### JOURNAL PUBLICATION CITATION: Walker, E. J. & B. Gilbert. 2022. Extinction dynamics: the interplay of species traits and the spatial scales of metapopulation declines. *Ecology.*

### Data S1

### All novel code and data used in “Extinction dynamics: the interplay of species traits and the spatial scales of metapopulation declines” manuscript.

### Author(s)

Emma Walker  
University of Toronto  
Department of Ecology and Evolutionary Biology, University of Toronto, 25 Willcocks Street, Toronto, Ontario, Canada M5S 3B2  
emma.walker@mail.utoronto.ca

Benjamin Gilbert  
University of Toronto  
Department of Ecology and Evolutionary Biology, University of Toronto, 25 Willcocks Street, Toronto, Ontario, Canada M5S 3B2  
Benjamin.gilbert@utoronto.ca

### File list:

CODE:

For Simulations:

1. 50 patch test landscapes generation.r
2. 500 patch test landscape generation.r
3. CHECK FUNCTIONS2.r
4. Choose Alphas Function meanmins2.r
5. Create Landscape Function.r
6. Create Multiple Landscapes Function.r
7. Degrade and Destroy Multiple Landscapes Function.r
8. Destroy and Degrade a Landscape Function2 (P and Lm and SimEq)2.r
9. Destroy and Degrade a Landscape Function2 (P and Lm and SimEq)500 saving landscape data.r
10. Lambda M function.r
11. NNDist Distribution Summary Function.r
12. P and LM and SimEq in parallel 3 (50p).r
13. P and LM and SimEq in parallel 3 (500p).r
14. Pstar Function.r
15. Time to eq3.r
16. Vary Alpha Destroy and Degrade Function.r
17. Vary Alpha Destroy and Degrade Function for parallel.r

For Figures:

1. Hypothesis Figure Code.r
2. Manuscript Plots with Annotated Instructions.rmd
3. Manuscript Plots.r
4. Manuscript Plots 500.r
5. NNDISTS histograms of test landscapes code.r
6. Plotdata Median Lm Function.r
7. Plotdata Median P Function.r
8. Plotdata Median P1000 Function.r
9. Plotdata Median tex vs lm Function DEG.r
10. Plotdata Median tex vs lm DEST.r
11. Plotdata Median tp vs lm Function DEG.r
12. Plotdata Median tp vs lm Function DEST.r

DATA:

1. 2000reps\_PnLMnSimEq\_3\_50pnew2.csv
2. 10\_40reps\_ PnLMnSimEq\_3\_500pnew2.csv
3. 40\_70reps\_ PnLMnSimEq\_3\_500pnew2.csv
4. 70\_100reps\_ PnLMnSimEq\_3\_500pnew2.csv
5. 500\_p\_100landscapes\_clevel500.csv
6. 50\_p\_100landscapes\_clevel500.csv
7. 50\_p\_10landscapes\_clevel500.csv
8. alphas500p.csv
9. alphas50p.csv
10. clustered500p deltas sim.csv
11. clustered50p deltas sim.csv
12. random50p deltas sim.csv
13. regular50p deltas sim.csv

**Description**

**Code Files Description:**

For Simulations folder contains:

1. 50 patch test landscapes generation.r

 - R script for generating data describing as many landscapes as specified by ‘no.replicates’ of each landscape type “regular”, “random” or “clustered”, calculating summary statistics about the distribution of nearest neighbour interpatch distances in those respective landscape types, and choosing values for the dispersal parameter “alpha” (=1/(avg. dispersal distance of a species)) based on the average minimum nearest neighbor interpatch distance found across those landscapes generated. All variables set as they were for the generation of the 100 test landscapes of 50 patches, therefore, all code in this script can simply be run without changes to replicate or use of it. Further relevant details annotated throughout the code.

