison involves calculating the dissimilarity between the spatial signatures derived from both rasters. This approach is complementary to the pixel-based comparison that is based on comparing all the values for each dataset, as it allows us to evaluate the change in the landscape pattern.

To perform this comparison, we utilize the `lsp_compare()` function and provide our two datasets, `france2000` and `france`, along with specifying the type of spatial signature (`type`) and the chosen dissimilarity measure (`dist_fun`). Additionally, we have the option to obtain the results in one of several output classes. For instance, we can choose to use the `SpatRaster` class from the **terra** package by setting `output="terra"`.

```
lc_change1 <- lsp_compare(france2000, france,
                          type="cove", dist_fun="jensen-shannon",
                          output="terra")
lc_change1["dist"]
```

```
class       : SpatRaster
dimensions  : 1, 1, 1  (nrow, ncol, nlyr)
resolution  : 233300, 286000  (x, y)
extent      : 3568800, 3802100, 2619700, 2905700  (xmin, xmax, ymin, ymax)
coord. ref. : ETRS89-extended / LAEA Europe (EPSG:3035)
source(s)   : memory
name        :         dist
min value   : 0.0002682711
max value   : 0.0002682711
```

The above example compared the change in landscape patterns using the co-occurrence vector signature (`"cove"`) and the Jensen-Shannon distance (`"jensen-shannon"`) for the whole area. However, we can change the spatial scale of our analysis and look at the spatial pattern changes also for smaller subareas.

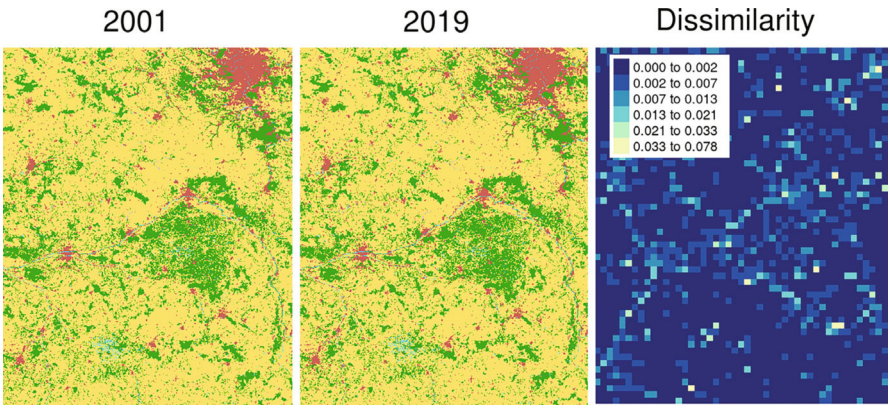


Fig. 9 (Left) land cover data for year 2000; (center) land cover data for year 2018; and (right) change in land cover landscape patterns between 2000 and 2018 for 50 by 50 cells areas (Jensen-Shannon distance)

This can be done by adding a value to the `window` parameter. In the example below, we set the `window` parameter to 50, meaning each subarea will consist of 50 by 50 cells from the input rasters, Fig. 9.

```
lc_change <- lsp_compare(france2000, france,
                         type="cove", dist_fun="jensen-shannon",
                         output="terra", window=50)
```

The highest values in the right panel of Fig. 9 represent areas with the most prominent change of land cover landscape patterns. They mainly relate to the change of herbaceous areas into shrubs. We can also find examples of the areas with the largest changes by subsetting only the local landscapes with dissimilarity above some threshold. Then, the `lsp_extract()` function can be used to extract a local landscape with a given `id`.

```
lc_change_df <- as.data.frame(lc_change)
subset(lc_change_df, dist>0.05)
```

```
      id na_prop_x na_prop_y       dist
1764 1764         0         0 0.05966520
2107 2107         0         0 0.07848007
```

```
compare_1 <- lsp_extract(c(france2000, france), window=50, id=2107)
plot(compare_1, legend = FALSE)
```

Figure 10 shows a local landscape with the largest change of land cover landscape patterns between 2000 and 2018. This location is an example of an area with a large loss of natural vegetation (to agriculture) and a small expansion of urban areas.

