Dear authors,

Thank you very much for your replies and the significant improvements you made upon the text. I most sincerely apologize for the delayed response, due to health issues.

I believe the article does not need to go through reviewers again; it is not far from a potential final state, which it can reach by a mild process of removal rather than addition (or it can undergo a longer process of gathering sufficient support for the claims that I am currently less convinced by)

- Addressing my main comment 1 below might simply require removing a few instances of mentioning temporal structure.

- Regarding my main comment 2, I would put it thus: while I feel that the central message that "structured IV is different" is interesting and comes across easily enough, the specifics of how it is different and why in your particular simulations are still a bit perplexing, and so I remain hesitant to trust certain particulars in your results and discussion.

Those particulars are probably not very relevant to your main message, so I think that just being a bit more prudent, and avoiding interpretations that are perhaps ecologically sensible but potentially wrong or misleading in the context of your simulations, would be enough to satisfy me.

Sincerely,
Matthieu Barbier

R: Dear recommender,

We thank you for your thorough review of our manuscript and your encouraging assessment. We answer below point by point to your remarks.

Looking forward to hearing from you,

Best,

The authors.

MAIN COMMENTS

1) There is one point I failed to explain properly in my previous comments: uIV, as you conceptualize it, should be effectively identical to sIV with many environmental factors changing very rapidly, so that each new individual that is born encounters a new environment unrelated to those that existed before.

Therefore, it cannot always be true that your claims regarding sIV hold even when the environment can change in time, or can be transposed from spatial structure to temporal

structure without further investigation. Constancy in time is likely an important ingredient for your results.

Indeed, implementing simulations with zero uIV but a temporally changing environment (+ abundance-dependent fecundity) was the most robust way I got diversity drops as large as yours in my attempt to replicate your results, see below.

R: We agree with the recommender that transposing our results obtained with a spatial structure to a temporal structure would require further investigation. As a consequence, we removed any reference to time in the Methods and Results sections and in the first two subsections of the Discussion. We also added a sentence in the Methods section (l. 76-78) to clarify that we assumed a constant environment:

“We here assumed that the environmental variables do not vary in time and therefore restricted our experiment to environmental variation in space.”

We moreover emphasised that we restricted our analyses to spatial environmental variation in the Abstract, the Introduction (l. 44, l. Fig. 1), and in the Discussion (l. 365, l. 419-420, l. 421-422, l. 431-432). We also already called for additional investigation focusing on the temporal variation (l. 436-439):

"Another source of environmental variation that was not tackled in this study is temporal variation. This variation is often structured, at different temporal scales (seasons, years, El Niño/La Niña events, \textit{etc.}) and this structure should be accounted for in models by expliciting those temporal scales after detection in the data."

Nonetheless, in the last subsection of the Discussion entitled “Accounting for a high-dimensional environment in community dynamics models”, which is a subsection of perspectives and does not directly discuss our results, we kept the reference to time. Indeed, in this part of the text, we think that it is interesting to underline that if the spatial variability of the environment is a source of observed intraspecific variability and offers the opportunity for many species to outperform the others at specific sites, thus allowing species coexistence, this should also apply to the temporal variability of the environment (see also Girard-Tercieux 2023).

Regarding the assertion that if environmental factors change very rapidly in time, sIV would be similar to uIV, we do not fully agree with the recommender. Although we agree that the environment would appear as random in this case, the filtering of the species by the environment would still apply and the final community would be a result of these changes in the environment. Again, we agree with the recommender that testing this hypothesis would require further investigation.

Regarding the simulation tests performed by the recommender, we really thank him for this effort, which we considered with great care and interest. As explained below, the model implemented in his code differs from ours in important aspects. As a consequence, we also conclude that the effect of temporal variation requires further investigation, despite his tests.

2) I attempted to reproduce the simulations (see Python code below, following most of your methods except for the autocorrelation in environmental factors which should be of no consequence) and encountered at least two issues which might indicate that your simulations are perhaps not doing exactly what you meant for them to do:

2.1) least importantly: in the case of uIV, modelling \hat epsilon as an unbounded random variable (e.g. a normally-distributed one) doesn't just create IV, it also creates a trend towards higher and higher performances far beyond anything in the reference model, since we systematically kill individuals with lower values and draw new individuals that only remain in the long term if their performance is at least equal to and typically higher than other existing individuals'.

Over time, regardless of the fraction of sIV, the performances of surviving individuals are thus less and less related to the environment, and stem more and more from random numbers (hat epsilon) becoming improbably rare and high. In my simulations I was able to observe an increase toward massive values of performance (e.g. a mean of 5, starting from a max of 1).
The more extreme aspects of this issue can be solved, of course, by bounding the random variable (e.g. using a uniform distribution between -1 and 1) -- then the passage of time will typically just lead to all surviving individuals being as close as possible to the ceiling imposed by the random variable.

R: We agree that the unbounded probability distribution of epsilon can lead to a progressive shift toward higher performance in average, and this should occur faster with increasing proportion of uIV (i.e. increasing variance of epsilon's distribution), as already mentioned (l. 192-194). Although, as suggested by the recommender, this effect could have been partially limited with a bounded distribution, we made this choice in order to mimic what previous studies did when investigating the role of IV in community dynamics, i.e. implementing it as a probability distribution, which is, in the vast majority of studies, an unbounded distribution (normal or others). This side effect is now better acknowledged in the text l. 137-139:

"Note that this contribution, which mimics what was typically made in previous studies, can lead to a trend of increasing average performance over iterations in simulations of community dynamics, especially due to the unbounded, normal distribution of δi,j,m."

