Appendices to:

Legendre, P., Fortin, M.-J. & Borcard, D. (2015) Should the Mantel test be used in spatial analysis? *Methods in Ecology & Evolution*, **6**, xxx–xxx.

Appendix S1

Analysis of spatially correlated data and Mantel test: who has shown what?

This section reviews some of the papers that showed important characteristics of tests of significance in the presence of spatial correlation, including Mantel and partial Mantel tests. Simulation studies previous published in the ecological and statistical literatures have touched upon different aspects of the problem of analysing the correlation between spatially correlated data. This brief review of the literature puts our simulation results in the context of the results that are already known about the Mantel and partial Mantel tests and show the significance of our new findings in that context.

Spatial autocorrelation is often taken to mean any kind of spatial dependence (or spatial structure) in data. More formally, one can distinguish *induced spatial dependence*, which is the type of spatial dependence present in a variable due to the causal influence of an explanatory variable that is spatially structured, and *spatial autocorrelation in the strict sense* which is the spatial dependence that remains in the residuals after the effects of all pertinent explanatory variables have been taken into account (Legendre & Legendre 2012, Section 1.1).

• To our knowledge, Bivand (1980) was the first to publish a short series of simulation results showing that the test of significance of the correlation coefficient between two variables that were uncorrelated to each other but strongly spatially autocorrelated had inflated levels of type I error, meaning that the test rejected the null hypothesis more often than predicted by the α significance level. He also showed that when only one of the variables was autocorrelated, there was no such strong effect. In their well-cited book on spatial processes, Cliff & Ord (1981) included a figure (their Fig. 7.2) presenting some of the Bivand simulation results.

• Manly (1986) simulated pairs of spatially correlated multivariate data on points on a map. The data were then transformed into distance matrices **A** and **B** and tested for significance by regressing **A** on **B**; a third distance matrix **C** representing geographic distances was included in the regression equation. For the test, the statistic was the regression coefficient b_i, which was a partial regression coefficient since the **C** distances were also in the regression equation. The test of significance involved Mantel-like permutations of matrix **A**. The simulation results did not demonstrate any major effect of spatial correlation on the estimated values of the regression coefficients in simple (**A** ~ **B**) and multiple regression (**A** ~ **B** + **C**) on distance matrices. This may be due in part to the use of a suboptimal test statistic in the Manly (1986) paper. In the 1997 edition of his book (p. 180), Manly modified the testing procedure, recommending to use the pivotal statistic $t_i = b_i/\mathrm{SE}(b_i)$ where $\mathrm{SE}(b_i)$ is the standard error of b_i, instead of b_i as the test statistic in the permutation test.

• Smouse *et al.* (1986) suggested two ways of testing a partial Mantel statistic r_M(**AB.C**). Their first method was the same as the original regression method of Manly (previous point). The second was to compute the residuals of the partial correlations of **A** on **C** and of **B** on **C**, then

carry out a simple Mantel test between the two residual matrices. Later, Oden & Sokal (1992) used numerical simulations to compare three methods of partial Mantel analysis in the situation where two data sets were spatially autocorrelated, but not correlated to each other. They devised two ways of simulating spatially autocorrelated data. The first one implemented the isolation-by-distance model of population genetics, the second consisted in simulating spatial autocorrelation in matrices **A** and **B** using a spatial covariance matrix that specified the relationships among the points, which were exponentially declining with geographic distance. The simulations showed that the type I error rates of the three partial Mantel testing procedures were inflated in the presence of strong spatial autocorrelation.

• Legendre (2000) used simulations to compare the type I error rates and powers of four permutation methods used for testing the correlation among distance matrices in partial Mantel tests. His simulations did not involve spatially autocorrelated data. The other studies that involved spatially autocorrelated data had simulated vectors or matrices of spatially autocorrelated data, which were then turned into distance matrices and analysed with a matrix **C** of geographic relationships as covariables. In the Legendre (2000) simulations on the contrary, three distance matrices were independently generated and were then correlated to one another using a correlation model. In that way, the tests of significance were really about correlations *between the distance matrices*, on which the null hypothesis of the Mantel test is based; the Mantel test does not test a hypothesis about correlation in the raw data. The simulation results showed that three of the permutation methods under study were appropriate, to the exclusion of method 3 (which was the second method of Smouse *et al.*, 1986).

• Following a NCEAS working group[1], Legendre *et al.* (2002) published extensive simulation results showing that the test of significance of the Pearson correlation coefficient between two variables that were uncorrelated to each other but strongly spatially correlated (due either to induced spatial dependence or to true spatial autocorrelation) had inflated levels of type I error, which made the test invalid, and that this effect disappeared when only one of the variables was spatially correlated. These results confirmed the less extensive simulation results of Bivand (1980). They also showed that Dutilleul's (1993) modified *t*-test for the correlation coefficient, which takes the spatial correlation of the variables into account, effectively corrected for the spatial correlation in the data and produced results with correct levels of type I error.

• Castellano & Balletto (2002) used simulations to show that the type I error of the partial Mantel test was correct. These authors used incorrect partial testing procedures, so their conclusions remain doubtful. That paper started an exchange in the literature, which is discussed in Appendix 3 of the Legendre & Fortin (2010) paper.

• In 2005, Legendre *et al.* published a paper comparing canonical redundancy analysis (RDA) to Mantel tests for the analysis of simulated multivariate, spatially structured data. The questions were to determine (1) which method had the highest power to detect a relationship between the two data sets when they were spatially autocorrelated and (2) which method had the highest power to detect spatial structures in the data. The two data sets represented community

[1] Working Group "Integrating the Statistical Modeling of Spatial Data in Ecology", 1999-2000, supported by the National Center for Ecological Analysis and Synthesis (NCEAS), a Center funded by NSF (Grant # DEB-94-21535), the University of California at Santa Barbara, and the State of California, USA.

composition data and environmental variables. The simulation results showed that partial RDA had much higher power than partial Mantel tests (1) to detect relationships between the two data sets when they were related and (2) to detect the presence of spatial structures in the species (response) data. In a follow-up paper (2008), additional simulations showed identical results for community composition data generated using Hubbell's neutral model. The main conclusion of these papers was that the Mantel test is inappropriate to test hypotheses concerning correlations in raw data; its use should be restricted to the study of correlations between the distances in distance matrices. – These papers did not study the combination that was of interest in the Guillot & Rousset (2013) paper, i.e. two unrelated data sets that were both spatially autocorrelated. The 2005 paper showed, however, that simple RDA and the simple Mantel test had correct levels of type I error when the two data sets were unrelated to each other and one of them was spatially autocorrelated.

• The Legendre & Fortin (2010) paper was concerned with the relative powers of the Pearson and Mantel correlations (r and r_M) for the study of genetic data. Preliminary simulations that did not involve spatially correlated data showed (again) that both methods had correct levels of type I error. More importantly, the simulations showed that in tests of significance of the relationship between simple variables and multivariate data tables, the power of linear correlation, regression and canonical analysis was far greater than that of the Mantel test and derived forms, meaning that the former methods are much more likely than the latter to detect a relationship when one is present in the data. Examples of difference in power are given for the detection of spatial gradients. Furthermore, the Mantel test does not correctly estimate the proportion of the original data variation explained by spatial structures. The Mantel test should not be used as a general method for the investigation of linear relationships or spatial structures in univariate or multivariate data. Its use should be restricted to tests of hypotheses that can only be formulated in terms of distances. An example of a study where the hypotheses clearly and only involved distances is Le Boulengé *et al.* (1996).

• Finally, using simulations, Guillot & Rousset (2013) found that simple Mantel tests between two autocorrelated variables that were not correlated to each other had inflated rates of type I error, a result in the same line as those obtained for Pearson correlations by Bivand (1980) and by Legendre *et al.* (2002). That result is expected for two vectors or matrices of raw data that are not independent and identically distributed (abbreviated *i.i.d.*). For the same kind of data, they also found that partial Mantel tests did not adequately correct for the presence of spatial structures, so that the tests also had inflated type I error rates. The simulations carried out by Guillot & Rousset (*ibid*.) for partial Mantel tests are more detailed than those of Oden & Sokal (1992) and they confirm their conclusions. Guillot & Rousset must be commended for their effort, but one would have expected them to state that their conclusion had been published 21 years before, instead of claiming that result as their own.

Except for the Legendre (2000) paper, all the above-mentioned studies, including the Guillot & Rousset (2013) paper, simulated *raw data* that were spatially autocorrelated; they were then transformed into distance matrices for Mantel testing. Admittedly, that corresponds to the way most researchers use partial Mantel tests to assess the relationship between data sets while controlling the type I error rate inflation due to spatial autocorrelation. However, and as mentioned above, the partial Mantel test does not test the correlation between two data sets, but between the distances in two distance matrices. This explains why Oden & Sokal (1992) and Guillot & Rousset (2013) found that it did not correct adequately for spatial correlation in data

and had inflated type I error rates. The fault is not with the partial Mantel test but with the inadequacy of the data that were analysed using that method. In the Legendre (2000) simulations, on the contrary, the partial Mantel test was found to have correct type I error rates because the data that were subjected to it were distances that were intercorrelated (although not spatially correlated), and so they corresponded to the null hypothesis of the test.

References

Bivand, R. (1980) A Monte Carlo study of correlation coefficient estimation with spatially autocorrelated observations. – *Quaestiones Geographicae*, **6**, 5–10.

Castellano S, Balletto E (2002) Is the partial Mantel test inadequate? *Evolution* **56**, 1871–1873.

Cliff, A. D. & Ord, J. K. (1981) *Spatial processes – Models and applications.* Pion, London.

Dutilleul, P. (1993) Modifying the *t* test for assessing the correlation between two spatial processes. *Biometrics*, **49**, 305–314.

Guillot, G. & Rousset, F. (2013) Dismantling the Mantel tests. *Methods in Ecology and Evolution*, **4**,336-344. doi: 10.1111/2041-210x.12018.

Le Boulengé, É., Legendre, P., de le Court, C., Le Boulengé-Nguyen, P. & Languy, M. (1996) Microgeographic morphological differentiation in muskrats. *Journal of Mammalogy*, **77**, 684–701.

Legendre, P. (2000) Comparison of permutation methods for the partial correlation and partial Mantel tests. *Journal of Statistical Computation and Simulation*, **67**, 37–73.

Legendre, P., Borcard, D. & Peres-Neto, P. R. (2005) Analyzing beta diversity: partitioning the spatial variation of community composition data. *Ecological Monographs*, **75**, 435–450.

Legendre, P., Borcard, D. & Peres-Neto, P. R. (2008) Analyzing or explaining beta diversity: Comment. *Ecology*, **89**, 3238–3244.

Legendre, P., Dale, M. R. T., Fortin, M.-J., Gurevitch, J., Hohn, M. & Myers, D. (2002) The consequences of spatial structure for the design and analysis of ecological field surveys. *Ecography*, **25**, 601–615.

Legendre, P. & Fortin, M.-J. (2010) Comparison of the Mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. *Molecular Ecology Resources*, **10**, 831–844.

Legendre, P. & Legendre, L. (2012) *Numerical ecology. 3rd English edition.* Elsevier Science BV, Amsterdam.

Manly, B. F. J. (1986) Randomization and regression methods for testing for associations with geographical, environmental and biological distances between populations. *Researches on Population Ecology*, **28**, 201–218.

Manly, B. F. J. (1997) *Randomization, bootstrap and Monte Carlo methods in biology. 2nd edition.* Chapman and Hall, London.

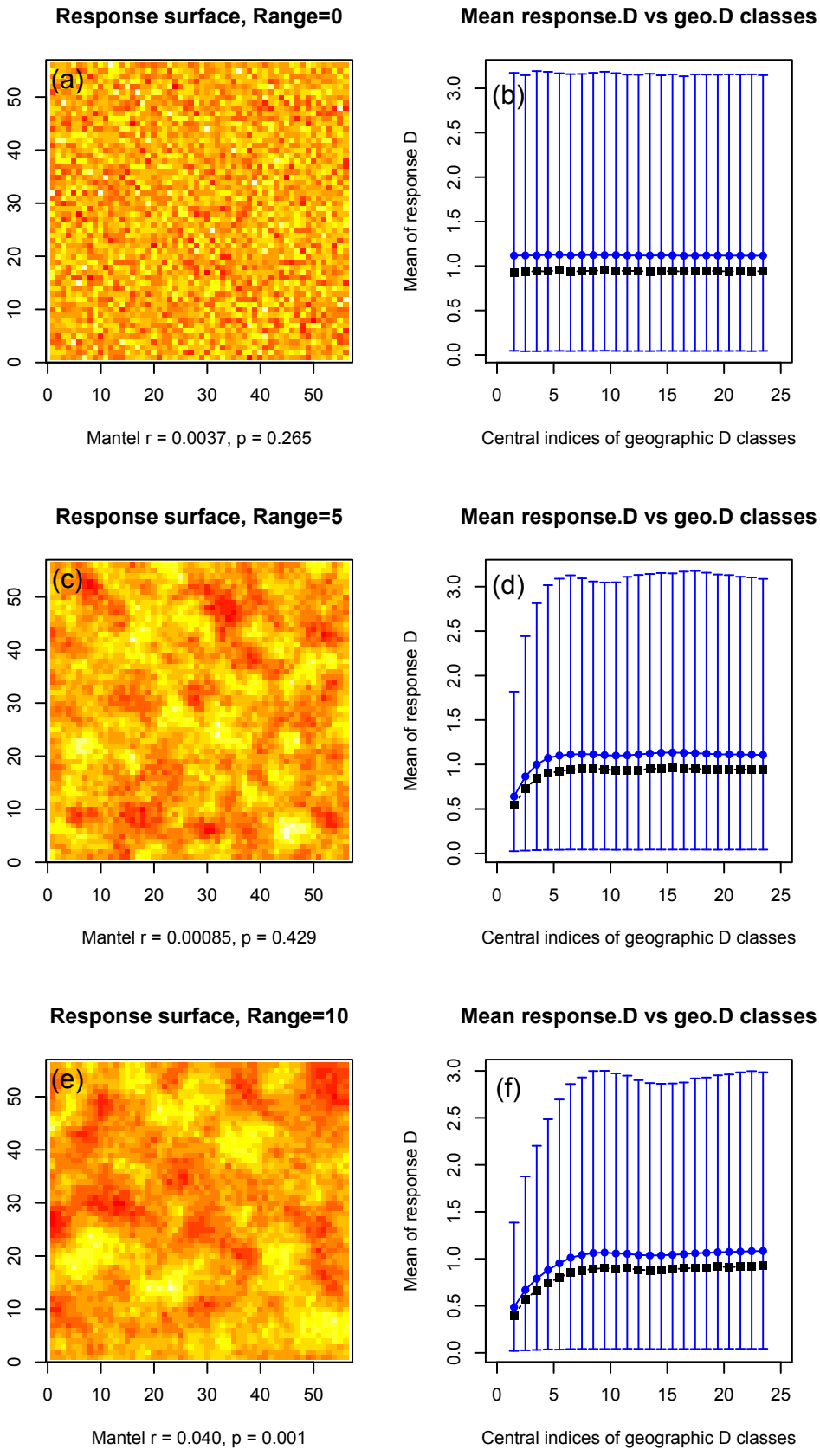
Oden, N. L. & Sokal, R. R. (1992) An investigation of three-matrix permutation tests. *Journal of Classification*, **9**, 275–290.

