Bryologische + Lichenologische Werkgroep

BLWG

The estimation of species richness of Dutch bryophytes between 1900 and 2011

Documentation of VBA-procedures based on the Frescalo program

Rienk-Jan Bijlsma

BLWG-report 15

(March 2013)

ISSN: 1571-5108

KNNV

BLWG-reports are issued by the Dutch Bryological and Lichenological Society (BLWG), Gouda, the Netherlands.

BLWG

The Dutch Bryological and Lichenological Society promotes and stimulates the study and conservation of bryophytes and lichens in the Netherlands.

Contents

Summary

In 2011 the Ministery of Economic Affairs, Agriculture and Innovation asked the BLWG to update the Dutch Red List of bryophytes. The Red List is derived from distribution data recorded between 1900 and 2011 on a quadrant basis (5 x 5 km squares). The Dutch recording grid for distribution data consists of 1476 quadrants.

The proper estimation of species richness for different time periods requires a method to correct for recording bias. We used the method and Frescalo program published recently by Hill (2012). This BLWG-report documents the implementation of the Frescalo program in Visual Basic for Applications (MS Access 2010) including modules for input/output and describes the corresponding database structure. Additional analyses evaluate parameter settings.

The Frescalo program estimates species richness by evaluating local frequencies in neighbourhoods of each quadrant. The VBA-version determines neighbourhoods by considering physical distance and abiotic similarity (features of soil, ground water and geomorphology).

Model output mainly depends on 1) the size of the neighbourhood used to estimate local species frequencies; 2) the proportion of local benchmark species used to estimate sampling intensity; 3) the expected mean neighbourhood frequency (assumed to be independent of species richness).

Frescalo output is used to calculate the expected number of quadrants per species for each decade between 1900 and 2011.

1. Introduction

In 2011 the Ministery of Economic Affairs, Agriculture and Innovation asked the BLWG to update the Dutch Red List of bryophytes (Siebel et al. 2006; reference year 2000). During the preparation stage of this project a new method to solve the 'recorder effort problem' for species occurrence data was published by Hill (2012) including the source code of his computer program Frescalo (Frequency Scaling Local) which calculates trend values. We concluded that this new method suited our needs perfectly: Frescalo enables the estimation of trends at time intervals given differences in recording effort between regions. In order to become familiar with the method and program and to find proper parameter values for the Dutch situation, the Fortran source code of Frescalo was converted into a VBA module (Visual Basic for Applications) in MS Access 2010 with the additional advantage that input/output could be structured using database tables and queries.

The Red List is derived from distribution data recorded between 1900 and 2011 on a quadrant basis (5 x 5 km squares). The Dutch recording grid for distribution data consists of 1476 quadrants (with at least 1 ha of land) ordered within 40 x 25 km sheets of the national topographic map 1:50,000. The new Red List will become available as BLWG-report 14 (Siebel et al. 2012).

The rationale of the method and the interpretation of parameters and results are explained and discussed by Hill (2012). This BLWG-report documents

- the implementation of Frescalo in VBA for MS Access 2010 including modules for input/output and corresponding database structure (chapter 2);
- analyses to evaluate parameter settings (chapter 3).

I thank Mark Hill for his help during the conversion of the Fortran source code and suggestions regarding the interpretation of the output. Red List co-authors Henk Siebel and Laurens Sparrius were involved in the sensitivity analyses. The Advisery Committee for the Red List project consisted of Dick Bal (Ministry of Economic Affairs), Arco van Strien (CBS/Statistics Netherlands) and Heinjo During and Bart van Tooren (BLWG).

2. VBA-version of the Frescalo program

The resources provided by Hill (www.brc.ac.uk/resources.htm) consist of three programs:

- sampdist_1.f: a program to calculate proximity of samples in Euclidean space;
- neighsim_1.f: a program to calculate similarity of grid squares;
- frescalo_1.f: a program to calculate trend values.

The first two programs were combined into the VBA module modNeighWeight (see 2.1 and Appendix 1).

The Frescalo program was rewritten as modFrescalo (see 2.2 and Appendix 2). The proper conversion of Frescalo from Fortran into VBA was checked thoroughly using British test data and corresponding output files provided by Hill along with the Fortran source code and executable. The Fortran source code contains several subroutines to select and order data within arrays. These routines have not been converted because the MS Access database functionality was used to perform these functions.

The module modNjtExpObs was added in order to analyse expected (calculated) and observed number of grid squares per species j and time period t (see 2.3 and Appendix 3).

2.1 modNeighWeight

Objective

A powerful aspect of the Hill-method is that the 'recorder effort problem' is solved by examining an ecologically defined neighbourhood of each quadrant and by evaluating the species' frequencies in these neighbourhoods. This makes the definition of neighbourhoods an essential step. Module modNeighWeight determines a neighbourhood for each quadrant and calculates weights for all quadrants relative to the target quadrant of the neighbourhood.

Module documentation

Hill uses physical distance and vascular plant similarity to define neighbourhoods. The current VBA-version of Frescalo uses abiotic similarity besides physical distance. Abiotic features related to soil, groundwater and geomorphology have been derived from the LKN-database (Landschaps-ecologische Kartering Nederland; De Waal 1992, Maas et al. 1994). Table 2.1 gives the relevant LKN-tables and corresponding fields. Attribute values at the km-square level were combined into single (presence) values per quadrant (field tblSampAtt.attvalue). The occurrence of salt water shores was added using the national classification of physical geographical regions (Bal & Looise 1997-2001). Quadrants with continuous surface water areas of at least 1 ha were extracted from the digital topographic map 1:10,000 (TOP10Vector, version 2006; Kadaster Topografische Dienst Nederland).

Table 2.1. Tables and fields from the LKN-database and corresponding features to define field tblSampAtt.attvalue used to calculate abiotic similarity between quadrants.

LKN-table	*Field name and values*	*Description of values*
bodemgt	bodem_c=9998	urban areas
bodemgt	bodem_c<9990 and bodem_c<>8000	soil features (excluding strongly antropogenic soils and foreign areas)
bodemgt	gt_c<10	ground water features (excluding open water)
grondwat	kwelkw_c is not null	seapage type
grondwat	vergrb_c is not null	type of vertical ground water movement
geomorf	geom_c<80000	geomorphological features (excluding foreign and urban areas, open water &c)

Input/Output tables

Tables 2.2 and 2.3 present descriptions of the input and output tables of modNeighWeight. The input tables have been prepared in a separate database from GIS-data for the Dutch recording grid.

Table 2.2. Description of input (I) and output (O) tables for module modNeighWeight.

I/O	*Table name*	*Description*	*Origin of input table*
I	tblSamp	geographic data of all locations (quadrants) of the national recording grid	user supplied (from GIS-data)
I	tblSampAtt	ecological attributes of all quadrants used to define neighbourhoods	user supplied (from GIS-data)
O	tblDistSimWght	distances, similarities and corresponding weights of all quadrants relative to their target quadrant (neighbourhood)	

Table 2.3. Fields in input and ouput tables for module modNeighWeight.

Table name	*Field name*	*Data type*	*Field description*
tblSamp	samp	text	unique quadrant code
	xsamp	double	x-coordinate (m) of centre of 5 x 5 km grid cell
	ysamp	double	y-coordinate (m) of centre of 5 x 5 km grid cell
	iduhok	double	quadrant code on topographic map 1:50,000 (reference aid not used by any program)
	irank	long integer	rank of quadrant samp in tblSamp (filled by modFrescalo)
	idat	long integer	number of records (species-time combinations) for quadrant samp (filled by modFrescalo)
	iitot	long integer	number of species (ever recorded) for quadrant samp (filled by modFrescalo)
tblSampAtt	samp	text	unique quadrant code
	attvalue	text	attribute value used to calculate ecological similarities within quadrant pairs (see Table 2.3 for actual values)
	iduhok	double	quadrant code on topographic map 1:50,000 (reference aid not used by any program)
	attribute	text	attribute class (reference aid not used by any program)
	nkmsq	integer	number of km-squares with attvalue within quadrant (reference aid not used by any program)
tblDistSimWght	samp1	text	unique quadrant code; target quadrant of neighbourhood
	samp2	text	unique quadrant code; quadrant in neighbourhood of samp1
	dist	single	distance (m) between samp1 and samp2 based on coordinates in tblSamp
	drank	integer	distance rank within 200 nearest quadrants of samp1 (drank=1 for dist=0)
	sim	single	Sørensen similarity between samp1 and samp2 based on attvalue in tblSampAtt
	srank	integer	similarity rank within set of nearest quadrants of samp1 (srank=1 for sim=1)
	amult	double	weight of samp2 within neighbourhood of target quadrant samp1
	smult	double	similarity component of amult (Hill: amult1)
	dmult	double	distance component of amult (Hill: amult2)

2.2 modFrescalo

Objective

The Frescalo program calculates sampling effort and probabilities of species occurrence based on frequencies in the neighbourhood of each quadrant as well as time factors based on sampling effort and probability of occurrence (subroutine tfcalc). See Hill (2012) and the file readme_frescalo.txt (www.brc.ac.uk) for further background information.

Module documentation

The original source code includes two important subroutines: fresca and tfcalc. In the VBA version subroutine fresca has been incorporated in the main program (Appendix 2, line 274 - 412) because this appeared more convenient in working with species ranks which were added to the output file tblFrescaFreq as well (see Input/output tables). Subroutine tfcalc has been converted into subprogram tfcalc (Appendix 2, line 645 - 733).

Model output mainly depends on the following three parameters (see Constant section, Appendix 2 line 85 - 97)

1. nghlim is the lower limit of neighbourhood weight and determines the size of the neighbourhood. It is the only parameter that was added to the Frescalo-program. Target quadrants have unit weight. The more distant or ecologically dissimilar a quadrant is, the lower its weight. nghlim=0.6 means that all quadrants with weight <0.6 don't contribute to the neighbourhood. The higher the parameter value, the smaller the neighbourhood and corresponding species pool. Higher parameter values result in more homogeneous neighbourhoods with less rare species. Note that weights are only assigned to the 100 abiotically most similar quadrants within the set of 200 spatially closest quadrants.
2. blmdef is the benchmark limit R^* in Hill (2012). It defines the proportion of the expected number of species in a neighbourhood used as benchmark species based on scaled species rank. $R^* = 0.1$ means that the 10% most common expected species in a neighbourhood functions as benchmark species. An important feature of Frescalo is that the set of benchmark species is not fixed in advance but determined by the program for each quadrant separately (see however below, Input/output tables). Sampling intensity s_{it} of quadrant *i* in time period *t* is defined as the proportion of recorded benchmark species in that period. When $s_{it} = 1$ the quadrant is supposed to be well recorded. So R^* determines

how easy quadrants will be considered sufficiently searched. Lower sampling intensities result in higher time factors and higher expected ('corrected') numbers of quadrants for the corresponding species (see chapter 3). This shows the importance of parameter blmdef. Note that the size of the neighbourhood determines the size of the species pool (see 1, parameter nghlim) and that a for given R^*, a neighbourhood with a large species pool is less likely to be considered well recorded than one with a smaller species pool.

3. phidef is the expected mean species frequency in well-sampled quadrants, denoted by Φ in Hill (2012). An important assumption of the method is that in well-sampled neighbourhoods Φ is fixed i.e. independent of species richness. However, Φ depends on the general size of neighbourhoods: larger neighbourhoods are more heterogeneous with more rare species resulting in a higher Φ.

Input/output tables

Tables 2.4 and 2.5 present descriptions of the input and output tables of modFrescalo.

Note that output tables from earlier runs are deleted without notice. You can save these tables by renaming them before starting a new run.

Table 2.4. Description of input (I) and output (O) tables for module modFrescalo.

