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

Last revision: *January 7, 2022*

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

Introduction

Neutral landscapes are increasingly used in ecological and evolutionary studies to provide a null expectation of the variance of a given metric over space.

Wide range of disciplines: from landscape genetics [], to spatial ecology [], and biogeography [].

As biodiversity science becomes increasingly concerned with temporal change and its consequences, its clear there is a gap generating neutral landscapes that change over time. In this ms we present how `NeutralLandscapes.jl` is orders of magnitudes faster than packages `nlmpy` (in python) or `NLMR` (in R). In addition we then present a novel method for generating landscape change with prescribed levels of spatial and temporal autocorrelation.

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 (**nlmpycite?**), which shows the capacity of the library to generate different types of neutral landscapes, and then apply masks and categorical classifcation to them.

Table of methods.

Model	nlmpy?	NLMR	Description	References	Aliases
No gradient	x	x	Each cell is drawn randomly		
Planar gradient	x	x	A gradient from low to high at a given angle		
Distance gradient	x	x	Each cell is the distance between that cell and a location		
Random rectuangular cluster	x	x	Covers the plane in random rectanges until covered		
Random element nearest-neighbor	x	x*	Discrete categories based on distance a set of `n` random points		`nlm_mosaictess`, k-means

Model	nlmpy?	NLMR	Description	References	Aliases
Random cluster nearest-neighbor	x	x	Starts with n seed points and grows clusters probabilistically		
Random curds		x			
Gaussian Field		x			
Diamond-square			Improvement on Diamond-square and fractal brownian motion.		
Perlin noise	x		Method for noise with “smoother” features than DS/MPD		
Mosaic random field		x			

[Figure 1 about here.]

What methods have been called different things but are actually the same thing?

Interoperability

Ease of use with other julia packages

Mask of neutral variable masked across quebec in 3 lines.

```
using NeutralLandscapes
using SimpleSDMLayers

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")
```

[Figure 2 about here.]

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).

[Figure 3 about here.]