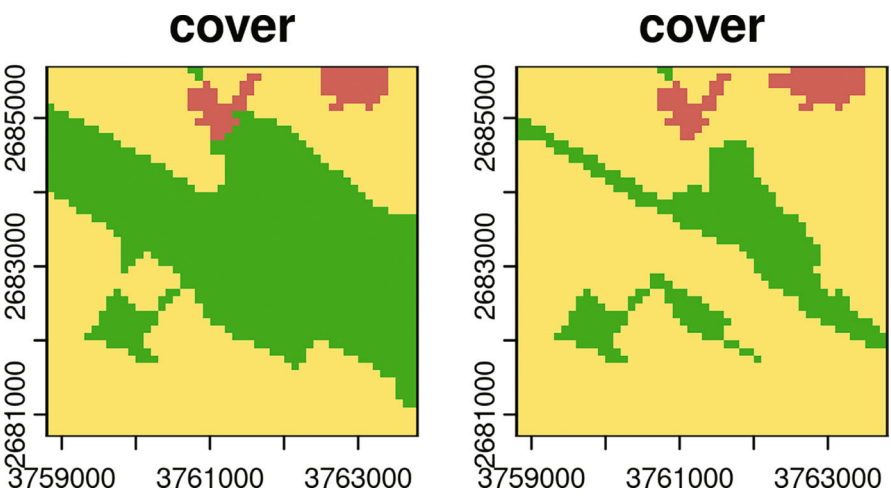


Fig. 10 An example of a local landscape of 5 by 5 km with the largest change of land cover landscape patterns between 2000 and 2018

1.2.3 Landscape Patterns' Search

Another example of pattern-based spatial analysis is searching for areas with similar landscape patterns to a query one. Pattern-based search is a comparison of a spatial pattern of a given local landscape to many subareas of a larger raster.

Here, we start by reading polygon data with our study area and cropping the `france` raster to its borders. Thus, the `france_study_area` object is a raster storing LULC data for the study area.

```
study_area <- read_sf("data/france_study_area.gpkg")
france_study_area <- crop(france, study_area, mask=TRUE)
```

Search can be performed with the `lsp_search()` function, while providing two raster datasets (an area of interest and a larger area), the signature type, distance measure, and the window size.

```
nlcd_search <- lsp_search(france_study_area, france,
                          type="cove", dist_fun="jensen-shannon",
                          output="terra", window=50)
```

```
# result not shown
plot(nlcd_search[["dist"]])
```

Its output is a **terra** object with three raster layers: a unique `id`, the proportion of cells in an area with missing values (`na_prop`), and the calculated dissimilarity