I/O	*Table name*	*Description*	*Origin of input table*
I	tblSamp	geographic data of all locations (quadrants) of the national recording grid	user supplied (from GIS-data)
I	tblDistSimWght	distances, similarities and corresponding weights of all quadrants relative to their target quadrant (neighbourhood)	modNeighWeight
I	tblSampSpecTime	species list per quadrant per time period	user supplied (from distribution data)
I	tblNotBenchSpec	list of species not suitable as benchmarks	user supplied
O	tblFrescaStat	statistics report per location/quadrant (Hill: samples.txt)	
O	tblFrescaFreq	listing of rescaled frequencies for species per location/quadrant (Hill: frequencies.txt)	
O	tblTrend	listing of time factors for species (Hill: trends.txt)	

Table 2.5. Fields in input and ouput tables for module modFrescalo.

Table name	*Field name*	*Data type*	*Field description*
tblSamp	see Table 2.1	see Table 2.1	see Table 2.1
tblDistSimWght	see Table 2.1	see Table 2.1	see Table 2.1
tblSampSpecTime	samp	text	unique quadrant code
	spec	text	unique code of species present in quadrant samp in period time
	time	long integer	time period, specified as (first year of) a class (e.g. 1970)
tblNotBenchSpec	spec	text	unique code of species in tblSpec (derived from tblSampSpecTime) unsuitable as benchmark species
tblFrescaStat	samp	text	unique quadrant code
	no_spp	integer	number of species recorded in quadrant samp
	phi_in	single	initial value of phi, the frequency-weighted mean frequency
	alpha	single	sampling effort multiplier
	wgt_n2	single	effective number of weights in neighbourhood of samp
	phi_out	single	value of phi after scaling; constant, if the algorithm has converged
	spnum_in	single	sum of neighbourhood frequencies before rescaling
	spnum_out	single	estimated species richness, i.e. sum of neighbourhood freguencies after rescaling
	iter	integer	number of iterations for algorithm to converge
tblFrescaFreq	samp	text	unique quadrant code
	spec	text	unique species code
	jrank	integer	rank of species spec in tblSpec (derived from tblSampSpecTime)
	pres	integer	record of species in quadrant samp (1 = recorded, 0 = not recorded)
	freq	single	frequency of species in neighbourhood of quadrant samp
	freq1	single	estimated probability of occurrence, i.e. frequency of species after rescaling
	sd_freq1	single	standard error of freq1 calculated on the assumption that freq is a binomial variate with standard error sqrt(freq*(1-freq)/wgt_n2)
	rank	integer	rank of frequency in neighbourhood of quadrant samp
	rank1	integer	rescaled rank, defined as rank/estimated species richness
	qijt	single	-ln(1-sit*f'ij) with recording intensity sit=1 (complete survey), see Hill eqn 3

Table name	*Field name*	*Data type*	*Field description*
tblTrend	spec	text	unique species code
	jrank	integer	rank of species spec in tblSpec (derived from tblSampSpecTime)
	time	long integer	time period, specified as (first year of) a class (e.g. 1970)
	trank	integer	rank of time period in tblTime (derived from tblSampSpecTime)
	tfactor	double	time factor, the estimated relative frequency of species at the time
	stdev	double	standard deviation of the time factor, given that spt (defined below) is a weighted sum of binomial variates
	count	integer	number of occurrences of species at the time period
	spt	double	number of occurrences given reduced weight of locations having very low sampling effort
	est	double	estimated number of occurrences; this should be equal to spt if the algorithm has converged
	n000	integer	number of locations with non-zero probability of the species occurring
	n098	integer	number of locations for which the probability of occurrence was estimated as greater than 0.98

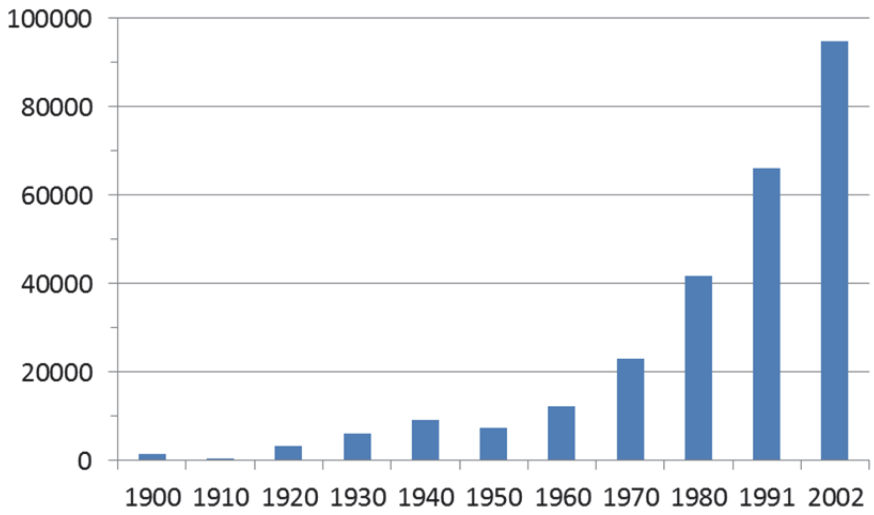

Figure 2.1. Total number of analysed quadrant-species combinations per time period (from tblSampSpecTime). The years mark the beginning of each period. The last period is 2002-2011.

tblSampSpecTime provides presence data of bryophytes for each quadrant and time period. This table has been derived from presence data of species for each year between 1900 and 2011, selected from the national BLWG-database. For several difficult species (e.g. within *Bryum*, *Cephaloziella*, *Sphagnum*) only records based on herbarium collections have been accepted. These annual data were grouped into presence data per decade with 2002-2011 as the last decade (the target decade of the new Red List): 1900-1909, 1910-1919, ..., 1980-1990 (11 year!), 1991-2001 (11 year!), 2002-2011 (Figure 2.1). Table 2.6 shows a selection of records from tblSampSpecTime.

Tabel 2.6. Example records from table tblSampSpecTime.

samp	*spec*	*time*
0146	SYNTRPAP	2002
0146	SYNTRRSL	1991
0146	SYNTRRSL	2002
0146	THAMNALO	1970
0147	AMBLSSER	1970
0147	AMBLSSER	1991
0147	AMBLSSER	2002

tblNotBenchSpec is a list of species to be excluded as benchmark species beforehand. Frescalo determines benchmark species for each quadrant separately, based on scaled species rank: a certain proportion of the most common species (ever recorded) in the neighbourhood (see above, Module documentation). However, implicit in the use of benchmark species is the assumption that they represent rather common species for all periods in the analysis. This makes species which are common in certain decades but with strong positive or negative trends over the whole period unsuitable. The invasive *Campylopus introflexus* from the southern hemisphere is a good example: it arrived in the 1960s and is a very common species nowadays, very likely (but erroneously) to be selected as local benchmark species in many neighbourhoods.

The list in tblNotBenchSpec has been prepared by setting a common threshold to scaled species frequencies for the periods 1900-1949, 1950-1979 and 1980-2011. Species frequencies for these periods were obtained from annual presence data. These frequencies were ranked (with unit rank for the most common species) and scaled by the total number of species observed in each period. Finally, benchmark species were defined as species with scaled rank <0.1 in all three periods, i.e. as

species which for all three periods belong to the 10% most common species. Consequently, tblNotBenchSpec contains all other species.
Note that Frescalo does not exclude non-benchmark species from the calculation of sampling effort (Appendix 2 line 517). The program only downweights these species with weight bwght=0.001 relative to bwght=1 for potential benchmark species (see Appendix 2 line 222 - 225).

Tables with intermediate output are deleted by the program but can be saved by making comment lines of the source codes lines where the tables are dropped (Appendix 2 lines 637-740). The tables tblSampef en tblBenchIJ are perhaps the most interesting. The former gives the sampling effort (sampef: the proportion of observed benchmark species) for all quadrants and time periods. The latter indicates for each quadrant (neighbourhood) whether a species is used as benchmark species or not (ibench=0/1).

2.3 modNjtExpObs

Objective

The calculation of Red List categories requires corrected numbers of quadrants per species for recent and historical periods. These numbers are not explicitly provided by Frescalo but can easily be derived from Frescalo output (tblFrescaFreq and tblTrend). Module modNjtExpObs delivers both the observed and corrected (expected) number of quadrants per species for all time periods (decades). Output can be exported to Excel and viewed as a pivot chart.

Module documentation

The probability that species *j* is recorded in quadrant *i* in period *t* is modeled as (Hill 2012 eqn 2)

$$P_{ijt} = 1 - e^{-Q_{ijt}x_{jt}} \text{ with } Q_{ijt} = -\ln\left(1 - s_{it}f'_{ij}\right) \quad (1)$$

where x_{jt} is the time factor for species *j* in period *t* (tfactor in tblTrend; see Table 2.5), s_{it} is the sampling intensity in neighbourhood *i* for period *t* and f'_{ij} is the estimated probability of occurrence of species *j* in neighbourhood *i* (freq1 in tblTrend; see Table 2.5). The expected number of quadrants N_{jt} for each species and time period is found by assuming that all quadrants are well recorded in all periods, i.e. when $s_{it} = 1$ for all *i* and *t*:

$$N_{jt} = \sum_i P_{ijt} = \sum_i \left(1 - e^{-Q_{ij} x_{jt}}\right) \text{ with } Q_{ij} = -\ln\left(1 - f'_{ij}\right) \quad (2)$$

Values for Q_{ij} have been added to the Frescalo output (tblFrescaFreq, field qijt; see Table 2.5).

The role of the time factor x_{jt} becomes obvious from equation (2) by setting $x_{jt} = 1$ which results in $N_{jt} = \sum_i f'_{ij}$.[1]

Input/output tables

Tables 2.7 and 2.8 present descriptions of the input and output tables of modNjtExpObs.

Table 2.7. Description of input (I) and output (O) tables for module modNjtExpObs.

I/O	*Table name*	*Description*	*Origin of input table*
I	tblFrescaFreq	listing of rescaled frequencies for species per location/quadrant (Hill: frequencies.txt)	modFrescalo
I	tblTrend	listing of time factors for species (Hill: trends.txt)	modFrescalo
O	tblSpecTimeNjt	expected (corrected) and observed numbers of quadrants per species per time period	

Table 2.8. Fields in input and ouput tables for module modNjtExpObs.

Table name	*Field name*	*Data type*	*Field description*
tblFrescaFreq	see Table 2.5	see Table 2.5	see Table 2.5
tblTrend	see Table 2.5	see Table 2.5	see Table 2.5
tblSpecTimeNjt	spec	text	unique species code
	jrank	integer	rank of species spec in tblSpec (derived from tblSampSpecTime)
	time	long integer	time period, specified as (first year of) a class (e.g. 1970)
	trank	integer	rank of time period in tblTime (derived from tblSampSpecTime)
	njtexp	single	expected (corrected) number of quadrants
	njtobs	integer	observed number of quadrants

[1] The estimation procedure for the time factor is not correctly documented by Hill (2012: 200). Instead, x_{jt} is found iteratively as soon as $\sum_i [1 - \exp(-Q_{ijt}\, x_{jt})]$ equals the total number of observed records of species *j* in time period *t* (see Appendix 2 line 699).

The output table can be exported to Excel and used as input for a pivot chart (Figure 2.2).