Generating dynamic neutral landscapes

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

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

Models of change

Directional

Temporally autocorrelation

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

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

Spatial autocorrelation

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

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

Spatiotemporal autocorrelation

Rescaling to mimic real data

Discussion

References

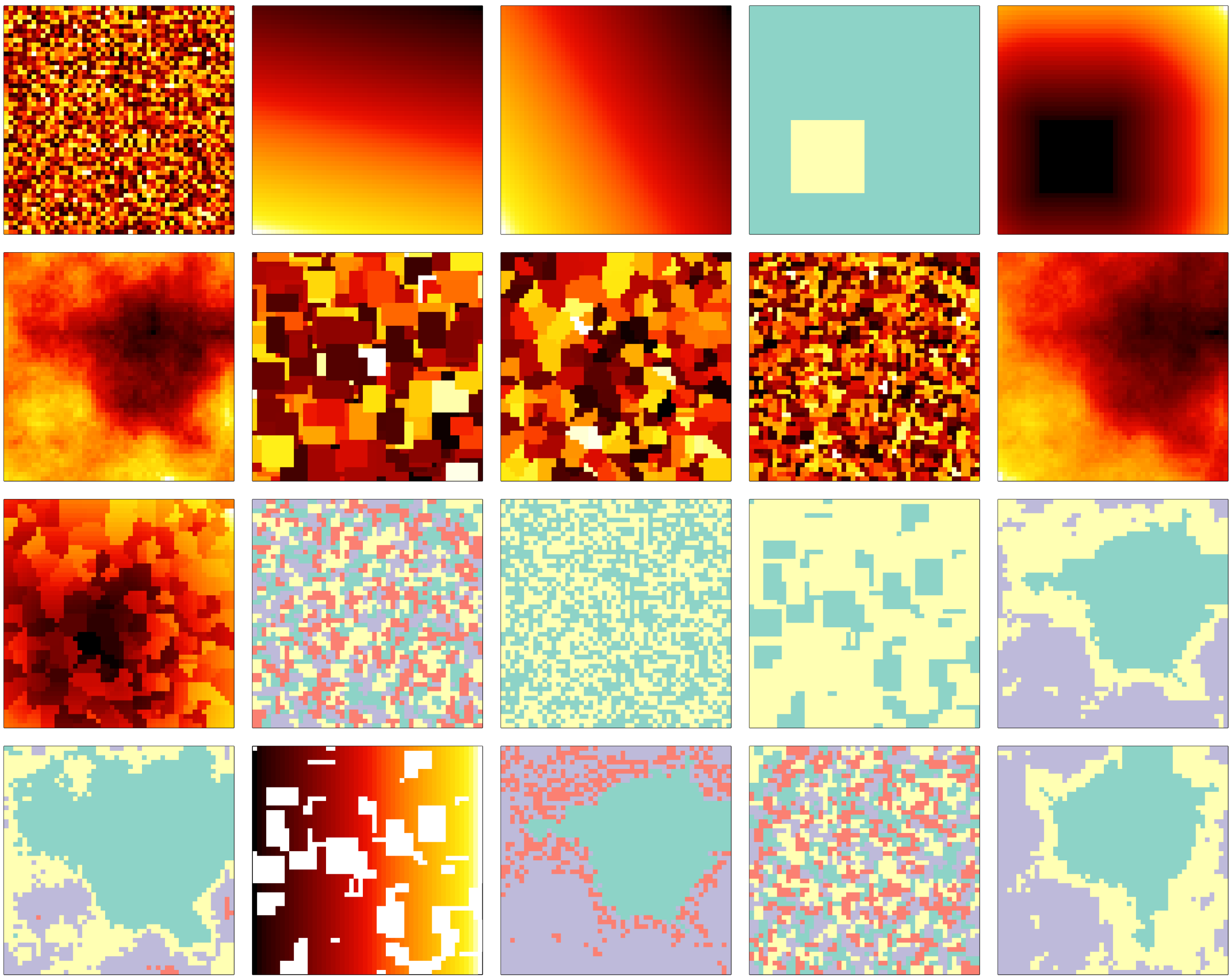

Figure 1: Recreation of the figure in `nlmpy` paper and the source, supplied in less than 40 lines of code.

