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: *August 19, 2022*

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

Introduction

- what are neutral landscapes

Neutral landscapes are increasingly used in ecological and evolutionary studies to provide a null expectation spatial variation of some measurement.

Different properties of landscapes (elevation, land cover, temperature) vary across space in non-random ways. Tobler's law of geography: "everything is related to everything else, but near things are more related than distant things"

Originally based around methods for simulating spatially auto-correlated data (Gardner *et al.* 1987; Milne 1992).

they have seen use in a wide range of fields in ecology and evolution: from landscape genetics (Storfer *et al.* 2007), to spatial ecology (Tinker *et al.* 2004; Remmel & Fortin 2013), and biogeography (Albert *et al.* 2017).

The two most popular libraries used to generate neutral landscapes are `NLMR` in (the `R` language) (Sciaini *et al.* 2018) and `NLMpy` (in Python; Etherington *et al.* 2015). Here we present `NeutralLandscapes.jl`, a new package for neutral landscape simulation in the `Julia` language. So, why create another package?

Here we demonstrate that `NeutralLandscapes.jl` is orders of magnitude faster than previous neutral landscape packages.

In addition, `NeutralLandscapes.jl` implements several novel methods for simulating environmental change with temporal variation. 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. Our model allows users to simulate time-series of any `NeutralLandscape` layer, and which can produce an arbitrary distribution of change across every spatial cell, with provided 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 Etherington *et al.* (2015), which shows the capacity of the library to generate different types of neutral landscapes, and then apply masks and categorical classifcation to them.

[Figure 1 about here.]

Further, `NL.jl` provides methods for interacting with other `julia` packages, and functions for rescaling

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))                   # ------ TODO
qcmask[findall(isnothing, quebec.grid)] .= false   # should both of these lines be possible only using mas