Note that, while the variance of epsilon's distribution (Vj) is species-specific, and thus varies across species, we did not observe the case where one species (potentially the one with the highest variance) dominates the whole landscape in the end.

2.2) most importantly: I still don't understand exactly why you get extinctions (and especially as many as you do), and I don't know how much of it is *truly* about uIV.

Your Perfect Knowledge model allows many species to coexist even if there is a single environmental dimension -- it is enough that they have different optima and thus that each wins in different sites. Regardless of IV, there is no *deterministic* reason (such as "not

enough niche partitioning") for any species going extinct in your model, especially not due to decreasing the number of modelled environmental factors.

This leads to the suspicion that a lot of what you see in IK models is due to having the wrong functional form in Eq 2, ending up with an artefactual site-independent competitive hierarchy because different species get different intercepts beta_0,j and this becomes really impactful when n_obs is small compared to K (otherwise, why would adding uIV ever increase diversity?)

In my simulations, simply replacing sIV by uIV but without the model misspecification, I was unable to reproduce such large diversity drops for anything like the parameters mentioned in the main text and SI -- in particular, when I tried Fixed Fecundity, no species ever died out over such a large landscape with so little mortality and so many propagules. With 1% mortality, at most 6 individuals die each time step, while 200 propagules are released, so stochastic effects are pretty weak. In your setup I would expect every species to be best in ~30 sites out of 600, thus dying out with fixed fecundity and low uIV would basically require a species being absent from all these 30 sites at the same time while better species were present in all 10 other sites where it is sending its propagules.

For me, abundance-dependent fecundity was necessary for species to go extinct at all, and it had to be combined with totally random mortality to reach low species numbers (e.g. 5). Thus it's not impossible to get these numbers, but it requires having almost gone to a neutral model, and then the fraction of uIV vs sIV doesn't matter much.

Thus, while I am willing to agree with your general message, the quantitative details are perplexing and suggest a larger role of model misspecification than of sIV versus uIV (the importance of misspecification is potentially a valid point to make, but not one that really serves your message!), so I don't know exactly what is robust among the detailed analyses you provide in results and discussion.

I think it is fine to present these results as observations (= you decided on a certain simulation plan, it gives such and such results, that's a fact) but stay cautious about interpretations (e.g. in terms of niche partitioning and how it interacts with IV) and general statements.

R: We sincerely thank the recommender for the dedication he showed to the improvement of our paper by building this model. Based on his code below, we see two main differences with what we implemented.

First, environmental variables were normally distributed in our case (l. 73), contrary to what is implemented in the code below (*i.e.* a uniform distribution for environmental variables: E=np.random.uniform(0,1,(K,M))). As a result, species favourable habitats are not equally frequent in the landscape in our case (which is clearly stated l. 74-75 and l. 204-205), so we would not expect species to perform best in ~30 sites out of 600, as this average estimate hides lots of variability across species. This normal distribution actually results in a species hierarchy in habitat suitability at the landscape scale. Specifically, a few species can become extinct even in the perfect knowledge model, where there is no inference step, because some species have no site where they perform best (see e.g. l. 205-207). While this is also possible using a uniform distribution, it is much less likely.

Second, when we compared simulation outcomes with species' performances computed with the Perfect Knowledge and the Imperfect Knowledge models, differences can result from two main aspects: a) the model mis-specification (*i.e.* the use of a quadratic function with an intercept instead of a distance), and b) the number of observed dimensions. Changes in the number of observed dimensions actually result in a change in the proportion of uIV, but also in the number of variables that can be used to compute the site-dependent (or environment-dependent) part of the performance. This is now clearly stated l. 143-149:

"Note that, when comparing simulation outcomes with performance computed with the Perfect Knowledge and the Imperfect Knowledge models, differences can actually result from two main aspects: (i) the model mis-specification (i.e. the use of a quadratic function including an intercept, Eq.2, Eq.3, Eq.4, instead of a distance, Eq.1), and (ii) the number of observed dimensions. Change in the latter actually results in a change in the proportion of uIV, but also in the number of variables that can be used to compute the site-dependent (or environment-dependent) part of performance."

The way performance is computed in the code by the recommender below (calc_perf) does not reflect the fact that an increase in uIV should occur simultaneously with a decrease in the number of dimensions used (the site-dependent part in the recommender's code is computed as if we knew all the dimensions, but multiplied by (1-uIV)).

We acknowledge however that our comparison of the Imperfect Knowledge model with 15-dimensions and the Perfect Knowledge model was only a partial test of the effect of model misspecification, which could vary as the number of observed dimensions (n_obs) decreases. The effect of model misspecification can be more thoroughly assessed by comparing simulations with the two different model shapes and various n_obs < K. A rapid (i.e. with less replicates and environment (E)*species (S) configurations than what was performed in the manuscript) implementation of this approach is presented below, building on the Python code provided by the recommender.

More specifically, we compared four models. In each of these models:
i) Performance is not computed using a quadratic function as in the manuscript's Imperfect Knowledge models, but rather using a distance to the species optima on the observed dimensions, as in the Perfect Knowledge.
ii) The species intercept and variance are computed using the unobserved dimensions. To do so, for each species, we computed the mean and variance of residuals, residuals being the difference between the actual performance and the performance computed with observed dimensions on each site only (both being computed as a distance to the species' optima; see code below).
iii) Environmental values are drawn in either a uniform distribution [0,1] as in the recommender's code or a normal distribution as in the manuscript, and species optima are drawn in a uniform distribution [0,1].

The four models are as follows:

- **m1:** performance only depends on observed dimensions, without species intercept nor uIV.