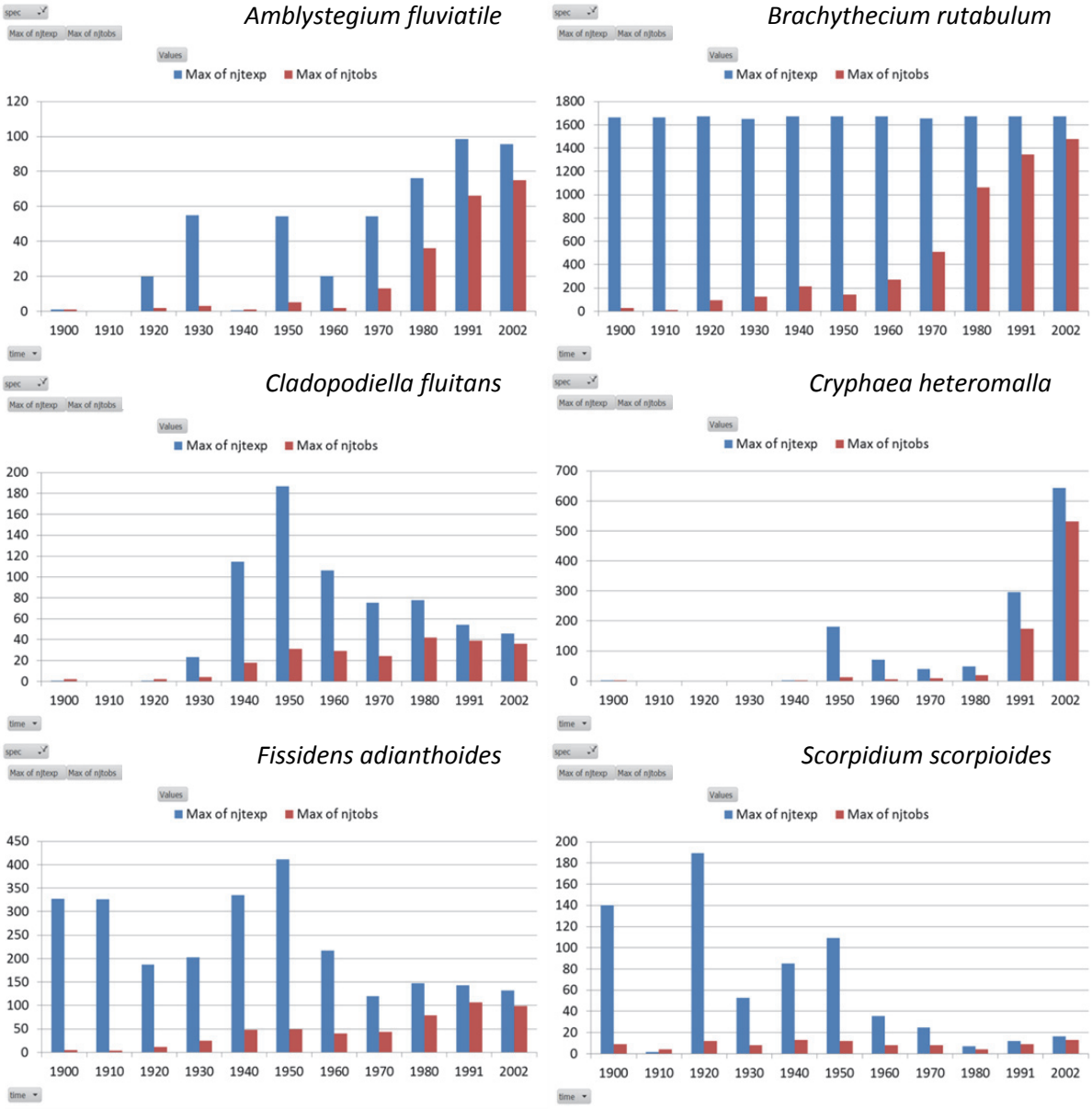

Figure 2.2. Examples of output from modNjtObsExp (exported to Excel and converted into a pivot chart). See chapter 3 for the parameter values used.

3. Parameter values

Values for the relevant parameters Φ, R^* (blmdef) and nghlim (see 2.2 Module documentation for modFrescalo) have been obtained first by considering the expected number of species per quadrant. Because, in our opinion, several quadrants in polder areas never contained more than 40-50 species, we decided to reduce neighbourhood size until the lower limit of the expected number of species was in this range (Figure 3.1). This resulted in nghlim=0.6. Because the distribution of neighbourhood size for nghlim=0.6 shows neighbourhoods with only seven quadrants (Figure 3.2), this parameter value appears the largest possible. Higher values result in neighbourhood sizes too small to be relevant for the estimation of species frequencies.

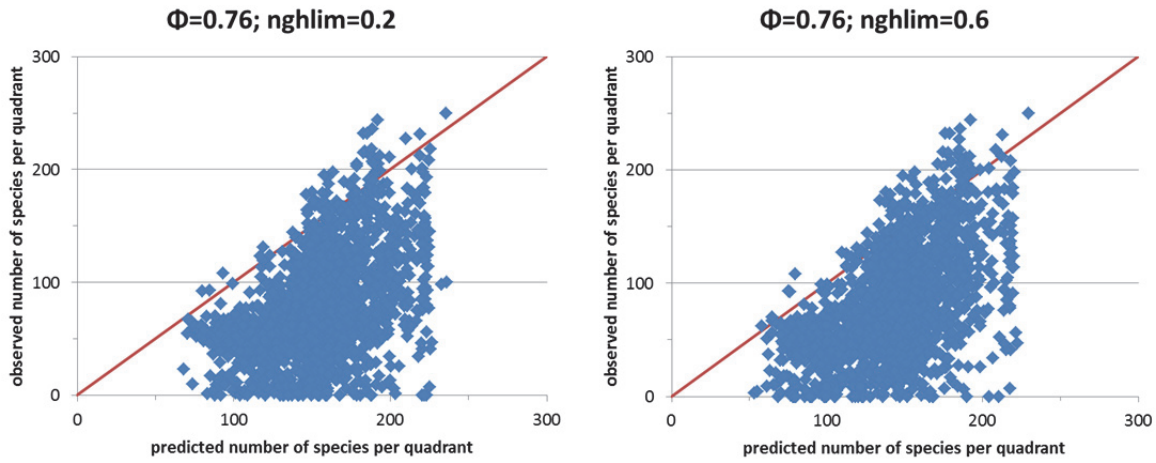

Figure 3.1. Expected (predicted) and observed numbers of bryophyte species for all 1476 quadrants in the Netherlands based on records between 1900 en 2011. The parameter nghlim (lower limit of neighbourhood weight) determines neighbourhood size: smaller neighbourhoods (higher values of nghlim) result in smaller species pools per neighbourhood and a lower lower limit for the expected number of species per quadrant.

The value of Φ was adjusted to Φ = 0.76 such that the expected and observed numbers of species correspond for relatively species poor quadrants. We accept that the number of species in relatively species-rich quadrants is underestimated (see Figure 3.1, nghlim=0.6).

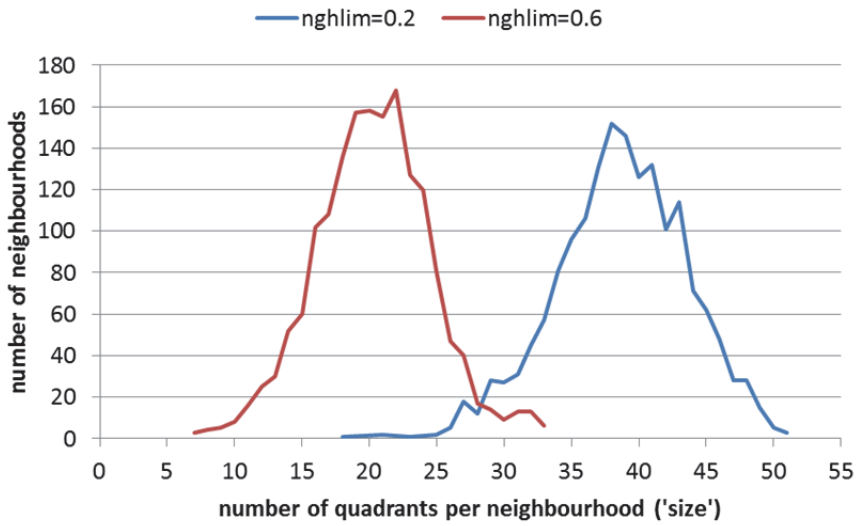

Figure 3.2. Distributions of neighbourhood size for two values of nghlim (lower limit of neighbourhood weight). Higher values result in smaller neighbourhoods.

Figure 3.3 shows what different values for nghlim actually mean for the size and spatial distribution of neighbourhoods. The value nghlim=0.6 gives rather small and compact neighbourhoods, most clearly seen in the coastal dune area (target quadrant Zandvoort). For nghlim=0.2 neighbourhoods become more fragmented but still acceptable. Values below 0.2 lose ecological significance.

The benchmark limit blmdef (R^*) has a large influence on the average expected number of species per quadrant. Figure 3.4 shows how relative large values of R^* result in larger numbers of benchmark species, lower sampling intensities and eventually larger time factors and expected numbers of species per quadrant. Since we observed that expected species numbers tend to be overestimated, especially for historical periods, the low value of R^*=0.05 was chosen for the calculation of trends. Figure 3.5 illustrates how different combinations of nghlim and R^* affect the standardization of frequencies and the scaling of species ranks for species-poor (Nieuw Pekela, Houtribdijk) and species-rich (Zandvoort, Rheden) quadrants. After all the method works remarkably well even for rather extreme parameter setting (nghlim=0.6, R^*=0.05) applied to species-poor neighbourhoods.

Conclusion: for the calculation of trend values for the Dutch Red List the parameter values Φ = 0.76, nghlim = 0.6 and blmdef (R^*) = 0.05 seem most appropriate.