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
)
```

[Figure 2 about here.]

Rescaling to mimic real data

Figure: Real temp (left) / Rescaled NL (right) , same unit bar

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

Two types of temporal change: (1) null change, where the is random variation in each cell across time but the mean value across all cells stays constant (with some variation around this constant due to randomness in change generation)

Null

Take an arbitrary distribution (from `Distributions.jl`) and set its mean value to 0, and apply draws from that distribution to each cell at each timestep.

Directional

Can take an arbitrary distribution of values and set its expected-value to be the primary input into a change model—the mean amount of change at each timestep. This can also be parameterized to a be a variable list of mean change at each corresponding timestep.

Temporally autocorrelation We generate temporally autocorrelated change using the method. We take an arbitrary distribution A

r: rate, υ: variability, U matrix of draws from standard Normal$(0, 1)$. Here υ replects the amount of temporal autocorrelation.

The value of a given cell (i, j) with value M_{ij}

$$M_{ij}(t+1) = f_T(M_{ij}(t)) = r + \upsilon A_{ij}$$

Results in an expected value of change of r per timestep with variance υ.

Spatial autocorrelation Generate a matrix δ with a NL generator.

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

Z-score is arbitrary and can be replaced with any dist.

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

Spatiotemporal autocorrelation Finally, to implement change this is both spatially and temporally autocorrelated

$f_{ST}(M_{ij}) = r + \upsilon \cdot [Z(\delta)]_{ij}$

[Figure 3 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.

[Figure 4 about here.]

How many lines of code, and what language is that code in for each pkg? `NLMR` contains 893 lines of R, and 376+51 lines of C++. `nlmpy` contains 386 lines of python. Julia contains 664 lines of (non-test) `julia`. Note these numbers refer only to lines of code and not comments.

Example: fitting a neutral landscape to an empirical spatial dataset using generative modeling

Here we use approximate bayesian computation to estimate the parameter of autocorrelation H for an empirical raster of temperature data.