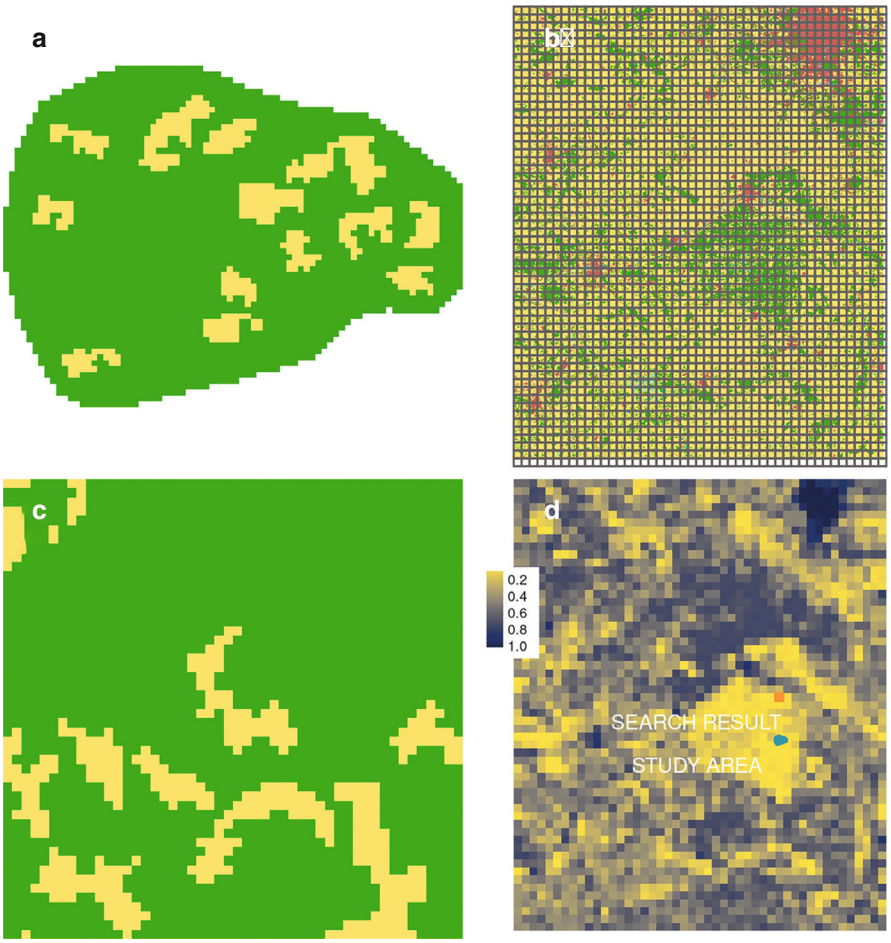


Fig. 11 (**a**) Area of interest, (**b**) search area divided into a set of regular local landscapes, (**c**) a local landscape with the most similar land cover spatial pattern to the area of interest, and (**d**) dissimilarity between land cover landscape patterns of the area of interest and the search area

(`dist`) (Fig. 11d). The areas with the smallest dissimilarity values are the ones that have the most similar spatial pattern to the area of interest.

```
nlcd_search_df <- as.data.frame(nlcd_search)
subset(nlcd_search_df, dist<0.001)
```

```
      id na_prop         dist
1303 1303       0 0.0003939899
1677 1677       0 0.0007520309
1867 1867       0 0.0007956902
```

To extract a selected local landscape, the `lsp_extract()` function can also be used here (Fig. 11c).

```
search_1 <- lsp_extract(france, window=50, id=1303)
```

1.2.4 Landscape Patterns' Regionalization and Clustering

Spatial structures derived with `lsp_signature()` may also be converted to spatial objects of `sf`, `stars`, and `terra` classes. The `lsp_add_terra()` function, for example, takes a spatial signature object and converts it into a **terra** raster object.

```
france_cove2 <- lsp_signature(france, type="cove", window=10)
france_cove_terra <- lsp_add_terra(france_cove2, metadata=FALSE)
france_cove_terra
```

```
class       : SpatRaster
dimensions  : 286, 234, 15  (nrow, ncol, nlyr)
resolution  : 1000, 1000  (x, y)
extent      : 3568800, 3802800, 2619700, 2905700  (xmin, xmax, ymin, ymax)
coord. ref. : ETRS89-extended / LAEA Europe (EPSG:3035)
source(s)   : memory
names       : X1,        X2, X3,        X4,        X5, X6, ...
min values  :  0, 0.0000000,  0, 0.0000000, 0.0000000,  0, ...
max values  :  1, 0.3191489,  1, 0.3055556, 0.3944444,  1, ...
```

This new object has several raster layers representing spatial signature elements. This opens new possibilities for using the information of landscape patterns in other workflows related to spatial regionalization, clustering, and machine learning.