- **m2:** performance depends on observed dimensions to which an uIV is added, but without species intercept.
- **m3:** performance depends on observed dimensions to which a species intercept is added, but without uIV (this model corresponds to the Imperfect knowledge model without uIV in our manuscript).
- **m4:** performance depends on observed dimensions to which a species intercept and an uIV are added (this model corresponds to the Imperfect knowledge model with uIV in our manuscript).

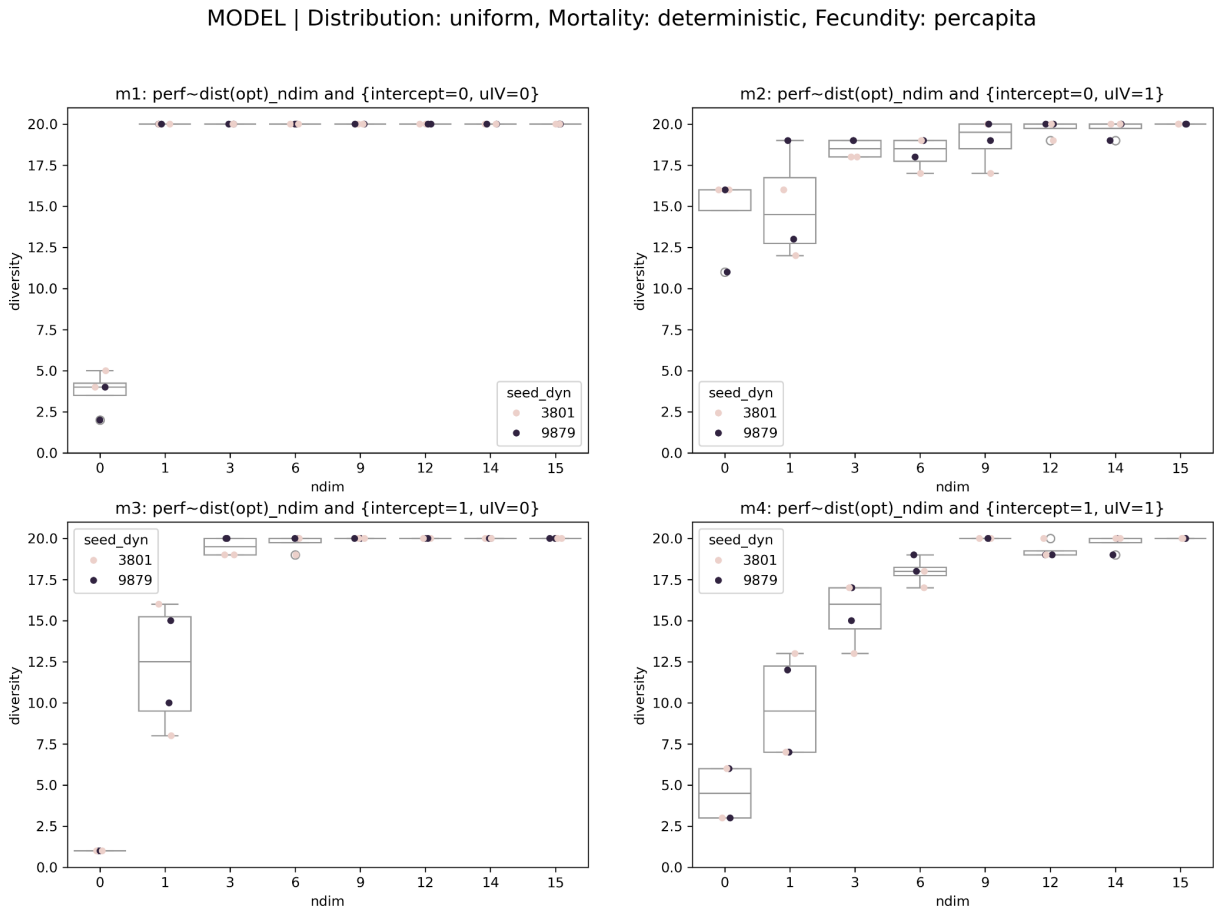

Figure R1: Model comparison with a uniform distribution for environmental variables. As in the main text of the manuscript, mortality here is deterministic and fecundity is abundance-dependent. For each of the four models (m1, m2, m3, m4), species richness at the end of simulations is shown as a function of the number of observed dimensions (ndim, previously named n_obs). For each ndim, four simulations were made, with two different random seeds for demographic processes (seed_dyn) and two different random seeds for initial conditions (leading to four combinations of random chains for demographic processes and initial conditions).

Results with uniformly distributed environment variables are shown in Fig. R1. m1 with 0 observed dimensions is neutral and the low species richness is due to random drift. In contrast, with m2 with 0 observed dimensions, uIV is high (cf. Fig. 2 of the manuscript) and centred on 0, enabling high-performance individuals to “rescue” species. Comparing m1 with m3 on the one hand, and m2 with m4 on the other hand shows that using intercepts

significantly reduces the species richness only when the number of observed dimensions is low (< 3).
Comparing m3 and m4 is analogous to what we have done in our manuscript. Like in our main results, the effect of adding uIV on species richness is positive at low numbers of observed dimensions (ndim < 3), then becomes negative as the number of observed dimensions increases, then fades as uIV becomes smaller (cf. Fig. 2 of the manuscript). This effect is also visible when comparing m1 and m2 (positive effect when ndim = 0).