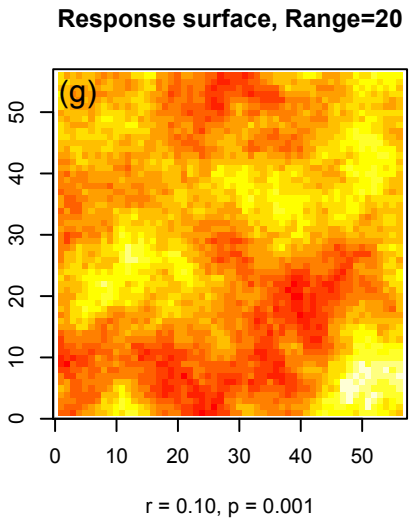
Smouse, P. E., J. C. Long & R. R. Sokal. 1986. Multiple regression and correlation extensions of the Mantel test of matrix correspondence. *Syst. Zool.* 35: 627–632.

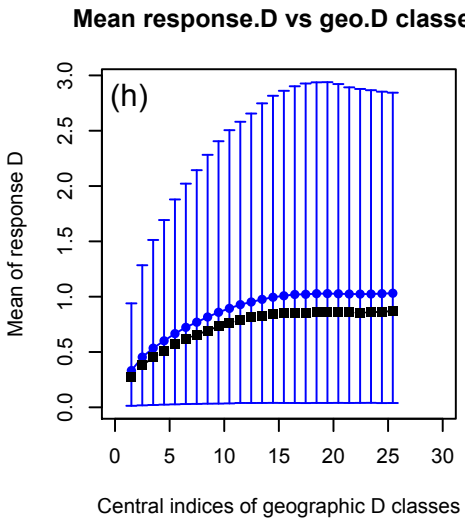
Appendix S2

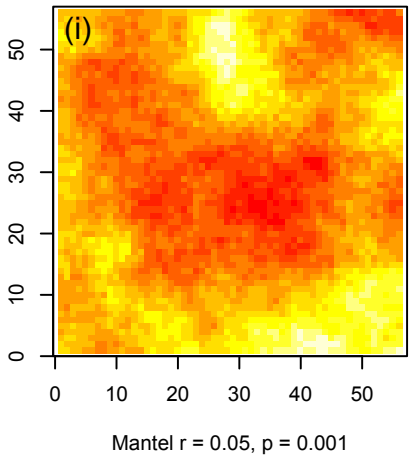
ANALYSIS OF SIMULATED RANDOM AUTOCORRELATED SURFACES

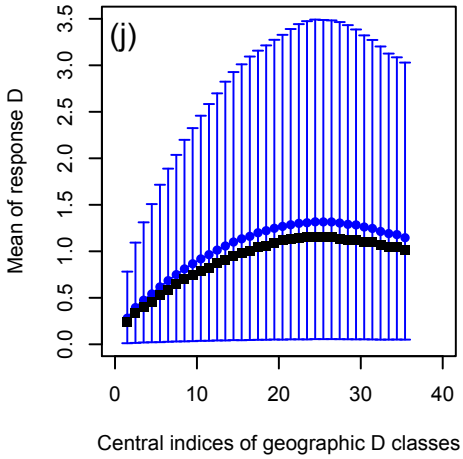
Random autocorrelated variables were generated by random Gaussian field simulations on a (56 × 56 pixel) grid with autocorrelation controlled by a spherical variogram with range values of {0,5,10,20,30,40,50,60,70} pixel units. The values at 100 points forming a regular grid were sampled from each surface, with horizontal and vertical spacing of 5 pixel units. Examples of simulated surfaces are shown and analysed in Fig. S2.1 of this appendix.

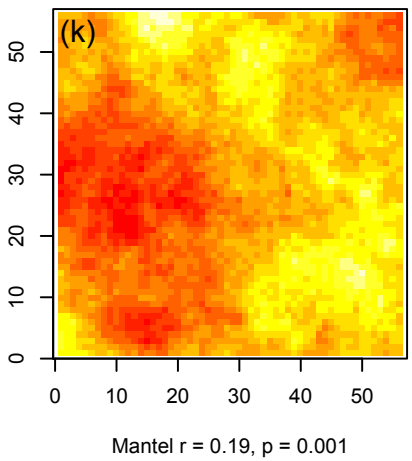

Response surface, Range=0
Mean response.D vs geo.D classes
(a)
(b)
Mean of response D
Mantel r = 0.0037, p = 0.265
Central indices of geographic D classes
Response surface, Range=5
Mean response.D vs geo.D classes
(c)
(d)
Mean of response D
Mantel r = 0.00085, p = 0.429
Central indices of geographic D classes
Response surface, Range=10
Mean response.D vs geo.D classes
(e)
(f)
Mean of response D
Mantel r = 0.040, p = 0.001
Central indices of geographic D classes

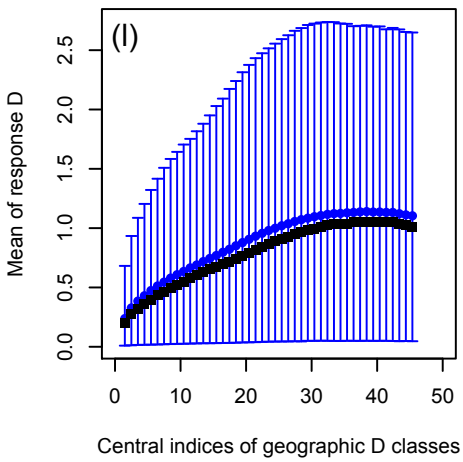

Response surface, Range=20
(g)
0
10
20
30
40
50
r = 0.10, p = 0.001


Mean response.D vs geo.D classes
(h)
Mean of response D
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
5
10
15
20
25
30
Central indices of geographic D classes


Response surface, Range=30
(i)
0
10
20
30
40
50
Mantel r = 0.05, p = 0.001


Mean response.D vs geo.D classes
(j)
Mean of response D
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
10
20
30
40
Central indices of geographic D classes


Response surface, Range=40
(k)
0
10
20
30
40
50
Mantel r = 0.19, p = 0.001


Mean response.D vs geo.D classes
(l)
Mean of response D
0.0
0.5
1.0
1.5
2.0
2.5
0
10
20
30
40
50
Central indices of geographic D classes

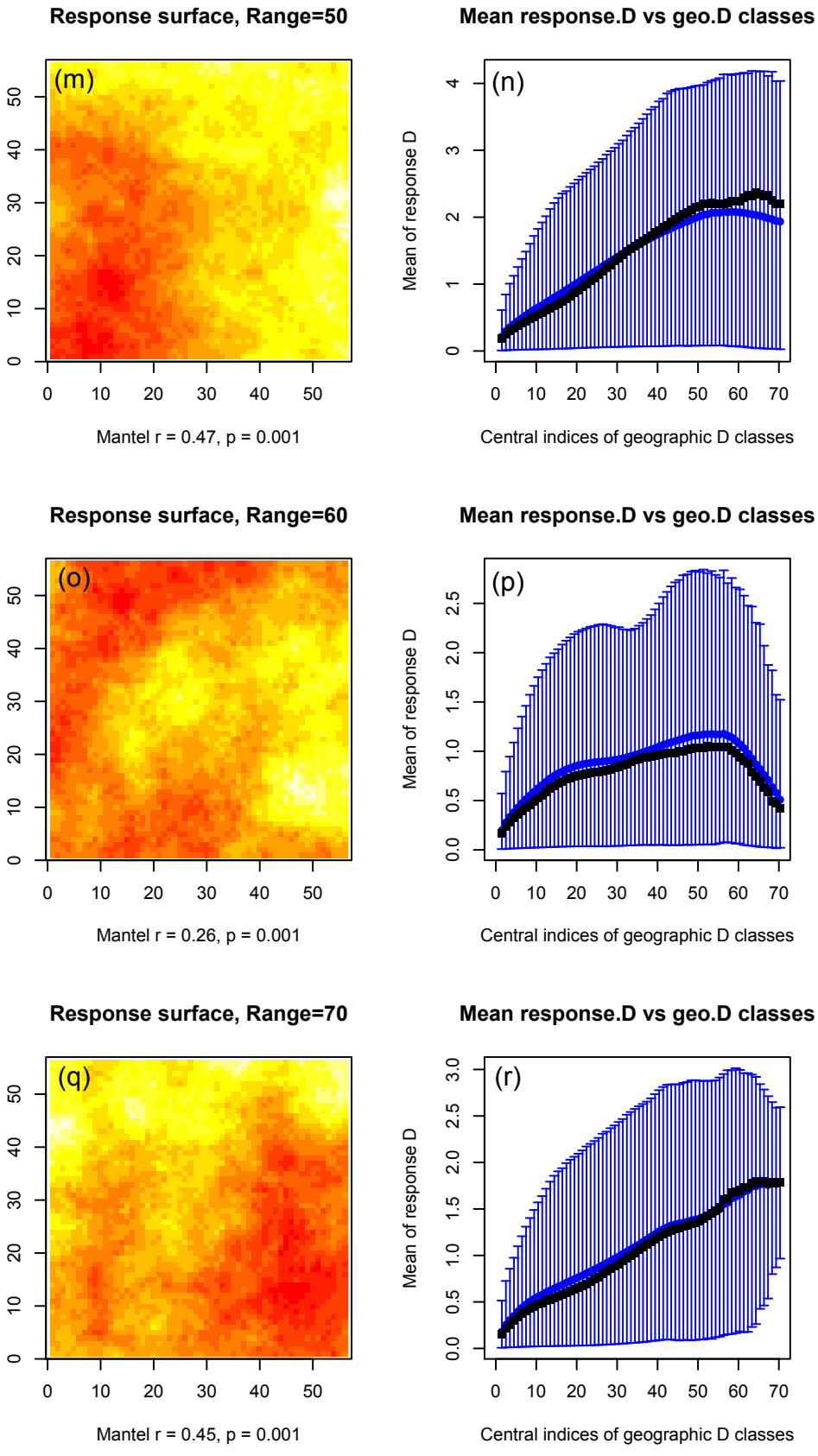

Response surface, Range=50
(m)
Mantel r = 0.47, p = 0.001
Mean response.D vs geo.D classes
(n)
Mean of response D
Central indices of geographic D classes
Response surface, Range=60
(o)
Mantel r = 0.26, p = 0.001
Mean response.D vs geo.D classes
(p)
Mean of response D
Central indices of geographic D classes
Response surface, Range=70
(q)
Mantel r = 0.45, p = 0.001
Mean response.D vs geo.D classes
(r)
Mean of response D
Central indices of geographic D classes

Fig. S2.1. For each value of variogram range (Range = 0 to 70), one of the possible simulated surfaces is shown on the left, using a colour scale from dark red (low values) to pale yellow (high values). Right: a plot similar to Fig. 1c of the main paper is presented, where the means (circles) and medians (squares) of the dissimilarities computed from the simulated values of the response variable are shown as a function of the geographic distance classes. Each mean value is accompanied by its empirical 95% coverage interval. Mantel *r* and *p*-values of the **D**-**D** comparisons are shown underneath the response surface maps.

Appendix S3

SERIES 3 SIMULATIONS INVOLVING DELAUNAY TRIANGULATIONS

The purpose of the third series of simulations was to reproduce circumstances where the number of pairwise comparisons among sites is reduced to reflect species predation avoidance behaviour or limited dispersal abilities in fragmented landscapes (Braunisch *et al.* 2010). In such cases, species move through fragmented landscapes using nearby patches in a stepping-stone fashion instead of moving across inhospitable areas over long distances. A reduced number of pairwise steps among sites can also occur in translocation experiment field studies (Bélisle *et al.* 2001); likewise, in studies of invasive species that have not invaded the whole study area or the spread of genetically modified organisms (GMO) that hybridize with wild or unmodified cultivated forms in contact zones. To represent a limited number of edges linking sites, different types of networks have been developed, going from the minimum spanning tree (Gower & Ross 1969; Urban & Keitt 2001), to the more complex relative neighbourhood graph (Toussaint 1980), Gabriel graph (Gabriel & Sokal 1969; Naujokaitis-Lewis *et al.* 2013), and finally the Delaunay triangulation (Dirichlet 1850) and its landscape equivalent, the minimum planar graph (Fall *et al.* 2007). The first four form a nested series of networks with increasing connectedness (Toussaint 1980). By far the most commonly used algorithm in landscape ecology and genetics is the Delaunay network (e.g. Goldberg & Waits 2010; Koen *et al.* 2012) to represent the stepping-stone behaviour of species (Saura *et al.* 2014). These Delaunay networks can then be truncated to match the species dispersal ability.

So we carried out a third series of simulations to study the power of truncated Delaunay networks. The edges of a Delaunay triangulation represent the spatial relationships between points. For each simulated surface, a random sample of $N = 50$ points was selected from the regular grid of 100 points, and a Delaunay triangulation and a matrix of dbMEM eigenfunctions were computed for the selected points.

Then, a "graph distance matrix" was computed along the edges of each Delaunay triangulation using function delaunay.distance() of the **spatstat** package (Baddeley *et al.* 2014). The distance was *the number of edges* along the shortest path between any pair of points in the connection network. In studies as those described above, researchers may choose to truncate the distance matrix, replacing any value larger than a selected threshold by a large distance, which can be the largest distance actually found in the matrix max.D (that was the case in our simulations), or by some larger distance chosen by the user. The following values of truncation threshold were used: *thresh* = {1, 3, 5, 10}). Since no graph distance was ever larger than 10 in our simulations, the simulations with *thresh* = 10 used the full graph distance matrix without any truncation. The autocorrelated data sets were analysed with respect to geography through a dbMEM analysis using the full set of eigenfunctions modelling positive spatial correlation, as in series 1 simulations, and a Mantel test using the truncated Delaunay "graph distance matrix".

Fig. S3.1 shows the mean rejection rates obtained in the various simulations, for different amounts of SA (variogram ranges, above) and different truncation thresholds (*thresh* = {1, 3, 5, 10}). The following observations can be made:

• For Delaunay graph distance matrices truncated at distance 1, the results of the Mantel test had the same rejection rates as the dbMEM method for all amounts of SA in the response data. In

these matrices, only two distance classes are present: 1 and max.D. The latter was the largest value encountered in the Delaunay graph distance matrix; it varied from 7 to 10 in our simulations, depending on the selected subset of 50 points. This form of Mantel test is the same as that conducted for the first distance class in a Mantel correlogram. This equivalence of the two tests was encountered in the series 2 simulation results. More about this in the Discussion of the main paper.

• For truncation levels larger than 1 (i.e. *thresh* = {3, 5, 10}), the power of the Mantel test was always much lower than that of the dbMEM test, except when no spatial autocorrelation was present in the data, which occurred when the range of the variogram was smaller than (range = 0) or equal to (range = 5) the spacing between the points of the regular grid; that spacing determined the closest possible spacing of pairs of points in the Delaunay triangulation.

The point where the power of the dbMEM test comes near 1 in Fig. S3.1a (i.e. when range = 20) cannot be compared to that same point in Figs. 2a and 3a (where power is nearly 1 at range = 10) because 100 data points were used in Figs. 2a and 3a whereas Fig. S3.1a used only 50 points.

Fig. S3.1b shows that the R^2 of the Mantel test is always smaller than the R^2 and R^2_{adj} of the dbMEM analysis. They are both equal to 0 when there is no spatial structure in the data, i.e. when the range of the variogram was 0 or 5 in our simulations. This simply illustrates the well-established fact that the R^2 of regression or canonical analysis is unrelated to the R^2 of the Mantel test, as discussed in a previous paper (Legendre & Fortin 2010).

References

Baddeley, A., Turner, R. & Rubak, E. (2014) *spatstat: Spatial point pattern analysis, model-fitting, simulation, tests.* R package version 1.38-1. http://cran.r-project.org/package=spatstat.

Bélisle, M., Desrochers, A. & Fortin, M.-J. (2001) Landscape barriers to forest birds: a test with homing experiments. *Ecology*, **82**, 1893–1904.

