Test

Author Name

Summary

Predicting where species should be found in space is a common question in ecology and biogeography. Species distribution models (SDMs), for instance, aim to predict where environmental conditions are suitable for a given species, often on continuous geographic scales. Such analyses require the use of geo-referenced data on species distributions coupled with climate or land cover information, hence a tight integration between environmental data, species occurrence data, and spatial coordinates. Thus, it requires an efficient way to access these different data types within the same software, as well as a flexible framework on which to build various analysis workflows. Here we present `SimpleSDMLayers.jl` and `GBIF.jl`, two packages in the *Julia* language implementing a simple framework and type-system on which to build SDM analyses, as well as providing access to popular data sources for species occurrences and environmental conditions.

Statement of need

Species distribution modeling (SDM) is an increasingly growing field in ecology and biogeography, with many applications in biodiversity assessment, management, and conservation [@Araujo2019StaDis]. Most SDM models aim at predicting a species distribution in space based on environmental data and information on where the species was previously seen. Hence, SDM studies require a tight and efficient integration between geo-referenced environmental and species occurrence data. However, such data are complex to handle and often require different software: climate and land use data are stored as layers in raster files, then visualized and manipulated in specialized GIS (geographic information systems) software, while occurrence data are stored in tables and spreadsheets, then manipulated in data analysis and statistics-oriented tools or programming languages. Therefore, there is a need for efficient tools to manipulate bioclimatic data, specifically oriented towards species distribution modeling.

In recent years, *R* [@RCoreTeam2020RLan] has become the most widely used programming language in ecology, especially in spatial ecology studies [@Lai2019EvaPop]. Hence, many efficient packages and tools for species distribution modeling have been developed in *R*. For instance, the package

`raster` [@Hijmans2020RasGeo] can be used to manipulate raster format data (for example climatic or land use data), `dismo` [@Hijmans2017DisSpe] implements many SDM models and provides access to common climatic data sources, and `rgbif` [@Chamberlain2020RgbInt] provides access to the GBIF database, a common source of species occurrence data in SDM studies. In comparison, few specific SDM resources currently exist for the *Julia* language [@Bezanson2017JulFre], although SDM models could benefit from its speed, efficiency and ease of writing (removing the need to rewrite functions in other languages for higher performance, as in *R*). There are currently packages such as `GDAL.jl` and `ArchGDAL.jl` to manipulate raster data; however, these are lower-level implementations than what is typically used by most ecologists, and they lack support for common layer manipulation. generalized linear models (GLM.jl), random forests (DecisionTrees.jl), neural networks (Flux.jl), and other commonly used models have excellent implementations in *Julia*, although not oriented towards species distribution modeling and raster format data.

`SimpleSDMLayers.jl` is a package to facilitate manipulation of geo-referenced raster data in *Julia*, specifically aimed at species distribution modeling. It is a higher-level implementation, building on `ArchGDAL.jl`, that is easier to use and provides support for common SDM operations (see *Feature Overview* section below). The package implements simple type structures to manipulate the input and output data of SDM models, and is meant to be a flexible framework on which to build more complex analyses. While it does not implement SDM models in itself, we believe the package is a step that will make *Julia* more popular for species distribution modeling, and lead to the development of more complete implementations. `SimpleSDMLayers.jl` also offers built-in access to some of the most common data sources in SDM studies, such as the WorldClim 2.1 climatic data, which is the most common source of climate data in SDM studies [@Booth2014BioFir]. The package is also tightly integrated with `GBIF.jl`, which allows easy access to the GBIF database, a common data source for species occurrences. Both `SimpleSDMLayers.jl` and `GBIF.jl` are part of the *EcoJulia* organization, whose aim is to integrate a variety of packages for ecological analyses in *Julia*.

Basic structure

The core structure implemented in `SimpleSDMLayers.jl` is the `SimpleSDMLayer` type, with two variants, `SimpleSDMPredictor` and `SimpleSDMResponse`, depending if the layer is meant to be mutable or not.

A `SimpleSDMLayer` element is made of a `grid` field, which contains the raster data as a simple `Array` (matrix) of any type, easily manipulable. It also contains the fields `left`, `right`, `bottom`, and `top`, representing the bounding coordinates of the layer.