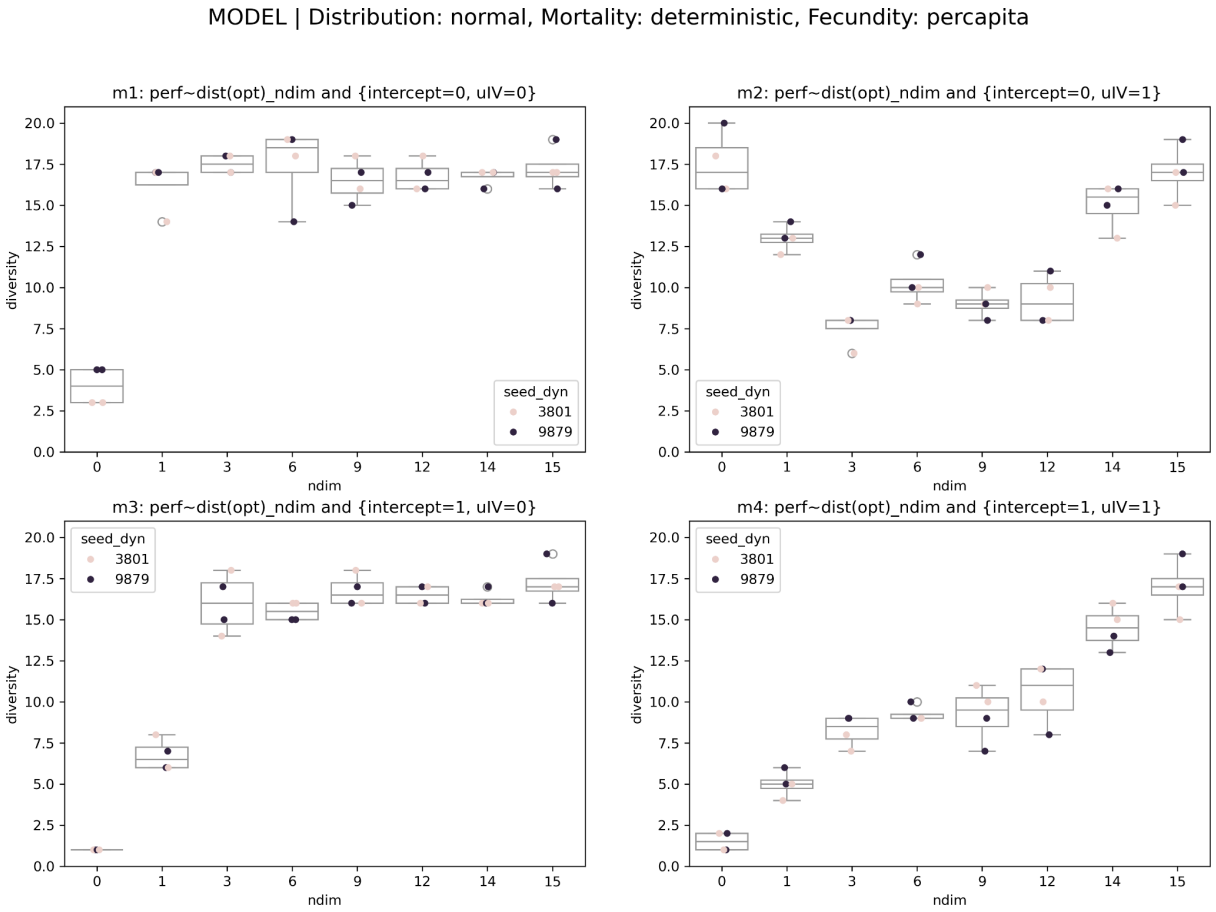

Figure R2: Model comparison with a normal distribution for environmental variables. See legend of Fig. R1.

The same patterns are found when environmental variables follow a normal distribution (as in our manuscript; Fig. R2). However, since more species have a small favourable habitat in this case, species can become extinct even with m1 when ndim > 0 (diversity < 20). In m2 and m4, the effect of uIV is stronger.
These results (which we refer the reader to in the main text, l. 153) answer three of the concerns raised by the recommender:

1) The absence of a drop in species richness in his simulations: it can be observed here.

2) The influence of the intercept on our results: we agree that this effect is underestimated in our manuscript, and it is now better acknowledged in the main text (l. 290-294):

"While the intercepts included in the Imperfect knowledge models resulted in a species hierarchy leading to a drop in species richness at low numbers of observed dimensions, this effect quickly weakened as the number of observed dimensions increased. Overall this effect alone did not explain the observed patterns and our results were qualitatively robust to the use of a distance rather than a quadratic function in Imperfect knowledge models (see Reviews)."

Indeed, under a low number of observed environmental variables, using an intercept can lead to a species hierarchy, which reduces species richness. However, this effect weakens and becomes negligible when the number of observed dimensions increases. Importantly, the effect of the intercept alone does not explain the observed patterns in species richness, which are also attributable to uIV - the scope of our study. Moreover, ecologists rarely implement models without an intercept, which motivates our choice, as explained l. 113-115:

"The use of an intercept to estimate species performance in averaged environmental conditions is also a common practice among ecologists."

3) Using a polynomial rather than a distance model: this choice does not have an important effect on our qualitative results.

More generally, although the implementation of different mortality and fecundity types in our simulation framework aimed at minimising the dependence of our results to modelling choices, we acknowledge that our results remain inevitably dependent on the overall model structure and other choices, even if these choices were motivated (such as the distribution of environmental variables or the model mis-specification). Our study thus provides a test given these choices and this is now reemphasized l. 364, l. 278, l. 280.

3) Finally, a very minor comment that sprang from Simon Blanchet's mention of drift for small populations: it is clear that you are setting aside genetic drift (or genetic anything) here, but it is worth mentioning the notion of *ecological* drift i.e. small populations going extinct due to chance events, which is probably the main cause of extinctions here except model misspecification. Such drift is initially more likely when you start with one individual per species (you will automatically lose one out of 20 species at the first time step if your mortality is larger than 5% for instance), but it is not clear that these transient effects are of much import here, I would guess not in most of your simulations.