Braunisch, V., Segelbacher, G. & Hirzel, A. (2010) Modelling functional landscape connectivity from genetic population structure: a new spatially explicit approach. *Molecular Ecology*, **19**, 3664–3678.

Dirichlet, G.L. (1850) Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal für die reine und angewandte Mathematik*, **40**, 209–234.

Fall, A., Fortin, M.-J., Manseau, M. & O'Brien, D. (2007) Spatial Graphs: Principles and Applications for Habitat Connectivity. *Ecosystems*, **10**, 448–461.

Gabriel, K.R. & R. R. Sokal, R.R. (1969) A new statistical approach to geographic variation analysis. *Systematic Zoology*, **18**, 259–278.

Goldberg, C.S. & Waits, L.P. (2010) Comparative landscape genetics of two pond-breeding amphibian species in a highly modified agricultural landscape. *Molecular Ecology*, **19**, 3650–3663.

Gower, J.C. & Ross, G.J.S. (1969) Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, **18**: 54–64.

Koen, E.L., Bowman, J., Garroway, C.J., Mills, S.C. & Wilson, P.J. (2012). Landscape resistance and American marten gene flow. *Landscape Ecology*, **27**, 29–43.

Legendre, P. & Fortin, M.-J. (2010) Comparison of the Mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. *Molecular Ecology Resources*, **10**, 831–844.

Naujokaitis-Lewis, I., Rico, Y., Lovell, J., Fortin, M.-J., & Murphy, M. (2013) Implications of incomplete networks on estimation of landscape genetic connectivity. *Conservation Genetics*, **14**, 287–298.

Saura, S., Bodin, Ö. & Fortin, M.-J. (2014) Stepping stones are crucial for species' long-distance dispersal and range expansion through habitat networks. *Journal of Applied Ecology*, **51**, 171–182.

Toussaint, G. (1980) The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, **12**: 261–268.

Urban, D. & Keitt, T. (2001) Landscape connectivity: a graph-theoretic perspective. *Ecology*, **82**, 1205–1218.

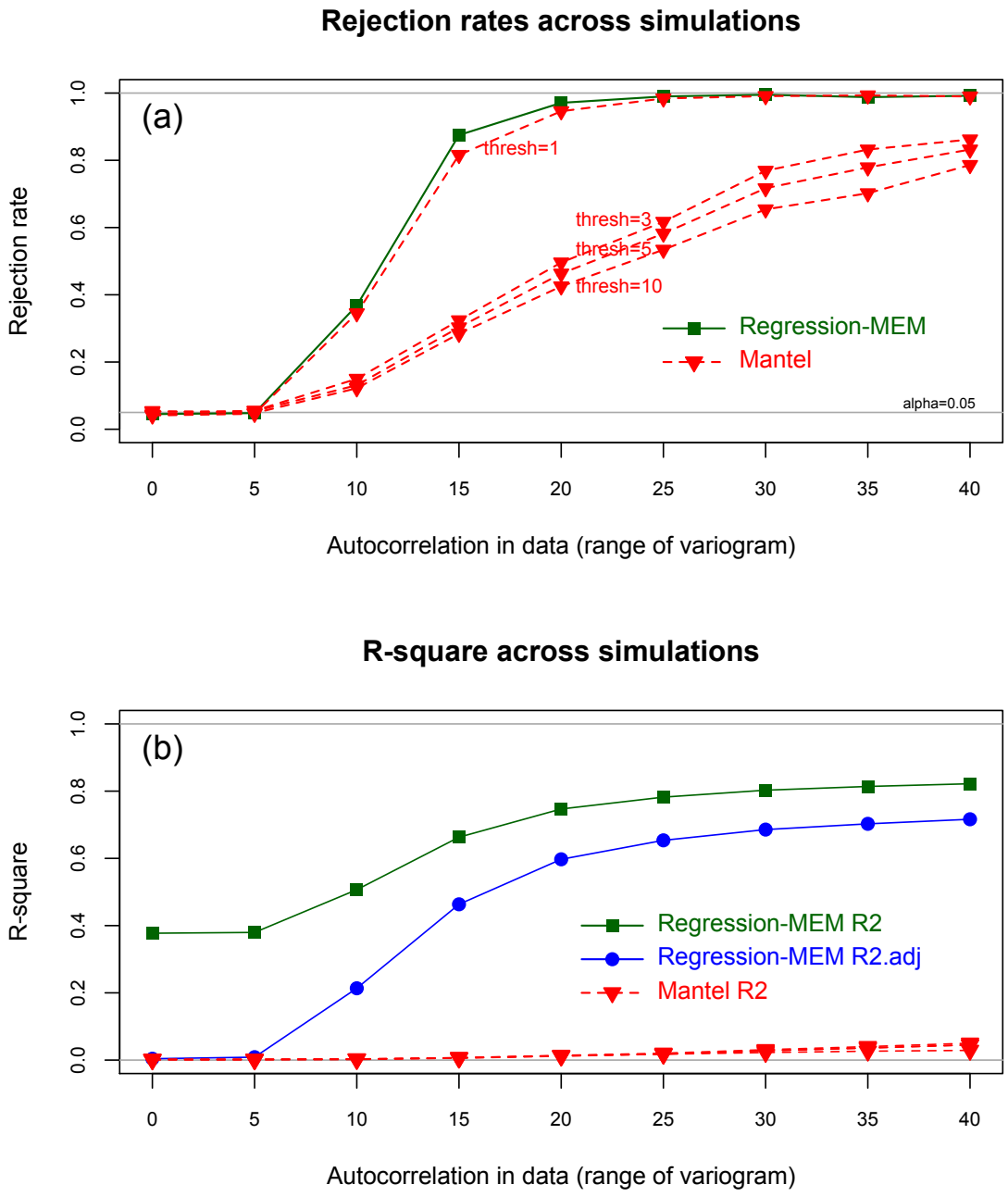


Fig. S3.1. (a) Rejection rates (i.e. number of rejections of H_0 divided by the number of simulations, which was 1000) of the regression–MEM and Mantel tests as a function of the degree of autocorrelation (variogram range) in the simulated data involving Delaunay triangulations with various truncation levels (*thresh* = {1, 3, 5, 10}; *thresh* = 10 produces no truncation). (b) Median *R*-squares of the two methods of analysis. The median adjusted *R*-square (R^2_{adj}) of the regression–MEM test, which is an unbiased estimate of the explained variation, is also shown. No R^2_{adj} statistic is available for Mantel tests.

Appendix S4

REGRESSION ON A GEOGRAPHIC DISTANCE MATRIX DOES NOT CONTROL FOR SA IN DATA

This Appendix offers a proof-by-example that regression on a geographic distance matrix does not control for the spatial correlation that may be found in response data.

Legendre *et al.* (2005) have already shown that spatial variation is, at best, weakly captured by regressing a response distance (or dissimilarity) matrix on a geographic distance matrix, whatever its form: D(XY), D(3rd degree polynomial of X and Y), or ln(D(XY)), where XY is a matrix of geographic coordinates. In this Appendix, we address the other facet of this question, which is: *Does regression on a geographic distance matrix remove the spatial correlation in response data, producing residuals without spatial correlation?*

We will approach this question using example data. We wrote an R function, called gen.SA.data(), that readers can use to generate a variable of simulated spatially autocorrelated data over a geographic surface (regular grid). The function is provided at the end of this Appendix. The calculation steps are the following:

1. Check that the data do indeed contain spatially correlated data. To that aim, we apply dbMEM analysis to the response data using multiple linear regression, since there is a single response variable. If the data were multivariate, redundancy analysis (RDA) would be used instead of multiple regression. MEM analysis has been proven capable of identifying spatial correlation in response data (i.e. spatial structures of various kinds, be they the result of a process producing spatial autocorrelation in data, or the result of spatial dependence induced by explanatory variables); this has been demonstrated in Borcard & Legendre (2002), Borcard *et al.* (2004), Dray *et al.* (2006) and Legendre & Legendre (2012, Chapter 14). We make note of the R-square, adjusted R-square and p-value of the MEM analysis as proof of the presence of spatial correlation in the generated data. The function outputs a data matrix ("Surface") from which a colour map of the generated data can be drawn.

2. The function then carries out Mantel tests of the generated response data vs. geographic distances expressed in various forms: raw geographic distances (D(XY)), square-rooted distances (sqrt(D(XY))), and $\log_e$-transformed distances (ln(D(XY))). The R-square and p-value of each Mantel analysis is noted.

3. From the output of the function, one can recuperate the generated data and the grid coordinates and compute distance matrices (see the *Example run* below). One can then compute a regression of the response distances on the geographic distances for each form: raw, square-rooted and log distances.

4. We will focus on the fitted values and the residuals of these regressions. The residuals are not really distances (about half of the values are negative), but they can still be tested for the presence of spatial correlation. The residual distances are regressed on the set of MEM eigenfunctions recuperated from the gen.SA.data() function. Testing is carried out using the test of significance proposed by McArdle & Anderson (2001), which uses a response distance matrix directly for the calculation. We examine the R-square, adjusted R-square and p-value of the permutation tests of both the fitted distances and the residual distances.

5. If the test of the residual distances is not significant, it means that no significant spatial correlation has been identified in the residual distances. If the test is significant, and especially, if the adjusted R-square is high, we must conclude that spatial correlation is still present in the response data after residualization by regression on the various kinds of geographic distance matrices. If the test is significant but the adjusted R-square is lower than that of the regression of the generated data on MEM eigenfunctions, it means that regression on the geographic distance matrix has controlled the spatial correlation in the response data less efficiently than regression on MEM eigenfunctions.

The function and additional calculations will now be run to provide an example of the analysis. Readers are invited to run the function again (it will generate a different spatially autocorrelated surface every time) and check that the results are similar to those reported here.

Example run

1. Load the files containing the gen.SA.data() function, the dbRDA.D() function, and the necessary R packages. The list of packages and their locations are given in the *Details* paragraph of the gen.SA.data() function documentation.

Functions gen.SA.data.R and dbRDA.D.R are included in this Appendix, from which they can be copied to text files and loaded to the R console.

2. Function gen.SA.data() generates spatially autocorrelated data on a surface (square grid) and analyses it using MEM (by multiple linear regression) and Mantel tests.

Generate a 20×20 spatially autocorrelated (SA) surface (400 points) with variogram range = 5.
Only the MEM eigenfunctions modelling positive SA are kept for regression analysis.

res = gen.SA.data(nx=20, range=5, var=1, nperm1=999, nperm2=999)

```
summary(res)
           Length Class      Mode
R2.dbMEM     2    -none-     numeric
p.dbMEM      2    -none-     numeric
R2.Mantel    3    -none-     numeric
p.Mantel     3    -none-     numeric
Surface    400    -none-     numeric
surf       400    -none-     numeric
grid.coord   2    data.frame list
dbMEM      189    data.frame list
```

dim(res$dbMEM)
# [1] 400 189

res$R2.dbMEM # R-square, adjusted R-square of MEM analysis by regression
[1] 0.9176085 0.8434561
res$p.dbMEM # Parametric and permutational p-values
[1] 1.188061e-59 1.000000e-03

The presence of significant spatial autocorrelation in the response data is confirmed

res$R2.Mantel # Mantel R-square for geographic D, sqrt(D), and ln(D)
[1] 0.0008160981 0.0012585326 0.0020933597
res$p.Mantel # Mantel p-values for geographic D, sqrt(D), and ln(D)

```
# [1] 0.038   0.012   0.003

# Mantel analysis explains a very small fraction of the response distance matrix variation
```

3. Plot a map of the original random autocorrelated surface

```
require(graphics)

image(1:20, 1:20, t(res$Surface), main="Map of response surface, Range=5", xlim=c(0,21),
ylim=c(0,21), xlab="", ylab="", asp=1)
```

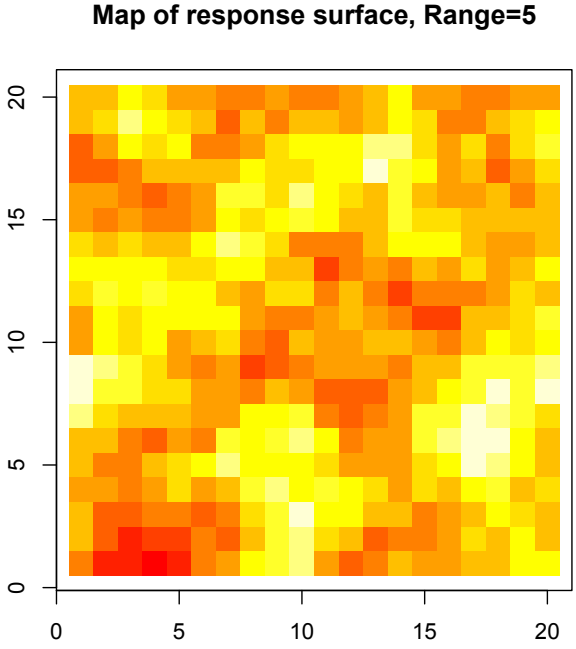


Fig. S4.1. Map of the response surface generated on a 20×20 grid for this example.

4. Check that the response data, expressed in distance matrix form, displays significant spatial structure. This analysis will confirm that the dbRDA.D() function produces the same R-square, adjusted R-square and p-value as regular regression (results are shown above).

```
# Load function dbRDA.D() to compute RDA following McArdle & Anderson (2001), using a
dissimilarity matrix and a matrix of explanatory variables.

res0.dbRDA = dbRDA.D(dist(res$surf), res$dbMEM, nperm=999, compute.eig=FALSE)
# Rank of X centred = 189

res0.dbRDA$Rsquare          # R-square and adjusted R-square; same results as above
# [1] 0.9176085   0.8434561
res0.dbRDA$P.perm           # Permutational p-value
# [1] 0.001
```

5. Compute the fitted and residual distances and test them for spatial correlation

```
# First, compute vectors containing the two distance matrices, unfolded
```

```
Y.D.vec = as.vector(dist(res$surf))                # Response data D, vector length = 79800

XY.D.vec = as.vector(dist(res$grid.coord))         # Geographic D, vector length = 79800

# Regress response distances (Y.D) on simple geographic distances (XY.D) in vector form

res.lm.D = lm(Y.D.vec ~ XY.D.vec)

summary(res.lm.D)$r.squared                        # Same as Mantel R-square above
# [1] 0.0008160981
```

5.1. Fitted distances

```
fitted.D = fitted(res.lm.D)                        # Length of the vector: 79800

range(fitted.D)
# [1] 1.039841    1.161839

# Turn the fitted dissimilarities into a matrix with class "dist"

fitted.D.mat <- as.dist(matrix(NA,400,400))
fitted.D.mat[] <- fitted.D

# Do the fitted distances contain SA? dbRDA.D of the fitted distance matrix against MEM.

res.dbRDA.fit = dbRDA.D(fitted.D.mat, res$dbMEM, nperm=999, compute.eig=FALSE)

res.dbRDA.fit$Rsquare                              # Fitted D: R-square and adjusted R-square
# [1] 0.51806244    0.08431864

res.dbRDA.fit$P.perm                               # Permutational p-value
[1] 0.001

# The fitted distances do account for a small albeit significant amount of SA, which was present
in the generated data.
```

5.2. Residual distances

```
resid.D = residuals(res.lm.D)                      # Length of the vector: 79800

range(resid.D)
# [1] -1.146697    4.388709

length(which(resid.D < 0)) / (200*399)             # 0.5735589
# In this particular example, 57.4% of the residual "distances" were negative.

hist(resid.D)

# Turn the residual dissimilarities into a matrix with class "dist"

resid.D.mat = as.dist(matrix(NA,400,400))
resid.D.mat[] = resid.D

# Do the residual distances contain SA? dbRDA of the residual distance matrix against MEM.
```

```
# Load function dbRDA.D() to compute RDA following McArdle & Anderson (2001), using a dissimilarity matrix and a matrix of explanatory variables.

res.dbRDA = dbRDA.D(resid.D.mat, res$dbMEM, nperm=999, compute.eig=FALSE)

summary(res.dbRDA)

res.dbRDA$Rsquare                    # Residual D: R-square and adjusted R-square
# [1] 0.7151880   0.4588571
res.dbRDA$P.perm                     # Permutational p-value
# [1] 0.001

# The residual distances do contain significant SA in substantial amount.
```

Readers are invited to regress the response distances (Y.D) on the square-rooted (sqrt(XY.D)) and log-transformed (ln(XY.D)) geographic distances, compute matrices of fitted and residual distances from the regressions, and test these distance matrices for presence of SA by MEM analysis using the dbRDA.D() function, following the script provided above where Y.D is regressed on XY.D. The detailed steps are not presented to save space. They have been computed, however, and the results are assembled in Table S4.1 for the random autocorrelated response data generated during our run of the gen.SA.data() function. The adjusted R-squares of the dbMEM.D analyses against MEM eigenfunctions are the statistics of interest because they represent unbiased estimates of the variance of the response data explained by the MEM eigenfunctions. The results are also presented in the form of a graph (Fig. S4.2).