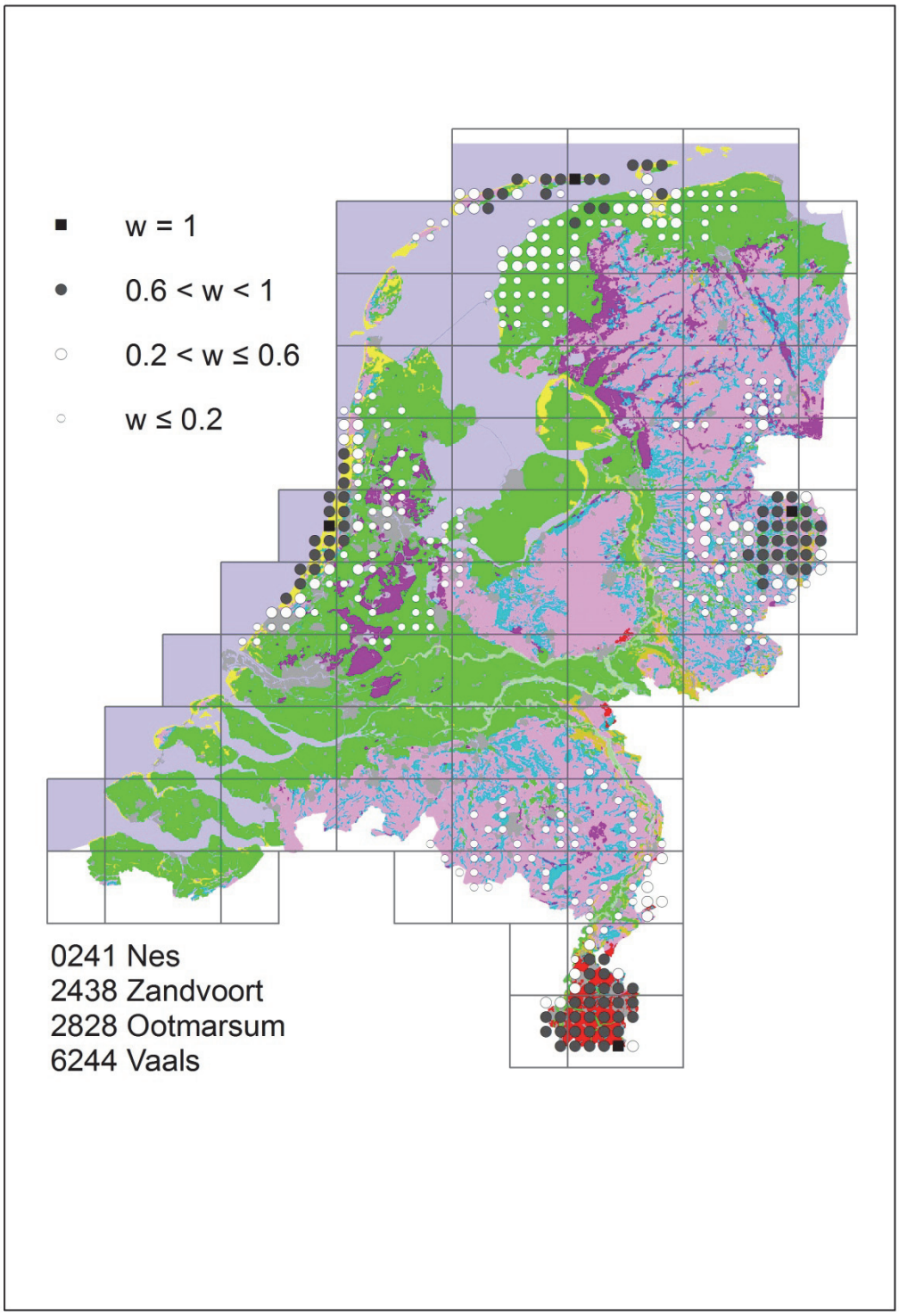

Figure 3.3. Examples of neighbourhoods for four quadrants (Nes, Zandvoort, Ootmarsum, Vaals) with symbols for classes of neighbourhood weight. A lower limit of 0.6 results in compact neighbourhoods.

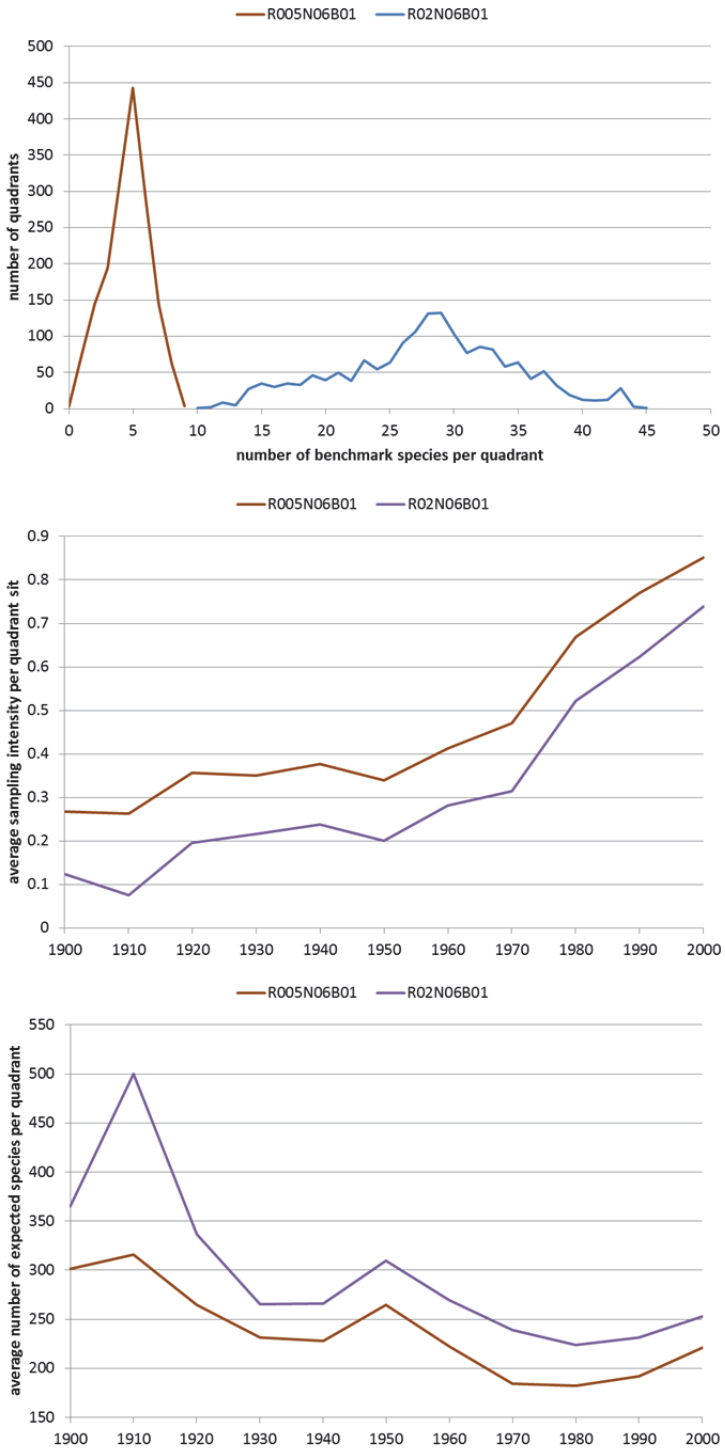

Figure 3.4. The benchmark limit blmdef (R^*) further determines the expected number of species per quadrant. Higher values (R^*=0.2: blue) relative to R^*=0.05 (red) result in much more benchmark species per quadrant (above), corresponding lower sampling intensities per quadrant (middle) and higher time factors causing higher expected numbers of species per quadrant (below).

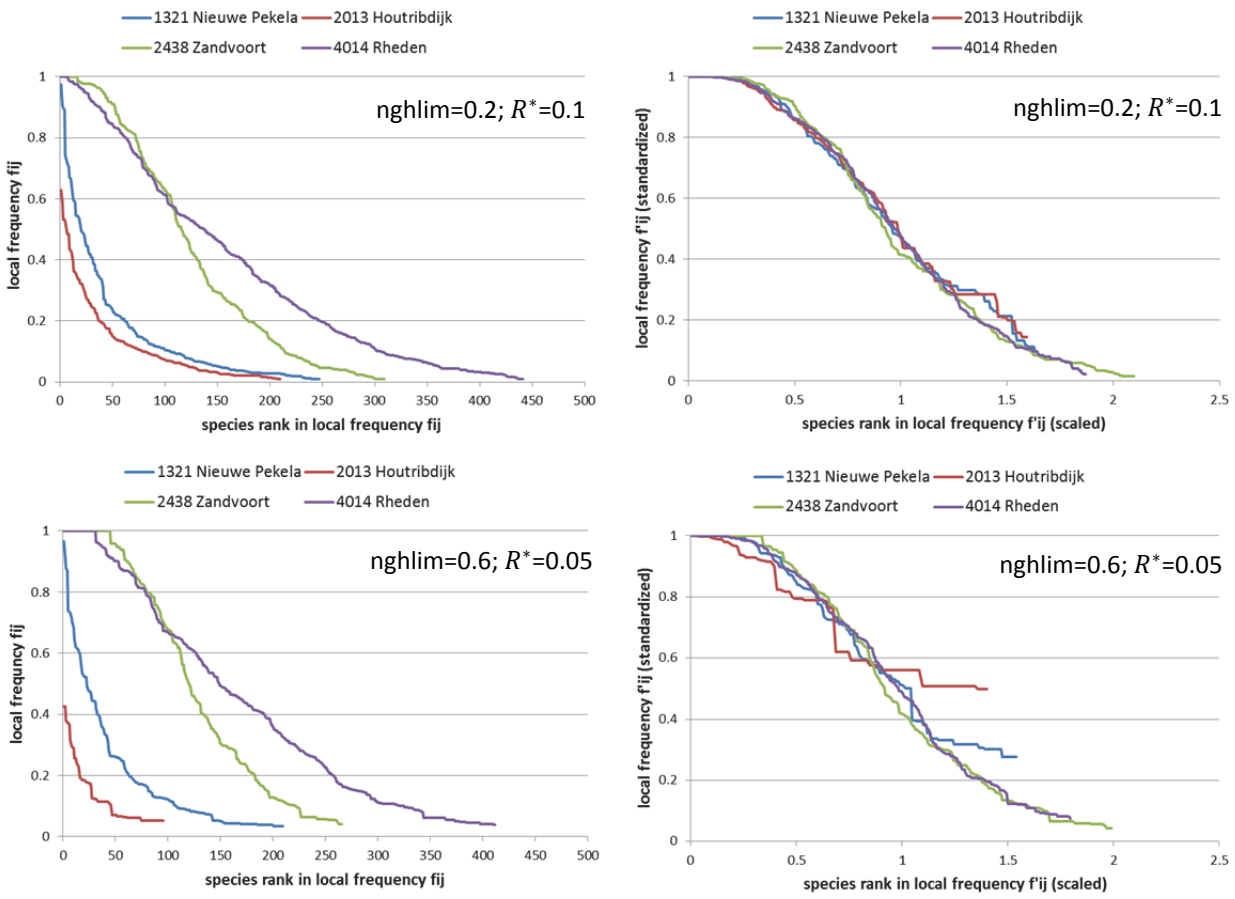

Figure 3.5. The effect of the parameters nghlim (lower neighbourhood limit) and R^* (lower benchmark limit) on the standardization of species frequencies and the scaling of species ranks. Left-hand figures: unstandardized and unscaled. Right-hand figures: standardized and scaled. Note that higher values of nghlim result in more truncated distribution curves and that for species-poor quadrants (Nieuwe Pekela, Houtribdijk) the combination of a high nghlim=0.6 and a low R^*=0.05 results in a rather poor standardization.

Technical remark

The model output for *Scorpidium scorpioides* in Figure 2.2 shows a very low value for the expected number of quadrants in 1910 compared to 1900 and 1920. This pattern occurs rather often and is caused by a poor convergence of expected and observed numbers of species in procedure tfcalc during the iterative process of fitting the time factor: in these cases the fields spt and est in tblTrend can differ by several orders of magnitude instead of being equal. In fact this lack of convergence is caused by the inclusion of an ad hoc weight in the case of very low sampling intensities (<0.0995) (Appendix 2 line 681).

The calculation of sampling intensities in the present model depends on the occurrence of rather common (benchmark) species in each time period. Historical records are generally based on herbarium collections. Unfortunately but understandably, common species have been far less often collected in the past than rare species. This explains why in historical periods low sampling intensities occur even in quadrants with a substantial number of rare species (suggesting a rather thorough search). Hill (2012: 204) recognized this problem as well and suggests future improvements of the model by using partial lists of rare species instead of benchmark species.

References

Bal, D. & B.J. Looise. 1997-2001. Fysisch-geografische regio's van Nederland. Arc/Info-bestand. Expertisecentrum LNV, Wageningen.

Hill, M.O. 2012. Local frequency as a key to interpreting species occurrence data when recording effort is not known. Methods in Ecology and Evolution 3: 195-205.

Maas, G.J., R.W. de Waal & H.P. Wolfert. 1994. Landschapsecologische kartering van Nederland: geomorfologie; toelichting bij het databestand GEOMORF. DLO-Staring Centrum Rapport 335, Wageningen.

Siebel, H.N., R.J. Bijlsma & D. Bal. 2006. Toelichting Rode Lijst Mossen. Rapport DK nr. 2006/034. Directie Kennis, Ministerie van LNV, Ede.

Siebel, H.N., R.J. Bijlsma & L.B. Sparrius. 2012. Basisrapport voor de Rode Lijst mossen. BLWG-rapport 14. BLWG, Oude Tonge (www.blwg.nl/mossen/onderzoek/rapporten/ index.aspx)

Waal, R.W. de. 1992. Landschapsecologische kartering van Nederland: bodem en grondwatertrappen; toelichting bij het databestand BODEMGT. DLO-Staring Centrum Rapport 132, Wageningen.