1. 500 patch test landscape generation.r

 - (Nearly identical to the above but set to create landscapes of 500 rather than 50 patches) R script for generating data describing as many landscapes as specified by ‘no.replicates’ of each landscape type “regular”, “random” or “clustered”, calculating summary statistics about the distribution of nearest neighbour interpatch distances in those respective landscape types, and choosing values for the dispersal parameter “alpha” (=1/(avg. dispersal distance of a species)) based on the average minimum nearest neighbor interpatch distance found across those landscapes generated. All variables set as they were for the generation of the 100 test landscapes of 500 patches, therefore, all code in this script can simply be run without changes to replicate or use of it. Further relevant details annotated throughout the code.

1. CHECK FUNCTIONS2.r

- Just an R script that was used while building the code for the project to check the various functions the project uses from time to time. It is not necessary to make use of this to replicate our work but may be useful for understanding some of the functions created and used. Further relevant details annotated throughout the code.

1. Choose Alphas Function meanmins2.r

- This function chooses alpha values for simulation based on each landscape type's distribution of nndists (ie. the levels of alpha representing different species but of equivalent dispersal ability within the given landscape, controlling for the fact that clustering will have smaller nearest neighbour distances by way of the clustering algorithm, while regular will have larger nndists – that is to say it isolates for the effect of regularity in patch spacing). This function takes in a dataframe containing the summary stats of nndists for all three landscape types (aka output from the NNDist Distribution Summary Function.r function) and requires specification of the max x and y coordinates that were given to the landscapes these data were generated from (ie. “landscape.limit”). Further relevant details annotated throughout the code.

1. Create Landscape Function.r

- This function creates a landscape of a specified size with a specified number of patches, with a patch distribution of specified level of clustering or uniformity.“n.patches” specifies the number of patches in the desired landscape. “landscape.limit” specifies the size of desired landscape (max possible x and y coordinate of a patch). “landscape.type” specifies whether the desired landscape's patches should be randomly, clustered or more regularly distributed (accepts either "clustered", "regular" or "random"). “no.runs” specifies the number of iterations for which the clustering algorithm is run for, specifying the level of clustering. Details of this algorithm are outlined in the main manuscript in the description of how landscapes were generated. Further relevant details annotated throughout the code.

1. Create Multiple Landscapes Function.r

- This function uses the “Create Landscape Function.r” to create multiple landscapes and store all the data describing each landscape in one common dataframe. Again, “n.patches” specifies the number of patches in the desired landscape. “landscape.limit” specifies the size of desired landscape (max possible x and y coordinate of a patch). “no.runs” specifies the number of iterations for which the clustering algorithm is run for, specifying the level of clustering. And finally an additional parameter “no.replicates” specifying how many landscapes of each type to create and where “landscape.types” may contain a list of landscape types to be created (accepting "clustered", "regular" or "random" as possible options). Details of this algorithm are outlined in the main manuscript in the description of how landscapes were generated. Further relevant details annotated throughout the code.

1. Degrade and Destroy Multiple Landscapes Function.r

- This function uses the “Create Landscape Function.r” to create landscapes and either the “Destroy and Degrade a Landscape Function2 (P and Lm and SimEq)2.r or Destroy and Degrade a Landscape Function2 (P and Lm and SimEq)500 saving landscape data.r (depending on which you choose to source) to simulate both a degradation and destruction scenario, on separate copies of each landscape generated. Again, “n.patches” specifies the number of patches in the desired landscape. “landscape.limit” specifies the size of desired landscape (max possible x and y coordinate of a patch). “no.runs” specifies the number of iterations for which the clustering algorithm is run for, specifying the level of clustering. “landscape.type” specifies whether the desired landscape's patches should be randomly, clustered or more regularly distributed (accepts either "clustered", "regular" or "random"). “no.runs” specifies the number of iterations for which the clustering algorithm is run for, specifying the level of clustering. Details of this algorithm are outlined in the main manuscript in the description of how landscapes were generated. “a” stands for the model parameter “alpha” and specifies 1/(average dispersal distance of the species). “scaled.lm” specifies to which value “lambda\_M” shall be commonly scaled to across pristine landscapes (prior to any destruction or degradation). Details of how destruction and degradation are simulated, the metapopulation model and what metapopulation responses are tracked are outlined in the main manuscript’s methods. Further relevant details annotated throughout the code.