Table S4.1. Adjusted R-squares resulting from analysis against MEM eigenfunctions of the fitted and residual response distances, obtained by regression on three transformations of the geographic distances: raw geographic distances D(XY), square-rooted distances sqrt(D(XY)), and $\log_e$-transformed distances ln(D(XY)).

	Fitted.D_adj.R2	Residual.D_adj.R2
D(XY)	0.08432	0.45886
sqrt(D(XY))	0.14596	0.51057
ln(D(XY))	0.26962	0.60410

Firstly and foremost, the graph shows that the fitted distances account for a rather small fraction (blue symbols) of the spatial structure of the generated data that can be modelled by MEM eigenfunctions (black horizontal line). A surprisingly high amount of spatial correlation can be modelled by the same MEM eigenfunctions from the residual distances. Regression on any of the three types of geographic distance matrices has not controlled for the spatial structure in the response data since it has left a lot of that structure in the residual distances. Secondly, the graph shows that the square-root and log transformations of the geographic distances, before regression (or Mantel test), offer a slightly better performance in terms of capturing the spatial

autocorrelation of the response data than the original representation of the geographic relationships as a simple distance matrix computed from the geographic coordinates, D(XY), although we observe the surprising consequence that there is also more spatial autocorrelation left in the residuals with the transformed distances. This seems to be due to a distortion of the distances when regressing D(Y) on sqrt(D(XY)) or ln(D(XY)), which causes the appearance of spurious broad-scaled spatial structures in both the fitted values and the residuals. These structures can be identified as broad-scaled because they can be modelled by the first few largest-scaled MEM eigenfunctions. These eigenfunctions model the broad-scaled structures generated in the fitted values and residuals of the regressions on sqrt(D(XY)) and ln(D(XY)) more strongly (meaning: higher R^2) than in the fitted values and residuals of the regression on D(XY). Thus, regression on distance matrices is no better using sqrt(D(XY)) or ln(D(XY)) than using D(XY). A more detailed investigation of this phenomenon would be in order.

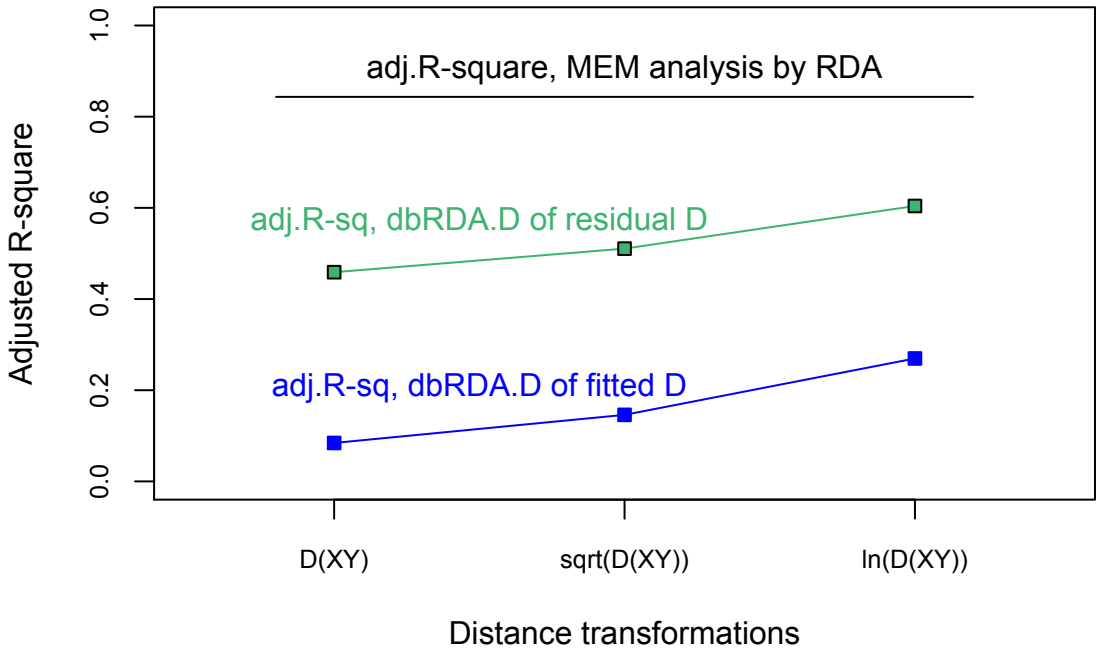


Fig. S4.2. Adjusted R-squares resulting from analysis of the fitted (blue) and residual response distances (green), obtained by regression on three transformations of the geographic distances (abscissa), against MEM eigenfunctions: raw geographic distances D(XY), square-rooted distances sqrt(D(XY)), and $\log_e$-transformed distances ln(D(XY)). The adjusted R-square of the original regression analysis of the generated data against the MEM eigenfunctions is also presented (black horizontal line in the graph); this is the amount of explained variance that can be extracted from the response data by MEM analysis.

In any case, using three representations of the geographic relationships by distance matrices that are commonly used to represent spatial relationships in Mantel tests, the example has shown that the residual distances *did* contain significant spatial autocorrelation (SA). Hence, regression of the response distances on the geographic distance matrices did not "remove" or "control for"

spatial correlation in the response data, as is often assumed by users of Mantel tests or regression on distance matrices.

Readers are invited to generate new response data with the function gen.SA.data() and analyse them at leisure.

References

Borcard, D. & Legendre, P. (2002) All-scale spatial analysis of ecological data by means of principal coordinates of neighbour matrices. *Ecological Modelling*, **153**, 51–68.

Borcard, D., Legendre, P., Avois-Jacquet, C. & Tuomisto, H. (2004) Dissecting the spatial structure of ecological data at multiple scales. *Ecology*, **85**, 1826–1832.

Dray, S., P. Legendre, P. & Peres-Neto, P. R. (2006) Spatial modelling: a comprehensive framework for principal coordinate analysis of neighbour matrices (PCNM). *Ecological Modelling*, **196**, 483-493.

Legendre, P., Borcard, D. & Peres-Neto, P. R. (2005) Analyzing beta diversity: partitioning the spatial variation of community composition data. *Ecological Monographs*, **75**, 435–450.

Legendre, P. & Fortin, M.-J. (2010) Comparison of the Mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. *Molecular Ecology Resources*, **10**, 831–844.

Legendre, P. & Legendre, L. (2012) *Numerical ecology. 3rd English edition.* Elsevier Science BV, Amsterdam.

McArdle, B. H. & Anderson, M. J. (2001) Fitting multivariate models to community data: a comment on distance-based redundancy analysis. *Ecology*, **82**, 290–297.

Generation of random SA data on a square grid

Description

This function generates random spatially autocorrelated (SA) data on a square grid with nx points in the X and Y directions and analyses it with dbMEM eigenfunctions and Mantel tests.

Usage

```
gen.SA.data(nx=20, surf=NULL, range=5, var=1, nperm1=0, nperm2=999)
```

Arguments

nx	Number of rows and columns of the regular grid; n = nx^2.
surf	Vector with values for each point of the response surface. The vector may be a surf vector produced during a previous run of the function, or contain real univariate data about a surface forming a regular square grid. Default: surf=NULL.
range	Range parameter of the variogram for generation of SA on the surface.
var	Total variance of the simulated data.
nperm1	Number of permutations for dbMEM analysis by regression. Default: nperm1=0.
nperm2	Number of permutations for the Mantel tests. Default: nperm2=999.

Details

Required packages:

• Load the {RandomFields} and {vegan} available from CRAN.

• Load package {PCNM}. The source code (for Linux) and compiled Windows file are available from https://r-forge.r-project.org/R/?group_id=195. A Mac OSX file compiled for R 3.0.x is available on http://adn.biol.umontreal.ca/~numericalecology/Rcode/.

If "surf" is NULL, a surface is simulated by Gaussian random fields controlled by a spherical variogram:

If "surf" is not NULL and a "surf" vector of the correct size is provided (length=nx^2), containing values for each point of the response surface, the dbMEM and Mantel analyses will be carried out on that surface.

Value

Function `gen.SA.data` returns a list containing the following results:

`R2.dbMEM`	R-square and adjusted R-square of dbMEM analysis by regression.
`p.dbMEM`	p-values (parametric and permutational) of dbMEM analysis by regression.
`R2.Mantel`	Mantel correlations squared with geo.D, sqrt(geo.D) and log(geo.D).
`p.Mantel`	Mantel test p-values with geo.D, sqrt(geo.D) and log(geo.D).
`Surface`	The generated surface presented as a matrix.
`surf`	The generated data presented as a vector.
`grid.coord`	A matrix with grid coordinates, from which D.geo can be recomputed.
`dbMEM`	The matrix of dbMEM spatial eigenvectors.

References

Legendre, P. & Legendre, L. (2012) *Numerical Ecology, 3rd English edition*. Elsevier Science BV, Amsterdam.

Schlather, M., Malinowski, A., Oesting, M., Boecker, D., Strokorb, K., Engelke, S., Martini, J., Menck, P., Gross, S., Burmeister, K., Manitz, J., Singleton, R., Pfaff, B. and R Core Team (2014). *RandomFields: Simulation and Analysis of Random Fields.* R package version 3.0.10. http://CRAN.R-project.org/package=RandomFields

Author

Pierre Legendre, Département de sciences biologiques, Université de Montréal
License: GPL (>=2)

Example

Generate a new random 20x20 spatially autocorrelated (SA) surface with variogram range = 5.

res = gen.SA.data(nx=20, range=5, var=1, nperm1=999, nperm2=999)

Reanalyse the surface generated during the previous run. A new surface is not generated if parameter `surf` is not `NULL`. The `surf` vector provided must be of the correct length, nx^2.

res2 = gen.SA.data(nx=20, surf=res$surf, range=5, var=1, nperm1=999, nperm2=999)

The vector may be a `surf` vector produced during a previous run of the function, as in this example, or contain univariate data on some other surface forming a regular square grid.

```
gen.SA.data <- function(nx=20, surf=NULL, range=5, var=1, nperm1=0, nperm2=999)
{
# Load packages
    require(RandomFields)
    require(vegan)
    require(PCNM)
    cat("Function RFsimulate() of RandomFields 3.0.10 is used with R 3.0\n")
# Package Random Fields options
    RFoptions(spConform=FALSE)
    RFoptions(seed=NULL)
#
a <- system.time({             # How much time for the calculations?

    coord <- 1:nx
    n <- nx^2
    grid.coord <- expand.grid(coord, coord)
    colnames(grid.coord) <- c("Easting", "Northing")
#
# If no surface is provided, one is generated here
    if(is.null(surf)) {
        surf     <- vector(mode="numeric", length=n)
        ## Simulate autocorrelated surface with spherical variogram model
        model = RMspheric(var=var, scale=range)
        Surface <- RFsimulate(model = model, x=coord, y=coord, grid=TRUE)
        ### If a nugget is sought, argument err.model in RFsimulate
        ### can be used: err.model=RMnugget(var=var)
        #
        for(k in 1:n) surf[k] <- Surface[grid.coord[k,2], grid.coord[k,1]]
        } else {
# else, the "surf" data provided are used for the analysis
        if(length(surf) != n) stop("The 'surf' vector provided is not of length
nx^2")
        Surface <- matrix(surf,nx,nx,byrow=TRUE)
        }
    Y.D <- dist(surf)
#
# Construct dbMEM eigenfunctions. Keep the eigenfunctions modelling positive SA
    geo.D <- dist(grid.coord)
    dbMEM.grid <- PCNM(geo.D,thresh=1,dbMEM=TRUE)
    dbMEM <- dbMEM.grid$vectors
#
# dbMEM analysis by regression. No selection of the eigenfunctions
    lm.res <- lm(surf ~ .,data=as.data.frame(dbMEM))
    R2.dbMEM <- c(summary(lm.res)$r.squared, summary(lm.res)$adj.r.squared)
    F.vec <- summary(lm.res)$fstatistic
    #
    # Compute the regression parametric p.value
    p.dbMEM.temp <- pf(F.vec[1], F.vec[2], F.vec[3], lower.tail=FALSE)[[1]]
    #
    if(nperm1==0) {
        # Output only the parametric regression test p.value
        p.dbMEM <- c(p.dbMEM.temp, NA)
        } else {
        # Output the parametric and permutation test p.values
        p.dbMEM <- c(p.dbMEM.temp,
R2.test.perm(surf,dbMEM,nperm=nperm1,dbMEM=TRUE)$P)
        }
# Mantel test using vegan's mantel() function
    res.Mantel1 <- mantel(Y.D, geo.D, permutations=nperm2)
    res.Mantel2 <- mantel(Y.D, sqrt(geo.D), permutations=nperm2)
```

```
    res.Mantel3 <- mantel(Y.D, log(geo.D), permutations=nperm2)
    R2.Mantel <-
        c(res.Mantel1$statistic^2, res.Mantel2$statistic^2,
res.Mantel3$statistic^2)
    p.Mantel <- c(res.Mantel1$signif, res.Mantel2$signif, res.Mantel3$signif)
})
a[3] <- sprintf("%2f",a[3])
cat("Calculation time =",a[3]," sec",'\n')
#
list(R2.dbMEM=R2.dbMEM, p.dbMEM=p.dbMEM, R2.Mantel=R2.Mantel, p.Mantel=p.Mantel,
Surface=Surface, surf=surf, grid.coord=grid.coord, dbMEM=as.data.frame(dbMEM))
}

R2.test.perm <- function(Y, X, nperm=999, dbMEM=FALSE)
#
# Permutation test for R2 statistic in regression or RDA.
#
# Parameters of the function --
# Y : Matrix or vector of response data.
# X : Matrix of explanatory data, e.g. a file of dbMEM.
# nperm : Number of permutations for the test.
# dbMEM=FALSE: Normal computation of statistics for regression or RDA:
#              Centre X, QR decomposition of X, compute adjusted R-square.
# dbMEM=TRUE : Do not centre X since the matrix of dbMEM is centred.
#              Do not compute the adjusted R-square.
# License: GPL-2
# Author:: Pierre Legendre, June 2014
{
    Y.c <- scale(Y,center=TRUE,scale=FALSE)           # Centre Y
    SS.Y <- sum(Y.c^2)
    X <- as.matrix(X)
    n <- nrow(X)
    if(!dbMEM) X <- scale(X,center=TRUE,scale=FALSE)  # Centre X if not dbMEM
    Q <- qr(X)                                        # QR decompostion of X
    Yfit.X <- qr.fitted(Q, Y.c)                       # Compute fitted values
# Compute statistics
    Rsquare <- sum(Yfit.X^2)/SS.Y
    m <- Q$rank
    residualDF <- n-m-1
    F <- (Rsquare*residualDF)/((1-Rsquare)*m)
    if(dbMEM) { adjRsq <- NA
        } else {
        totalDF <- n-1
        adjRsq <- 1-((1-Rsquare)*totalDF/residualDF) }
# Permutation test of F
    if(nperm > 0) {
        nPGE <- 1
        for(i in 1:nperm) {
            YfitPerm <- qr.fitted(Q, sample(Y.c))
            RsquarePerm <- sum(YfitPerm^2)/SS.Y
            FPerm <- (RsquarePerm*residualDF)/((1-RsquarePerm)*m)
            if(FPerm >= F) nPGE <- nPGE+1
            }
        P <- nPGE/(nperm+1)
        } else { P <- NA }
#
if(dbMEM) { out <- list(P=P)
    } else {
    out <- list(Rsquare=Rsquare, F=F, P=P, adjRsq=adjRsq, nperm=nperm, m=m,
    residualDF=residualDF) }
}
```

dbRDA *F*-test for response data in dissimilarity matrix form

Description

Compute the dbRDA *F*-test of significance between response data represented by a Euclidean or non-Euclidean dissimilarity matrix and a matrix of explanatory variables, using the method of McArdle and Anderson (2001).