Appendix 1. VBA source code modNeighWeight

```
'-----------------------------------------------------------------------------------------------------
'  NEIGHWEIGHT
‘  a program to calculate weights for grid squares in a
'  neighbourhood of each grid square based on physical distance and attribute
'  ('species') similarity
'  written by Rienk-Jan Bijlsma march 2012
'  compatible with the FRESCALO-program written by Mark Hill
'-----------------------------------------------------------------------------------------------------
'** Input
'-- tblSamp: list of quadrants with coordinates and national grid code; quadrant
'  (field samp; string), x- and y-coordinate of quadrant samp in m (fields xsamp,
'  ysamp; double), quadrant code on topographic map 1:50,000 (field iduhok; double);
'  fields irank (rank of quadrant in tblSamp; long integer), idat (number of records
'  i.e. species-time combinations for quadrant samp; long integer) en iitot (number
'  of distinct species ever recorded in quadrant samp; long integer) are filled by
'  the Frescalo program; user provided
'-- tblSampAtt: ecological attributes of quadrants used to define neighbourhoods;
'  quadrant code (field samp; string), attribute value used to calculate ecological
'  similarities within quadrant pairs (field attvalue; string), quadrant code on
'  topographic map 1:50,000 (field iduhok; double), attribute class (field attribute;
'  string), number of km-squares with attvalue in quadrant (field nkmsq; integer);
'  user supplied
'
'** Output
'-- tblDistSimWght: weights for quadrants in neighbourhoods relative to a target
'  quadrant; samp1 (target quadrant; string), samp2 (quadrant in
'  neighbourhood of samp1), dist (distance between samp1 and samp2 in m),
'  drank (rank of dist), sim (Sorensen similarity index for attributes of
'  samp1 and samp2), srank (rank of sim), amult (weight for samp2 based on
'  dmult and smult), dmult (weight of samp2 based on drank), smult (weight
'  of samp2 based on srank)
'-----------------------------------------------------------------------------------------------------
Option Compare Database
Option Explicit
Option Base 1

'** Const section
Const neigh As Integer = 200        'number of locations per neighbourhood

Private Sub NeighWeight()

'** Declarations
Dim d As Single                     'distance (m) between samp1 and samp2
Dim dbCurrent As Database           'database object
Dim i As Integer                    'running index
Dim ncom As Integer                 'number of common attribute values
Dim qry As String                   'string representing a SQL-query
Dim rstOut As Recordset             'recordset for writing tblDistSimWght
```

```
51  Dim rstLoc1 As Recordset            'recordset for reading tblSamp
52  Dim rstLoc2 As Recordset            'recordset for reading tblSamp
53  Dim rstSamp As Recordset            'recordset for reading tblSamp
54  Dim rstTmp As Recordset             'recordset for general use
55  Dim samp1 As String, samp2 As String        'unique codes for target quadrant and
56                                              ' quadrant in neighbourhood
57  Dim tot1 As Single, tot2 As Single  'number of attribute values ('species')
58  Dim x1 As Single, y1 As Single      'coordinates samp1 (target)
59  Dim x2 As Single, y2 As Single      'coordinates samp2
60
61  '** Initialisations
62  Set dbCurrent = CurrentDb()
63  '-- initialise index irank
64  Set rstTmp = dbCurrent.OpenRecordset("tblSamp", dbOpenDynaset)
65  With rstTmp
66     Do Until .EOF
67        .Edit
68        .Fields("irank") = .AbsolutePosition + 1
69        .Update
70        .MoveNext
71     Loop
72  End With
73  rstTmp.Close
74
75  '** Calculations
76  '-- calculate distance between each location pair (samp1, samp2) where
77  '   samp1=(x1,y1), samp2=(x2,y2) from tblSampAtt
78  Set rstLoc1 = dbCurrent.OpenRecordset("tblSamp", dbOpenSnapshot)
79  Set rstLoc2 = dbCurrent.OpenRecordset("tblSamp", dbOpenSnapshot)
80  On Error Resume Next
81  dbCurrent.TableDefs.Delete "tblDistSimWghtTmp"
82  qry = "CREATE TABLE tblDistSimWghtTmp (samp1 string, samp2 string, dist single," & _
83  "tot1 double,tot2 double,sim double)"
84  dbCurrent.Execute qry, dbFailOnError
85  Set rstTmp = dbCurrent.OpenRecordset("tblDistSimWghtTmp", dbOpenDynaset)
86  With rstLoc1
87     Do Until .EOF
88        samp1 = !samp: x1 = !xsamp: y1 = !ysamp
89        With rstLoc2
90           Do Until .EOF
91              samp2 = !samp: x2 = !xsamp: y2 = !ysamp
92              d = Sqr((x1 - x2) ^ 2 + (y1 - y2) ^ 2)
93              rstTmp.AddNew
94              rstTmp!samp1 = samp1: rstTmp!samp2 = samp2: rstTmp!dist = d
95              rstTmp.Update
96              .MoveNext
97           Loop
98           .MoveFirst
99        End With
100       .MoveNext
101    Loop
102 End With
103 rstLoc1.Close
```

```
104     rstLoc2.Close
105     rstTmp.Close
106     dbCurrent.Execute "CREATE INDEX isamp1 ON tblDistSimWghtTmp(samp1)"
107     dbCurrent.Execute "CREATE INDEX isamp2 ON tblDistSimWghtTmp(samp2)"
108
109     '-- assign ranks to dist for each location pair: dist=0 for drank=1 and
110     '   store neigh (default 200) nearest pairs of locations per location
111     On Error Resume Next
112     dbCurrent.TableDefs.Delete "tblDistSimWght"
113     qry = "CREATE TABLE tblDistSimWght (samp1 string,samp2 string,dist single," & _
114     "drank short,sim single,srank short,amult double,dmult double,smult double)"
115     dbCurrent.Execute qry, dbFailOnError
116     Set rstOut = dbCurrent.OpenRecordset("tblDistSimWght", dbOpenDynaset)
117     Set rstSamp = dbCurrent.OpenRecordset("tblSamp", dbOpenSnapshot)
118     With rstSamp
119        Do Until .EOF
120           qry = "SELECT samp1,samp2,dist FROM tblDistSimWghtTmp WHERE " & _
121           "samp1=""" & !samp & """" & " ORDER BY dist ASC"
122           Set rstTmp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
123           For i = 1 To neigh
124              rstOut.AddNew
125              rstOut!samp1 = rstTmp!samp1: rstOut!samp2 = rstTmp!samp2
126              rstOut!dist = rstTmp!dist: rstOut!drank = i
127              rstOut.Update
128              rstTmp.MoveNext
129           Next i
130           rstTmp.Close
131           .MoveNext
132        Loop
133     End With
134     rstSamp.Close
135     rstOut.Close
136     dbCurrent.Execute "CREATE INDEX isamp1 ON tblDistSimWght(samp1)"
137     dbCurrent.Execute "CREATE INDEX isamp2 ON tblDistSimWght(samp2)"
138
139     '-- find total number of attributes per location: ntot in tblNtotTmp
140     On Error Resume Next
141     dbCurrent.TableDefs.Delete "tblNtotTmp"
142     qry = "SELECT tblSampAtt.samp,count(*) AS ntot INTO tblNtotTmp " & _
143     "FROM tblSampAtt GROUP BY tblSampAtt.samp"
144     dbCurrent.Execute qry, dbFailOnError
145     dbCurrent.Execute "CREATE INDEX isamp ON tblNtotTmp(samp)"
146
147     '-- update total number of attribute values (‘species’) for each pair of locations:
148     '   tot1 and tot2 in tblDistSimWghtTmp
149     qry = "UPDATE tblDistSimWghtTmp INNER JOIN tblNtotTmp ON tblDistSimWghtTmp.samp1 = " & _
150     "tblNtotTmp.samp SET tblDistSimWghtTmp.tot1 = tblNtotTmp.ntot"
151     dbCurrent.Execute qry, dbFailOnError
152     qry = "UPDATE tblDistSimWghtTmp INNER JOIN tblNtotTmp ON tblDistSimWghtTmp.samp2 = " & _
153     "tblNtotTmp.samp SET tblDistSimWghtTmp.tot2 = tblNtotTmp.ntot"
154     dbCurrent.Execute qry, dbFailOnError
155
156     '-- find number of common species for each pair of locations: ncom
```

```
'-- calculate Sorensen similarity index: tblDistSimWght.sim
On Error Resume Next
dbCurrent.Execute "CREATE INDEX isamp ON tblSampAtt(samp)"
On Error Resume Next
dbCurrent.Execute "CREATE INDEX iattvalue ON tblSampAtt(attvalue)"
Set rstOut = dbCurrent.OpenRecordset("tblDistSimWghtTmp", dbOpenDynaset)
With rstOut
   Do Until .EOF
      samp1 = !samp1: samp2 = !samp2
      If samp1 <> samp2 Then
         tot1 = !tot1: tot2 = !tot2
         qry = "SELECT count(*) AS ncom FROM tblSampAtt AS a INNER " & _
         "JOIN tblSampAtt AS b ON a.attvalue = b.attvalue WHERE " & _
         "a.samp = """ & samp1 & """" & " AND b.samp = """ & samp2 & """"
         Set rstTmp = dbCurrent.OpenRecordset(qry, dbOpenSnapshot)
         ncom = rstTmp!ncom
         rstTmp.Close
         .Edit
         rstOut!sim = CSng(2 * ncom) / CSng(tot1 + tot2)
         .Update
      End If
      .MoveNext
   Loop
End With
rstOut.Close

'-- update tblDistSimWght.sim from blDistSimTmp
qry = "UPDATE tblDistSimWght INNER JOIN tblDistSimWghtTmp ON (tblDistSimWght.samp2 = " & _
"tblDistSimWghtTmp.samp2) AND (tblDistSimWght.samp1 = tblDistSimWghtTmp.samp1) SET " & _
"tblDistSimWght.sim=tblDistSimWghtTmp.sim"
dbCurrent.Execute qry, dbFailOnError
qry = "UPDATE tblDistSimWght SET dist=0,sim=1 WHERE samp1=samp2"
dbCurrent.Execute qry, dbFailOnError

'-- assign ranks to similarities for each location pair: sim=1 > srank=1
Set rstSamp = dbCurrent.OpenRecordset("tblSamp", dbOpenSnapshot)
With rstSamp
   Do Until .EOF
      qry = "SELECT * FROM tblDistSimWght WHERE " & _
      "samp1=""" & !samp & """" & " ORDER BY sim DESC"
      Set rstOut = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
      With rstOut
         i = 1
         Do Until .EOF
            .Edit
            rstOut!srank = i
            .Update
            .MoveNext
            i = i + 1
         Loop
      End With
      rstOut.Close
      .MoveNext
```

```
210         Loop
211     End With
212     rstSamp.Close
213
214     '-- calculate weights for grid squares in neighbourhoods
215     qry = "UPDATE tblDistSimWght SET smult=(1-(csng(srank-1)/100)^2)^4," & _
216     '"dmult=(1-(csng(drank-1)/200)^2)^4"
217     dbCurrent.Execute qry, dbFailOnError
218     qry = "UPDATE tblDistSimWght SET amult=smult*dmult"
219     dbCurrent.Execute qry, dbFailOnError
220
221     '-- delete locations from neighbourhoods with srank>100 or weight<=0.00005
222     qry = "DELETE * FROM tblDistSimWght WHERE srank>100 or amult<=0.00005"
223     dbCurrent.Execute qry, dbFailOnError
224     '-- delete temporary tables
225     dbCurrent.Execute "DROP TABLE tblDistSimWghtTmp"
226     dbCurrent.Execute "DROP TABLE tblNtotTmp"
227
228     End Sub
```

