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 2, 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	Reference	Aliases
No gradient	x	x			
Planar gradient	x	x			
Distance gradient	x	x			
Random rectuangular cluster	x	x			
Random element nearest-neighbor	x	x*			`nlm_mosaictess`, k-means
Random cluster nearest-neighbor	x	x			
Random curds		x			
Gaussian Field		x			
Perlin noise	x				
Diamond-square					

Model	nlmpy?	NLMR	Description	Reference	Aliases
Mosaic random field		x			

In

[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.

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).

Fig 2: Benchmark comparison of selected methods in each of the three languages

In fig 2 we should a selection of neutral landscape generators (random, edge gradient, perlin noise, distance-gradient)

MPD comparison

Why use this particular generator as the comparison? It's slow. So slow that NLMR implements it in C++. (NLMR implements both MPD, neighbor, randrect, and random neighboorhood in c++). Still these three algorithms, which cosinsts of 3/16 of NLMR's alg implementations, constitute 33% of its codebase.

In this section we show our implementation of MPD is faster than NLMR's C++ MPD scales until the asyptotic limit imposed by the $O(n^2)$ scaling of the raster

[Figure 2 about here.]

Generating dynamic neutral landscapes

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

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

Discussion

References

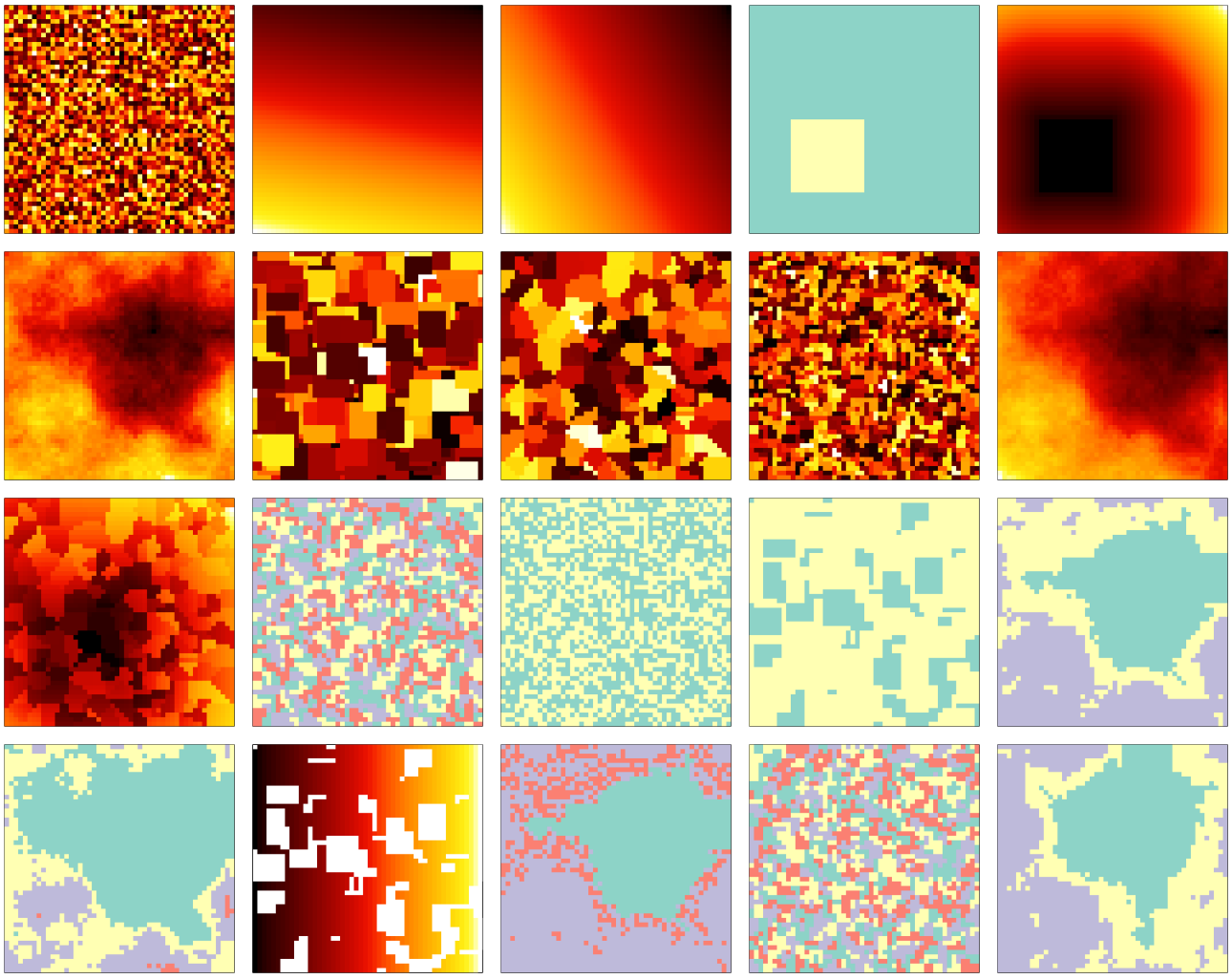

Figure 1: Recreation of the figure in `nlmpy` paper and the source, supplied in less than 40 lines of code.

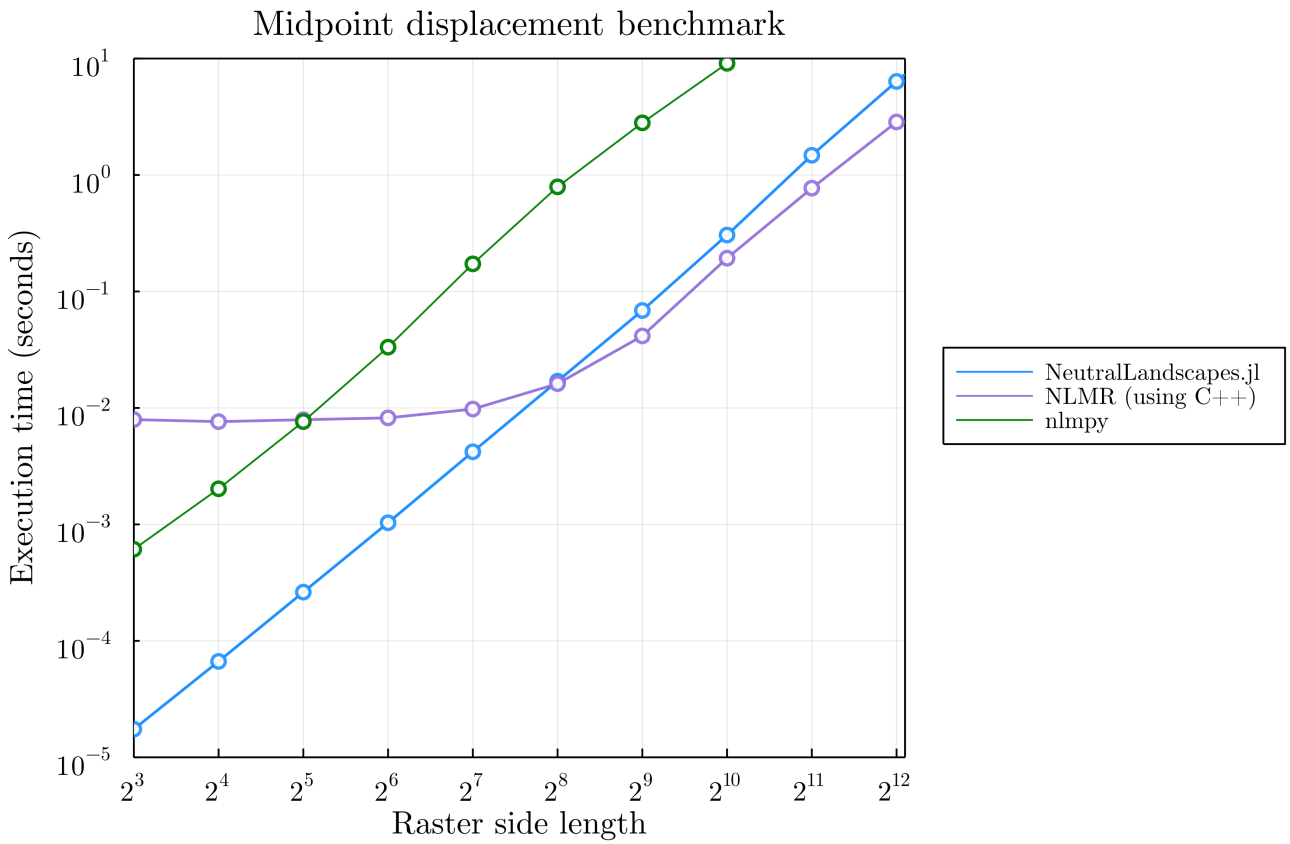

Figure 2: Comparison of speed of generating a midpoint displacement neutral landscape (y-axis) against raster size (measured as length of the size of a square raster, x-axis)