Usage

```
dbRDA.D(D, X, nperm=999, option=3, compute.eig=FALSE, coord=FALSE,
        rda.coord=2, pos.RDA.val=FALSE)
```

Arguments

`D`	Dissimilarity matrix (class `matrix` or `dist`) representing the response data. D may be Euclidean or non-Euclidean.
`X`	Rectangular matrix of explanatory variables for the RDA (class `data.frame` or `matrix`). Factors must be recoded as dummy variables or Helmert contrasts.
`option`	`=1` : Original McArdle-Anderson (2001) equation 4. Slow, not recommended. `=2` : McArdle-Anderson equation, simplified. `=3` : Least-squares solution after orthogonalization of X.
`compute.eig`	`=TRUE` : the eigenvalues and eigenvectors of D are computed. Do *not* use with very large matrices (slow).
`coord`	`=TRUE` : compute the principal coordinates corresponding to the positive eigenvalues of D. This option requires `compute.eig=TRUE`.
`rda.coord`	Number of RDA ordination coordinates to compute, for example 2 (default value).
`pos.RDA.val`	`=TRUE` : store only positive RDA eigenvalues in the output list. `=FALSE` : store all RDA eigenvalues in the output list.

Details

Compute the dbRDA *F*-test of significance. The response is represented by a Euclidean or non-Euclidean dissimilarity matrix; X is a matrix of explanatory variables, as in regular RDA.

The *F*-statistic is obtained without prior computation of the eigenvalues and eigenvectors of the dissimilarity matrix, hence no correction has to be made to eliminate the negative eigenvalues. Three computation methods are available, all derived from McArdle and Anderson (2001).

The eigenvalues and eigenvectors of D are computed if `compute.eig=TRUE`. If `coord=TRUE`, the principal coordinates corresponding to the positive eigenvalues of D are computed.

The function may fail to produce a meaningful RDA test of significance and ordination axes if D is extremely non-Euclidean. This is the case with some forms of genomic distances.

Computation options:

option=1 — The original *F*-statistic of McArdle and Anderson (2001), eq. 4:

F = SSYhat / sum(diag(I.minus.H %*% G %*% I.minus.H))

Degrees of freedom are added to this equation when writing the output list.

option=2 — Simplified equation:

F = SSYhat/(SSY-SSYhat)

option=3 — Orthogonalize matrix X by PCA before computing projector H. No inversion.

Compute SSYhat, then F = SSYhat/(SSY-SSYhat)

Options 2 and 3 are equivalent; they require half the computing time of option 1.

Value

Function `dbRDA.D` returns a list containing the following results:

`F`	*F*-statistic.
`Rsquare`	R-square and adjusted R-square statistics.
`P.perm`	Permutational p-value of RDA R-square (test based on F).
`SS.total`	Trace of Gower-centred matrix G. The trace is equal to the total sum of squares of Y and to the sum of the eigenvalues of D.
`PCoA.values`	Eigenvalues (if they are computed, i.e. if `compute.eig=TRUE`).
`PCoA.vectors`	Principal coordinates for the positive eigenvalues of D.
`RDA.values`	RDA eigenvalues.
`RDA.rel.values`	RDA relative eigenvalues.
`RDA.cum.values`	RDA cumulative relative eigenvalues.
`RDA.coord`	Ordination coordinates of objects on selected RDA axes.

References

Legendre, P. (2014) Interpreting the replacement and richness difference components of beta diversity. *Global Ecology and Biogeography*, **23**, 1324–1334.

Legendre, P. & Legendre, L. (2012) *Numerical Ecology, 3rd English edition*. Elsevier Science BV, Amsterdam.

McArdle, B.H. & Anderson, M.J. (2001) Fitting multivariate models to community data: a comment on distance-based redundancy analysis. *Ecology*, **82**, 290–297.

Author

Pierre Legendre, Département de sciences biologiques, Université de Montréal
License: GPL (>=2)

Example

```
# Load function dbRDA.D()

# 1. Analysis of mite data with the percentage difference (alias Bray-Curtis) dissimilarity

require(vegan)
data(mite)
data(mite.env)                          # The first 2 environmental variables are quantitative
sel = c(14,24,31,41,49,64)              # Select 6 sites for the example
mite.BC = vegdist(mite[sel,], "bray")   # The D matrix will produce one negative eigenvalue
res = dbRDA.D(mite.BC, mite.env[sel,1:2], nperm=999, compute.eig=TRUE)
res$Rsquare
res$P.perm
# Plot the ordination on PCoA axes 1 and 2
plot(res$RDA.coord)
text(res$RDA.coord, labels=rownames(mite.env[sel,]), pos=3)

# 2. Compare RDA to dbRDA.D results using Euclidean distance

mite.hel = decostand(mite, "hellinger")
mite.hel.D = dist(mite.hel)
# RDA solution
rda.out = rda(mite.hel ~ SubsDens+WatrCont, data=mite.env)
RsquareAdj(rda.out)
anova(rda.out, step=1000, perm.max=1000)
# dbRDA.D solution
dbRDA.out = dbRDA.D(mite.hel.D, mite.env[,1:2], nperm=999, compute.eig=TRUE)
dbRDA.out$F
dbRDA.out$Rsquare
dbRDA.out$P.perm
```

```
dbRDA.D <- function(D, X, nperm=999, option=3, compute.eig=FALSE, coord=FALSE,
rda.coord=2, positive.RDA.values=FALSE)
{
    D <- as.matrix(D)
    X <- as.matrix(X)
    n <- nrow(D)
    epsilon <- .Machine$double.eps
#
# Gower centring, matrix formula. Legendre & Legendre (2012), equation 9.42
    One <- matrix(1,n,n)
    mat <- diag(n) - One/n
    G <- -0.5 * mat %*% (D^2) %*% mat
    SSY <- sum(diag(G))
    # LCBD <- diag(G)
#
# Principal coordinate analysis after eigenvalue decomposition of D
    if(compute.eig) {
        eig <- eigen(G, symmetric=TRUE)
        values <- eig$values     # All eigenvalues
        vectors <- eig$vectors   # All eigenvectors, scaled to lengths 1
        if(coord) {
            select <- which(values > epsilon)
            princ.coord <- vectors[,select] %*% diag(sqrt(values[select]))
            } else { princ.coord <- NA }
        } else {
        values <- princ.coord <- NA
        }
#
# Compute projector matrix H ("hat" matrix in the statistical literature)
    X.c <- scale(X, center=TRUE, scale=FALSE)   # Centre matrix X
    m <- qr(X.c, tol=1e-6)$rank                 # m = rank of X.c
    cat("Rank of X centred =",m,"\n")
    if(m==1) {
        H <- (X.c[,1] %*% t(X.c[,1]))/((t(X.c[,1]) %*% X.c[,1])[1,1])
        } else {
        if(option<3) {
            # if(det(t(X.c)%*%X.c)<epsilon) stop ('Collinearity detected in X')
            if(m < ncol(X.c)) stop ('Collinearity detected in X')
            H <- X.c %*% solve(t(X.c) %*% X.c) %*% t(X.c)
            #
            # option=3: compute projector H from orthogonalized X; no inversion
            } else {
            X.eig <- eigen(cov(X.c))
            k <- length(which(X.eig$values > epsilon))
            X.ortho <- X.c %*% X.eig$vectors[,1:k]  # F matrix of PCA
            XprX <- t(X.ortho) %*% X.ortho
            H <- X.ortho %*% diag(diag(XprX)^(-1)) %*% t(X.ortho)
            }
    }
#
# Compute the F statistic: McArdle & Anderson (2001), equation 4 modified
    HGH <- H %*% G %*% H
    SSYhat <- sum(diag(HGH))
    #
    if(option==1) {
        I.minus.H <- diag(n) - H
        den1 <- sum(diag(I.minus.H %*% G %*% I.minus.H))
        F <- SSYhat/den1     # F statistic without the degrees of freedom
        Rsquare <- F/(F+1)
    } else {
```

```
        F <- SSYhat/(SSY-SSYhat)  # F statistic without the degrees of freedom
        Rsquare <- SSYhat/SSY     # or equivalent: Rsquare <- F/(F+1)
    }
    RsqAdj <- 1-((1-Rsquare)*(n-1)/(n-1-m))
#
# Permutation test of F
    if(nperm > 0) {
        nGE=1
        for(i in 1:nperm) {
            order <- sample(n)
            Gperm <- G[order, order]
            H.Gperm.H <- H %*% Gperm %*% H
            SSYhat.perm <- sum(diag(H.Gperm.H))
            #
            if(option==1) {
                den <- sum(diag(I.minus.H %*% Gperm %*% I.minus.H))
                F.perm <- SSYhat.perm/den
            } else {
                F.perm <- SSYhat.perm/(SSY-SSYhat.perm)
            }
            if(F.perm >= F) nGE=nGE+1
            }
        P.perm <- nGE/(nperm+1)
        } else { P.perm <- NA }
#
# Compute RDA ordination coordinates
    if(rda.coord > 0) {
        HGH.eig <- eigen(HGH, symmetric=TRUE)
        # kk <- length(which(HGH.eig$values > epsilon))
        RDA.values <- HGH.eig$values
        rel.eig <- RDA.values/SSY
        cum.eig <- cumsum(rel.eig)
        kk <- length(which(rel.eig > epsilon))
        if(positive.RDA.values) {
            RDA.values <- RDA.values[1:kk]
            rel.eig <- rel.eig[1:kk]
            cum.eig <- cum.eig[1:kk]
            }
        k <- min(rda.coord, kk)
        if(k >= 2) {
        RDA.coord <-sweep(HGH.eig$vectors[,1:k],2,sqrt(RDA.values[1:k]),FUN="*")
            } else {
            RDA.coord <- NA
            cat("k =",k," -- Fewer than two RDA eigenvalues > 0\n")
            }
        } else { RDA.values <- rel.eig <- cum.eig <- RDA.coord <- NA }
#
list(F=F*(n-m-1)/m, Rsquare=c(Rsquare,RsqAdj), P.perm=P.perm, SS.total=SSY,
PCoA.values=values, PCoA.vectors=princ.coord, RDA.values=RDA.values/(n-1),
RDA.rel.values=rel.eig, RDA.cum.values=cum.eig, RDA.coord=RDA.coord)
}
```

Appendix S5

SOFTWARE USED IN THE SIMULATIONS

This Appendix presents the software used in the simulations reported in section "Simulations involving spatially autocorrelated data: comparison of Mantel test and dbMEM analysis" of the paper. A separate R function was written for each series of simulations. This function was called by a set of R commands that produced output files. These files of R commands were run in batch mode.

Files shows in the following pages:

Series 1 simulations involving all pairwise geographic distances (see paper, Fig. 2)

- Simulation function: file LFB.simul1.R
- Running the simulation function: file run.LFB.simul1.batch.txt

Series 2 simulations involving truncated geographic distance matrices (see paper, Fig. 3)

- Simulation function: file LFB.simul2.R
- Running the simulation function: files run.LFB.simul2.batch1.txt, run.LFB.simul2.batch2.txt, run.LFB.simul2.batch3.txt, run.LFB.simul2.batch4.txt

Series 3 simulations involving Delaunay triangulations (see Appendix S3, Fig. S3.1)

- Simulation function: file LFB.simul3.R
- Running the simulation function: files run.LFB.simul3.batch1.txt, run.LFB.simul3.batch2.txt, run.LFB.simul3.batch3.txt, run.LFB.simul3.batch4.txt