Appendix 2. VBA source code modFrescalo

```
'-----------------------------------------------------------------------------------------------------------
'  FRESCALO
'  a program to calculate trend values
'  written by Mark Hill march 2011
'
'-- M.O. Hill. 2012. Local frequency as a key to interpreting species
'  occurrence data when recording effort is not known. Methods in Ecology and
'  Evolution 2, 195-205
'-- Additional Supporting Information: Data S1. Tests and sensitivity analysis for
'  Frescalo output
'-- Fortran source code www.brc.ac.uk
'-----------------------------------------------------------------------------------------------------------
'  This version converted from FORTRAN and written in Visual Basic for Applications
‘  (MS Office Pro 2010) by Rienk-Jan Bijlsma march 2012
‘
'  Note that this version has no I/O interface. Input is read from and output is
'  written to specific tables (see **Input and **Output); parameters can be
'  altered in the constant section (**Const).
'-----------------------------------------------------------------------------------------------------------
'** Input:
'-- tblSamp: list of quadrants with coordinates and national grid code; quadrant
'  (field samp; string), x- and y-coordinate in Dutch RD-format (fields xsamp,
'  ysamp; double), quadrant code (field iduhok; double); the fields irank (rank of
'  quadrant in tblSamp; long integer), idat (number of records i.e. species-time
'  combinations for quadrant samp; long integer) en iitot (number of distinct
'  species ever recorded in quadrant samp; long integer) are filled by the program;
'  user provided
'-- tblDistSimWght: weights for quadrants in neighbourhoods relative to a target
'  quadrant; target quadrant (field samp1; string), neighbouring quadrant (field
'  samp2; string), weight based on abiotic similarity and distance (field amult;
'  double); output from modNeighWeight
'-- tblSampSpecTime: a species list per quadrant per time period; quadrant (field
'  samp; string), species (field spec; string), time period (field time; integer);
‘  user provided
'-- tblNotBenchSpec: a list of species not suitable as benchmarks; species (field
'  spec; string); user provided
'
‘** Output:
'-- tblFrescaStat: statistics report per location/sample(Hill: samples.txt);
'  quadrant (field samp; string), number of species in location (field no_spp;
'  integer), initial value of phi, the frequency-weighted mean frequency (field
'  phi_in; single), sampling effort multiplier (field alpha; single), 'effective
'  number' n2 for the neighbourhood weights: this is small if there are few
'  similar quadrants close to the target quadrant (field wgt_n2; single), value
'  of phi after scaling; constant, if the algorithm has converged (field phi_out;
'  single), sum of neighbourhood frequencies before rescaling (field spnum_in;
'  single), estimated species richness, i.e. sum of neighbourhood freguencies
'  after rescaling (field spnum_out; single), number of iterations for algorithm
```

```
'  to converge (field iter; integer)
'-- tblFrescaFreq: listing of rescaled frequencies for species per location/sample
'  (Hill: frequencies.txt); quadrant (field samp; string), name of species (field
'  spec; string), species rank in tblSpec derived from tblSampSpecTime (added
'  field jrank; integer), record of species in location (1 = recorded, 0 = not
'  recorded)(field pres; integer), frequency of species in neighbourhood of
'  location (field freq; single), estimated probability of occurrence, i.e.
'  frequency of species after rescaling (field freq1; single), standard error of
'  freq1 calculated on the assumption that freq is a binomial variate with
'  standard error sqrt(freq*(1-freq)/wgt_n2)(field sd_freq1; single), rank of
'  frequency in neighbourhood of location (field rank; integer), rescaled rank,
'  defined as rank/estimated species richness (field rank1; integer),
‘  -ln(1-sit*f'ij with sit=1 (complete survey), see Hill eqn 3 (added field
‘  qijt; single)
'-- tblTrend: listing if time factors for species (Hill: trends.txt); name of
'  species (field spec; string); species rank in tblSpec derived from
'  tblSampSpecTime (added field jrank; integer), time period, specified as a
'  class (e.g. 1970)(field time; long integer), rank of time period in tblTime
'  derived from tblSampSpecTime (added field trank; integer), time factor, the
'  estimated relative frequency of species at the time (field tfactor; double),
'  standard deviation of the time factor, given that spt (defined below) is a
'  weighted sum of binomial variates (field stdev; double), number of occurrences
'  of species at the time period (field count; integer), number of occurrences
'  given reduced weight of locations having very low sampling effort (field spt;
'  double), estimated number of occurrences; this should be equal to spt if the
'  algorithm has converged (field est; double), number of locations with non-zero
'  probability of the species occurring (field n000; integer), number of locations
'  for which the probability of occurrence was estimated as greater than 0.98
'  (field n098; integer)'
'-------------------------------------------------------------------------------------------------
Option Compare Database
Option Explicit
Option Base 1

'** Constant section
Const blmdef As Double = 0.05       'default limit benchmark species
Const fmax As Double = 0.99999      'a value less than 1 to ensure that logs
                                    ' of (1-f) can be taken
Const fmin As Double = 0.0000000001         'used to ensure that frequencies do not
                                            ' sum to zero in small datasets
Const irepmx As Integer = 100       'maximum number of iterations
Const mm As Integer = 4000          'maximum number of samples (quadrants)
Const nghlim As Single = 0.6        'lower limit of neighbourhood weight
Const nn As Integer = 2000          'maximum number of species
Const nnt As Integer = 100          'maximum number of time periods
Const phidef As Double = 0.76       'default value of phi, the local frequency
Const tol As Double = 0.0003        'the convergence limit for rescaling

Private Sub Frescalo()

'** Declarations
Dim abtot(mm) As Double             'number of benchmark species in i
Dim alpha As Double, alph As Double         'sampling effort multiplier
```

```
104     Dim an2 As Double, an21 As Double           'effective species number
105     Dim blim As Double                          'benchmark limit
106     Dim bwght(nn) As Double             'benchmark weights for species (1/0.001)
107     Dim dbCurrent As Database           'database object
108     Dim est As Double                   'estimated sum of Pijt (esttot in tfcalc)
109     Dim f(nn) As Double                 'observed frequencies
110     Dim ff(nn) As Double                'rescaled frequencies
111     Dim fff(mm) As Double               'rescaled frequences used by tfcalc
112     Dim ffff(mm) As Double              'phi1 for each i
113     Dim f4 As Double, f5 As Double      'used to find stdev of ffij
114     Dim ffij(mm, nn) As Double                  'frequencies of species in neighbourhoods
115     Dim ffsd As Double, fffsd As Double         'used to find stdev of ffij
116     Dim frij As Double, ffrij As Double         'Hill: fij en ffij in subroutine Fresca
117     Dim i As Integer                    'running index for quadrant/location
118     Dim ibench(mm, nn) As Integer       '0/1: species j in i is a local benchmark
119     Dim ic1 As Integer, ic2 As Integer  'tfcalc output
120     Dim id As Integer                   'running index
121     Dim idtji As Long                   'running index in reordered data set
122     Dim iit As Integer, iiit As Integer 'running index for time period
123     Dim iitx As Integer                 'reference index for time period
124     Dim iocc(mm) As Integer             '1 if species found at location i at a time
125     Dim itot As Integer                 'total number of species in a quadrant
126     Dim ir As Integer                   'iteration number
127     Dim j As Integer, jj As Integer     'running index for species
128     Dim jrank(nn) As Integer            'index for species from tblSpec.jrank
129     Dim jtot As Integer                 'sum(iocc(i)) for given species j (in tfcalc)
130     Dim jx As Integer                   'reference index for species
131     Dim lendat(nn, nnt) As Integer      'number of records with species j in period iit
132     Dim msg As String                   'return value message box
133     Dim ndtji As Long                   'number of input data records for quadrants
134     Dim nsamp As Integer                'total number of quadrants (=m in Hill)
135     Dim nspec As Integer                'total number of species (=n in Hill)
136     Dim ntime As Integer                'total number of time periods (=t in Hill)
137     Dim phi As Double, phi1 As Double           'expected value of species frequency
138     Dim phi985 As Double                'idem 98.5 percentile value
139     Dim phibig As Double                'expected frequency in well-sampled neighbourhoods
140     Dim qry As String                   'string representing a SQL-query
141     Dim rank1 As Double                 'scaled species rank j/spnum
142     Dim rstFstat As Recordset           'recordset for writing tblFrescaStat
143     Dim rstFfreq As Recordset           'recordset for writing tblFrescaFreq
144     Dim rstSamp As Recordset            'recordset for reading tblSampSpecTime
145     Dim rstTmp As Recordset             'recordset for general use
146     Dim rstTrend As Recordset           'recordset for writing tblTrend
147     Dim samp As String                  'quadrant code (e.g. IVON-code 5x5 km-square)
148     Dim sampef(mm, nnt) As Double               'sampling effort for quadrant i in period iit
149     Dim sd As Double                    'standard error of tf
150     Dim sdij As Double, sdfij As Double         'standard error of ffij
151     Dim smpint(mm) As Double            'sampef(i,iit) for particular iit
152     Dim sp(nn) As String                'species list from tblSpec
153     Dim spec As String                  'species code
154     Dim spnum As Double, spnum1 As Double   'species number
155     Dim spt As Double                   'species total for a time period
156     Dim tf As Double                    'time factor
```

```
157     Dim tim(nnt) As Long                    'ordered time periods
158     Dim time As Long                        'time period
159     Dim tot As Double                       'sum ffij
160     Dim tot2 As Double                      'sum ffij*ffij
161     Dim txt1 As String, txt2 As String      'text for msgbox
162     Dim sumwi As Double, sumwi2 As Double     'factors defining fij
