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 1, 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: landscape genetics to 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). We then present a novel method for generating landscape change with prescribed levels of spatial and temporal autocorrelation, and demonstrate that it works

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?**)

Table of methods.

[Figure 1 about here.]

```
using NeutralLandscapes, Plots
siz = 50, 50

Fig1a = rand(NoGradient(), siz) # Random NLM
Fig1b = rand(PlanarGradient(), siz) # Planar gradient NLM
Fig1c = rand(EdgeGradient(), siz) # Edge gradient NLM
Fig1d = falses(siz)
Fig1d[10:25, 10:25] .= true # Mask example
Fig1e = rand(DistanceGradient(findall(vec(Fig1d))), siz) # Mask example
Fig1f = rand(MidpointDisplacement(0.75), siz) # Mask example
```

```
Fig1g = rand(RectangularCluster(4, 8), siz)
Fig1h = rand(NearestNeighborElement(200), siz)
Fig1i = rand(NearestNeighborCluster(0.4), siz)
Fig1j = blend([Fig1f, Fig1c])
Fig1k = blend(Fig1h, Fig1e, 1.5)
Fig1l = classify(Fig1i, ones(4))
Fig1m = classify(Fig1a, [1-0.5, 0.5])
Fig1n = classify(Fig1g, [1-0.75, 0.75])
Fig1o = classify(Fig1f, ones(3))
Fig1p = classify(Fig1f, ones(3), Fig1d)
Fig1q = rand(PlanarGradient(90), siz, mask = Fig1n .== 2) #TODO mask as keyword + should mask be matrix or
Fig1r = ifelse.(Fig1o .== 2, Fig1m .+ 2, Fig1o)
Fig1s = rotr90(Fig1l)
Fig1t = Fig1o'

class = cgrad(:Set3_4, 4, categorical = true)
c2, c3, c4 = class[1:2], class[1:3], class[1:4]

gr(color = :fire, ticks = false, framestyle = :box, dpi=500, colorbar = false)
plot(
   heatmap(Fig1a),      heatmap(Fig1b),      heatmap(Fig1c),      heatmap(Fig1d, c = c2), heatmap(Fig1e
   heatmap(Fig1f),      heatmap(Fig1g),      heatmap(Fig1h),      heatmap(Fig1i),      heatmap(Fig1j)
   heatmap(Fig1k),      heatmap(Fig1l, c = c4), heatmap(Fig1m, c = c2), heatmap(Fig1n, c = c2), heatmap(F
   heatmap(Fig1p, c = c4), heatmap(Fig1q),      heatmap(Fig1r, c = c4), heatmap(Fig1s, c = c4), heatmap(F
     layout = (4,5), size = (1600, 1270)
)

savefig("./figures/figure1.png", )
```

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

Fig 3 scale comparison

[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

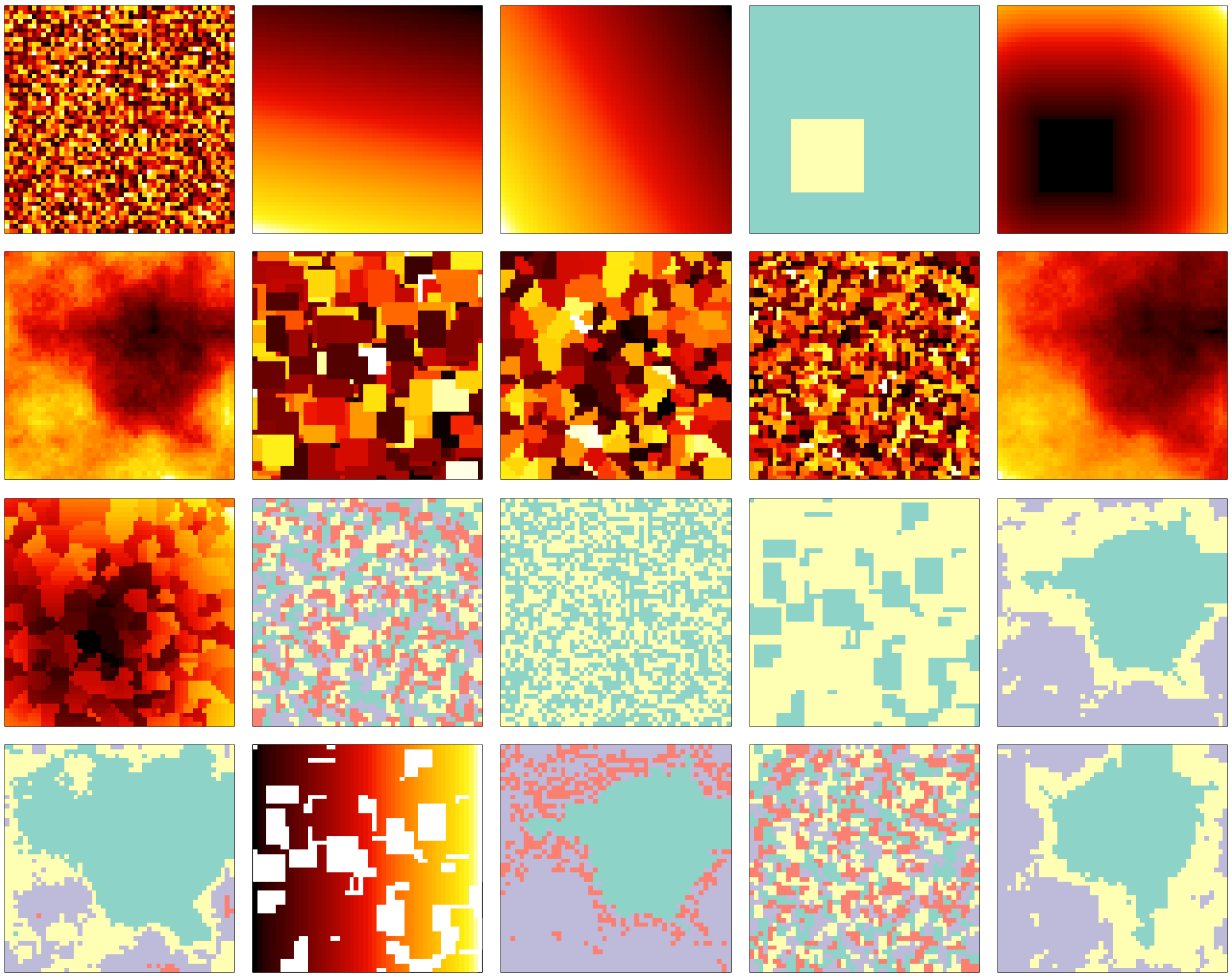

Figure 1: Recreation of the figure in `nlmpy` paper and the source

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)