

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

This work is released by its authors under a CC-BY 4.0 license



Last revision: *January 2, 2022*

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

1 Introduction

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

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

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

10 Software Overview

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

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

15 Table of methods.

16 In

17 [Figure 1 about here.]

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

19 Interoperability

20 Ease of use with other julia packages

21 Mask of neutral variable masked across quebec in 3 lines.

22 **Benchmark comparison to nlmpy and NLMR**

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

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

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

31 **MPD comparison**

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

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

37 [Figure 2 about here.]

38 **Generating dynamic neutral landscapes**

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

41
$$M_t = f(M_{t-1})$$

42 **Discussion**

43 **References**

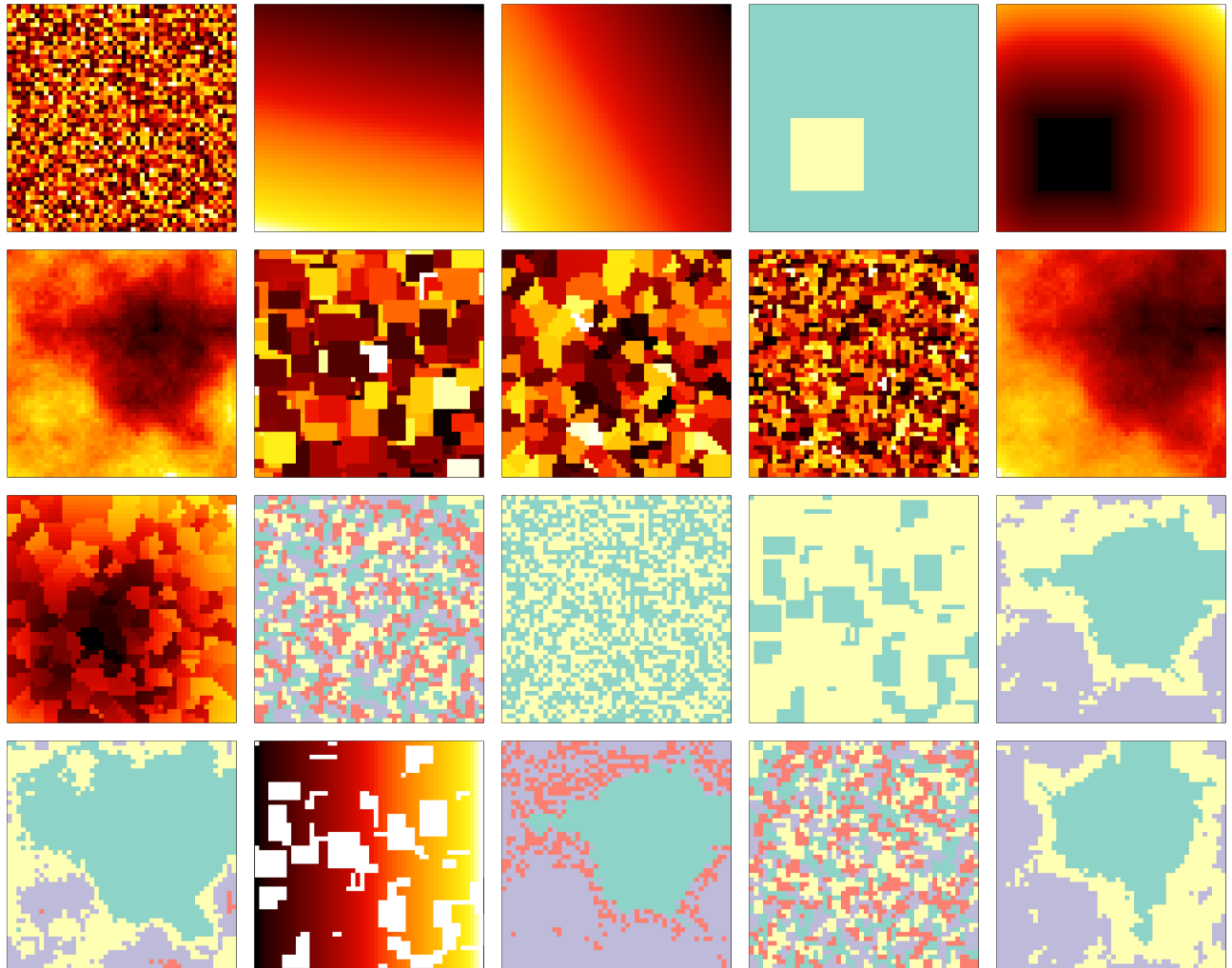


Figure 1: Recreation of the figure in n1mpy paper and the source, supplied in less than 40 lines of code.