1. Destroy and Degrade a Landscape Function2 (P and Lm and SimEq)2.r

- This function takes a given landscape (as generated by the “Create Landscape Function.r” function) and separately performs two scenarios in which all patches are either destroyed or degraded until no habitat remains, outputting the metapopulation persistence capacity (as calculated using Lambda M function.r function) , average probability of patch occupancy at equilibrium across all patches (as calculated using Pstar Function.r function), average number of patches occupied for last 50 of 1000 timesteps of a simulation of the metapopulation through time (as simulated using Time to eq3.r), and a timestep at which the metapopulation went extinct (also included in the output of the Time to eq3.r function). “landscape” specifies a dataframe created by the create landscape function which specifies patch locations, areas, and and an interpatch distance matrix. “a” stands for the model parameter “alpha” and specifies the 1/(average dispersal distance) for a species. “delta” species the ratio of the extinction to colonization rate of a species (e/c). Details of how destruction and degradation are simulated, the metapopulation model and what metapopulation responses are tracked are outlined in the main manuscript’s methods. Further relevant details annotated throughout the code.

1. Destroy and Degrade a Landscape Function2 (P and Lm and SimEq)500 saving landscape data.r

- This function is nearly identical to the one above just a minor adjustment to handle working with landscapes with much larger numbers of patches. Rather than randomly removing one patch at a time and calculating and simulating the effects of this each time for the destruction scenario, these calculations and simulations are only performed following the random removal of 10 patches to save time and computation.

1. Lambda M function.r

- This function calculates lambda\_M for a given metapopulation in a given landscape. “landscape” specifies a dataframe created by the create landscape function which specifies patch locations, areas, and and an interpatch distance matrix (as generated by the “Create Landscape Function.r” function). “a” stands for the model parameter “alpha” and specifies the 1/(average dispersal distance) for a species. “delta” species the ratio of the extinction to colonization rate of a species (e/c). Further relevant details annotated throughout the code.

1. NNDist Distribution Summary Function.r

- This function calculates the summary statistics of the distribution of nearest neighbour distances across multiple landscapes created using the ‘50 patch test landscapes generation.r’ or ‘500 patch test landscapes generation.r’ functions. ‘landscape.data’ provides the data of the multiple landscapes created. ‘no.replicates’ specifies the number of landscapes that were created. ‘n.patches’ specifies the number of patches within those landscapes created. NOTE: only enter landscapes of one type/level of clustering, hence, use the output data from the create multiple landscapes function selected for one given landscape type. Further relevant details annotated throughout the code.

1. P and LM and SimEq in parallel 3 (50p).r

- This R script runs all of the functions simulating and calculating metapopulations responses to both destruction and degradation scenarios and tabulates all of this data into one common dataframe for analysis. Essentially, it simply runs the Degrade and Destroy Multiple Landscapes Function.r in parallel with all the appropriate parameter combinations we desired investigation of in this manuscript (drawing on the Vary Alpha Destroy and Degrade Function for parallel.r to break these up and pass the relevant parameter info to each core). It splits this task up over the 48 cores we, the authors, had available to us. Thus, if you have a minimum of 48 cores available to you, you may simply run this script as is to replicate our experiments (allow several days – a week or two to run). Without availability of this minimum computational power and time, however, neither will you be able to run this script nor would you have the time to fully replicate our experiments across the full parameter space we have explored (not split into parallel this would take months to years). Thus there is no point to providing a version of this code not in parallel as it simply would not execute fast enough to be worthwhile. However, feel free to either A) create a similar script to split the task of running the Degrade and Destroy Multiple Landscapes Function.r in parallel with all the parameter combinations you desire to investigate that best makes use of the computational resources available to you or B) use the Vary Alpha Destroy and Degrade Function.r to generate data for a slice of the parameter space we explored.

\*Note: this script is optimized for a high number of replicate simulations and low number of patches within landscapes. For visa versa see below.