The simulation functions call upon packages RandomFields, vegan and spatstat available on CRAN, and package PCNM available on http://r-forge.r-project.org/R/?group_id=195 for Linux and Windows versions, and on http://adn.biol.umontreal.ca/~numericalecology/FonctionsR/ for a Mac OSX compiled version.

```
LFB.simul1 <- function(nsim=100, spacing=3, range=5, mean=0.0, nugget=0.0, var=1, nperm1=99,
nperm2=99, R.option=2)
#
# Task: Simulation function for the Legendre, Fortin & Borcard paper.
# Map sizes are chosen to harbour 100 sampling units (n=100) with spacing of
# {1,2,3,4,5} with a buffer zone of 5 pixels all around the sampled area.
#
# Parameters of the function --
#
# nsim : number of surfaces to be simulated.
# spacing = {1,2,3,4,5} for size of simulation surface = {20,29,38,47,56}.
# range : range of the variogram for generation of SA on the surface.
# mean  : mean of the simulated data
# nugget: nugget parameter
# var   : total variance of the simulated data; var = nugget + sill.
# nperm1 : number of permutations for dbMEM analysis.
# nperm2 : number of permutations for Mantel analysis.
#
# Simulation of Gaussian random fields controlled by a spherical variogram:
# R.option=2 : With R 2.15, use function GaussRF() of RandomFields 2.0.66
# R.option=3 : With R 3.0, use function RFsimulate() of RandomFields 3.0.10
#
# Permutational test in dbMEM analysis: see function R2.test.perm().
#
# Value (output list) --
#
# rej.dbMEM : rejection rates, dbMEM analysis, parametric and permutation tests.
# rej.Mantel : rejection rates, Mantel analysis, permutation test.
# R.dbMEM : R-square & adj. R-square of indiv. simulations dbMEM anal. (matrix).
# p.dbMEM : p-values (param. & perm.) of individual simul. dbMEM anal. (matrix).
# r.Mantel : Mantel correlation of individual simulations with geoD and sqrt(geo.D).
# p.Mantel : p-values of individual simulations, Mantel with geoD and sqrt(geo.D).
# param : vector listing the run parameters: {nsim,spacing,range,nperm1,nperm2}.
# geo.D : geographic distance matrix for the grid sample, shown only if nsim=1.
# Y.D : response distance matrix for the grid sample, shown only when nsim=1.
# Surface : matrix with the whole response surface, shown only when nsim=1.
#           A map can be plotted using image(x$Surface)
#
# References --
#
# Schlather, M., P. Menck, R. Singleton, B. Pfaff and R Core team (2013).
#    RandomFields: Simulation and analysis of random fields.
#    R package version 2.0.66. http://CRAN.R-project.org/package=RandomFields
#
# Schlather, M., A. Malinowski, M. Oesting, D. Boecker, K. Strokorb, S. Engelke,
#    J. Martini, P. Menck, S. Gross, K. Burmeister, J. Manitz, R. Singleton,
#    B. Pfaff and R Core Team (2014). RandomFields: Simulation and Analysis
#    of Random Fields. R package version 3.0.10.
#    http://CRAN.R-project.org/package=RandomFields
#
# License: GPL-2
# Authors: Pierre Legendre and D. Borcard, June 2014
{
require(RandomFields)
require(vegan)
require(PCNM)
param <-c(nsim=nsim, spacing=spacing, range=range, nperm1=nperm1, nperm2=nperm2)
#
if(R.option==2) {
    cat("Function GaussRF() of RandomFields 2.0.66 is used with R 2.15\n")
    } else if(R.option==3) {
    cat("Function RFsimulate() of RandomFields 3.0.10 is used with R 3.0\n")
    RFoptions(spConform=FALSE)
    RFoptions(seed=NULL)
    } else { stop("Error in R.option parameter") }
```

```
#
# Set up the simulation surface and the sampling grid parameters
size <- 9*spacing+11     # size = {20,29,38,47,56} depending on "spacing"
coo <- 1:size    # Coordinates of points on simulation surface in each direction
#
# Coordinates of the sample of n=100 points on the surface
grid.coord <- expand.grid(seq(6,(size-5),spacing), seq(6,(size-5),spacing))
n <- nrow(grid.coord)
geo.D <- dist(grid.coord)
#
# Construct dbMEM eigenfunctions
    dbMEM.grid <- PCNM(geo.D,thresh=spacing,dbMEM=TRUE)
    dbMEM <- as.data.frame(dbMEM.grid$vectors)
    Q <- qr(as.matrix(dbMEM))
#
# Prepare matrix and vectors to receive the simulation results
surf     <- vector(mode="numeric", length=n)
R.dbMEM  <- matrix(NA,nsim,2)
colnames(R.dbMEM) <- c("R.square","adj.R.square")
p.dbMEM  <- matrix(NA,nsim,2)
colnames(p.dbMEM) <- c("Parametric.P","Permutational.P")
r.Mantel <- matrix(NA,nsim,2)
p.Mantel <- matrix(NA,nsim,2)
colnames(r.Mantel) <- c("Mantel(Y.D,geo.D)","Mantel(Y.D,sqrt(geo.D))")
colnames(p.Mantel) <- c("Mantel(Y.D,geo.D)","Mantel(Y.D,sqrt(geo.D))")
###
# Main simulation loop
a <- system.time({             # How much time for the simulations?
#
for(kk in 1:nsim) {
    if(range==0) {
        surf <- rnorm(n)
        } else {
        ## Simulate autocorrelated surface with spherical variogram model
        if(R.option==2) {
            Surface <- GaussRF(coo, coo, model="spherical", grid=TRUE,
                param=c(mean=mean, var=var, nugget=nugget, scale=range))
            }
        if(R.option==3) {
            model = RMspheric(var=var, scale=range)
            Surface <- RFsimulate(model = model, x=coo, y=coo, grid=TRUE)
            ### DB: if a nugget is sought, argument err.model in RFsimulate
            ###     can be used: err.model=RMnugget(var=var)
            }
        #
        for(k in 1:n) surf[k] <- Surface[grid.coord[k,2], grid.coord[k,1]]
        }
    Y.D <- dist(surf)
    #
    # dbMEM analysis by regression. No selection of the eigenfunctions
    lm.res <- lm(surf ~ .,data=dbMEM)
    R.dbMEM[kk,] <- c(summary(lm.res)$r.squared, summary(lm.res)$adj.r.squared)
    F.vec <- summary(lm.res)$fstatistic
    # Parametric test results (for the time being)
    p.dbMEM[kk,1] <- pf(F.vec[1], F.vec[2], F.vec[3], lower.tail=FALSE)
    if(nperm1==0) { p.dbMEM[kk,2] <- NA } else {
        p.dbMEM[kk,2]<-R2.test.perm(surf,dbMEM,nperm=nperm1,Q=Q, dbMEM=TRUE)$P }
    # Mantel test using vegan's mantel() function
    res.Mantel1 <- mantel(Y.D, geo.D, permutations=nperm2)
    res.Mantel2 <- mantel(Y.D, sqrt(geo.D), permutations=nperm2)
    r.Mantel[kk,1] <- res.Mantel1$statistic
    r.Mantel[kk,2] <- res.Mantel2$statistic
    p.Mantel[kk,1] <- res.Mantel1$signif
    p.Mantel[kk,2] <- res.Mantel2$signif
    }
rej.dbMEM <-
```

```
    c(length(which(p.dbMEM[,1]<=0.05)),length(which(p.dbMEM[,2]<=0.05)))/nsim
rej.Mantel <- c(length(which(p.Mantel[,1]<=0.05)),length(which(p.Mantel[,2]<=0.05)))/nsim
#
})
a[3] <- sprintf("%2f",a[3])
cat("Simulation time =",a[3]," sec",'\n')
#
# Save Surface, geo.D and Y.D only if a single simulation has been produced
if(nsim==1) {Surface<-Surface; geo.D=geo.D; Y.D<-Y.D}
    else {Surface<-NULL; geo.D=NULL; Y.D<-NULL}
#
out <- list(rej.dbMEM=rej.dbMEM, rej.Mantel=rej.Mantel, R.dbMEM=R.dbMEM, p.dbMEM=p.dbMEM,
r.Mantel=r.Mantel, p.Mantel=p.Mantel, param=param, geo.D=geo.D, Y.D=Y.D, Surface=Surface)
}

R2.test.perm <- function(Y, X, nperm=999, Q=NULL, dbMEM=FALSE)
# Permutation test for R2 statistic in regression or RDA
# Some operations on X=dbMEM are transferred to the main function LFB.simul1()
# in the case of dbMEM analysis.
#
# Parameters of the function --
# Y : Matrix or vector of response data.
# X : Matrix of explanatory data, e.g. a file of dbMEM.
# nperm : Number of permutations for the test.
# dbMEM=FALSE: Normal computation of statistics for regression or RDA:
#              Centre X, QR decomposition of X, compute adjusted R-square.
# dbMEM=TRUE : Do not centre X since the matrix of dbMEM is centred.
#              Do not compute the adjusted R-square.
# License: GPL-2
# Author:: Pierre Legendre, June 2014
{
Y.c <- scale(Y,center=TRUE,scale=FALSE)    # Centre Y
SS.Y <- sum(Y.c^2)
X <- as.matrix(X)
n <- nrow(X)
if(!dbMEM) {
    X <- scale(X,center=TRUE,scale=FALSE)  # Centre X
    Q <- qr(X)                             # QR decompostion of X
    }
Yfit.X <- qr.fitted(Q, Y.c)
# Compute statistics
Rsquare <- sum(Yfit.X^2)/SS.Y
m <- Q$rank
residualDF <- n-m-1
F <- (Rsquare*residualDF)/((1-Rsquare)*m)
if(dbMEM) { adjRsq <- NA } else {
    totalDF <- n-1
    adjRsq <- 1-((1-Rsquare)*totalDF/residualDF)
    }
# Permutation test of F
if(nperm > 0) {
    nPGE <- 1
    for(i in 1:nperm) {
        YfitPerm <- qr.fitted(Q, sample(Y.c))
        RsquarePerm <- sum(YfitPerm^2)/SS.Y
        FPerm <- (RsquarePerm*residualDF)/((1-RsquarePerm)*m)
        if(FPerm >= F) nPGE <- nPGE+1
        }
    P <- nPGE/(nperm+1)
    } else { P <- NA }
#
if(dbMEM) { out <- list(P=P) }
else {out <- list(Rsquare=Rsquare, F=F, P=P, adjRsq=adjRsq, nperm=nperm, m=m,
residualDF=residualDF) }
}
```

File **run.LFB.simul1.batch.txt** — Batch run for series 1 simulations

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul1.R"
source("LFB.simul1.R")

res1.5.0 = LFB.simul1(nsim=1000, spacing=5, range=0, nperm1=999, nperm2=999)
#
res1.5.5 = LFB.simul1(nsim=1000, spacing=5, range=5, nperm1=999, nperm2=999)
#
res1.5.10 = LFB.simul1(nsim=1000, spacing=5, range=10, nperm1=999, nperm2=999)
#
res1.5.15 = LFB.simul1(nsim=1000, spacing=5, range=15, nperm1=999, nperm2=999)
#
res1.5.20 = LFB.simul1(nsim=1000, spacing=5, range=20, nperm1=999, nperm2=999)
#
res1.5.25 = LFB.simul1(nsim=1000, spacing=5, range=25, nperm1=999, nperm2=999)
#
res1.5.30 = LFB.simul1(nsim=1000, spacing=5, range=30, nperm1=999, nperm2=999)
#
res1.5.35 = LFB.simul1(nsim=1000, spacing=5, range=35, nperm1=999, nperm2=999)
#
res1.5.40 = LFB.simul1(nsim=1000, spacing=5, range=40, nperm1=999, nperm2=999)
#
save(res1.5.0,res1.5.5,res1.5.10,res1.5.15,res1.5.20,res1.5.25,res1.5.30,res1.5.35,res1.5.40,
file="simul1,spacing=5.9_runs_sqrt.RData")
```

```
LFB.simul2 <- function(nsim=100, spacing=5, range=10, thresh=20, maxD=NULL, mean=0.0,
nugget=0.0, var=1, nperm1=99, nperm2=99, R.option=2)
#
# Task: Simulation function for the Legendre, Fortin & Borcard paper.
# Map sizes are chosen to harbour 100 sampling units (n=100) with spacing of
# {1,2,3,4,5} with a buffer zone of 5 pixels all around the sampled area.
#
# In the LFB.simul2 simulations, autocorrelated surfaces are generated with different
# range values, as in the LFB.simul1 function. However, instead of considering the whole
# matrix of geographic distances among points, the distances larger than the truncation
# distance ("thresh") are changed to the largest distance in the data set (maxD). A Mantel
# test is then computed.
#
# Details --
#
# Different values of "thresh" will be used in separate runs. Graphs will be drawn for the
# different "thresh" values, showing the rejection rates and the R-squares of dbMEM and
# Mantel analyses as a function of the autocorrelation range values.
#
# Parameters of the function --
#
# nsim : number of surfaces to be simulated.
# spacing = {1,2,3,4,5} for size of simulation surface = {20,29,38,47,56}.
#           This is also the truncation threshold used by function PCNM.
# range : range of the variogram for generation of SA on the surface.
# thresh: truncation distance for Mantel tests; {5,10,15,20} in our simulations.
# maxD  : distance to use when geo.D[geo.D > thresh] <- maxD.
#         For spacing = 5, maxD = 63.63961.
# mean  : mean of the simulated data.
# nugget: nugget parameter.
# var   : total variance of the simulated data; var = nugget + sill.
# nperm1 : number of permutations for dbMEM analysis.
# nperm2 : number of permutations for Mantel analysis.
#
# Simulation of Gaussian random fields controlled by a spherical variogram:
# R.option=2 : With R 2.15, use function GaussRF() of RandomFields 2.0.66
# R.option=3 : With R 3.0, use function RFsimulate() of RandomFields 3.0.10
#
# Permutational test in dbMEM analysis: see function R2.test.perm().
#
# Value (output list) --
#
# rej.dbMEM : rejection rates, dbMEM analysis, parametric and permutation tests.
# rej.Mantel : rejection rates, Mantel analysis, permutation test.
# R.dbMEM : R-square & adj. R-square of indiv. simulations dbMEM anal. (matrix).
# p.dbMEM : p-values (param. & perm.) of individual simul. dbMEM anal. (matrix).
# r.Mantel : Mantel correlation of individual simulations (vector) with geoD.
# p.Mantel : p-values of individual simulations, Mantel analysis (vector).
# param : vector listing the run parameters: {nsim,spacing,range,nperm1,nperm2}.
# geo.D : geographic distance matrix for the grid sample, shown only if nsim=1.
# Y.D : response distance matrix for the grid sample, shown only when nsim=1.
# Surface : matrix with the whole response surface, shown only when nsim=1.
#           A map can be plotted using image(x$Surface)
#
# References --
#
# Schlather, M., P. Menck, R. Singleton, B. Pfaff and R Core team (2013).
#    RandomFields: Simulation and analysis of random fields.
#    R package version 2.0.66. http://CRAN.R-project.org/package=RandomFields
#
# Schlather, M., A. Malinowski, M. Oesting, D. Boecker, K. Strokorb, S. Engelke,
#    J. Martini, P. Menck, S. Gross, K. Burmeister, J. Manitz, R. Singleton,
#    B. Pfaff and R Core Team (2014). RandomFields: Simulation and Analysis
```

```
#    of Random Fields. R package version 3.0.10.
#    http://CRAN.R-project.org/package=RandomFields
#
# License: GPL-2
# Authors: Pierre Legendre and D. Borcard, July 2014
{
require(RandomFields)
require(vegan)
require(PCNM)
param <-c(nsim=nsim, spacing=spacing, range=range, thresh=thresh, nperm1=nperm1,
nperm2=nperm2)
#
if(R.option==2) {
    cat("Function GaussRF() of RandomFields 2.0.66 is used with R 2.15\n")
    } else if(R.option==3) {
    cat("Function RFsimulate() of RandomFields 3.0.10 is used with R 3.0\n")
    RFoptions(spConform=FALSE)
    RFoptions(seed=NULL)
    } else { stop("Error in R.option parameter") }
#
# Set up the simulation surface and the sampling grid parameters
size <- 9*spacing+11     # size = {20,29,38,47,56} depending on "spacing"
coo <- 1:size    # Coordinates of points on simulation surface in each direction
#
# Coordinates of the sample of n=100 points on the surface
grid.coord <- expand.grid(seq(6,(size-5),spacing), seq(6,(size-5),spacing))
n <- nrow(grid.coord)
geo.D <- dist(grid.coord)
# cat("maxD =",maxD,"\n")
if(is.null(maxD)) maxD <- max(geo.D)
cat("maxD =",maxD,"\n")
#
# Construct dbMEM eigenfunctions
    dbMEM.grid <- PCNM(geo.D,thresh=spacing,dbMEM=TRUE)
    dbMEM <- as.data.frame(dbMEM.grid$vectors)
    Q <- qr(as.matrix(dbMEM))
# Truncate geographic matrix D for Mantel test
geo.D[geo.D > thresh] <- maxD
# Prepare matrix and vectors to receive the simulation results
surf     <- vector(mode="numeric", length=n)
R.dbMEM  <- matrix(NA,nsim,2)
colnames(R.dbMEM) <- c("R.square","adj.R.square")
p.dbMEM  <- matrix(NA,nsim,2)
colnames(p.dbMEM) <- c("Parametric.P","Permutational.P")
r.Mantel <- vector(mode="numeric", length=nsim)
p.Mantel <- vector(mode="numeric", length=nsim)
###
# Main simulation loop
a <- system.time({             # How much time for the simulations?
#
for(kk in 1:nsim) {
    if(range==0) {
        surf <- rnorm(n)
        } else {
        ## Simulate autocorrelated surface with spherical variogram model
        if(R.option==2) {
            Surface <- GaussRF(coo, coo, model="spherical", grid=TRUE,
                param=c(mean=mean, var=var, nugget=nugget, scale=range))
            }
        if(R.option==3) {
            model = RMspheric(var=var, scale=range)
            Surface <- RFsimulate(model = model, x=coo, y=coo, grid=TRUE)
            ### DB: if a nugget is sought, argument err.model in RFsimulate
            ###     can be used: err.model=RMnugget(var=var)
```