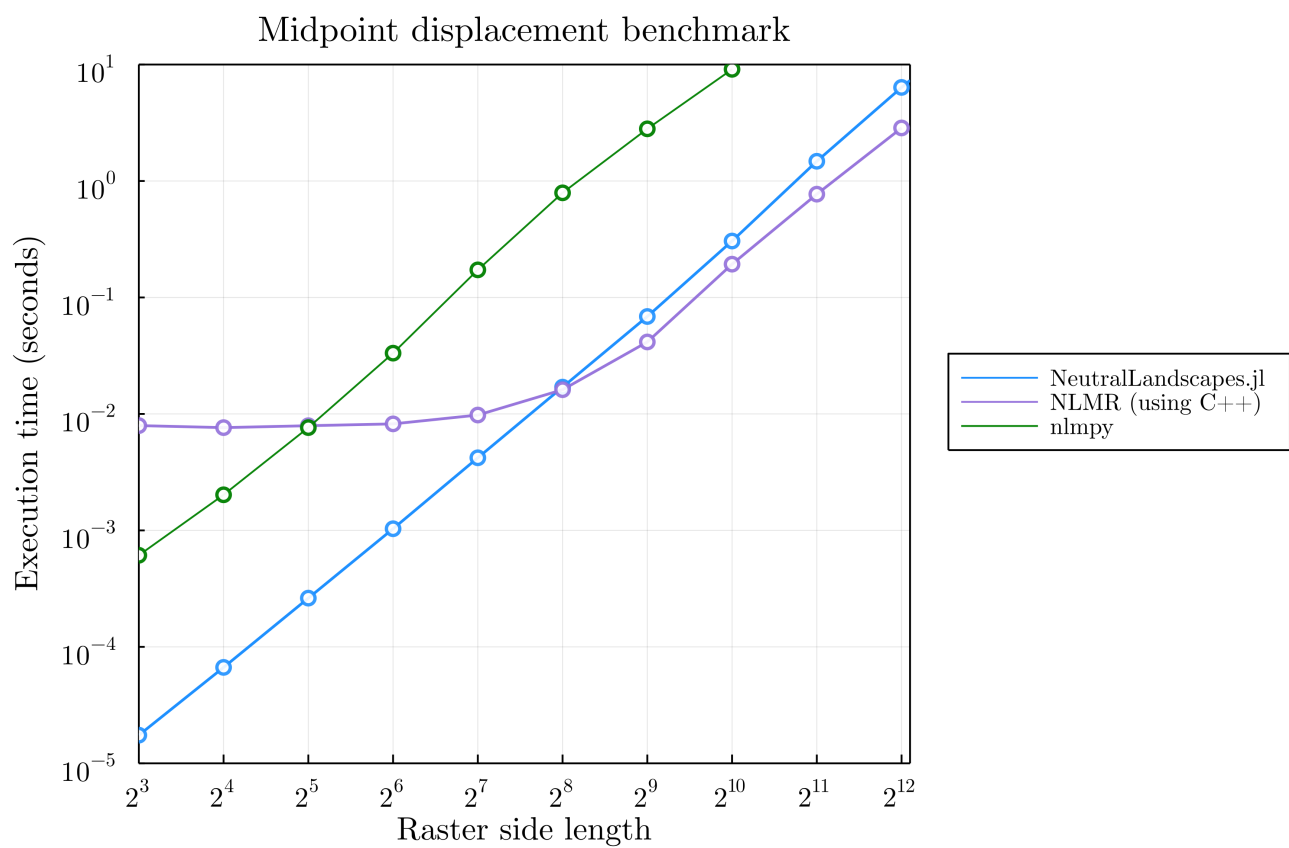


Figure 2: Comparison of speed of generating a midpoint displacement neutral landscape (y-axis) against raster size (measured as length of the size of a square raster, x-axis)