1. P and LM and SimEq in parallel 3 (500p).r

- This R script is nearly identical to the one above, just optimized for a high number of patches within each landscape keeping a low number of replicate simulations. Again, provided you have similar or greater computational resources available to you, you may simply rerun this script to replicate work applied to landscapes of 500 patches.

1. Pstar Function.r

- This function calculates the probability individual patches are occupied at equilibrium. “landscape” specifies a dataframe created by the create landscape function which specifies patch locations, areas, and and an interpatch distance matrix (as generated by the “Create Landscape Function.r” function). “a” stands for the model parameter “alpha” and specifies the 1/(average dispersal distance) for a species. “delta” species the ratio of the extinction to colonization rate of a species (e/c). “iterations” specifies the number of iterations for which the iterative function f (see equations 2 & 3 of the main manuscript) is iterated. The greater the number of iterations the greater the accuracy of P\*, while less iterations results in lower accuracy. Further relevant details annotated throughout the code.

1. Time to eq3.r

- This function simulates a metapopulation’s dynamics through time in a given landscape. “landscape” specifies a dataframe created by the create landscape function which specifies patch locations, areas, and and an interpatch distance matrix (as generated by the “Create Landscape Function.r” function). “a” stands for the model parameter “alpha” and specifies the 1/(average dispersal distance) for a species. “delta” species the ratio of the extinction to colonization rate of a species (e/c). “p.initial” to it’s initial occupancy of patches in the given landscape. “avg.p” to the average occupancy expected at equilibrium within that landscape (as calculated using the “Pstar Function.r”. Further relevant details annotated throughout the code.

1. Vary Alpha Destroy and Degrade Function.r

- This function simulates the response of metapopulations of differing dispersal abilities across multiple different landscapes in response to both destruction and degradation scenarios within a given landscape type (ie. landscapes with either “regular”, “random” or “clustered” patch distributions) (by using the Degrade and Destroy Multiple Landscapes Function.r function for a set of alpha values). “alphas” requires a vector of values specifying 1/(the avg. dispersal ability of the species). “n.patches” specifies the number of patches desired in the generated landscapes. “landscape limit” specifies the size of landscapes generated (max x and y coord of patches). “landscape.type” specifies whether landscapes should be "clustered", "random" or "regular" in patch distribution. “no.runs” specifies the level of clustering (number of times the clustering algorithm is iterated for). “n.landscapes” specifies the number of landscapes the metapopulations will be simulated within and that will be put through the degradation and destruction scenarios. “scaled.lm” specifies the common value to which the lambda.M of metapopulations in pristine landscapes will be scaled to. Further relevant details annotated throughout the code.

1. Vary Alpha Destroy and Degrade Function for parallel.r

- This function does the same as the above function but breaks this task up amongst cores to be run in parallel. It is only intended for use within the P and LM and SimEq in parallel 3 (50p).r and P and LM and SimEq in parallel 3 (500p).r scripts.

For Figures folder contains:

1. Hypothesis Figure Code.r

- This script simply plots a figure based on our general hypotheses of how degradation and destruction should impact a species’ lambda\_M, where degradation follows our analytical result and only the extremes of destruction are predicted. Further relevant details annotated throughout the code.

1. Manuscript Plots with Annotated Instructions.rmd

- This R Notebook creates all of the figures featured in our main manuscript with detailed instructions on how each figure is generated from the data output by our simulations.

1. Manuscript Plots.r

- This script contains only the code portion of the ‘Manuscript Plots with Annotated Instructions.rmd” as was used to create the main manuscript figures.

1. Manuscript Plots 500.r

- This script is a minor variation of ‘Manuscript Plots.r’ for handling data from landscapes with a larger number of patches (500). Further relevant details annotated throughout the code.

1. NNDISTS histograms of test landscapes code.r

- This script simply plots histograms of the distribution of nearest neighbor interpatch distances across multiple landscapes created using the ‘50 patch test landscapes generation.r’ or ‘500 patch test landscapes generation.r’ functions. It also runs the ‘NNDist Distribution Summary Function.r’ to calculate the summary statistics of the distribution of nearest neighbour distances across those multiple landscapes. Further relevant details annotated throughout the code.