163     Dim wn2 As Double                       'effective number of weights in neighbourhood
164
165     '** Initialisations
166     phibig = phidef
167     blim = blmdef
168     Set dbCurrent = CurrentDb()
169
170     '-- create temporary tblSampSpec (samp, spec): species lists per quadrant
171     On Error Resume Next
172     dbCurrent.TableDefs.Delete "tblSampSpec"
173     qry = "SELECT a.samp, a.spec INTO tblSampSpec FROM tblSampSpecTime As a " & _
174     "INNER JOIN tblSamp ON a.samp = tblSamp.samp GROUP BY a.samp, a.spec"
175     dbCurrent.Execute qry, dbFailOnError
176
177     '-- create temporary tblSampIdat: # records per quadrant
178     On Error Resume Next
179     dbCurrent.TableDefs.Delete "tblSampIdat"
180     qry = "SELECT tblSamp.samp, count(*) AS idat INTO tblSampIdat FROM tblSamp " & _
181     "INNER JOIN tblSampSpecTime ON tblSamp.samp =  tblSampSpecTime.samp " & _
182     "GROUP BY tblSamp.samp"
183     dbCurrent.Execute qry, dbFailOnError
184
185     '-- create temporary tblSampIdatIItot: # records and # distinct species per quadrant
186     On Error Resume Next
187     dbCurrent.TableDefs.Delete "tblSampIdatIItot"
188     qry = "SELECT tblSampSpec.samp, tblSampIdat.idat, Count(*) AS iitot " & _
189     "INTO tblSampIdatIItot FROM tblSampSpec INNER JOIN tblSampIdat ON " & _
190     "tblSampSpec.samp = tblSampIdat.samp GROUP BY tblSampSpec.samp, tblSampIdat.idat"
191     dbCurrent.Execute qry, dbFailOnError
192
193     '-- update fields idat and iitot in tblSamp
194     qry = "UPDATE tblSamp as a INNER JOIN tblSampIdatIItot as b ON a.samp = " & _
195     "b.samp SET a.idat=b.idat,a.iitot=b.iitot"
196     dbCurrent.Execute qry, dbFailOnError
197
198     '-- insert list of distinct species into tblSpec and initialise jrank, bwght
199     '   exclude species occurring only in quadrants without weights
200     '   note: columns jocctmp, ftmp and fftmp function as temporary storage
201     On Error Resume Next
202     dbCurrent.TableDefs.Delete "tblSpec"
203     qry = "SELECT a.spec,cint(0) as jrank,cdbl(1) as bwght,cint(0) as jocctmp," & _
204     "cdbl(0) as ftmp,cdbl(0) as fftmp INTO tblSpec FROM tblSampSpecTime as a " & _
205     "INNER JOIN tblSamp ON a.samp = tblSamp.samp GROUP BY a.spec"
206     dbCurrent.Execute qry, dbFailOnError
207     Set rstTmp = dbCurrent.OpenRecordset("tblSpec", dbOpenDynaset)
208     With rstTmp
209        Do Until .EOF
```

```
        j = .AbsolutePosition + 1
        sp(j) = !spec
        .Edit
        .Fields("jrank") = j
        .Update
        nspec = j
        .MoveNext
      Loop
  End With
  rstTmp.Close
  dbCurrent.Execute "CREATE INDEX ispec ON tblSpec(spec)"

  '-- downweight species that are unsuitable as benchmarks
  qry = "UPDATE tblSpec as a INNER JOIN tblNotBenchSpec as b ON a.spec =" & _
  "b.spec SET a.bwght=0.001"
  dbCurrent.Execute qry, dbFailOnError

  '-- insert list of distinct timeperiods into tblTime and initialise trank
  On Error Resume Next
  dbCurrent.TableDefs.Delete "tblTime"
  qry = "SELECT a.time,cint(0) as trank INTO tblTime " & _
  "FROM tblSampSpecTime as a GROUP BY a.time"
  dbCurrent.Execute qry, dbFailOnError
  Set rstTmp = dbCurrent.OpenRecordset("tblTime", dbOpenDynaset)
  With rstTmp
    Do Until .EOF
      iit = .AbsolutePosition + 1
      tim(iit) = !time
      .Edit
      .Fields("trank") = iit
      .Update
      ntime = iit
      .MoveNext
    Loop
  End With
  rstTmp.Close

  '** Calculations
  '1* calculate observed frequencies fij of species spec in the neighbourhood of
  '   samp and insert into tblFij
  '-- create temporary tblSumWiAij
  On Error Resume Next
  dbCurrent.TableDefs.Delete "tblSumWiAij"
  qry = "SELECT a.samp1, b.spec, Sum(a.amult) AS sumwiaij, Count(*) AS nij INTO " & _
  "tblSumWiAij FROM tblDistSimWght as a INNER JOIN tblSampSpec as b ON " & _
  "a.samp2 = b.samp WHERE a.amult>" & nghlim & " GROUP BY a.samp1, b.spec"
  dbCurrent.Execute qry, dbFailOnError

  '-- create temporary tblSumWii
  On Error Resume Next
  dbCurrent.TableDefs.Delete "tblSumWi"
  qry = "SELECT a.samp1, Sum(a.amult) AS sumwi, Sum(a.amult*a.amult) AS sumwi2 " & _
  "INTO tblSumWi FROM tblDistSimWght as a WHERE a.amult>" & nghlim & " GROUP BY a.samp1"
```

```
263     dbCurrent.Execute qry, dbFailOnError
264     dbCurrent.Execute "CREATE INDEX isamp ON tblSumWi(samp)"
265
266     '-- create tblFij: observed frequencies of species in neighbourhoods
267     On Error Resume Next
268     dbCurrent.TableDefs.Delete "tblFij"
269     qry = "SELECT a.samp1,a.spec,a.sumwiaij/(tblSumWi.sumwi+1.0E-10) AS fij " & _
270     "INTO tblFij FROM tblSumWiAij as a INNER JOIN tblSumWi ON a.samp1 = tblSumWi.samp1"
271     dbCurrent.Execute qry, dbFailOnError
272     dbCurrent.Execute "CREATE INDEX ispec ON tblFij(spec)"
273
274     '2* Fresca-procedure for each quadrant/location in tblSamp
275     '--  make tables for Fresca-output
276     On Error Resume Next
277     dbCurrent.TableDefs.Delete "tblFrescaStat"
278     dbCurrent.Execute "CREATE TABLE tblFrescaStat(samp string,no_spp short," & _
279     "phi_in single,alpha single,wgt_n2 single, phi_out single,spnum_in single," & _
280     "spnum_out single,iter short)", dbFailOnError
281     Set rstFstat = dbCurrent.OpenRecordset("tblFrescaStat", dbOpenDynaset)
282     On Error Resume Next
283     dbCurrent.TableDefs.Delete "tblFrescaFreq"
284     dbCurrent.Execute "CREATE TABLE tblFrescaFreq(samp string,spec string," & _
285     "jrank short, pres short,freq single,freq1 single,sdfreq1 single,rank short," & _
286     "rank1 single,qijt single)", dbFailOnError
287     Set rstFfreq = dbCurrent.OpenRecordset("tblFrescaFreq", dbOpenDynaset)
288
289     '-- determine total number of quadrants nsamp
290     qry = "SELECT samp,irank,iitot FROM tblSamp"
291     Set rstSamp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
292     rstSamp.MoveLast
293     nsamp = rstSamp.AbsolutePosition + 1
294
295     '-- rewind recordset samp before reading and processing all quadrants
296     rstSamp.MoveFirst
297     With rstSamp
298        Do Until .EOF
299           i = !irank
300           samp = !samp
301           itot = !iitot
302
303           'determine wn2: the effective number of weights in neighbourhood i
304           qry = "SELECT sumwi,sumwi2 FROM tblSumWi WHERE samp1=""" & samp & """"
305           Set rstTmp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
306           sumwi = rstTmp.Fields("sumwi")
307           sumwi2 = rstTmp.Fields("sumwi2")
308           rstTmp.Close
309           wn2 = sumwi * sumwi / (sumwi2 + 0.000000000001)
310
311           dbCurrent.Execute "UPDATE tblSpec SET jocctmp=0,ftmp=0.0,fftmp=0.0"
312           'update tblSpec.ftmp from tblFij for quadrant i (=observed fij)
313           qry = "UPDATE tblFij as a INNER JOIN tblSpec as b ON a.spec = b.spec " & _
314           "SET b.ftmp=fij WHERE a.samp1=""" & samp & """"
315           dbCurrent.Execute qry, dbFailOnError
```

```
316         'update tblSpec.jocctmp=1 if spec occurs in quadrant i
317         qry = "UPDATE tblSpec as a INNER JOIN tblSampSpecTime as b ON a.spec " & _
318         "= b.spec SET a.jocctmp = 1 where b.samp=""" & samp & """"
319         dbCurrent.Execute qry, dbFailOnError
320
321     '-- call to original subroutine fresca
322     '   convert the rank-frequency data f to rescaled values ff
323         qry = "SELECT spec,jrank,ftmp,fftmp FROM tblSpec"
324         Set rstTmp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
325         With rstTmp
326            For j = 1 To nspec
327               f(j) = !ftmp                    'f(j)=ffij(i,j)
328               If (f(j) > fmax) Then f(j) = fmax
329               If (f(j) < fmin) Then f(j) = fmin
330               ff(j) = -Log(1 - f(j))
331               .Edit
332               'trick to circumvent ORDER DESC?!
333               .Fields("fftmp") = -f(j) + j * 0.000000000001
334               .Update
335               .MoveNext
336            Next j
337         End With
338         rstTmp.Close
339
340         alpha = 1#
341         For ir = 1 To irepmx
342            tot = 0#
343            tot2 = 0#
344            For j = 1 To nspec
345               ffrij = 1 - Exp(-ff(j) * alpha)
346               tot = tot + ffrij
347               tot2 = tot2 + ffrij ^ 2
348            Next j
349            phi = tot2 / tot
350            an2 = tot * tot / tot2  '=effective species number (not used)
351            spnum = tot
352            If ir < 20 Then
353               alpha = alpha * Exp(1.86 * (Log(1 - phi) - Log(1 - phibig)))
354            Else
355               alpha = alpha * phibig / phi
356            End If
357            If ir = 1 Then
358               phi1 = phi
359               an21 = an2
360               spnum1 = tot
361            End If
362            If Abs(phi - phibig) < tol Then GoTo AlphaFound
363         Next ir
364     AlphaFound:
365         alph = alpha
366         If alpha > 999.99 Then alph = 999.99
367         rstFstat.AddNew
368         rstFstat!samp = samp: rstFstat!no_spp = itot: rstFstat!phi_in = phi1:
```

```
        rstFstat!alpha = alph: rstFstat!wgt_n2 = wn2: rstFstat!phi_out = phi
        rstFstat!spnum_in = spnum1: rstFstat!spnum_out = spnum: rstFstat!iter = ir
        rstFstat.Update

        qry = "SELECT spec,bwght,jrank,jocctmp,ftmp FROM tblSpec " & _
        "ORDER BY fftmp ASC"
        Set rstTmp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
        With rstTmp
           For j = 1 To nspec
              jrank(j) = !jrank
              spec = !spec
              jj = jrank(j)
              ffij(i, jj) = !ftmp           'still observed fij
              bwght(jj) = !bwght
              frij = !ftmp
              If (frij > fmax) Then frij = fmax
              ffrij = 1 - Exp(alpha * Log(1 - frij))
              sdfij = Sqr(frij * (1 - frij) / wn2)
              f4 = frij + sdfij
              If (1 - f4 < 0.000000000001) Then f4 = 1 - 0.000000000001
              f5 = frij - sdfij
              ffsd = 1 - Exp(alpha * Log(1 - f4))
              fffsd = 1 - Exp(alpha * Log(1 - f5))
              sdij = 0.5 * (ffsd - fffsd)
              ff(jj) = ffrij
              If (frij > 0.00005) Then
                 rank1 = j / spnum
                 rstFfreq.AddNew
                 rstFfreq!samp = samp: rstFfreq!spec = !spec: rstFfreq!jrank = jj
                 rstFfreq!pres = !jocctmp: rstFfreq!freq = !ftmp
                 rstFfreq!freq1 = ffrij: rstFfreq!sdfreq1 = sdij
                 rstFfreq!rank = j: rstFfreq!rank1 = rank1
                 If ffrij < 0.98 Then       'Qijt for sit=1 (Hill eqn 3)
                    rstFfreq!qijt = -Log(1# - ffrij)
                 Else
                    rstFfreq!qijt = -Log(1# - 0.98)
                 End If
                 rstFfreq.Update
              End If
              .MoveNext
           Next j
        End With
        rstTmp.Close
'-- return from original subroutine fresca