```
            }
        #
        for(k in 1:n) surf[k] <- Surface[grid.coord[k,2], grid.coord[k,1]]
        }
    Y.D <- dist(surf)
    #
    # dbMEM analysis. No selection of the eigenfunctions
    lm.res <- lm(surf ~ .,data=dbMEM)
    R.dbMEM[kk,] <- c(summary(lm.res)$r.squared, summary(lm.res)$adj.r.squared)
    F.vec <- summary(lm.res)$fstatistic
    # Parametric test resuts (for the time being)
    p.dbMEM[kk,1] <- pf(F.vec[1], F.vec[2], F.vec[3], lower.tail=FALSE)
    if(nperm1==0) { p.dbMEM[kk,2] <- NA } else {
        p.dbMEM[kk,2]<-R2.test.perm(surf,dbMEM,nperm=nperm1,Q=Q, dbMEM=TRUE)$P }
    # Mantel test using vegan's mantel() function
    res.Mantel <- mantel(Y.D, geo.D, permutations=nperm2)
    r.Mantel[kk] <- res.Mantel$statistic
    p.Mantel[kk] <- res.Mantel$signif
    }
rej.dbMEM <-
    c(length(which(p.dbMEM[,1]<=0.05)),length(which(p.dbMEM[,2]<=0.05)))/nsim
names(rej.dbMEM) <- c("p.param", "p.perm")
rej.Mantel <- length(which(p.Mantel <= 0.05))/nsim
#
})
a[3] <- sprintf("%2f",a[3])
cat("Simulation time =",a[3]," sec",'\n')
#
# Save Surface, geo.D and Y.D only if a single simulation has been produced
if(nsim==1) {Surface<-Surface; geo.D=geo.D; Y.D<-Y.D}
    else {Surface<-NULL; geo.D=NULL; Y.D<-NULL}
#
out <- list(rej.dbMEM=rej.dbMEM, rej.Mantel=rej.Mantel, R.dbMEM=R.dbMEM, p.dbMEM=p.dbMEM,
r.Mantel=r.Mantel, p.Mantel=p.Mantel, param=param, geo.D=geo.D, Y.D=Y.D, Surface=Surface)
}

R2.test.perm <- function(Y, X, nperm=999, Q=NULL, dbMEM=FALSE)
# Permutation test for R2 statistic in regression or RDA
# Some operations on X=dbMEM are transferred to the main function LFB.simul1()
# in the case of dbMEM analysis.
#
# Parameters of the function --
# Y : Matrix or vector of response data.
# X : Matrix of explanatory data, e.g. a file of dbMEM.
# nperm : Number of permutations for the test.
# dbMEM=FALSE: Normal computation of statistics for regression or RDA:
#              Centre X, QR decomposition of X, compute adjusted R-square.
# dbMEM=TRUE : Do not centre X since the matrix of dbMEM is centred.
#              Do not compute the adjusted R-square.
# License: GPL-2
# Author:: Pierre Legendre, June 2014
{
Y.c <- scale(Y,center=TRUE,scale=FALSE)    # Centre Y
SS.Y <- sum(Y.c^2)
X <- as.matrix(X)
n <- nrow(X)
if(!dbMEM) {
    X <- scale(X,center=TRUE,scale=FALSE)  # Centre X
    Q <- qr(X)                             # QR decompostion of X
    }
Yfit.X <- qr.fitted(Q, Y.c)
# Compute statistics
Rsquare <- sum(Yfit.X^2)/SS.Y
m <- Q$rank
```

```
residualDF <- n-m-1
F <- (Rsquare*residualDF)/((1-Rsquare)*m)
if(dbMEM) { adjRsq <- NA } else {
    totalDF <- n-1
    adjRsq <- 1-((1-Rsquare)*totalDF/residualDF)
    }
# Permutation test of F
if(nperm > 0) {
    nPGE <- 1
    for(i in 1:nperm) {
        YfitPerm <- qr.fitted(Q, sample(Y.c))
        RsquarePerm <- sum(YfitPerm^2)/SS.Y
        FPerm <- (RsquarePerm*residualDF)/((1-RsquarePerm)*m)
        if(FPerm >= F) nPGE <- nPGE+1
        }
    P <- nPGE/(nperm+1)
    } else { P <- NA }
#
if(dbMEM) { out <- list(P=P) }
else {out <- list(Rsquare=Rsquare, F=F, P=P, adjRsq=adjRsq, nperm=nperm, m=m,
residualDF=residualDF) }
}
```

File **run.LFB.simul2.batch1.txt** — Batch run for series 2 simulations, thresh=5

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul2.R"
source("LFB.simul2.R")

res2.5.0.5 = LFB.simul2(nsim=1000, spacing=5, range=0, thresh=5, nperm1=0, nperm2=999)
#
res2.5.5.5 = LFB.simul2(nsim=1000, spacing=5, range=5, thresh=5, nperm1=0, nperm2=999)
#
res2.5.10.5 = LFB.simul2(nsim=1000, spacing=5, range=10, thresh=5, nperm1=0, nperm2=999)
#
res2.5.15.5 = LFB.simul2(nsim=1000, spacing=5, range=15, thresh=5, nperm1=0, nperm2=999)
#
res2.5.20.5 = LFB.simul2(nsim=1000, spacing=5, range=20, thresh=5, nperm1=0, nperm2=999)
#
res2.5.25.5 = LFB.simul2(nsim=1000, spacing=5, range=25, thresh=5, nperm1=0, nperm2=999)
#
res2.5.30.5 = LFB.simul2(nsim=1000, spacing=5, range=30, thresh=5, nperm1=0, nperm2=999)
#
res2.5.35.5 = LFB.simul2(nsim=1000, spacing=5, range=35, thresh=5, nperm1=0, nperm2=999)
#
res2.5.40.5 = LFB.simul2(nsim=1000, spacing=5, range=40, thresh=5, nperm1=0, nperm2=999)
#
save(res2.5.0.5,res2.5.5.5,res2.5.10.5,res2.5.15.5,res2.5.20.5,res2.5.25.5,res2.5.30.5,res2.5.35.5,
res2.5.40.5, file="simul2,spacing=5,thresh=5.RData")
```

File **run.LFB.simul2.batch2.txt** — Batch run for series 2 simulations, thresh=10

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul2.R"
source("LFB.simul2.R")

res2.5.0.10 = LFB.simul2(nsim=1000, spacing=5, range=0, thresh=10, nperm1=0, nperm2=999)
#
res2.5.5.10 = LFB.simul2(nsim=1000, spacing=5, range=5, thresh=10, nperm1=0, nperm2=999)
#
res2.5.10.10 = LFB.simul2(nsim=1000, spacing=5, range=10, thresh=10,nperm1=0,nperm2=999)
#
res2.5.15.10 = LFB.simul2(nsim=1000, spacing=5, range=15, thresh=10,nperm1=0,nperm2=999)
#
res2.5.20.10 = LFB.simul2(nsim=1000, spacing=5, range=20, thresh=10,nperm1=0,nperm2=999)
#
res2.5.25.10 = LFB.simul2(nsim=1000, spacing=5, range=25, thresh=10,nperm1=0,nperm2=999)
#
res2.5.30.10 = LFB.simul2(nsim=1000, spacing=5, range=30, thresh=10,nperm1=0,nperm2=999)
#
res2.5.35.10 = LFB.simul2(nsim=1000, spacing=5, range=35, thresh=10,nperm1=0,nperm2=999)
#
res2.5.40.10 = LFB.simul2(nsim=1000, spacing=5, range=40, thresh=10,nperm1=0,nperm2=999)
#
save(res2.5.0.10,res2.5.5.10,res2.5.10.10,res2.5.15.10,res2.5.20.10,res2.5.25.10,res2.5.30.10,
res2.5.35.10,res2.5.40.10, file="simul2,spacing=5,thresh=10.RData")
```

File **run.LFB.simul2.batch3.txt** — Batch run for series 2 simulations, thresh=15

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul2.R"
source("LFB.simul2.R")

res2.5.0.15 = LFB.simul2(nsim=1000, spacing=5, range=0, thresh=15, nperm1=0, nperm2=999)
#
res2.5.5.15 = LFB.simul2(nsim=1000, spacing=5, range=5, thresh=15, nperm1=0, nperm2=999)
#
res2.5.10.15 = LFB.simul2(nsim=1000, spacing=5, range=10, thresh=15,nperm1=0,nperm2=999)
#
res2.5.15.15 = LFB.simul2(nsim=1000, spacing=5, range=15, thresh=15,nperm1=0,nperm2=999)
#
res2.5.20.15 = LFB.simul2(nsim=1000, spacing=5, range=20, thresh=15,nperm1=0,nperm2=999)
#
res2.5.25.15 = LFB.simul2(nsim=1000, spacing=5, range=25, thresh=15,nperm1=0,nperm2=999)
#
res2.5.30.15 = LFB.simul2(nsim=1000, spacing=5, range=30, thresh=15,nperm1=0,nperm2=999)
#
res2.5.35.15 = LFB.simul2(nsim=1000, spacing=5, range=35, thresh=15, nperm1=0,nperm2=999
#
res2.5.40.15 = LFB.simul2(nsim=1000, spacing=5, range=40, thresh=15,nperm1=0,nperm2=999)
#
save(res2.5.0.15,res2.5.5.15,res2.5.10.15,res2.5.15.15,res2.5.20.15,res2.5.25.15,res2.5.30.15,
res2.5.35.15,res2.5.40.15, file="simul2,spacing=5,thresh=15.RData")
```

File **run.LFB.simul2.batch4.txt** — Batch run for series 2 simulations, thresh=20

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul2.R"
source("LFB.simul2.R")

res2.5.0.20 = LFB.simul2(nsim=1000, spacing=5, range=0, thresh=20, nperm1=0, nperm2=999)
#
res2.5.5.20 = LFB.simul2(nsim=1000, spacing=5, range=5, thresh=20, nperm1=0, nperm2=999)
#
res2.5.10.20 = LFB.simul2(nsim=1000, spacing=5, range=10, thresh=20,nperm1=0,nperm2=999)
#
res2.5.15.20 = LFB.simul2(nsim=1000, spacing=5, range=15, thresh=20,nperm1=0,nperm2=999)
#
res2.5.20.20 = LFB.simul2(nsim=1000, spacing=5, range=20, thresh=20,nperm1=0,nperm2=999)
#
res2.5.25.20 = LFB.simul2(nsim=1000, spacing=5, range=25, thresh=20,nperm1=0,nperm2=999)
#
res2.5.30.20 = LFB.simul2(nsim=1000, spacing=5, range=30, thresh=20,nperm1=0,nperm2=999)
#
res2.5.35.20 = LFB.simul2(nsim=1000, spacing=5, range=35, thresh=20,nperm1=0,nperm2=999)
#
res2.5.40.20 = LFB.simul2(nsim=1000, spacing=5, range=40, thresh=20,nperm1=0,nperm2=999)
#
save(res2.5.0.20,res2.5.5.20,res2.5.10.20,res2.5.15.20,res2.5.20.20,res2.5.25.20,res2.5.30.20,
res2.5.35.20,res2.5.40.20, file="simul2,spacing=5,thresh=20.RData")
```

```
LFB.simul3 <- function(nsim=100, spacing=5, range=10, N=50, thresh=NULL, maxD=NULL,
new.Del=TRUE, mean=0.0, nugget=0.0, var=1, nperm1=99, nperm2=99, R.option=3)
#
# Task: Simulation function for the Legendre, Fortin & Borcard paper.
# Map sizes are chosen to harbour 100 sampling units (n=100) with spacing of
# {1,2,3,4,5} with a buffer zone of 5 pixels all around the sampled area.
#
# In the LFB.simul3 simulations, autocorrelated surfaces are generated with different
# range values, as in the LFB.simul1 function. In this function, however, N points are
# selected at random from among the 100 points of the regular sampling grid. A Delaunay
# triangulation is constructed among these points. New distances are computed among these
# N points; the distance is the minimum number of Delaunay segments separating two points.
#
# Details --
#
# For the N selected points, a Delaunay triangulation is computed, as well as a matrix of
# dbMEM eigenfunctions. If "thresh" is not NULL, the matrix of Delaunay graph distances is
# truncated, which means that any value larger than a selected threshold is replaced by
# the largest distance actually found in the matrix, max.D. The simulated response values
# for the N selected points are kept in a vector. The Mantel test is computed for the
# truncated matrix of Delaunay graph distances.
#
# Different values of "thresh" will be used in separate runs. Graphs will be drawn for the
# different "thresh" values, showing the rejection rates and the R-squares of dbMEM and
# Mantel analyses as a function of the autocorrelation range values.
#
# Permutation test in dbMEM analysis are carried out by function R2.test.perm.
#
# Simulation of Gaussian random fields controlled by a spherical variogram:
# R.option=2 : With R 2.15, use function GaussRF() of RandomFields 2.0.66
# R.option=3 : With R 3.0, use function RFsimulate() of RandomFields 3.0.10
#
# Parameters of the function --
#
# nsim : number of surfaces to be simulated.
# spacing = {1,2,3,4,5} for size of simulation surface = {20,29,38,47,56}.
# range : range of the variogram for generation of SA on the surface.
# N     : Number of points to sample on the regular grid.
# thresh: truncation D of Delaunay triangulation for Mantel tests.
# maxD  : distance to use when geo.D[geo.D > thresh] --
#         Use the maximum distance on the grid if maxD=NULL: sqrt(9^2*2)*5
#         Use the user-defined value of maxD if maxD is not NULL.
# new.Del=TRUE : New Delaunay triangulation and MEM file for every SA surface.
#        =FALSE: A single Delaunay triangulation and MEM file for all simulations.
# mean  : mean of the simulated data
# nugget: nugget parameter
# var   : total variance of the simulated data; var = nugget + sill (PL check).
# nperm1 : number of permutations for dbMEM analysis.
# nperm2 : number of permutations for Mantel analysis.
#
# Value (output list) --
#
# rej.dbMEM : rejection rates, dbMEM analysis, parametric and permutation tests.
# rej.Mantel : rejection rates, Mantel analysis, permutation test.
# R.dbMEM : R-square & adj. R-square of indiv. simulations dbMEM anal. (matrix).
# p.dbMEM : p-values (param. & perm.) of individual simul. dbMEM anal. (matrix).
# r.Mantel : Mantel correlations of individual simulations (vector) with geoD.
# p.Mantel : p-values of individual simulations, Mantel analysis (vector).
# param : vector of run parameters: {nsim,spacing,range,thresh,nperm1,nperm2}.
# Del.D : Delaunay distance matrix for the grid sample, shown only if nsim=1.
# Y.D : response distance matrix for the grid sample, shown only when nsim=1.
# Surface : matrix with the whole response surface, shown only when nsim=1.
#           A map can be plotted using image(x$Surface)
```