1. Plotdata Median Lm Function.r

- This function calculates the lambda\_M values plotted in Figure 2 (binning destruction and calculating median lambda.M for percent habitat loss via destruction -using the median because it reflects that the data distribution is heavily skewed, 95% confidence intervals calculated by bootstrapping). Further relevant details annotated throughout the code.

1. Plotdata Median P Function.r

- This function calculates the average P\* values plotted in Figure 3 (binning destruction and calculating median average P\* for percent habitat loss via destruction and degradation -using the median because it reflects that the data distribution is heavily skewed, 95% confidence intervals calculated by bootstrapping). Further relevant details annotated throughout the code.

1. Plotdata Median P1000 Function.r

- This function calculates the average P1000 values plotted in Figure 3 (binning destruction and calculating median average P1000 for percent habitat loss via destruction and degradation -using the median because it reflects that the data distribution is heavily skewed, 95% confidence intervals calculated by bootstrapping). Further relevant details annotated throughout the code.

10. Plotdata Median tex vs lm Function DEG.r

- This function calculates the median time to extinction for a given decrease in lambda\_M for different dispersal abilities (alpha) plotted in Figure 3 panel F. It also calculates 95% confidence intervals that are not used in plotting. Further relevant details annotated throughout the code.

1. Plotdata Median tex vs lm DEST.r

- This function calculates the median time to extinction for a given decrease in lambda\_M for different dispersal abilities (alpha) plotted in Figure 3 panel E. It also calculates 95% confidence intervals that are not used in plotting. Further relevant details annotated throughout the code.

1. Plotdata Median tp vs lm Function DEG.r

- This function calculates the median time to equilibrium for a given decrease in lambda\_M for different dispersal abilities (alpha) plotted in Figure 3 panel D. It also calculates 95% confidence intervals that are not used in plotting. Further relevant details annotated throughout the code.

13. Plotdata Median tp vs lm Function DEST.r

- This function calculates the median time to equilibrium for a given decrease in lambda\_M for different dispersal abilities (alpha) plotted in Figure 3 panel C. It also calculates 95% confidence intervals that are not used in plotting. Further relevant details annotated throughout the code.

**Instructions for use:**

Note: change all setwd() filepaths to be appropriate for your personal computer and file system.

Run‘50 patch test landscapes generation.r’ (or the 500 patch equivalent) to generate test landscapes to look at the nearest neighbour interpatch distance distributions in and choose dispersal parameters values (alpha).

Run ‘NNDISTS histograms of test landscapes code.r’ to look at the distribution of nearest neighbour interpatch distances.

Run‘P and LM and SimEq in parallel 3 (50p).r’ (or the 500 patch equivalent) to perform the simulations and collect their data. Note: You need access to a server with sufficient cores for this. See notes on this script for your options if you do not have access to a server with sufficient cores.

Run ‘Manuscript Plots with Annotated Instructions.rmd’ (or the 500 patch equivalent) to generate the figures and save related data tables of delta values we provided in the supplementary information.

You may check out the outputs of some of the functions the simulations use by running ‘CHECK FUNCTIONS2.r’.

**The overall structure of the code files and how their outputs feed into each other is as follows:**

Diagram

Description automatically generated

**Where section A uses the following code files in the following structure:**

Diagram

Description automatically generated

**B uses:**

Graphical user interface, diagram

Description automatically generated

**and C uses:**

Diagram

Description automatically generated

(or similar files for 500 patches)

**Data Files Description:**

**The following simulation data files are all structured the same:**

1. 2000reps\_PnLMnSimEq\_3\_50pnew2.csv

- All simulated data for 2000 replicates of each parameter combination simulated in our main manuscript with 50 patches in each landscape and lambda\_M\_pristine =20.