A regionalization example below uses the **supercells** package to group adjacent local areas with similar landscape patterns. We can apply the **supercells** algorithm (Nowosad and Stepinski, 2022), which expects four essential arguments: input raster data, `k`: a number of regions desired by the user, `compactness`: a compactness value, where larger values cause clusters to be more compact, and `dist_fun`: a distance function.

```
library(supercells)
france_sc <- supercells(france_cove_terra, k=200, compactness=0.6,
                        dist_fun="jensen-shannon", metadata=FALSE)
```

The output here is a spatial vector with a set of polygons, where each polygon represents a larger area with a certain level of homogeneity of its landscape patterns (Fig. 12a).

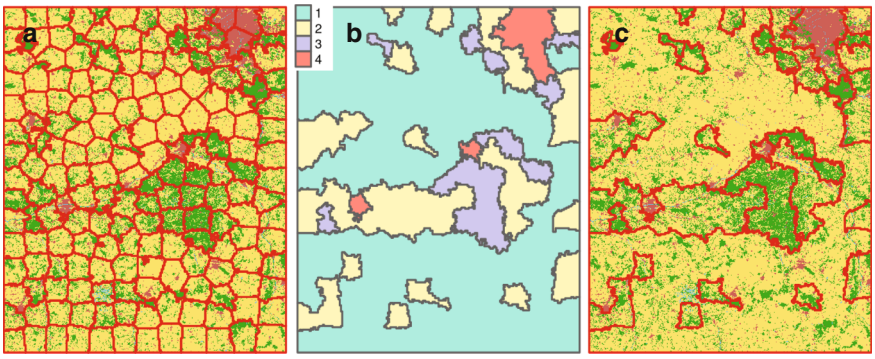


Fig. 12 (**a**) Study area divided into supercells of homogeneous land cover landscape patterns, (**b**) supercells grouped into four clusters using the hierarchical clustering method, and (**c**) borders of the polygons derived with the hierarchical clustering method

```
plot(france) # result not shown
plot(st_geometry(france_sc), add=TRUE, border="red")
```

You may notice that while the land cover landscape patterns are homogeneous internally, there are several examples of similar adjacent polygons. In cases like this, we could merge such polygons, for example, using a clustering method, such as hierarchical clustering or k-means. Firstly, we need to drop the geometry column from the `france_sc` object, as it is not needed for the clustering, and then calculate the value distances between the landscape patterns of each supercell. Next, we can apply the hierarchical clustering method with the `hclust()` function. Based on the dendrogram visualization (`plot(hc)`), we can select the number of clusters: Here we choose four clusters. Finally, we can assign a cluster number to each supercell using the `cutree()` function.

```
france_sc_df <- st_drop_geometry(france_sc)
france_sc_dist <- philentropy::distance(france_sc_df,
                                        method="jensen-shannon",
                                        as.dist.obj=TRUE)
hc <- hclust(france_sc_dist)
france_sc$k <- cutree(hc, 4)
```

The hierarchical clustering method assigns a cluster number to each, previously derived supercell. Thus, to obtain a separate polygon for each cluster, we need to perform a post-processing step: `aggregate()` the clusters to dissolve the borders between clusters.

```
france_sc2 <- aggregate(france_sc, by=list(france_sc$k), FUN=mean)
```

The dissolved areas of clusters are shown on panel b of Fig. 12, while panel c shows the final created polygons. The first cluster represents mostly agricultural areas, the third cluster represents mostly forested areas, and the fourth cluster represents mostly urban areas. The second cluster is a bit more complex, as it contains a mosaic of forested and agricultural areas. The result seems mostly satisfactory, where most created polygons are internally homogeneous and visibly distinct from their neighbors. At the same time, there are a few areas in which homogeneity needs to be improved. This suggests that the result could still be improved, for example, by using a larger number of initial supercells or changing the number of expected clusters.

1.3 Conclusions

In conclusion, this chapter has highlighted the significance of categorical raster data, specifically land use or land cover (LULC) data, in ecological studies. We have explored the role of landscape metrics and pattern-based spatial analysis in quantifying and analyzing landscape patterns, using the R packages **landscapemetrics** and **motif**.

