Understanding the spatial variation of species interactions

1 Introduction to the research problem

Interactions between species support most of ecosystem services and properties. In the recent years, community ecologists have recognized the importance of variation in species interactions over time and space. Although this variation has been adequately described, there is very little understanding of how different ecological and biogeographic mechanisms interact to generate it. This is due, in part, to the lack of conceptual and theoretical models, and in part to the lack of an appropriate methodology to analyse the already existing datasets. We will use this allocation to solve this second point, through the development, test, and analyse of permutational null models. These new stastistical tools will allow to understand why species interactions vary over large (continental/worldwide) scales, by identifying the ecological and biogeographical mechanisms involved and quantifying their contributions. These tools will (i) generate new knowledge about the macroecological structure of species interactions and (ii) contribute to the growth of the numerical toolkit available to ecologists wanting to tackle big-data problems.

2 Research justification

Research problem

Species interactions are the backbone of all ecosystems. Recent empirical and theoretical results show that from knowledge of species interactions, one can predict ecosystem functions, potential for coexistence, and resilience to invasion. Yet ecologists have often neglected the fact that species interactions vary over time, and space, despite many well described examples of this; juvenile fishes may be preys of larger species, that they will later consume when they have grown enough; bacteria can have no effect on healthy hosts, but become pathogenic when the host immunity is compromised; pollinators may neglect some flower genus entirely when other, prefered, ones are present. In all ecological communities that have been studied for variation in species interactions so far, important variation in species interactions has been found. This calls for the investigation of two questions: *why* and *how* do interactions vary. In practice, this amount to (i) determining whether this variability is signal or noise, and (ii) understanding the mechanisms involved.

The research projects for which I submit this application will focus on the following question: *how does the spatial variation of species presence/absence interacts with the spatial variation of species interactions to determine local community structure.* Some of my previous research shows that (i) there is no relationship between the drivers of species distribution and of interactions distribution, and (ii) some ecological systems vary at higher rates than other as a function of the identity of species present. Empirical work – either the collection of new data, or the re-analysis of existing datasets – has been hampered by the fact that there is currently no statistical framework to adress these questions; while it is possible to *describe* the variation of species interactions, *testing* how significant it is from chance expectation cannot be done for lack of proper methodology.

Methodology

The usual methodology applied to questions of this family relies on null models – models that generate a permutation or a randomization of the data under a set of pre-determined hypotheses represented as permutational constraints. For example, drawing species at random to maintain the smae richness in each location, or drawing species at random to maintain the number of sites occupied by each species. These (pseudo)-random data are then compared to the empirical data, to assess the probability that empirical data would have been generated by chance.

Using spatially or temporally replicated species interaction networks, we will first implement efficient species distribution and species interaction null models. Some of my previous research suggested that (i) species interaction networks are intrinsically hard (computationaly) to randomize, requiring to develop efficient permutation strategies, and (ii) there is a long tail of un-commonly observed / un-commonly interacting species, requiring exhaustive sampling of the parameter space to properly capture. In addition, because the observation of species interaction networks (locally) is the result of several ecological processes (joint absence of species, then interaction), there is a need to implement different scenarios for analysis.

Although the use of null models is typically associated to a frequentist statistics approach (the proportion of randomized data with more structure than the empirical data is a proxy for the p-value), we will use elements of Approximate Bayesian Computation. This approach will allow use to determine the posterior distribution of which *model* fits the data the best, and allow us to make use of prior information in the prior probabilities of models choice.