1. 10\_40reps\_ PnLMnSimEq\_3\_500pnew2.csv

- All simulated data for the first 10-40 replicates of each parameter combination simulated in our main manuscript with 500 patches in each landscape and lambda\_M\_pristine =20

1. 40\_70reps\_ PnLMnSimEq\_3\_500pnew2.csv

- All simulated data for the first 40-70 replicates of each parameter combination simulated in our main manuscript with 500 patches in each landscape and lambda\_M\_pristine =20

1. 70\_100reps\_ PnLMnSimEq\_3\_500pnew2.csv

- All simulated data for the first 70-100 replicates of each parameter combination simulated in our main manuscript with 500 patches in each landscape and lambda\_M\_pristine =20

**Providing the following in each:**

**The first half of the dataframe contains variables and information pertaining to the landscape in which each simulation was performed provided by the following:**

‘Landscape.type’ provides whether patches follow a more ‘regular’, ‘random’ or ‘clustered’ distribution.

‘initial.landscape.patch.ID’ provides a unique ID number to each patch in a landscape.

‘initial.landscape.A’ provides the area or carrying capacity of each patch.

‘initial.landscape.x.coord’ provides the x coordinate location of each patch in the landscape.

‘initial.landscape.y.coord’ provides the y coordinate location of each patch in the landscape.

**The second half of the dataframe contains the variables measured throughout each simulation.**

‘order.lost’ ranks patches according to which was first removed in the simulation of the destruction scenario on a copy of the landscape.

‘lambda.M.r’ provides the value of lambda\_M following the destruction of patches in each simulation.

‘eq.size.r’ please ignore as this output was not used in our (unverified value, please verify what it may represent if interested in this output).

‘eq.p.r’ please ignore as this output was not used in our study (unverified value, please verify what it may represent if interested in this output).

‘avg.p.r’ provides the average patch occupancy at equilibrium (P\*) following the destruction of patches in each simulation.

‘time.to.eq.r’ provides the number of timesteps it took for the simulated metapopulation’s size reach average P\* in the landscape following the destruction of patches in each simulation.

‘time.to.p1000.r’ provides the number of timesteps it took for the simulated metapopulation’s to go extinct following the destruction of patches in each simulation.

‘sim.eq.size.r’ provides the average patch occupancy across the last 50 of 1000 timesteps following the destruction of patches in each simulation.

‘percent.habitatloss’ provides a percentage of how much habitat was lost in each simulation.

‘lambda.M.e’ provides the value of lambda\_M following degradation of the landscape in each simulation.

‘eq.size.e’ please ignore as this output was not used in our study (unverified value, please verify what it may represent if interested in this output).

‘eq.p.e’ please ignore as this output was not used in our study (unverified value, please verify what it may represent if interested in this output).

‘avg.p.e’ provides the average patch occupancy at equilibrium (P\*) following degradation of the landscape in each simulation.

‘time.to.eq.e’ provides the number of timesteps it took for the simulated metapopulation’s size reach average P\* following degradation of the landscape in each simulation.

‘time.to.p1000.r’ provides the number of timesteps it took for the simulated metapopulation’s to go extinct following degradation of the landscape in each simulation.

‘sim.eq.size.r’ provides the average patch occupancy across the last 50 of 1000 timesteps following degradation of the landscape in each simulation.

‘delta’ provides the value of delta (e/c; the extinctintion to colonization rate ratio) used in the simulation.

‘alpha’ provides the value of the dispersal parameter alpha (1/(average dispersal distance)) used in the the simulation.

‘rep.no’ is a unique number given to each replicate simulation of a given set of parameter values.

**The following test landscape data files are all structured the same:**

1. 500\_p\_100landscapes\_clevel500.csv

- landscape data pertaining to 100 replicate test landscapes of each landscape type (with ‘regular’, ‘random’ and ‘clustered’ patch distributions) containing 500 patches.

1. 50\_p\_100landscapes\_clevel500.csv

- landscape data pertaining to 100 replicate test landscapes of each landscape type (with ‘regular’, ‘random’ and ‘clustered’ patch distributions) containing 50 patches.