To illustrate this structure, the following code loads a layer of WorldClim 2.1

climate data, which also shows how easily this can be done in a single call. By default, this will return a layer with the values for the whole world if no bounding coordinates are specified.

```
# Load package
using SimpleSDMLayers

# Get world temperature data
temperature = worldclim(1)
```

```
SimpleSDMLayers.SimpleSDMPredictor{Float32}(Union{Nothing,
↪  Float32}[-31.017
105f0 -31.62153f0 ... -32.81253f0 -31.620333f0; -30.391916f0
↪  -31.63478f0 ...
-32.81005f0 -30.995281f0; ... ; nothing nothing ... nothing
↪  nothing; nothing
nothing ... nothing nothing], -180.0, 180.0, -90.0, 90.0)
```

The raster values can be displayed by calling the `grid` field.

```
# Display data grid
temperature.grid
```

```
1080×2160 Array{Union{Nothing, Float32},2}:
 -31.0171    -31.6215    -31.6227    ...  -32.8129    -32.8125
 ↪  -31.6203
 -30.3919    -31.6348    -31.6341        -32.8092    -32.8101
 ↪  -30.9953
 -33.4822    -34.1494    -34.1493        -35.4658    -35.4633
 ↪  -34.1374
 -33.6104    -34.2875    -34.2865        -35.596     -35.5931
 ↪  -34.2528
 -33.7199    -34.4041    -34.4014        -35.6932    -35.691
 ↪  -34.3311
 -33.8224    -34.5184    -34.5162    ...  -35.8037    -35.7996
 ↪  -34.4165
 -31.6613    -32.3194    -32.3184        -33.5133    -33.5101
 ↪  -32.2032
 -31.7635    -32.4307    -33.7036        -34.9522    -33.6282
 ↪  -32.3038
 -33.7063    -36.0738    -39.2075        -40.6438    -37.3938
 ↪  -34.3026
 -33.9768    -34.7016    -35.8662        -37.2408    -36.0364
 ↪  -34.5988

    nothing     nothing     nothing         nothing     nothing
    ↪  nothing
```

```
nothing     nothing     nothing         nothing     nothing
↪  nothing
nothing     nothing     nothing         nothing     nothing
↪  nothing
nothing     nothing     nothing         nothing     nothing
↪  nothing
nothing     nothing     nothing   ...     nothing     nothing
↪  nothing
nothing     nothing     nothing         nothing     nothing
↪  nothing
nothing     nothing     nothing         nothing     nothing
↪  nothing
nothing     nothing     nothing         nothing     nothing
↪  nothing
nothing     nothing     nothing         nothing     nothing
↪  nothing
```

`SimpleSDMLayers.jl` then makes it very simple to plot and visualize the layer as a map using `Plots.jl` (Figure 1).

```
using Plots
plot(temperature)
```

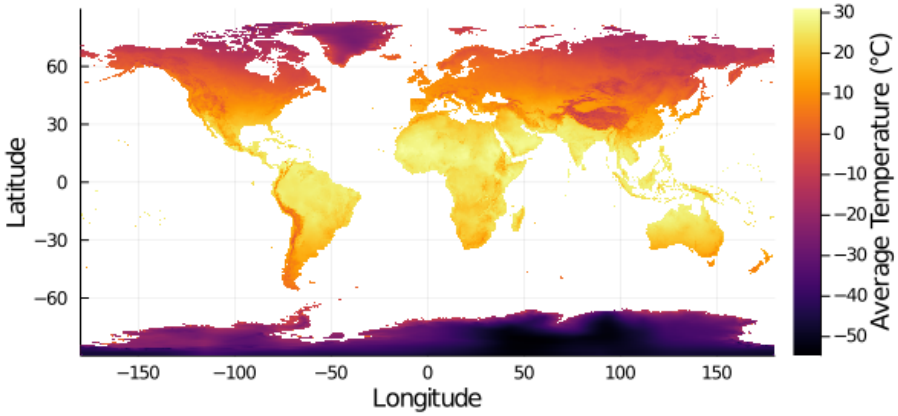

Figure 1: Map of the average annual temperature data from WorldClim 2.1, accessed as a layer through SimpleSDMLayers.jl

Feature overview

`SimpleSDMLayers.jl` implements the following features:

- **Overloads for common functions**: The `SimpleSDMLayer` types are implemented along with overloads for many common functions and operations, such as subsetting, changing values, copying, and iterating. Therefore, the layers and the raster values stored in the `grid` field can be manipulated as easily as any `Array`, without losing their spatial aspect.
- **Statistical operations on layer values**: Common operations can be performed directly on the layer values without worrying about the underlying structure (for example, sum, minimum, maximum, mean, median).
- **Statistical operations on multiple layers**: Operations can also be performed between layers to produce a new layer, just as `Arrays`, as long as they share the same coordinates. For instance, two layers can be added or subtracted, and calling `mean()` will produce a new layer with the mean value per pixel.
- **Spatial operations**: `SimpleSDMLayers.jl` implements spatial operations such as clipping a layer to given coordinates, coarsening the resolution by grouping values, and performing sliding window operations given a certain radius.
- **Datasets supported**: The package provides access to climate data at different resolutions from WorldClim 2.1 and CHELSA, as well as land cover data from EarthEnv. Custom raster data can be loaded as well.
- **Plotting recipes**: Default recipes are implemented for the `SimpleSDMLayer` types, allowing to directly map them, view the grid data as histograms and density plots, or compare layers as 2-dimensional histograms.
- **Integration with GBIF.jl (and DataFrames.jl)**: `SimpleSDMLayer.jl` is well integrated with `GBIF.jl`, allowing to clip layers based on the occurrence data, as well as to map occurrences by displaying them over the layers. Both packages also offer an integration with `DataFrames.jl` to easily convert environmental and occurrence data to a table format.