Landscape metrics, computed using the **landscapemetrics** package, have proven to be valuable tools in spatial ecology and landscape ecology. They allow us to quantify the composition and configuration of spatial landscape characteristics, enabling a deeper understanding of environmental processes and changes. By calculating metrics at various levels and visualizing them, researchers can gain insights into the patterns and structures present in the landscape. However, it is crucial to select metrics purposefully, aligned with research questions and hypotheses, to avoid the pitfall of "metric fishing" (Gustafson, 2019).

Moreover, pattern-based spatial analysis, facilitated by the **motif** package, offers a robust framework for analyzing and comparing landscape patterns. By considering local groups of adjacent cells as landscape patterns and calculating spatial signatures, we can capture both composition and configuration information. This approach allows us to compare landscape patterns over time, identify areas with similar patterns, and group similar areas together. The **motif** package's ability to handle large raster datasets is particularly advantageous.

The chapter has provided practical examples and code snippets demonstrating the calculation of landscape metrics and spatial signatures, as well as their visualization. We have shown how to use these techniques to analyze landscape patterns, explore changes over time, and extract local landscapes with significant pattern variations. By combining the power of landscape metrics and pattern-based spatial analysis, researchers can gain a comprehensive understanding of landscape dynamics, composition, and configuration. These techniques facilitate the investigation of the pattern-process link that is a central framework of many ecological fields, but specifically landscape ecology (Turner, 1989). For example, the pattern-process link using LULC data was studied for species distributions (Marshall et al., 2018), (functional) connectivity (Vogt et al., 2009), or genetic

population structure (Borthwick et al., 2020). However, patterns and processes are often in an interacting relationship influencing each other, and many drivers and processes act simultaneously—such as climate, disturbances, succession, biotic competition and dispersal, or human land use (Turner, 2005). The pattern-process link is a complex and challenging topic, and the analysis of landscape patterns is only one part of the puzzle. Nevertheless, landscape metrics and pattern-based spatial analysis are powerful tools to investigate the pattern-process link and to gain a better understanding of the landscape.

Both packages, being open source, offer extensive possibilities for customization and expansion. This enables researchers to tailor them to their specific inquiries and requirements. The **landscapemetrics** package, for instance, allows for the integration of new metrics through the utilization of diverse utility functions, thereby enhancing its capabilities. Similarly, the **motif** package can be expanded by incorporating user-defined spatial signatures, which can be used to measure similarities between different properties of landscape patterns. Consequently, contributions to both packages are highly encouraged and welcomed. You can actively participate in the development of these packages at https://github.com/r-spatialecology/landscapemetrics and https://github.com/Nowosad/motif.

Acknowledgments This work was supported by the Initiative of Excellence—Research University project at Adam Mickiewicz University, Poznan [grant number 107/07/POB1/0002]. We also thank the developers and contributors of the **landscapemetrics** and **motif** packages for their work and support.

References

Borthwick R, de Flamingh A, Hesselbarth MHK et al (2020) Alternative quantifications of landscape complementation to model gene flow in banded longhorn beetles [typocerus v. Velutinus (olivier)]. Front Genet 11:307. https://doi.org/10.3389/fgene.2020.00307

Chandra Pandey P, Koutsias N, Petropoulos GP et al (2021) Land use/land cover in view of earth observation: Data sources, input dimensions, and classifiers—a review of the state of the art. Geocarto Int 36:957–988. https://doi.org/10.1080/10106049.2019.1629647

Cushman SA, McGarigal K, Neel MC (2008) Parsimony in landscape metrics: strength, universality, and consistency. Ecol Indicat 8:691–703. https://doi.org/10.1016/j.ecolind.2007.12.002

European Environment Agency (EEA) (2023) Copernicus land monitoring service. CORINE land cover

Fassnacht KS, Cohen WB, Spies TA (2006) Key issues in making and using satellite-based maps in ecology: a primer. Forest Ecol Manag 222:167–181. https://doi.org/10.1016/j.foreco.2005.09.026