1. 50\_p\_10landscapes\_clevel500.csv

- landscape data pertaining to 100 replicate test landscapes of each landscape type (with ‘regular’, ‘random’ and ‘clustered’ patch distributions) containing 50 patches.

**Providing the following in each:**

**The first half of the dataframe contains variables and information pertaining to the landscape in which each simulation was performed provided by the following:**

‘Landscape.type’ provides whether patches follow a more ‘regular’, ‘random’ or ‘clustered’ distribution.

‘initial.landscape.patch.ID’ provides a unique ID number to each patch in a landscape.

‘initial.landscape.A’ provides the area or carrying capacity of each patch.

‘initial.landscape.x.coord’ provides the x coordinate location of each patch in the landscape.

‘initial.landscape.y.coord’ provides the y coordinate location of each patch in the landscape.

Followed by a distance matrix containing the Euclidean interpatch distances.

**The following alpha data files are all structured the same:**

1. alphas500p.csv

- values of alpha used for simulations within each landscape type (with ‘regular’, ‘random’ and ‘clustered’ patch distributions) containing 500 patches.

1. alphas50p.csv

- values of alpha used for simulations within each landscape type (with ‘regular’, ‘random’ and ‘clustered’ patch distributions) containing 50 patches.

**Providing the following in each:**

**The first half of the dataframe contains variables and information pertaining to the landscape in which each simulation was performed provided by the following:**

‘min’ provides a value of alpha representative of average dispersal distance of 1/16th the mean minimum nearest neighbour interpatch distance.

‘min.1 ‘provides a value of alpha representative of average dispersal distance of 1/8th the mean minimum nearest neighbour interpatch distance.

‘min.2’ provides a value of alpha representative of average dispersal distance of 1/4th the mean minimum nearest neighbour interpatch distance.

‘min.3’ provides a value of alpha representative of average dispersal distance equal to the mean minimum nearest neighbour interpatch distance.

‘min.4’ provides a value of alpha representative of average dispersal distance of 2x the mean minimum nearest neighbour interpatch distance.

‘min.5’ provides a value of alpha representative of average dispersal distance of 4x the mean minimum nearest neighbour interpatch distance.

‘min.6’ provides a value of alpha representative of average dispersal distance of 8x the mean minimum nearest neighbour interpatch distance.

‘a.limit’ provides a value of alpha representative of average dispersal distance equal to the extent of the landscape (ie. global dispersal).

‘landscape.type’ provides whether patches follow a more ‘regular’, ‘random’ or ‘clustered’ distribution.

**The following alpha data files are all structured the same:**

1. clustered500p deltas sim.csv

- provides summary statistics of the distribution of values for delta (e/c; the extinction to colonization rate ratio used for simulations within landscapes with ‘clustered’ patch distributions containing 500 patches.

1. clustered50p deltas sim.csv

- provides summary statistics of the distribution of values for delta (e/c; the extinction to colonization rate ratio) used for simulations within landscapes with ‘clustered’ patch distributions containing 500 patches.

1. random50p deltas sim.csv

- provides summary statistics of the distribution of values for delta (e/c; the extinction to colonization rate ratio) used for simulations within landscapes with ‘random’ patch distributions containing 50 patches.

1. regular50p deltas sim.csv

- provides summary statistics of the distribution of values for delta (e/c; the extinction to colonization rate ratio) used for simulations within landscapes with ‘regular’ patch distributions containing 50 patches.

**Providing the following in each:**

**The first half of the dataframe contains variables and information pertaining to the landscape in which each simulation was performed provided by the following:**

‘Min’ provides the minimum delta value across simulations.

‘1st.Q’ provides the first quartile delta value across simulations.

‘Med’ provides the median delta value across simulations.

‘Mean’ provides the mean delta value across simulations.

‘3rd.Q’ provides the third quartile delta value across simulations.

‘Max’ provides the maximum delta value across simulations.

‘a’ provides the value of the dispersal parameter alpha used across those simulations.

‘avg.disp’ provides the average dispersal distance that value of alpha is equivalent to in the landscape.