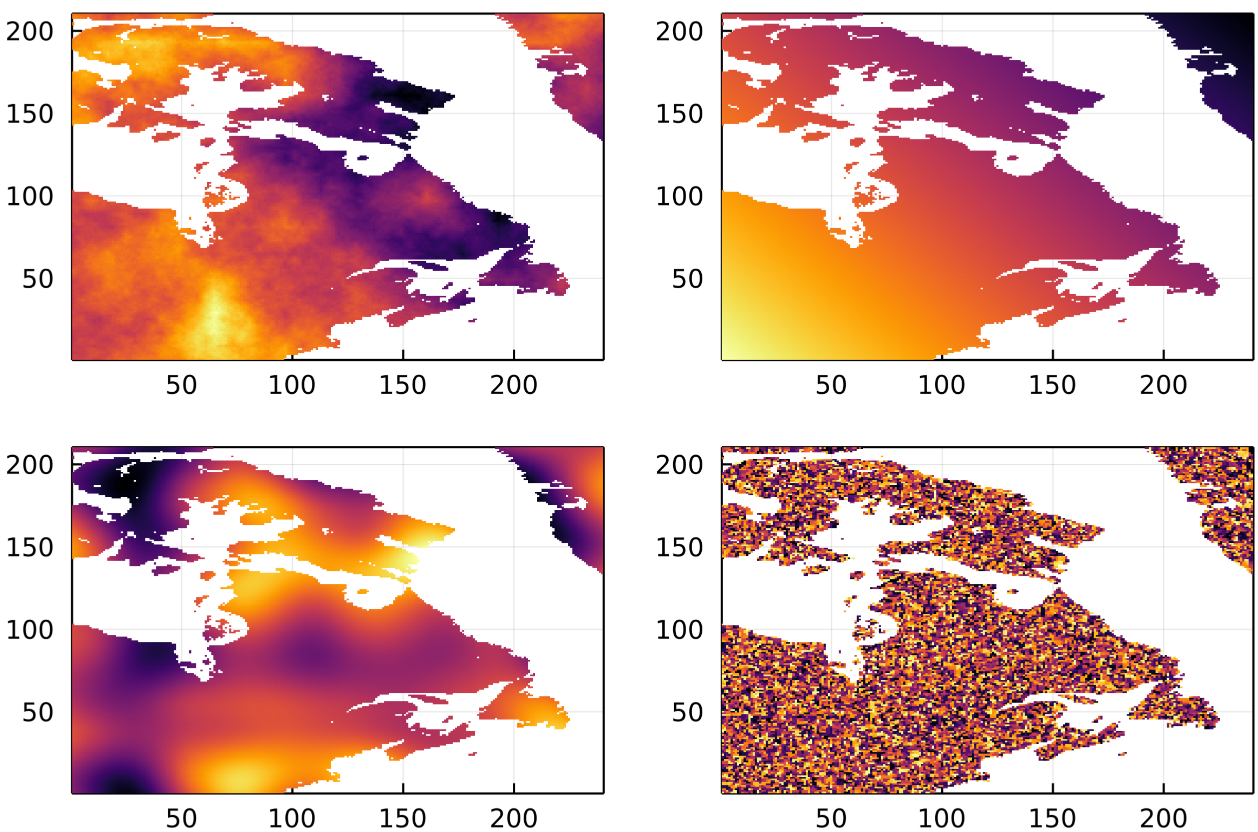

Figure 2: todo

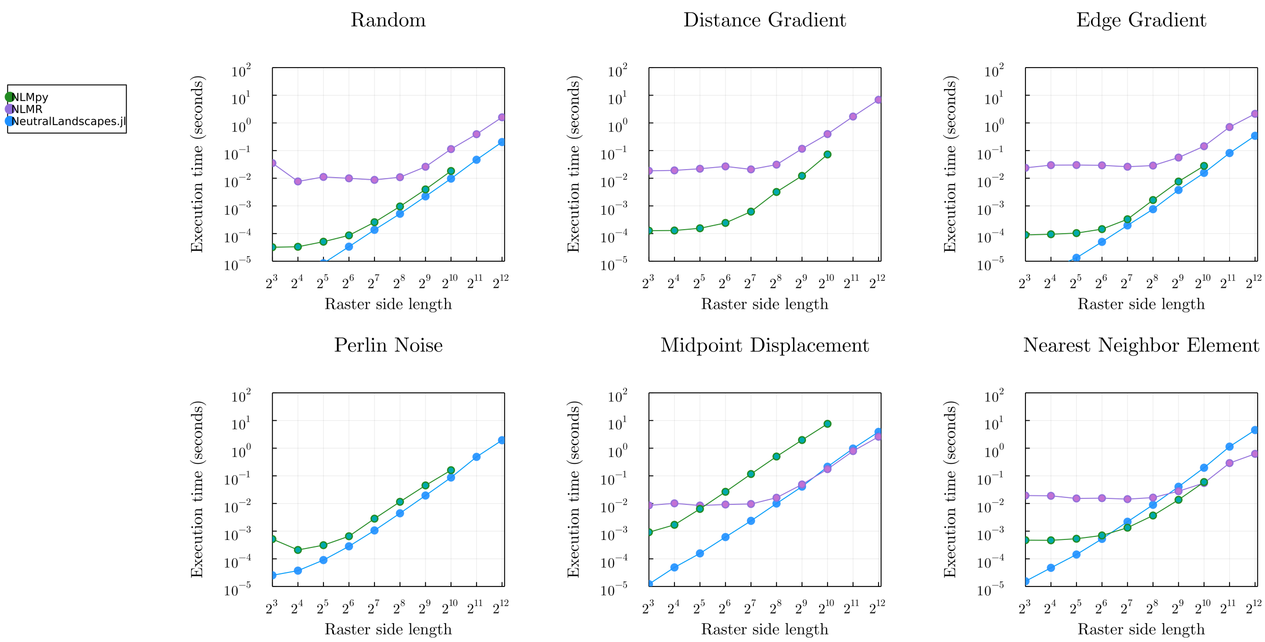

Figure 3: todo