Examples

Spatial operations

To illustrate a few of the spatial operations supported by `SimpleSDMLayers.jl`, the following code reuses the previous `temperature` layer, and shows how it is possible to : 1) clip the layer to a region of interest (Europe for instance); 2) coarsen the resolution by averaging groups of cells for large scale analyses; and 3) perform sliding window operations to aggregate values for each site based on a certain radius. Each of these operations can be performed in a single command and returns new layers, which can then be plotted as shown previously.

```
using Statistics
# Clip to Europe
temperature_europe = temperature[left = -11.2, right = 30.6,
↪  bottom = 29.1, top = 71.0]
# Coarsen resolution
temperature_coarse = coarsen(temperature_europe, Statistics.mean,
↪  (4, 4))
# Sliding window averaging
temperature_slided = slidingwindow(temperature_europe,
↪  Statistics.mean, 100.0)
```

GBIF integration

The following example shows how the integration between `SimpleSDMLayers.jl` and `GBIF.jl` allows to easily map the occurrences of any species in GBIF. The species represented in this example is the belted kingfisher (*Megaceryle alcyon*).

`GBIF.jl` first allows us to retrieve the latest occurrences from the GBIF database. Note that the element returned here uses the `GBIFRecords` format, which contains the metadata associated to each GBIF occurrence.

```
using GBIF
kingfisher = GBIF.taxon("Megaceryle alcyon", strict=true)
kf_occurrences = occurrences(kingfisher)
# Get at least 200 occurrences
while length(kf_occurrences) < 200
    occurrences!(kf_occurrences)
    @info "$(length(kf_occurrences)) occurrences"
end
kf_occurrences
```

```
GBIF records: downloaded 200 out of 100000
```

`SimpleSDMLayers.jl` then provides a simple integration between the occurrence data and the environmental layers. The layers can be clipped to the spatial extent of the occurrences in a single call using the `clip()` function. The occurrences' coordinates can also be extracted with `longitudes()` and `latitudes()`. Using these functions, we can easily create a map of the occurrences by overlaying them on top of the clipped environmental layer (Figure 2).

```
# Clip layer to occurrences
temperature_clip = clip(temperature, kf_occurrences)

# Plot occurrences
contour(temperature_clip, fill = true)
scatter!(longitudes(kf_occurrences), latitudes(kf_occurrences))
```

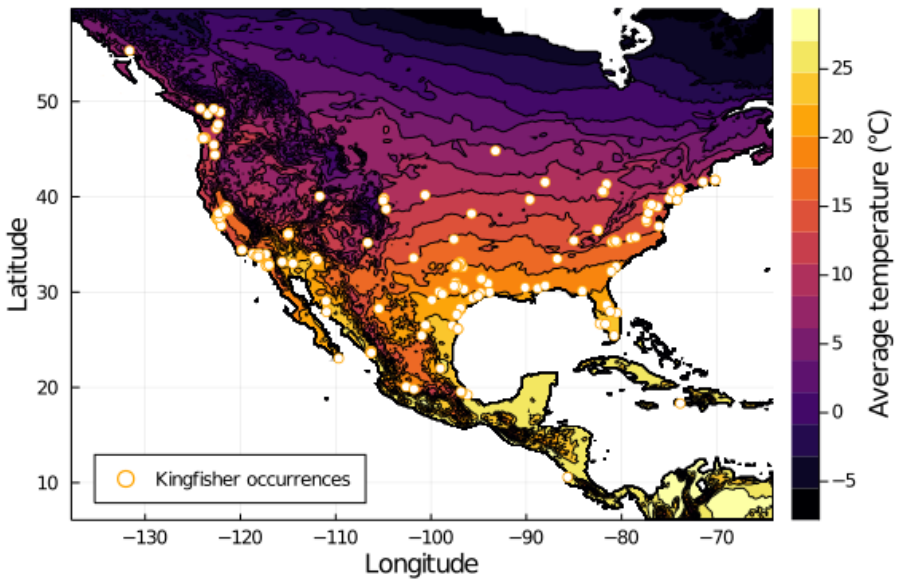

Figure 2: Latest belted kingfisher occurrences from the GBIF database displayed over the temperature data through the integration between SimpleSDMLayers.jl and GBIF.jl

Acknowledgements

We would like to thank all contributors to the *EcoJulia* organization for their help in developing this series of packages for ecological research. Funding was provided by Fonds de recherche du Québec - Nature et technologies (FRQNT) and the Computational Biodiversity Science and Services (BIOS^2) training program.