```
#
# References --
#
# Schlather, M., P. Menck, R. Singleton, B. Pfaff and R Core team (2013).
#    RandomFields: Simulation and analysis of random fields.
#    R package version 2.0.66. http://CRAN.R-project.org/package=RandomFields
#
# Schlather, M., A. Malinowski, M. Oesting, D. Boecker, K. Strokorb, S. Engelke,
#    J. Martini, P. Menck, S. Gross, K. Burmeister, J. Manitz, R. Singleton,
#    B. Pfaff and R Core Team (2014). RandomFields: Simulation and Analysis
#    of Random Fields. R package version 3.0.10.
#    http://CRAN.R-project.org/package=RandomFields
#
# License: GPL-2
# Authors: Pierre Legendre and D. Borcard, July 2014
{
require(spatstat)
require(RandomFields)
require(vegan)
require(PCNM)
param <-c(nsim=nsim, spacing=spacing, range=range, thresh=thresh, nperm1=nperm1,
nperm2=nperm2)
#
if(R.option==2) {
    cat("Function GaussRF() of RandomFields 2.0.66 is used with R 2.15\n")
    } else if(R.option==3) {
    cat("Function RFsimulate() of RandomFields 3.0.10 is used with R 3.0\n")
    RFoptions(spConform=FALSE)
    RFoptions(seed=NULL)
    } else { stop("Error in R.option parameter") }
#
# Set up the simulation surface and the sampling grid parameters
size <- 9*spacing+11     # size = {20,29,38,47,56} depending on "spacing"
coo <- 1:size    # Coordinates of points on simulation surface in each direction
#
# Coordinates of the sample of n=100 points on the surface
grid.coord <- expand.grid(seq(6,(size-5),spacing), seq(6,(size-5),spacing))
n <- nrow(grid.coord)
if(N > n) stop("N is larger than the number of points on the sample grid")
#
if(!new.Del) { # Use the same Delaunay triangulation and MEM file for all surfaces
    # Select N points for Delaunay triangulation
    vec.N = sort(sample(n, N))    # Selected N points for Delaunay triangulation
    print(vec.N)
    cat(vec.N,"\n")
    coord.N = grid.coord[vec.N,]  # Coordinates of the selected points
    # Delaunay triangulation -- D matrix in number of steps along the triangulation
    Del.D <- as.dist(delaunay.distance(ppp(coord.N[,1],coord.N[,2],xrange=c(1,size),
                     yrange=c(1,size))))
    if(is.null(maxD)) maxD. <- max(Del.D)
    cat("new.Del=FALSE: maxD. in Delaunay =",maxD.,"\n")
    # Truncate the Delaunay matrix D that will be used for the Mantel test
    Del.D[Del.D > thresh] <- maxD.
    # Construct dbMEM eigenfunctions
        geo.D <- dist(coord.N)
        dbMEM <- PCNM(geo.D,dbMEM=TRUE,silent=TRUE)
        dbMEM <- as.data.frame(dbMEM.N$vectors)
        Q <- qr(as.matrix(dbMEM))
    }
#
# Prepare matrix and vectors to receive the simulation results
surf     <- vector(mode="numeric", length=N)
R.dbMEM  <- matrix(NA,nsim,2)
colnames(R.dbMEM) <- c("R.square","adj.R.square")
```

```
p.dbMEM  <- matrix(NA,nsim,2)
colnames(p.dbMEM) <- c("Parametric.P","Permutational.P")
r.Mantel <- vector(mode="numeric", length=nsim)
p.Mantel <- vector(mode="numeric", length=nsim)
MaxD <- 0
###
# Main simulation loop
a <- system.time({             # How much time for the simulations?
#
for(kk in 1:nsim) {
    if(new.Del) { # New Delaunay triangulation and MEM file for every SA surface
        repeat {  ### try() : procedure to go around a bug in package deldir
            vec.N = sort(sample(n, N))    # Selected points for Delaunay triangulation
            coord.N = grid.coord[vec.N,]  # Coordinates of the selected points
            tmp <- ppp(coord.N$Var1/size,coord.N$Var2/size)
            # if(!is.ppp(tmp)) cat(kk,"No\n") else cat(kk,"Yes\n")
            tst = try(Del.D <- delaunay.distance(tmp), silent=TRUE)
            if(class(tst)=="matrix") {
                break } else { cat("kk =",kk,"### Error in delaunay.dist()\n") }
            }

        Del.D <- as.dist(Del.D)
        if(is.null(maxD)) maxD. <- max(Del.D)
        if(maxD. > MaxD) MaxD <- maxD.
        # cat(kk,"- maxD. =",maxD.,"\n")
        # Truncate the Delaunay matrix D that will be used for the Mantel test
        Del.D[Del.D > thresh] <- maxD.
        # Construct dbMEM eigenfunctions
            geo.D <- dist(coord.N)
            dbMEM.N <- PCNM(geo.D,dbMEM=TRUE,silent=TRUE)
            dbMEM <- as.data.frame(dbMEM.N$vectors)
            Q <- qr(as.matrix(dbMEM))
        }
    if(range==0) {
        surf <- rnorm(N)
        } else {
        ## Simulate autocorrelated surface with spherical variogram model
        if(R.option==2) {
            Surface <- GaussRF(coo, coo, model="spherical", grid=TRUE,
                param=c(mean=mean, var=var, nugget=nugget, scale=range))
            }
        if(R.option==3) {
            model = RMspheric(var=var, scale=range)
            Surface <- RFsimulate(model = model, x=coo, y=coo, grid=TRUE)
            ### DB: if a nugget is sought, argument err.model in RFsimulate
            ###     can be used: err.model=RMnugget(var=var)
            }
        #
        for(k in 1:N) surf[k] <- Surface[coord.N[k,2], coord.N[k,1]]
        }
    if(nsim==1) return(list(coord.N=coord.N, surf=surf, Del.D=Del.D, dbMEM=dbMEM, N=N))
    Y.D <- dist(surf)
    #
    # dbMEM analysis. No selection of the eigenfunctions
    lm.res <- lm(surf ~ .,data=dbMEM)
    R.dbMEM[kk,] <- c(summary(lm.res)$r.squared, summary(lm.res)$adj.r.squared)
    F.vec <- summary(lm.res)$fstatistic
    # Parametric test resuts (for the time being)
    p.dbMEM[kk,1] <- pf(F.vec[1], F.vec[2], F.vec[3], lower.tail=FALSE)
    if(nperm1==0) { p.dbMEM[kk,2] <- NA } else {
        p.dbMEM[kk,2]<-R2.test.perm(surf,dbMEM,nperm=nperm1,Q=Q, dbMEM=TRUE)$P }
    # Mantel test using vegan's mantel() function
    res.Mantel <- mantel(Y.D, Del.D, permutations=nperm2)
    r.Mantel[kk] <- res.Mantel$statistic
```

```
    p.Mantel[kk] <- res.Mantel$signif
    }
rej.dbMEM <-
    c(length(which(p.dbMEM[,1]<=0.05)),length(which(p.dbMEM[,2]<=0.05)))/nsim
names(rej.dbMEM) <- c("p.param", "p.perm")
rej.Mantel <- length(which(p.Mantel <= 0.05))/nsim
if(new.Del) cat("new.Del=TRUE: MaxD in Delaunay =",MaxD,"\n")
#
})
a[3] <- sprintf("%2f",a[3])
cat("\n","Simulation time =",a[3]," sec",'\n')
#
# Save Surface, Del.D and Y.D only if a single simulation has been produced
if(nsim==1) {Surface<-Surface; Del.D<-Del.D; Y.D<-Y.D}
    else {Surface<-NULL; Del.D<-NULL; Y.D<-NULL}
#
out <- list(rej.dbMEM=rej.dbMEM, rej.Mantel=rej.Mantel, R.dbMEM=R.dbMEM, p.dbMEM=p.dbMEM,
r.Mantel=r.Mantel, p.Mantel=p.Mantel, param=param, Del.D=Del.D, Y.D=Y.D, Surface=Surface)
}

R2.test.perm <- function(Y, X, nperm=999, Q=NULL, dbMEM=FALSE)
# Permutation test for R2 statistic in regression or RDA
# Some operations on X=dbMEM are transferred to the main function LFB.simul1()
# in the case of dbMEM analysis.
#
# Parameters of the function --
# Y : Matrix or vector of response data.
# X : Matrix of explanatory data, e.g. a file of dbMEM.
# nperm : Number of permutations for the test.
# dbMEM=FALSE: Normal computation of statistics for regression or RDA:
#              Centre X, QR decomposition of X, compute adjusted R-square.
# dbMEM=TRUE : Do not centre X since the matrix of dbMEM is centred.
#              Do not compute the adjusted R-square.
# License: GPL-2
# Author:: Pierre Legendre, June 2014
{
Y.c <- scale(Y,center=TRUE,scale=FALSE)    # Centre Y
SS.Y <- sum(Y.c^2)
X <- as.matrix(X)
n <- nrow(X)
if(!dbMEM) {
    X <- scale(X,center=TRUE,scale=FALSE)  # Centre X
    Q <- qr(X)                             # QR decompostion of X
    }
Yfit.X <- qr.fitted(Q, Y.c)
# Compute statistics
Rsquare <- sum(Yfit.X^2)/SS.Y
m <- Q$rank
residualDF <- n-m-1
F <- (Rsquare*residualDF)/((1-Rsquare)*m)
if(dbMEM) { adjRsq <- NA } else {
    totalDF <- n-1
    adjRsq <- 1-((1-Rsquare)*totalDF/residualDF)
    }
# Permutation test of F
if(nperm > 0) {
    nPGE <- 1
    for(i in 1:nperm) {
        YfitPerm <- qr.fitted(Q, sample(Y.c))
        RsquarePerm <- sum(YfitPerm^2)/SS.Y
        FPerm <- (RsquarePerm*residualDF)/((1-RsquarePerm)*m)
        if(FPerm >= F) nPGE <- nPGE+1
        }
    P <- nPGE/(nperm+1)
```

```
    } else { P <- NA }
#
if(dbMEM) { out <- list(P=P) }
else {out <- list(Rsquare=Rsquare, F=F, P=P, adjRsq=adjRsq, nperm=nperm, m=m,
residualDF=residualDF) }
}
```

File **run.LFB.simul3.batch1.txt** — Batch run for series 3 simulations, thresh=1

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul3.R"
source("LFB.simul3.R")

res3.5.0.1 = LFB.simul3(nsim=1000, spacing=5, range=0, thresh=1, nperm1=999, nperm2=999)
#
res3.5.5.1 = LFB.simul3(nsim=1000, spacing=5, range=5, thresh=1, nperm1=999, nperm2=999)
#
res3.5.10.1 = LFB.simul3(nsim=1000, spacing=5, range=10, thresh=1,nperm1=999,nperm2=999)
#
res3.5.15.1 = LFB.simul3(nsim=1000, spacing=5, range=15, thresh=1,nperm1=999,nperm2=999)
#
res3.5.20.1 = LFB.simul3(nsim=1000, spacing=5, range=20, thresh=1,nperm1=999,nperm2=999)
#
res3.5.25.1 = LFB.simul3(nsim=1000, spacing=5, range=25, thresh=1,nperm1=999,nperm2=999)
#
res3.5.30.1 = LFB.simul3(nsim=1000, spacing=5, range=30, thresh=1,nperm1=999,nperm2=999)
#
res3.5.35.1 = LFB.simul3(nsim=1000, spacing=5, range=35, thresh=1,nperm1=999,nperm2=999)
#
res3.5.40.1 = LFB.simul3(nsim=1000, spacing=5, range=40, thresh=1,nperm1=999,nperm2=999)
#
save(res3.5.0.1,res3.5.5.1,res3.5.10.1,res3.5.15.1,res3.5.20.1,res3.5.25.1,res3.5.30.1,res3.5.35.1,
res3.5.40.1, file="simul3,spacing=5,thresh=1.RData")
```

File **run.LFB.simul3.batch2.txt** — Batch run for series 3 simulations, thresh=3

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul3.R"
source("LFB.simul3.R")

res3.5.0.3 = LFB.simul3(nsim=1000, spacing=5, range=0, thresh=3, nperm1=999, nperm2=999)
#
res3.5.5.3 = LFB.simul3(nsim=1000, spacing=5, range=5, thresh=3, nperm1=999, nperm2=999)
#
res3.5.10.3 = LFB.simul3(nsim=1000, spacing=5, range=10, thresh=3,nperm1=999,nperm2=999)
#
res3.5.15.3 = LFB.simul3(nsim=1000, spacing=5, range=15, thresh=3,nperm1=999,nperm2=999)
#
res3.5.20.3 = LFB.simul3(nsim=1000, spacing=5, range=20, thresh=3,nperm1=999,nperm2=999)
#
res3.5.25.3 = LFB.simul3(nsim=1000, spacing=5, range=25, thresh=3,nperm1=999,nperm2=999)
#
res3.5.30.3 = LFB.simul3(nsim=1000, spacing=5, range=30, thresh=3,nperm1=999,nperm2=999)
#
res3.5.35.3 = LFB.simul3(nsim=1000, spacing=5, range=35, thresh=3,nperm1=999,nperm2=999)
#
res3.5.40.3 = LFB.simul3(nsim=1000, spacing=5, range=40, thresh=3,nperm1=999,nperm2=999)
#
save(res3.5.0.3,res3.5.5.3,res3.5.10.3,res3.5.15.3,res3.5.20.3,res3.5.25.3,res3.5.30.3,res3.5.35.3,
res3.5.40.3, file="simul3,spacing=5,thresh=3.RData")
```

File **run.LFB.simul3.batch3.txt** — Batch run for series 3 simulations, thresh=5

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul3.R"
source("LFB.simul3.R")

res3.5.0.5 = LFB.simul3(nsim=1000, spacing=5, range=0, thresh=5, nperm1=999, nperm2=999)
#
res3.5.5.5 = LFB.simul3(nsim=1000, spacing=5, range=5, thresh=5, nperm1=999, nperm2=999)
#
res3.5.10.5 = LFB.simul3(nsim=1000, spacing=5, range=10, thresh=5,nperm1=999,nperm2=999)
#
res3.5.15.5 = LFB.simul3(nsim=1000, spacing=5, range=15, thresh=5,nperm1=999,nperm2=999)
#
res3.5.20.5 = LFB.simul3(nsim=1000, spacing=5, range=20, thresh=5,nperm1=999,nperm2=999)
#
res3.5.25.5 = LFB.simul3(nsim=1000, spacing=5, range=25, thresh=5,nperm1=999,nperm2=999)
#
res3.5.30.5 = LFB.simul3(nsim=1000, spacing=5, range=30, thresh=5,nperm1=999,nperm2=999)
#
res3.5.35.5 = LFB.simul3(nsim=1000, spacing=5, range=35, thresh=5, nperm1=999,nperm2=999
#
res3.5.40.5 = LFB.simul3(nsim=1000, spacing=5, range=40, thresh=5,nperm1=999,nperm2=999)
#
save(res3.5.0.5,res3.5.5.5,res3.5.10.5,res3.5.15.5,res3.5.20.5,res3.5.25.5,res3.5.30.5,res3.5.35.5,
res3.5.40.5, file="simul3,spacing=5,thresh=5.RData")
```

File **run.LFB.simul3.batch4.txt** — Batch run for series 3 simulations, thresh=10

```
# Load the necessary R packages

# Source the function for calculation, file "LFB.simul3.R"
source("LFB.simul3.R")

res3.5.0.10 = LFB.simul3(nsim=1000, spacing=5, range=0, thresh=10,nperm1=999,nperm2=999)
#
res3.5.5.10 = LFB.simul3(nsim=1000, spacing=5, range=5, thresh=10,nperm1=999,nperm2=999)
#
res3.5.10.10 =LFB.simul3(nsim=1000,spacing=5,range=10,thresh=10,nperm1=999,nperm2=999)
#
res3.5.15.10 =LFB.simul3(nsim=1000,spacing=5,range=15,thresh=10,nperm1=999,nperm2=999)
#
res3.5.20.10 =LFB.simul3(nsim=1000,spacing=5,range=20,thresh=10,nperm1=999,nperm2=999)
#
res3.5.25.10 =LFB.simul3(nsim=1000,spacing=5,range=25,thresh=10,nperm1=999,nperm2=999)
#
res3.5.30.10 =LFB.simul3(nsim=1000,spacing=5,range=30,thresh=10,nperm1=999,nperm2=999)
#
res3.5.35.10 =LFB.simul3(nsim=1000,spacing=5,range=35,thresh=10,nperm1=999,nperm2=999)
#
res3.5.40.10 =LFB.simul3(nsim=1000,spacing=5,range=40,thresh=10,nperm1=999,nperm2=999)
#
save(res3.5.0.10,res3.5.5.10,res3.5.10.10,res3.5.15.10,res3.5.20.10,res3.5.25.10,res3.5.30.10,
res3.5.35.10,res3.5.40.10, file="simul3,spacing=5,thresh=10.RData")
```
