NeutralLandscapes.jl: a library for efficient generation of neutral landscapes with temporal change

Michael D. Catchen [1,2]

[1] McGill University; [2] Québec Centre for Biodiversity Sciences

Correspondance to:
Michael D. Catchen — michael.catchen@mail.mcgill.ca

Soon to be a paper, maybe. TK authors, MKB,VB,RS,TP

Keywords:
landscape ecology
spatial ecology
neutral landscapes
simulation

1 Introduction

Neutral landscapes are increasingly used in ecological and evolutionary studies to provide a null expectation spatial variation of a given measurement. Originally developed to simulate the spatially autocorrelated data (**Gardner1987NeuMod?**; **Milne1992SpaAgg?**), the have seen use in a wide range of disciplines: from landscape genetics (**Storfer2007PutLan?**), to landscape and spatial ecology (**Tinker2004HisRan?**; **Remmel2013CatCla?**), and biogeography (**Albert2017BarDis?**).

The two primary packages used to simulate neutral landscapes are `NLMR` in (the `R` language) (**Sciaini2018NlmLan?**) and `NLMpy` (in Python; **Etherington2015NlmPyt?**). We present `NeutralLandscapes.jl`, a package in `Julia` for neutral landscapes which is faster than both above package. Here we demonstrate that `NeutralLandscapes.jl`, depending on the method, is orders of magnitude faster than previous neutral landscape packages. As biodiversity science becomes increasingly concerned with temporal change and its consequences, its clear there is a gap in methodology in generating neutral landscapes that change over time. In addition we present a novel method for generating landscape change with prescribed levels of spatial and temporal autocorrelation, which is implemented in `NeutralLandscapes.jl`

2 Software Overview

This software can generate neutral landscapes using several methods, enables masking and works with other julia packages.

fig. 1 shows a replica of Figure 1 from (**Etherington2015NlmPyt?**), which shows the capacity of the library to generate different types of neutral landscapes, and then apply masks and categorical classifcation to them.

2.1. Interoperability Ease of use with other julia packages

Mask of neutral variable masked across quebec in 3 lines.

```
using NeutralLandscapes
using SimpleSDMLayers
```

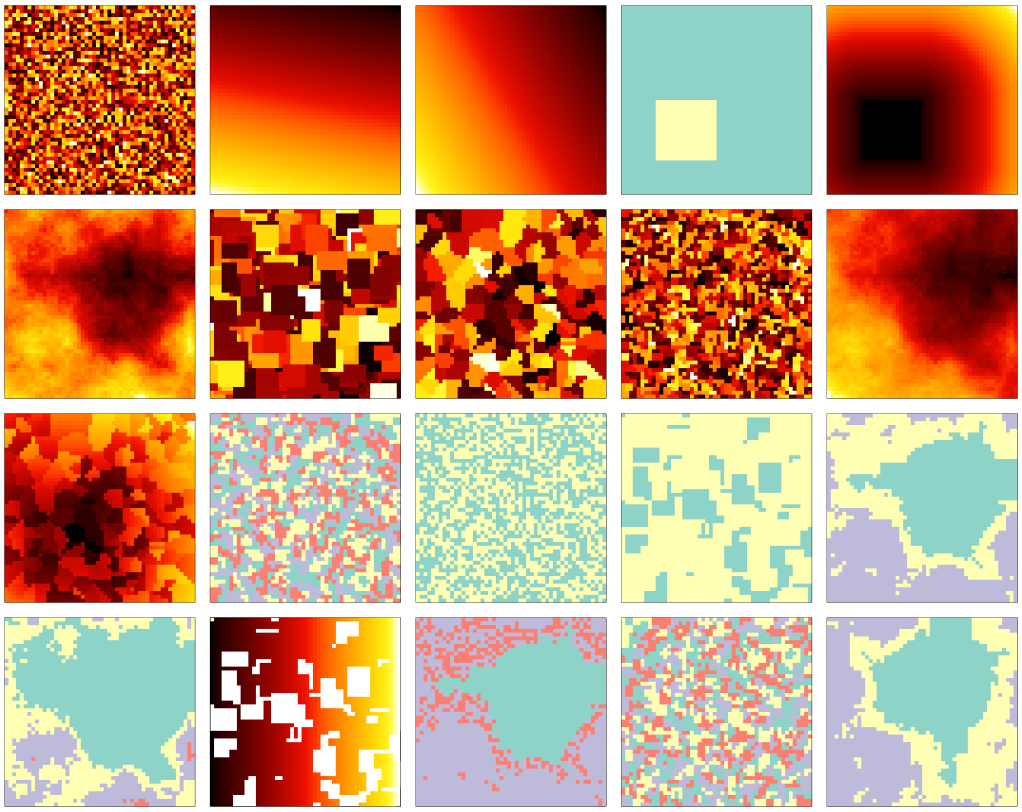

Figure 1 Recreation of the figure in `nlmpy` paper and the source, supplied in less than 40 lines of code.

```
quebec = SimpleSDMPredictor(WorldClim, BioClim; left=-90., right=-50., top=75., bottom=40.)
qcmask = fill(true, size(quebec))
qcmask[findall(isnothing, quebec.grid)] .= false

pltsettings = (cbar=:none, frame=:box)

plot(
    heatmap(rand(MidpointDisplacement(0.8), size(layer), mask=qcmask); pltsettings),
    heatmap(rand(PlanarGradient(), size(layer), mask=qcmask); pltsettings),
    heatmap(rand(PerlinNoise((4,4)), size(layer), mask=qcmask); pltsettings),
    heatmap(rand(NearestNeighborCluster(0.5), size(layer), mask=qcmask); pltsettings),
    dpi=400
)

savefig("interoperable.png")
```

3

Benchmark comparison to `nlmpy` and `NLMR`

It's fast. As the scale and resolution of raster data increases, neutral models must be able to scale to match those data dimensions. Here we provide two benchmark tests. First a comparison of the speed variety of methods from each `NeutralLandscapes.jl`, `NLMR`, and `nlmpy`. Second we compare these performance of each of these software packages as rasters become larger. We show that `Julia` even outperforms the `NLMR` via C++ implemention of a particularly slow neutral landscape method (midpoint displacement).