Why? What if we are interested in differentiating the processes that occur in this *real* landscape versus landscapes with *similar statistical properties*.

We take a raster of mean temp around the st lawrence lowlands in QC, and use ABC to estimate the value of H under the midpoint-displacement model.

We use the variogram as the loss function

[Figure 5 about here.]

Discussion

Why is it good that we've made this a faster thing to do? Why are models of temporal change necessary? What can simulation do for spatial ecology more generally?

What are questions we can address with NL.jl that was not possible before?

References

Albert, J.S., Schoolmaster, D.R., JR., Tagliacollo, V. & Duke-Sylvester, S.M. (2017). Barrier Displacement on a Neutral Landscape: Toward a Theory of Continental Biogeography. *Systematic Biology*, 66, 167–182.

Etherington, T.R., Holland, E.P. & O'Sullivan, D. (2015). NLMpy: A python software package for the creation of neutral landscape models within a general numerical framework. *Methods in Ecology and Evolution*, 6, 164–168.

Gardner, R.H., Milne, B.T., Turnei, M.G. & O'Neill, R.V. (1987). Neutral models for the analysis of broad-scale landscape pattern. *Landscape Ecology*, 1, 19–28.

Milne, B.T. (1992). Spatial Aggregation and Neutral Models in Fractal Landscapes. *The American Naturalist*, 139, 32–57.

Remmel, T.K. & Fortin, M.-J. (2013). Categorical, class-focused map patterns: Characterization and comparison. *Landscape Ecology*, 28, 1587–1599.

Sciaini, M., Fritsch, M., Scherer, C. & Simpkins, C.E. (2018). NLMR and landscapetools: An integrated environment for simulating and modifying neutral landscape models in R. *Methods in Ecology and Evolution*, 9, 2240–2248.

Storfer, A., Murphy, M.A., Evans, J.S., Goldberg, C.S., Robinson, S., Spear, S.F., *et al.* (2007). Putting the "landscape" in landscape genetics. *Heredity*, 98, 128–142.

Tinker, D., Romme, W.H. & Despain, D. (2004). Historic range of variability in landscape structure in subalpine forests of the Greater Yellowstone Area, USA. *Landscape Ecology*.

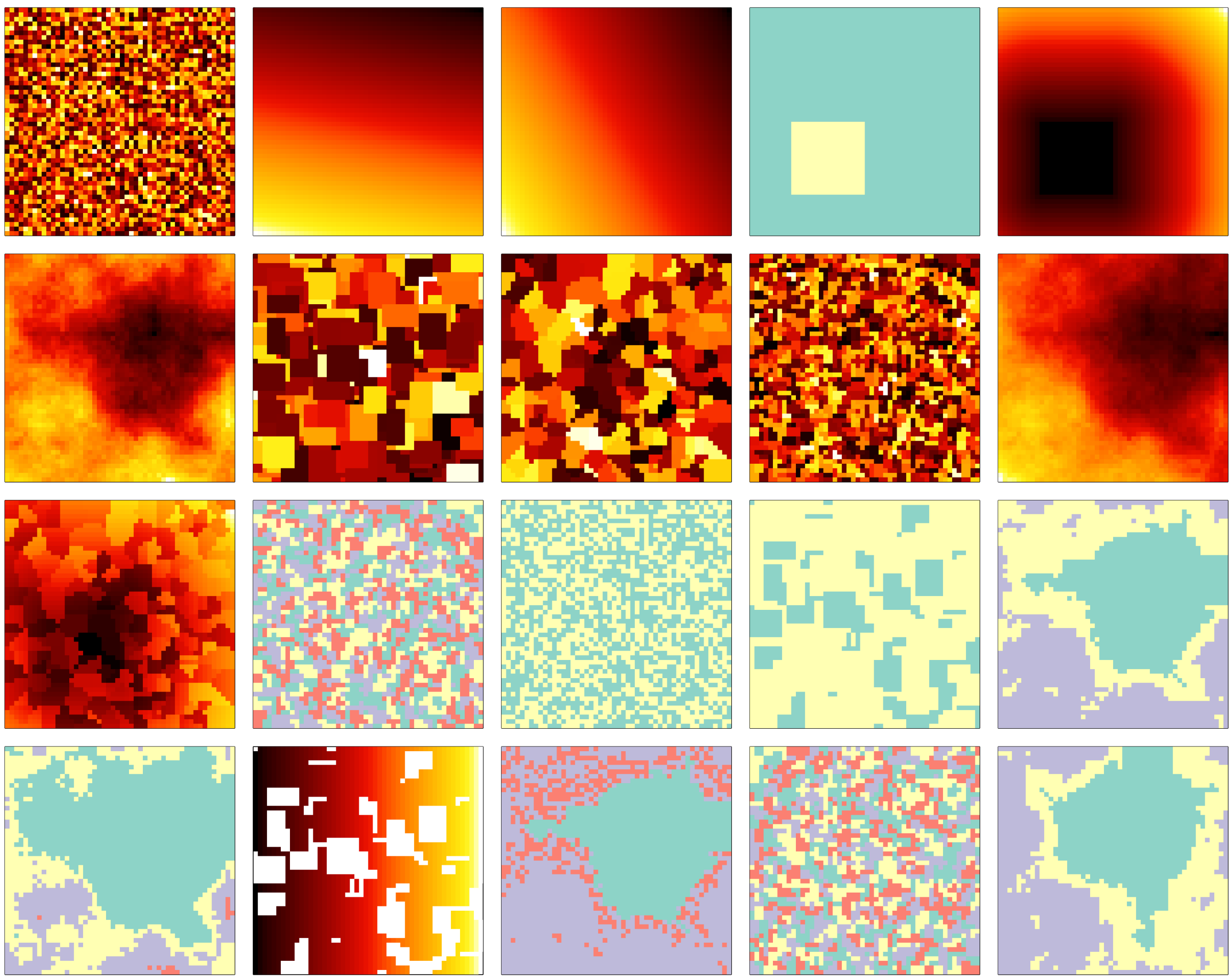

Figure 1: Recreation of the figure in `nlmpy` paper and the source, supplied in less than 40 lines of code.

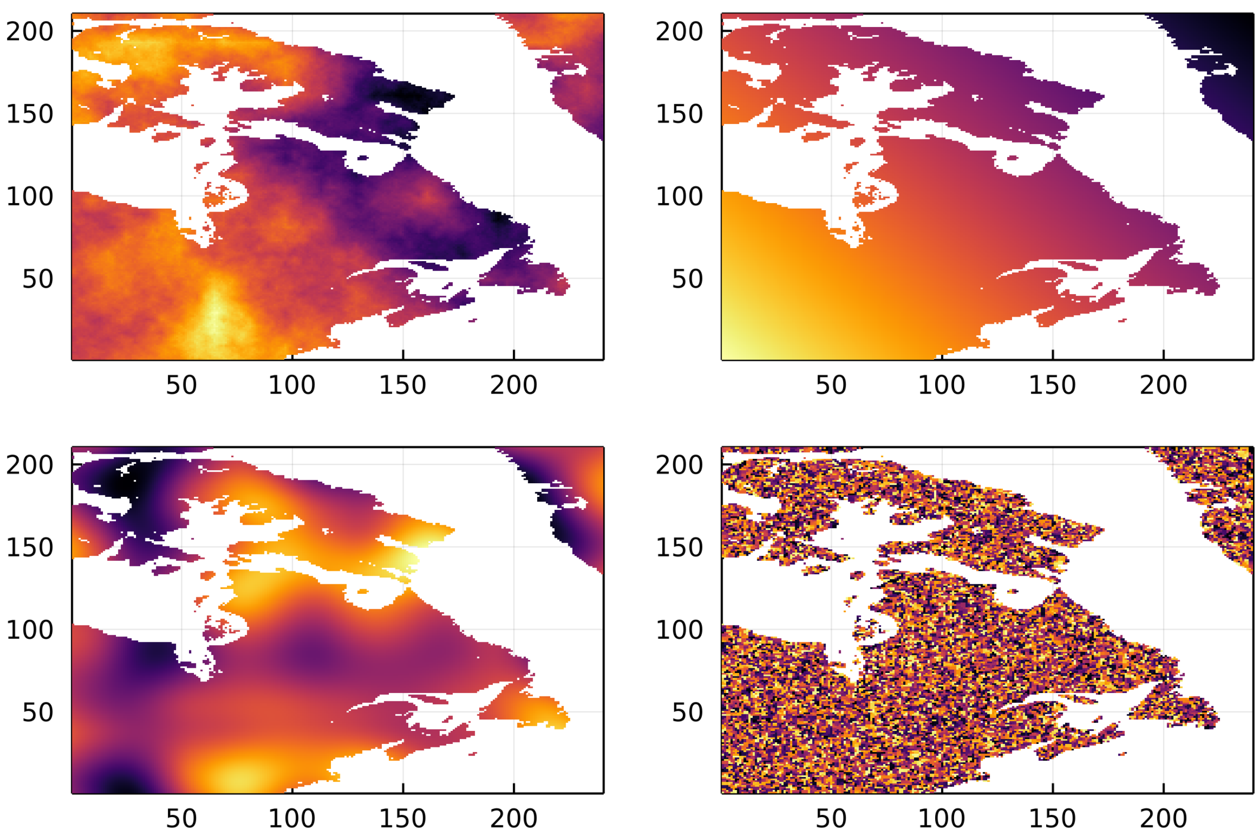

Figure 2: todo

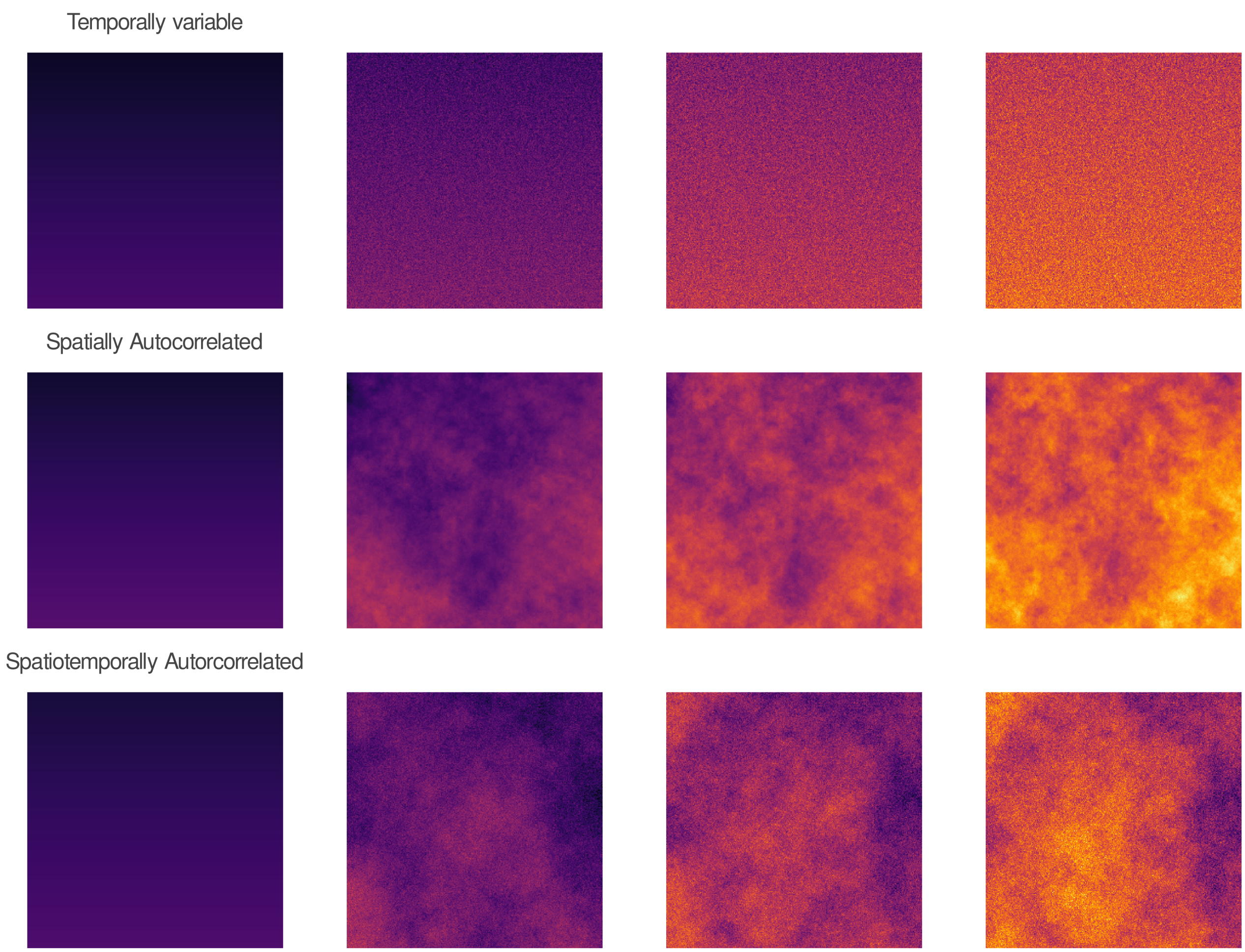

Figure 3: todo

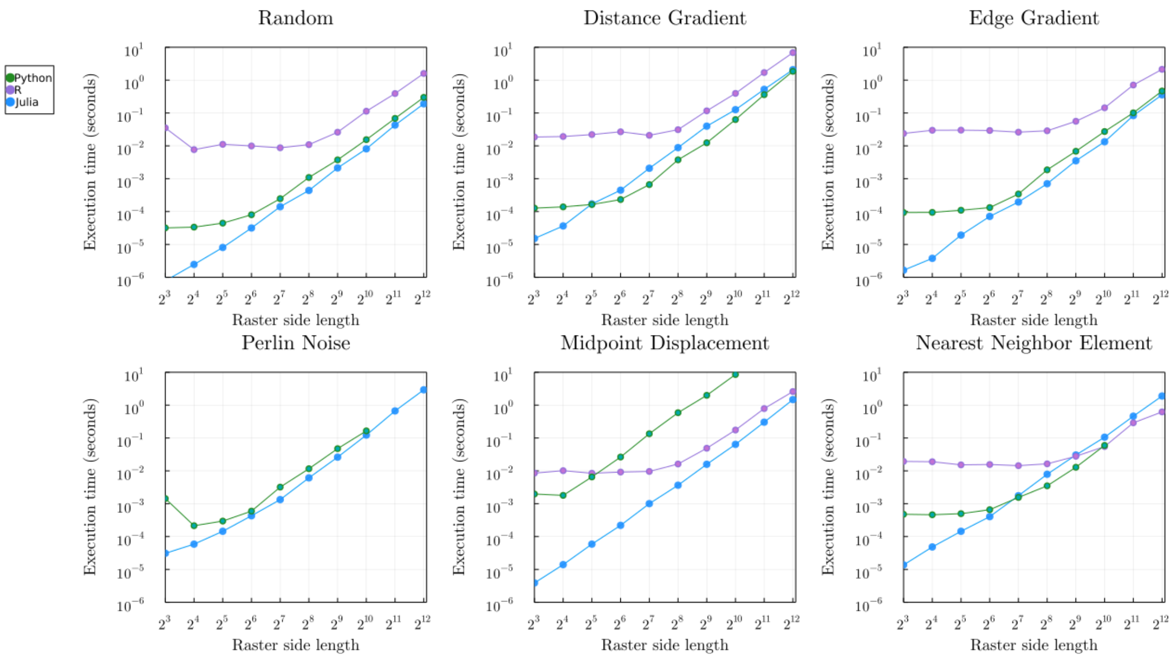

Figure 4: todo

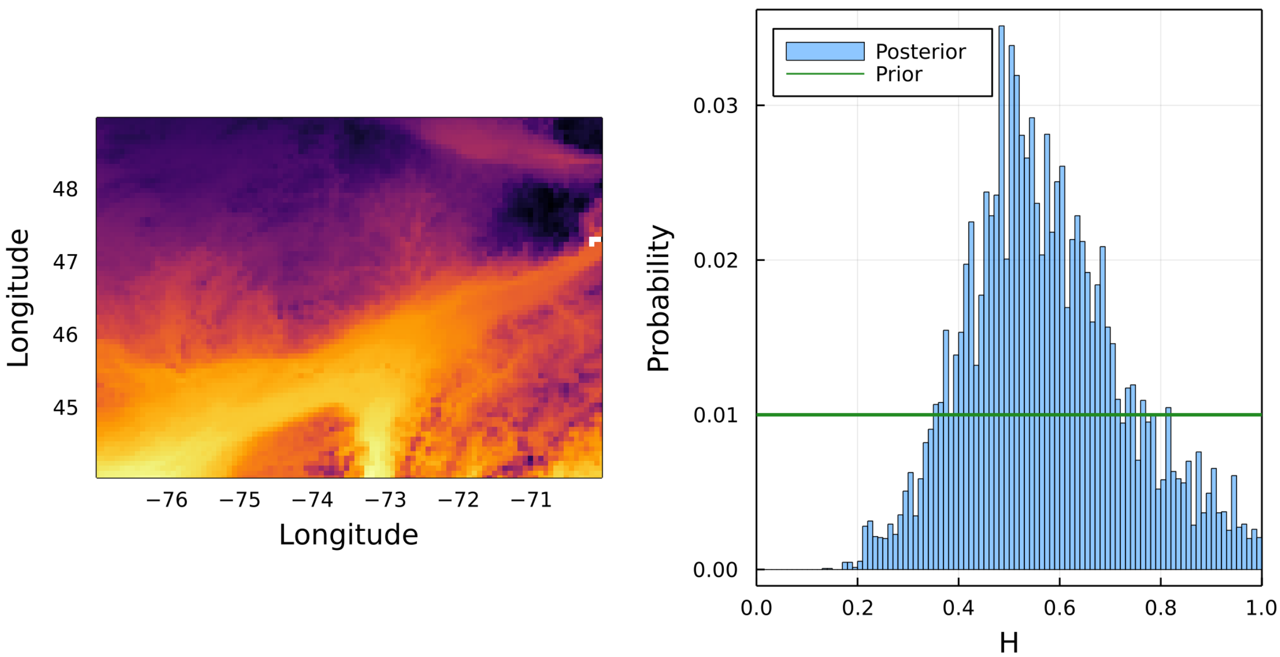

Figure 5: todo