Timeline

From January to March - conception and implementation of null models (no to minimal resource use)

From March to April - scaling tests and code optimization for parallel architecture (medium resource use)

From June to August - first runs on the complete dataset (high resource use)

September - Analysis of the results, adaptation of the statistical models and re-adjustemnt of the priors (no anticipated resource use)

From October to November - second runs on the complete dataset (high resource use)

December - data analysis and preparation of the manuscript (no anticipated resource use)

Specific goals

1. Develop a series of null models to understand the ecological mechanisms acting on the spatial structure of species interaction networks
2. Use ABC to allow model selection and evaluate the contribution of different sets of mechanisms to the data structure
3. Write a software note detailing our approach, with use cases
4. Train one MSc student into applications of HPC for large datasets in community ecology and biogeography

3 Technical justification

The problem we tackle is by design embarassingly parallel: each replicate of each model on each dataset is a single process. As such, we are less concerned with continued access to a moderate number of cores, than we are with short-term access to a large number of cores on which to launch several thousands of jobs.

3.1 Compute request

3.1.1 Code details

As with most emerging problems in ecology, there is no ready-made code that we can deploy. Everything we use is written in-house. To achieve maximal performance, we will use the Julia programming language (see next section). All code produced by the group will be released under the MIT license.

The code will be co-developped by a MSc student and I. I have extensive experience in developping software for the analysis of ecological networks, with a particular focus on code efficiency for large datasets (millions of interations). In terms of training, I will let the student write the code, review it, and iterate until we have a finished product. Julia is conducive to this approach – it offers C-like performance with python-like syntax, so even students with minimal programming abilities can rapidly be productive and deliver code with good runtime and resource usage.

Our code will perform the following tasks:

1. Take a spatially aggregated dataset, most likely as an annotated edge list (`location;node1;node2`
2. These data will be splitted into different blocks (*e.g.* either within `location`, withing values of `node1`, within `node1;node2` unique pairs, etc) in accordance with the ecological hypotheses.
3. Within these blocks, the data will be shuffled.
4. The reconstructed dataset will undego a series of checks: all interactions must be unique locally, the statistical properties of the original networks must be respected, etc.
5. Randomized datasets that pass the check will be accepted.

Steps 3-5 will be repeated until 10^4 replicates of each 10^1 models will have been generated for each 10^3 networks. Some of my preliminary tests revealed that

1. Using permutations on the annotated edge list decreased the number of rejected samples when compared to the usual successive Bernoulli trials of probabilisitc graphs.
2. Generating each replicate takes on average 2 seconds.

Using the estimates of number of models, networks, and replicates given above, this project should consume a maximum of 34 core/years.

3.1.2 Code performance and utilization

Our code will be written at 90+% in Julia (performance-blocking parts will be re-written in C or machine-optimized assembler). Julia is architecture agnostic, and can spawn instance on multiple cores within a node, thus allowing us to leverage the Calcul Quebec parallel architecture.

As we use in-house code, we will rely extensively on continuous integration, code coverage analysis, and exhaustive unit-testing. We will conduct scaling efficiency tests before deploying the code on the Compute Canada machines. That being said, embarassingly parallel problems tend to scale embarassingly well, and I routinely achieve close to linear speedup for large datasets (smaller datasets scale less well as IO become limiting).

This project will allow me to train the student in best practices in software engineering – version control, test-driven development, continuous integration. These are skills that are almost never taught in university courses, despite being extremely valuable as soon as software is involved.

3.1.3 Size of request

I have access to the HPC cluster of the Universite du Quebec a Rimouski. It will be used for performance testing and optimization, but (as I am not a UQAR researcher), I cannot use it for production. Data will be formatted and prepared on the group's database server, and will be uploaded to the Compute Canada machines in a format that is the most efficient for consumption.

This allocation is needed so that this particularly demanding project will not eat through the default allocation, that will be use by visiting students and colleagues to do routine simulations and data analysis.

3.1.4 Impact of a cut

Minimal: the analyses will involve very short bursts of very intense resource use – if less resources are available, the bursts will be slightly longer, or distributed across several machines.

3.2 Storage request

None needed.

Data will be curated locally, and send to the cluster in pre-formated form. Datasets are typically less than a few MiB, and the output is usually around 1-5 GiB. Output files will be downloaded back to our machines for further treatment, and removed from the cluster.