4

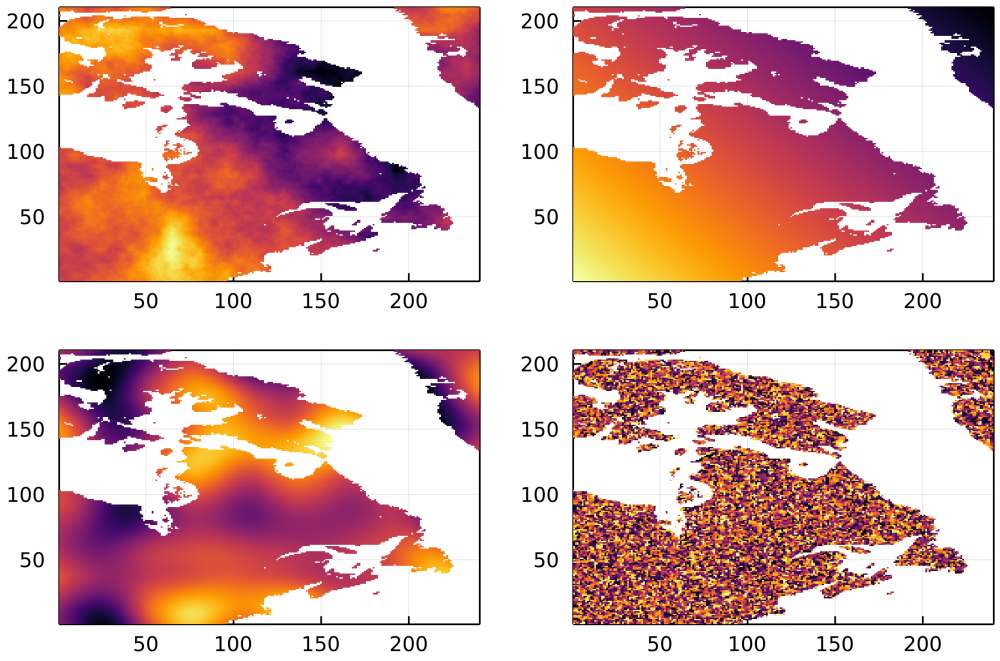

Figure 2 todo

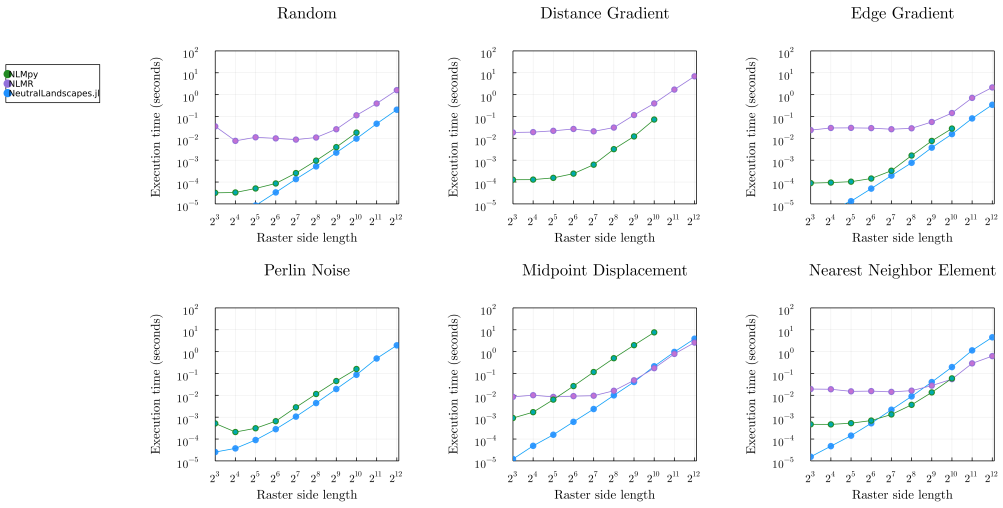

Figure 3 todo

Generating dynamic neutral landscapes

We implement methods for generating change that are temporally autocorrelated, spatially autocorrelated, or both.

$M_t = M_{t-1} + f(M(t-1))$

4.1. Models of change

4.1.1 Directional

4.1.2 ***Temporally autocorrelation*** r: rate, v: variability, U matrix of draws from standard Normal$(0, 1)$

$f_T(M_{ij}) = r + vU_{ij}$

4.1.3 ***Spatial autocorrelation*** r: rate, v: variability, $[Z(\delta)]_{ij}$: the (i, j) entry of the zscore of the δ matrix

$f_S(M_{ij}) = r + v \cdot [Z(\delta)]_{ij}$

4.1.4 ***Spatiotemporal autocorrelation*** $f_{ST}(M_{ij}) = r + v \cdot [Z(\delta)]_{ij}$

4.2. Rescaling to mimic real data

5 Discussion

6 References