R: We agree that this aspect is important and added a sentence l. 194-195:

“As population sizes were low, communities were subject to ecological drift (i.e. extinctions due to demographic stochasticity)”

WRITING COMMENTS

This is quite optional, more comfort than necessity, but parts of the discussion still feel like they might fare better in the (very short) Results section instead. The paragraphs starting

with "Compared to the reference communities simulated ", "When structured IV accounted for less", "When the proportion of structured IV increased, ", "Our results suggest that improving" ... tended to break the flow of the broader discussion, by mostly describing results and only briefly including a few additional thoughts and interpretations (some of which I am not necessarily convinced, see above)

R: We have shortened these paragraphs, in particular by removing specific references to models and figures, which were already made in the results section. These shorter paragraphs were kept however, as we feel they help contextualise the results and provide some important elements of discussion.

Eq1: d_i,j,m instead of d_m,j

R: Done

The sentence "Using the Imperfect knowledge models with nobs = K, we were able to test if the error due to this assumption was large" seems out of place, since I assume the assumption it refers to is the incorrect quadratic model Eq2; I would move it and/or clarify that the shape of Eq2 is the assumption in question.

R: As developed in our answer to comment 2.2, we removed this sentence to better clarify this point later (l. 143-153):

“Note that, when comparing simulation outcomes with performance computed with the Perfect Knowledge and the Imperfect Knowledge models, differences can actually result from two main aspects: (i) the model mis-specification (i.e. the use of a quadratic function including an intercept, Eq.2, Eq.3, Eq.4, instead of a distance, Eq.1), and (ii) the number of observed dimensions. Change in the latter actually results in a change in the proportion of uIV, but also in the number of variables that can be used to compute the site-dependent (or environment-dependent) part of performance. Moreover, the presence of an intercept in models (Eq.2, Eq.3, Eq.4) can lead to a hierarchy in species performances fostering species extinctions. We tested the magnitude of the effect of the intercept and of the model mis-specification, using an Imperfect knowledge model with a distance function, with and without intercepts, which did not substantially change our results (see Reviews and discussion below).”

It feels a bit confusing to have Eq3 and Eq4 one after the other with the same notations and without any text to separate them and explain what is what; I would put Eq3 after the first sentence of the preceding paragraph and Eq4 after the second (and ending the seocnd with something like "as modelled by a randomly drawn contribution to performance \hat epsilon_i,j,m").

R: Done

It is a bit confusing that Eq4 is exactly like Eq2 with a hat, when in one case epsilon is an estimated residual, and in the other, epsilon hat is a random variable drawn each time an individual is born in a simulation.
My recommendation would be to leave \hat epsilon_i,j,m ~ N(0,V_j) in eq 4, as this equation truly indicates a random variable drawn from a distribution, while Eq 2 could instead have something like V_j = var(epsilon_i,j,m) to explain that it is an "empirical" quantity estimated from the residuals.

R: We have updated the notation to differentiate better the different steps of the modelling procedure. In particular, we now use the \hat notation for every estimated parameter or predicted performance.

==========

```
import numpy as np, pandas as pd

mortality=['deterministic','random'][0]
fecundity=['fixed','dependent'][1]
deltaE=0. # How much the environment changes in each patch per turn


tmax=10000
J=20 #nb species
K=15 #nb env dim
M=625 #nb sites
uIV=0.9 #Fraction uIV
nprop=10 #nb propagules (initial, and each turn for fixed fecundity)
fecund = 0.5 # nb of propagules per capita (for abundance-dependent fecundity)
death=0.01 #Fraction of dead individuals per turn

E=np.random.uniform(0,1,(K,M)) #environment values
Eopt=np.random.uniform(0,1,(J,K)) #species environmental preferences

alld = np.sum( (Eopt.reshape((J,K,1)) - E.reshape((1,K,M)) )**2, axis=1 )**.5
mud,sigd=np.mean(alld),np.std(alld)

#Which species occupies each site
Occ=np.zeros(M,dtype='int')-1

#Initial condition
initial_pos=np.random.choice(np.where(Occ<0)[0],size=J*nprop,replace=True)
for j in range(J):
    Occ[initial_pos[j*nprop:(j+1)*nprop]]=j

#Computing performance for a new individual
def calc_perf(E,Eopt,unstructured):
```

```
    d=np.sum( (E-Eopt.T )**2, axis=0 )**.5
    return - (d-mud)/sigd *(1-unstructured) + np.random.uniform(-1,1,d.shape)* unstructured

#Performance at each site
Perf=calc_perf(E, Eopt[ Occ ],uIV)
Perf[Occ<0]=np.nan

table=[]
t=0
while t < tmax:
    print(t)

    ## For temprally changing environments, compute individuals' changes in performance
    if deltaE>0:
        prev= calc_perf(E, Eopt[ Occ ],0)
        E=np.clip(E+np.random.uniform(-deltaE,deltaE,E.shape), 0,1)
        next=calc_perf(E, Eopt[ Occ ],0)
        Perf=Perf+(next-prev)

    #Death
    occupied=np.where(Occ>=0)[0].astype('int')
    if mortality=='deterministic':
        #The worst x% die
        worst = occupied[np.argsort(Perf[occupied])[:int(M*death)]]
    elif mortality =='random':
        #x% die at random (just a test, not in the paper)
        worst=np.random.choice(occupied,size=int(M*death))
    Occ[worst]=-1
    Perf[worst]=np.nan

    #Fecundity
    if fecundity=='fixed':
        # Fixed fecundity
        propagule_sp=np.array([ j for j in range(J) for prop in range(nprop) if (Occ==j).any() ])
    elif fecundity == 'dependent':
        # Abundance-dependent fecundity
        propagule_sp=np.array([ j for j in range(J) for prop in range( int(np.sum(Occ==j)*fecund
) )  ])
    if len(propagule_sp)==0:
        break

propagule_pos=np.random.choice(np.where(Occ<0)[0],size=len(propagule_sp),replace=Tru
e)
    propagule_perf = calc_perf(E[:,propagule_pos],Eopt[propagule_sp],uIV)

    for pos in np.unique(propagule_pos):
        candidates=np.where(propagule_pos==pos)[0]
        best=candidates[np.argmax(propagule_perf[candidates])]
```

```
        Occ[pos]=propagule_sp[best]
        Perf[pos]=propagule_perf[best]


    # Summary outcomes
    x=np.array([np.sum(Occ==j) for j in range(J) ])
    p=x/np.sum(x)
    dic={'diversity':np.sum(x>0),'shannon':np.exp(-np.sum( p[p>0]*np.log(p[p>0]))) ,
'occupation':np.mean(Occ>=0),'mean_performance':np.nanmean(Perf)}
    table.append(dic)
    t+=1

results=pd.DataFrame(table)



#########################
# RESULTS

import matplotlib.pyplot as plt

plt.figure()
plt.subplot(221),plt.title('Occupation')
plt.plot(results['occupation'])
plt.subplot(222),plt.title('Diversity')
plt.plot(results['diversity'],label='number')
plt.plot(results['shannon'],label='shannon')
plt.legend()
plt.subplot(223),plt.title('Performance')
plt.plot(results['mean_performance'])
plt.subplot(224),plt.title('Final landscape occupation')
x=Occ.copy().astype('float')
x[x<0]=np.nan
plt.colorbar(plt.imshow(x.reshape((25,25)) ,cmap='jet'))
plt.show()
```