Fisher PF, Comber AJ, Wadsworth R (2005) Land use and land cover: Contradiction or complement. In: Fisher PF, Unwin DJ (eds) Re-Presenting GIS. Wiley, Hoboken

Floreano IX, De Moraes LAF (2021) Land use/land cover (LULC) analysis (2009–2019) with google earth engine and 2030 prediction using Markov-CA in the Rondônia state, Brazil. Environ Monit Assess 193:239. https://doi.org/10.1007/s10661-021-09016-y

Frazier AE, Kedron P (2017) Landscape metrics: past progress and future directions. Current Landscape Ecol Rep 2:63–72. https://doi.org/10.1007/s40823-017-0026-0

Fu P, Weng Q (2016) A time series analysis of urbanization induced land use and land cover change and its impact on land surface temperature with Landsat imagery. Remote Sens Environ 175:205–214. https://doi.org/10.1016/j.rse.2015.12.040

Gustafson EJ (1998) Quantifying landscape spatial pattern: what is the state of the art? Ecosystems 1:143–156. https://doi.org/10.1007/s100219900011

Gustafson EJ (2019) How has the state-of-the-art for quantification of landscape pattern advanced in the twenty-first century? Landscape Ecol 34:1–8. https://doi.org/10.1007/s10980-018-0709-x

Haralick RM, Shanmugam K, Dinstein I (1973) Textural features for image classification. IEEE Trans Syst Man Cybern SMC-3:610–621. https://doi.org/bdqvtn

Hesselbarth MHK, Sciaini M, With KA et al (2019) Landscapemetrics: an open-source R tool to calculate landscape metrics. Ecography 42:1648–1657. https://doi.org/10.1111/ecog.04617

Hesselbarth MHK, Nowosad J, Signer J, Graham LJ (2021) Open-source tools in R for landscape ecology. Current Landscape Ecol Rep 6:97–111. https://doi.org/10.1007/s40823-021-00067-y

Hijmans RJ (2019) Raster: Geographic data analysis and modeling. R package version 2.9-5. https://cran.r-project.org/package=raster

Hijmans RJ (2021) Terra: Spatial data analysis. R package version 1.0-10. https://cran.r-project.org/package=terra

Jasiewicz J, Netzel P, Stepinski T (2015) GeoPAT: a toolbox for pattern-based information retrieval from large geospatial databases. Comput Geosci 80:62–73. https://doi.org/f7fz7r

Kupfer JA (2012) Landscape ecology and biogeography: rethinking landscape metrics in a post-FRAGSTATS landscape. Progress Phys Geography 36:400–420. https://doi.org/10.1177/0309133312439594

Lai J, Lortie CJ, Muenchen RA et al (2019) Evaluating the popularity of R in ecology. Ecosphere 10:e02567. https://doi.org/10.1002/ecs2.2567

Lausch A, Blaschke T, Haase D et al (2015) Understanding and quantifying landscape structure - a review on relevant process characteristics, data models and landscape metrics. Ecol Model 295:31–41. https://doi.org/10.1016/j.ecolmodel.2014.08.018

Lovelace R, Nowosad J, Muenchow J (2019) Geocomputation with R. Chapman and Hall/CRC Press, Boca Raton

Manzoor SA, Griffiths G, Lukac M (2021) Land use and climate change interaction triggers contrasting trajectories of biological invasion. Ecol Indicat 120:106936. https://doi.org/10.1016/j.ecolind.2020.106936

Marshall L, Biesmeijer JC, Rasmont P et al (2018) The interplay of climate and land use change affects the distribution of EU bumblebees. Global Change Biol 24:101–116. https://doi.org/10.1111/gcb.13867

McGarigal K, Cushman SA, Ene E (2012) FRAGSTATS v4: Spatial pattern analysis program for categorical and continuous maps. Computer software program produced by the authors at the University of Massachusetts, Amherst. http://www.umass.edu/landeco/research/fragstats/fragstats.html