R: Python code used to test the effect of model misspecification on results. We built on the Python code provided by the recommender. The code is also available in a GitHub repository at https://github.com/ghislainv/uIV.

```
"""Theoretical model."""

import os
import multiprocessing as mp

import matplotlib.pyplot as plt
```

```
import numpy as np
import pandas as pd
import seaborn as sns
from tqdm import tqdm

# ====================================================
# Function to set initial conditions
# ====================================================


def set_initial_conditions(J=20, K=15, M=625,
                           nprop=10, distrib_env="uniform",
                           seed_init=1234):
    """Setting initial conditions.

    :param J: Number of species.
    :param K: Number of environmental dimensions.
    :param M: Number of sites.
    :param nprop: Number of initial propagules per species.
    :param distrib_env: Distribution for environmental
    variables. Either "normal" or "uniform".
    :param seed_init: Seed for initial conditions.

    :return: A dictionary with initial conditions.

    """

    # RNG
    rng_init = np.random.default_rng(seed_init)

    # Environment
    if distrib_env == "uniform":
    E = rng_init.uniform(0, 1, (K, M))
    elif distrib_env == "normal":
    E = rng_init.normal(0, 1, (K, M))
    # Normalize on [0, 1]
    E = (E - np.min(E)) / (np.max(E) - np.min(E))
    # Species optima
    Xopt = rng_init.uniform(0, 1, (J, K))

    # Initial occupancy
    Occ = np.zeros(M, dtype="int")-1
    initial_pos = rng_init.choice(
    np.where(Occ < 0)[0], size=J*nprop, replace=False)
    for jj in range(J):
    Occ[initial_pos[jj*nprop:(jj+1)*nprop]] = jj

    return {"Occ": Occ, "E": E, "Xopt": Xopt}
```

```
# ====================================================
# Function to compute propagule performance
# ====================================================


def calc_perf_propagules(E, Xopt, ndim, propagule_pos, propagule_sp,
                         mu_res, sig_res, mup, sigp,
                         sp_intercept=True, uIV=False,
                         rng_dyn=np.random.default_rng(1234)):
        """Computing propagule performances.

        The performance of a propagule depends on both the environment and
        the species optima. Residuals associated with the missing
        dimensions can be added to the performance computation. Per
        species residuals have a mean and a standard deviation. Either the
        mean or standard deviation of the species residuals or both can be
        taken into account to compute species and propagule performances.

        :param E: K x M np.array. Environment with K dimensions on M sites.
        :param Xopt: J x K np.array. Species optima for the K dimensions.
        :param ndim: Number of observed environmental dimensions.
        :param propagule_pos: Sites reached by propagules.
        :param propagule_sp: Species for each propagule.
        :param mu_res: Residual mean per sp. due to unobserved dimensions.
        :param sig_res: Residual std per sp. due to unobserved dimensions.
        :param mup: Mean performance of all species on all sites.
        :param sigp: Std of the performance of all species on all sites.
        :param sp_intercept: Add species intercept from missing dimensions.
        :param uIV: Add random IV from missing dimensions.
        :param rng_dyn: RNG for dynamics.

        :return: Propagule performances.

        """

        E_propagules = E[:ndim, propagule_pos]
        Xopt_propagules = Xopt[propagule_sp, :ndim]
        perf = np.sum((E_propagules - Xopt_propagules.T)**2, axis=0)**0.5
        if sp_intercept:
        perf += mu_res[propagule_sp]
        if uIV:
        perf += rng_dyn.normal(0, sig_res[propagule_sp], size=perf.shape)

        # Return the opposite of the distance (-)
        return -(perf-mup)/sigp
```

```
# ====================================================
# Function to sort and randomize between equal values
# ====================================================

def argsort_rand(x, rng_dyn=np.random.default_rng(1234)):
        """Sort and randomize between equal values"""
        # Draw random numbers of the same size as x
        y = rng_dyn.random(x.size)
        # Sort by y then by x with np.lexsort
        return np.lexsort((y, x))


def argmax_rand(x, rng_dyn=np.random.default_rng(1234)):
        """Random tie-breaking argmax"""
        return rng_dyn.choice(np.where(x == x.max())[0])


# ====================================================
# Function to simulate the community dynamics
# ====================================================

def simulate_dynamics(J=20, K=15, M=625, seed_init=1234,
                      death=0.01, nprop=10, fecund=0.5,
                      mortality="deterministic", fecundity="percapita",
                      distrib_env="uniform",
                      ndim=1, sp_intercept=False, uIV=False,
                      tmax=10000, seed_dyn=1234, verbose=True):
        """Simulating community dynamics.

        :param J: Number of species.
        :param K: Number of environmental dimensions.
        :param M: Number of sites.
        :param seed_init: Random seed for initial conditions.

        :param death: Fraction of dead individuals per turn.
        :param nprop:  Nb of propagules for fixed fecundity.
        :param fecund: Nb of propagules per capita (for
        abundance-dependent fecundity).

        :param mortality: Either "deterministic" or "random".
        :param fecundity: Either "fixed" or "percapita".
        :param distrib_env: Distribution for environmental
        variables. Either "normal" or "uniform".

        :param ndim: Number of observed dimensions (ndim <= K).
        :param sp_intercept: Add species intercept from missing dimensions.
        :param uIV: Add random IV from missing dimensions.
```

```
:param tmax: Maximal number of iterations.
:param seed_dyn: Random seed for dynamics.
:param verbose: Print messages and progress bar.

:return: Dataframe of results.

"""

# RNG simu
rng_dyn = np.random.default_rng(seed=seed_dyn)

# Initial conditions
init = set_initial_conditions(J, K, M, nprop,
                    distrib_env, seed_init)
Occ = np.copy(init["Occ"])
E = init["E"]
Xopt = init["Xopt"]

# Performance of species on all sites
perf_tot = np.sum((E[np.newaxis, ...] -
          Xopt[..., np.newaxis])**2, axis=1)**0.5
mup, sigp = np.mean(perf_tot), np.std(perf_tot)

# Residual mean and sd per species due to missing dimensions
E_dim = E[:ndim, :]
Xopt_dim = Xopt[:, :ndim]
perf_obs = np.sum((E_dim[np.newaxis, ...] -
          Xopt_dim[..., np.newaxis])**2, axis=1)**0.5
perf_res = perf_tot - perf_obs
mu_res = np.mean(perf_res, axis=1)
sig_res = np.std(perf_res, axis=1)

# Initial performances
Perf = calc_perf_propagules(E, Xopt, ndim,
                    np.arange(M), Occ,
                    mu_res, sig_res,
                    mup, sigp,
                    sp_intercept, uIV,
                    rng_dyn)
Perf[Occ < 0] = np.nan
table = []
x = np.array([np.sum(Occ == j) for j in range(J)])
p = x/np.sum(x)
dic = {"iteration": 0,
"diversity": np.sum(x > 0),
"shannon": np.exp(-np.sum(p[p > 0]*np.log(p[p > 0]))),
"occupation": np.mean(Occ >= 0),
```

```
"mean_performance": np.nanmean(Perf)}
table.append(dic)

# Iterations
for t in tqdm(range(tmax), disable=not verbose):

# Death
occupied = np.where(Occ >= 0)[0].astype("int")
worst = []
if mortality == "deterministic":
# The worst x% die
sorted_perf = argsort_rand(Perf[occupied], rng_dyn)
worst = occupied[sorted_perf[:int(M*death)]]
elif mortality == "random":
# x% die at random
worst = rng_dyn.choice(occupied, size=int(M*death))
Occ[worst] = -1
Perf[worst] = np.nan

# Fecundity
propagule_sp = []
if fecundity == "fixed":
# Fixed fecundity
propagule_sp = np.array([j for j in range(J)
                    for prop in range(nprop)
                    if (Occ == j).any()])
elif fecundity == "percapita":
# Abundance-dependent fecundity
propagule_sp = np.array([j for j in range(J)
                    for prop in
                    range(int(np.sum(Occ == j)*fecund))])
if len(propagule_sp) == 0:
break
propagule_pos = rng_dyn.choice(np.where(Occ < 0)[0],
                    size=len(propagule_sp), replace=True)
propagule_perf = calc_perf_propagules(E, Xopt, ndim,
                                propagule_pos, propagule_sp,
                                mu_res, sig_res,
                                mup, sigp,
                                sp_intercept, uIV,
                                rng_dyn)
for pos in np.unique(propagule_pos):
candidates = np.where(propagule_pos == pos)[0]
best = candidates[argmax_rand(propagule_perf[candidates])]
Occ[pos] = propagule_sp[best]
Perf[pos] = propagule_perf[best]

# Summary outcomes
```

```
        x = np.array([np.sum(Occ == j) for j in range(J)])
        p = x/np.sum(x)
        dic = {"iteration": t + 1,
        "diversity": np.sum(x > 0),
        "shannon": np.exp(-np.sum(p[p > 0]*np.log(p[p > 0]))),
        "occupation": np.mean(Occ >= 0),
        "mean_performance": np.nanmean(Perf)}
        table.append(dic)

        return {"dynamics": pd.DataFrame(table), "community": Occ}


# ======================================================
# Function to plot results
# ======================================================

def plot_results(data_frame, dist, mort, fec):
        """Plotting results."""

        # Data-frame
        df_exp = data_frame

        # Boxplots
        fig = plt.figure(figsize=(15, 10), dpi=300)
        fig.suptitle((f"MODEL | Distribution: {dist}, "
                f"Mortality: {mort}, Fecundity: {fec}"),
                fontsize=16)

        axes = plt.subplot(221)
        axes.set_title("m1: perf~dist(opt)_ndim and {intercept=0, uIV=0}")
        dm1 = df_exp.loc[(df_exp["intercept"] == 0) & (df_exp["uIV"] == 0)]
        sns.boxplot(x="ndim", y="diversity", data=dm1, color="white")