Netzel P, Nowosad J, Jasiewicz J et al (2018) GeoPAT 2: User's manual. https://doi.org/10.5281/zenodo.1185407

Nowosad J (2021) Motif: an open-source R tool for pattern-based spatial analysis. Landscape Ecol 36:29–43. https://doi.org/ghfsnh

Nowosad J, Stepinski TF (2018) Global inventory of landscape patterns and latent variables of landscape spatial configuration. Ecol Indicat 89:159–167. https://doi.org/10.1016/j.ecolind.2018.02.007

Nowosad J, Stepinski TF (2021) Pattern-based identification and mapping of landscape types using multi-thematic data. Int J Geograph Inf Sci 35:1634–1649. https://doi.org/gk437s

Nowosad J, Stepinski TF (2022) Extended SLIC superpixels algorithm for applications to non-imagery geospatial rasters. Int J Appl Earth Observ Geoinfor 112:102935. https://doi.org/10.1016/j.jag.2022.102935

Pebesma EJ (2018) sf: simple features for R. https://cran.r-project.org/package=sf

Pebesma EJ (2019) Stars: Scalable, spatiotemporal tidy arrays for R. R package version 0.3-1. https://cran.r-project.org/package=stars
Riitters K (2019) Pattern metrics for a transdisciplinary landscape ecology. Landscape Ecol 34:2057–2063. https://doi.org/gkb48v
Schindler S, Poirazidis K, Wrbka T (2008) Towards a core set of landscape metrics for biodiversity assessments: a case study from Dadia National Park, Greece. Ecol Indicat 8:502–514. https://doi.org/10.1016/j.ecolind.2007.06.001
Talukdar S, Singha P, Mahato S et al (2020) Land-use land-cover classification by machine learning classifiers for satellite observations—A review. Remote Sens 12:1135. https://doi.org/10.3390/rs12071135
Turner MG (1989) Landscape ecology: the effect of pattern on process. Ann Rev Ecol Syst 20:171–197
Turner MG (2005) Landscape ecology: what is the state of the science? Ann Rev Ecol Evol Syst 36:319–344. https://doi.org/10.1146/annurev.ecolsys.36.102003.152614
Uuemaa E, Mander Ü, Marja R (2013) Trends in the use of landscape spatial metrics as landscape indicators: a review. Ecol Indicat 28:100–106. https://doi.org/10.1016/j.ecolind.2012.07.018
Vogt P, Ferrari JR, Lookingbill TR et al (2009) Mapping functional connectivity. Ecol Indicat 9:64–71. https://doi.org/10.1016/j.ecolind.2008.01.011
Wang Z, Mountrakis G (2023) Accuracy assessment of eleven medium resolution global and regional land cover land use products: a case study over the conterminous United States. Remote Sens 15:3186. https://doi.org/10.3390/rs15123186
Wang J, Bretz M, Dewan MAA, Delavar MA (2022) Machine learning in modelling land-use and land cover-change (LULCC): Current status, challenges and prospects. Sci Total Environ 822:153559. https://doi.org/10.1016/j.scitotenv.2022.153559
Wickham H (2016) ggplot2: Elegant Graphics for Data Analysis. Springer, New York
Wickham H, François R, Henry L, Müller K (2019) Dplyr: A grammar of data manipulation. R package version 0.8.1. https://cran.r-project.org/package=dply
Wickham JD, Norton DJ (1994) Mapping and analyzing landscape patterns. Landscape Ecol 9:7–23. https://doi.org/fwz5bm
Wickham H, Vaughan D, Girlich M (2023) Tidyr: Tidy messy data. R package version 1.3.0 https://cran.r-project.org/package=tidyr
With KA (2019) Essentials of Landscape Ecology, 1st edn. Oxford University Press, Oxford
Wulder MA, Coops NC, Roy DP et al (2018) Land cover 2.0. Int J Remote Sens 39:4254–4284. https://doi.org/10.1080/01431161.2018.1452075