        sns.stripplot(x="ndim", y="diversity", data=dm1, hue="seed_dyn")
        axes.set_ylim(0, 21)

        axes = plt.subplot(222)
        axes.set_title("m2: perf~dist(opt)_ndim and {intercept=0, uIV=1}")
        dm1 = df_exp.loc[(df_exp["intercept"] == 0) & (df_exp["uIV"] == 1)]
        sns.boxplot(x="ndim", y="diversity", data=dm1, color="white")
        sns.stripplot(x="ndim", y="diversity", data=dm1, hue="seed_dyn")
        axes.set_ylim(0, 21)

        axes = plt.subplot(223)
        axes.set_title("m3: perf~dist(opt)_ndim and {intercept=1, uIV=0}")
        dm1 = df_exp.loc[(df_exp["intercept"] == 1) & (df_exp["uIV"] == 0)]
        sns.boxplot(x="ndim", y="diversity", data=dm1, color="white")
        sns.stripplot(x="ndim", y="diversity", data=dm1, hue="seed_dyn")
        axes.set_ylim(0, 21)
```

```
        axes = plt.subplot(224)
        axes.set_title("m4: perf~dist(opt)_ndim and {intercept=1, uIV=1}")
        dm1 = df_exp.loc[(df_exp["intercept"] == 1) & (df_exp["uIV"] == 1)]
        sns.boxplot(x="ndim", y="diversity", data=dm1, color="white")
        sns.stripplot(x="ndim", y="diversity", data=dm1, hue="seed_dyn")
        axes.set_ylim(0, 21)

        fig_out = f"outputs/results_{dist}_{mort}_{fec}.png"
        fig.savefig(fig_out)


# ============================================================
# Running the experiments with repetitions, intercept and uIV
# ============================================================

def run_experiment(distrib_env, mortality, fecundity, verbose=True):
        """Run experiment.

        :param distrib_env: Distribution for the environment.
        :param mortality: Either "deterministic" or "random".
        :param fecundity: Either "fixed" or "percapita".
        :param verbose: Print messages and progress bar.

        """

        # Output file
        f_out = f"outputs/results_{distrib_env}_{mortality}_{fecundity}.csv"

        # Run only if not output file
        if not os.path.isfile(f_out):

        # Model type with variable species parameters: intercept and variance
        sp_intercept = [False, True]
        sp_variance = [False, True]
        sp_int_var = [(i, j) for i in sp_intercept for j in sp_variance]

        # Dataframe to store results
        df = pd.DataFrame(columns=["seed_init", "seed_dyn", "mortality",
                           "fecundity", "intercept", "uIV", "ndim",
                           "iteration", "diversity", "shannon",
                           "occupation", "mean_perf"])

        # Message
        print((f"MODEL | Distribution: {distrib_env}, "
        f"Mortality: {mortality}, Fecundity: {fecundity}"))
        # Selection of dimensions to avoid iterating over all values of K
        ndim_sel = [0, 1, 3, 6, 9, 12, 14, 15]
```

```
        for i in ndim_sel:

        # Number of repetitions
        N_REP_INIT = 2
        N_REP_DYN = 2

        # Iterations
        TMAX = 10000

        # Seeds
        rng_exp = np.random.default_rng(1234)
        S_init = rng_exp.integers(10000, size=N_REP_INIT)
        S_dyn = rng_exp.integers(10000, size=N_REP_DYN)

        if verbose:
                print(f"-- Dimension: {i}")

        # Loop on repetitions
        for j in range(N_REP_INIT):
                for k in range(N_REP_DYN):
                # Loop on models including or not species parameters
                for (sp_int, sp_var) in sp_int_var:
                r = simulate_dynamics(ndim=i, sp_intercept=sp_int,
                                        uIV=sp_var,
                                        tmax=TMAX,
                                        mortality=mortality,
                                        fecundity=fecundity,
                                        distrib_env=distrib_env,
                                        seed_init=S_init[j],
                                        seed_dyn=S_dyn[k],
                                        verbose=verbose)
                rr = r["dynamics"].iloc[TMAX].tolist()
                ll = [S_init[j], S_dyn[k], mortality,
                        fecundity, sp_int, sp_var, i] + rr
                df.loc[len(df)] = ll

        # Saving results
        df.to_csv(f_out, index=False)

        # Plotting results
        plot_results(df, distrib_env, mortality, fecundity)


# ========================================
# Parallelize for demography and environment
# ========================================
```

```
# Model type with variable demographic processes
dist_env = ["uniform", "normal"]
mortality = ["deterministic", "random"]
fecundity = ["fixed", "percapita"]
args = [(i, j, k, False) for i in dist_env
        for j in mortality for k in fecundity]
ncpu_max = mp.cpu_count() - 1
ncpu = np.min([len(args), ncpu_max])

# Create and configure the process pool
with mp.Pool(processes=ncpu) as pool:
        # Issues tasks to process pool
        result = pool.starmap_async(run_experiment, args)
        # Wait for tasks to complete
        result.wait()
        # Process pool is closed automatically

# End of file
```