        ffff(i) = phi1
        abtot(i) = 0.0000001
        For j = 1 To nspec
           'now ffij becomes a rescaled frequency
           If (ffij(i, j) <> 0#) Then ffij(i, j) = ff(j)
           'tag benchmark species for neighbourhood i
           jj = jrank(j)
           ibench(i, jj) = 0
           rank1 = j / spnum
```

```
            If ((rank1 < blim) Or (j = 1)) Then
               ibench(i, jj) = 1
               abtot(i) = abtot(i) + bwght(jj)
            End If
         Next j
         .MoveNext
      Loop
   End With
   rstSamp.Close
   rstFstat.Close
   rstFfreq.Close

   '-- make tblBenchIJ with benchmark indication (0/1) per species per quadrant
   On Error Resume Next
   dbCurrent.TableDefs.Delete "tblBenchIJ"
   qry = "CREATE TABLE tblIBenchIJ (irank short,jrank short,ibench short)"
   dbCurrent.Execute qry, dbFailOnError
   Set rstTmp = dbCurrent.OpenRecordset("tblIbenchIJ", dbOpenDynaset)
   For i = 1 To nsamp
      For j = 1 To nspec
         rstTmp.AddNew
         rstTmp!irank = i: rstTmp!jrank = j: rstTmp!ibench = ibench(i, j)
         rstTmp.Update
      Next j
   Next i
   rstTmp.Close

   '-- test whether given value of phi appears to be unrealistically low
   On Error Resume Next
   dbCurrent.TableDefs.Delete "tblTmpF4"
   qry = "SELECT cint(1) as irank,cdbl(" & ffff(1) & ") as f4 INTO tblTmpF4"
   dbCurrent.Execute qry, dbFailOnError
   Set rstTmp = dbCurrent.OpenRecordset("tblTmpF4", dbOpenDynaset)
   For i = 2 To nsamp
      rstTmp.AddNew
      rstTmp!irank = i: rstTmp!f4 = ffff(i)
      rstTmp.Update
   Next i
   rstTmp.Close
   qry = "SELECT f4 FROM tblTmpF4 ORDER BY f4"
   Set rstTmp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
   For i = 1 To Int(0.985 * nsamp)
      rstTmp.MoveNext
   Next i
   phi985 = rstTmp!f4
   rstTmp.Close
   txt1 = "98.5 percentile of input phi " & Format(phi985, "0.000")
   txt2 = "; target value of phi " & Format(phibig, "0.000")
   msg = MsgBox(txt1 & txt2, vbOKOnly)
   If phibig < phi985 Then
      txt1 = "*** BEWARE *** Target value of phi may be too small"
      msg = MsgBox(txt1, vbOKCancel)
      If msg = vbCancel Then GoTo EndSub
```

```
475     End If
476
477     '3* Trend calculations by subroutine TFCALC
478     '--  initialise arrays
479     For iit = 1 To ntime
480        For i = 1 To nsamp
481           sampef(i, iit) = 0#
482        Next i
483        For j = 1 To nspec
484           lendat(j, iit) = 0
485        Next j
486     Next iit
487
488     '-- make temporary tblSampef
489     On Error Resume Next
490     dbCurrent.TableDefs.Delete "tblSampef"
491     qry = "CREATE TABLE tblSampef(irank short,trank short,sampef double)"
492     dbCurrent.Execute qry, dbFailOnError
493
494     '-- make tblTrend for output
495     On Error Resume Next
496     dbCurrent.TableDefs.Delete "tblTrend"
497     qry = "CREATE TABLE tblTrend(spec string,jrank short,[time] long,trank short," & _
498     "tfactor double,stdev double,count short,spt double,est double,n000 short, n098 short)"
499     dbCurrent.Execute qry, dbFailOnError
500
501     '-- fill arrays lendat, sampef from recordset corresponding to array ddjti in Hill
502     qry = "SELECT tblSampSpecTime.spec, tblSampSpecTime.time, tblSampSpecTime.samp, " & _
503     "tblSpec.jrank, tblTime.trank, tblSamp.irank FROM ((tblSampSpecTime INNER JOIN " & _
504     "tblSamp ON tblSampSpecTime.samp = tblSamp.samp) INNER JOIN tblSpec ON " & _
505     "tblSampSpecTime.spec = tblSpec.spec) INNER JOIN tblTime ON tblSampSpecTime.time = " & _
506     "tblTime.time ORDER BY tblSampSpecTime.spec, tblSampSpecTime.time, tblSampSpecTime.samp"
507     Set rstSamp = dbCurrent.OpenRecordset(qry, dbOpenDynaset)
508     With rstSamp
509        Do Until .EOF                 'ndtji records
510           i = !irank
511           samp = !samp
512           j = !jrank
513           spec = !spec
514           iit = !trank
515           time = !time
516           lendat(j, iit) = lendat(j, iit) + 1
517           sampef(i, iit) = sampef(i, iit) + ibench(i, j) * bwght(j) / abtot(i)
518           .MoveNext
519        Loop
520     End With
521
522     '--write sampling effort per quadrant and time period to tblSampef
523     Set rstTmp = dbCurrent.OpenRecordset("tblSampef", dbOpenDynaset)
524     For i = 1 To nsamp
525        For iit = 1 To ntime
526           rstTmp.AddNew
527           rstTmp!irank = i: rstTmp!trank = iit: rstTmp!sampef = sampef(i, iit)
```

```
        rstTmp.Update
      Next iit
    Next i
    rstTmp.Close

    '-- initialise indices in tfcalc procedure
    rstSamp.MoveLast
    ndtji = rstSamp.AbsolutePosition + 1
    idtji = 0
    jx = 1
    iitx = 1

    '-- find time factor for each species/time period
    Set rstTrend = dbCurrent.OpenRecordset("tblTrend", dbOpenDynaset)
    idtji = 0
    rstSamp.MoveFirst
    With rstSamp
      Do Until .EOF
        idtji = idtji + 1
        j = !jrank
        spec = !spec
        iit = !trank
        time = !time
        For i = 1 To nsamp
          iocc(i) = 0              'initialise iocc
          smpint(i) = sampef(i, iit)  'sampling effort for period iit
          fff(i) = ffij(i, j)        'rescaled frequencies for species j
        Next i
        i = !irank
        samp = !samp
        iocc(i) = 1
        For id = 1 To lendat(j, iit) - 1    'read remaining samples for j,iit
          idtji = idtji + 1
          If idtji > ndtji Then GoTo EndOfData
          .MoveNext
          i = !irank
          iocc(i) = 1
        Next id
        '-- calculate timefactor for species j and time period iit
        Call tfcalc(tf, sd, spt, jtot, est, iocc, smpint, fff, nsamp, ic1, ic2)
        '-- write output to tblTrend
        If sp(jx) < sp(j) Then         'write output for skipped species
          For iiit = iitx To ntime    'remaining periods for last species jx
            rstTrend.AddNew
            rstTrend!spec = sp(jx): rstTrend!jrank = jx
            rstTrend!time = tim(iiit): rstTrend!trank = iiit
            rstTrend!tfactor = 0#: rstTrend!stdev = 0#
            rstTrend!Count = 0: rstTrend!spt = 0#: rstTrend!est = 0#
            rstTrend!n000 = 0: rstTrend!n098 = 0
            rstTrend.Update
          Next iiit
          '-- theoretical situation: species in tblSpec but not in rstSamp
          For jj = jx + 1 To j - 1    'skipped species between jx en j
```

```
            For iiit = 1 To ntime   'for all periods
               rstTrend.AddNew
               rstTrend!spec = sp(jj): rstTrend!jrank = jj
               rstTrend!time = tim(iiit): rstTrend!trank = iiit
               rstTrend!tfactor = 0#: rstTrend!stdev = 0#
               rstTrend!Count = 0: rstTrend!spt = 0#: rstTrend!est = 0#
               rstTrend!n000 = 0: rstTrend!n098 = 0
               rstTrend.Update
            Next iiit
         Next jj
         jx = j
         iitx = 1
      End If
      If tim(iitx) < tim(iit) Then    'skipped periods current species
         For iiit = iitx To iit - 1  'until current period
            rstTrend.AddNew
            rstTrend!spec = sp(j): rstTrend!jrank = j
            rstTrend!time = tim(iiit): rstTrend!trank = iiit
            rstTrend!tfactor = 0#: rstTrend!stdev = 0#
            rstTrend!Count = 0: rstTrend!spt = 0#: rstTrend!est = 0#
            rstTrend!n000 = 0: rstTrend!n098 = 0
            rstTrend.Update
         Next iiit
         iitx = iit
      End If
      rstTrend.AddNew              'current species and period
      rstTrend!spec = sp(j): rstTrend!jrank = j
      rstTrend!time = tim(iit): rstTrend!trank = iit
      rstTrend!tfactor = tf: rstTrend!stdev = sd
      rstTrend!Count = jtot: rstTrend!spt = spt: rstTrend!est = est
      rstTrend!n000 = ic1: rstTrend!n098 = ic2:
      rstTrend.Update
      jx = j
      iitx = iit + 1
      .MoveNext
   Loop
EndOfData:
End With
rstSamp.Close
For iiit = iitx To ntime            'remaining periods current species
   rstTrend.AddNew
   rstTrend!spec = sp(jx): rstTrend!jrank = jx
   rstTrend!time = tim(iiit): rstTrend!trank = iiit
   rstTrend!tfactor = 0#: rstTrend!stdev = 0#
   rstTrend!Count = 0: rstTrend!spt = 0#: rstTrend!est = 0#
   rstTrend!n000 = 0: rstTrend!n098 = 0
   rstTrend.Update
Next iiit
rstTrend.Close

'4* Finish program
dbCurrent.Execute "UPDATE tblSpec SET jocctmp=0.0,ftmp=0.0,fftmp=0.0"
'-- create indices
```

```
634     dbCurrent.Execute "CREATE INDEX ispec ON tblFrescaFreq(jrank)"
635     dbCurrent.Execute "CREATE INDEX ispec ON tblTrend(jrank)"
636     '--  delete temporary tables
637     dbCurrent.Execute "DROP TABLE tblSampSpec, tblSampIdat, tblSampIdatIItot"
638     dbCurrent.Execute "DROP TABLE tblSumWiAij, tblSumWi"
639     dbCurrent.Execute "DROP TABLE tblFij, tblTmpF4"
640     dbCurrent.Execute "DROP TABLE tblSampef, tblBenchIJ"
641     EndSub:
642
643     End Sub
644
645     Private Sub tfcalc(tf As Double, sd As Double, sptot As Double, jtot As Integer, _
646        esttot As Double, iocc() As Integer, smpint() As Double, fff() As Double, _
647        m As Integer, ic1 As Integer, ic2 As Integer)
648
649     '-- Calculates a time factor tf, given observed species total sptot for that time
650     '   sd is the standard error
651     '   iocc(i) is 1 if species found at location i at the time, 0 otherwise
652     '   smpint(i) is the sampling intensity at location i and time t
653     '   fff(i) is the smoothed time-independent frequency of species at location i
654     '   ic1 is the number of samples for which there is nonzero probability of occurrence
655     '   ic2 is the number of cases where smptint*fff >0.98
656     '   weights are applied, downweighting cases where smpint<0.1 (no systematic sampling)
657
658     Dim esttt1 As Double
659     Dim estval As Double
660     Dim estvar As Double
661     Dim i As Integer
662     Dim k As Integer
663     Dim kmax As Integer
664     Dim pfac As Double
665     Dim plog As Double
666     Dim sptot1 As Double
667     Dim tf1 As Double
668     Dim wgt As Double
669
670     kmax = 100
671     tf = 1
672     For k = 1 To kmax
673        esttot = 0#
674        estvar = 0#
675        sptot = 0#
676        jtot = 0
677        ic1 = 0
678        ic2 = 0
679        For i = 1 To m
680           wgt = 1
681           If (smpint(i) < 0.0995) Then wgt = 10# * smpint(i) + 0.005
682           pfac = smpint(i) * fff(i)
683           'the probability of finding species i is multiplied by the sampling intensity
684           If (pfac > 0#) Then ic1 = ic1 + 1
685           If (pfac > 0.98) Then
686              pfac = 0.98
```

```
687             ic2 = ic2 + 1
688           End If
689           'this is a fudge to allow the next step of taking logs;
690           '  otherwise there is a danger that pfac=1
691           plog = -Log(1# - pfac)
692           estval = 1# - Exp(-plog * tf)
693           esttot = esttot + wgt * estval
694           estvar = estvar + wgt * wgt * estval * (1# - estval)
695           sptot = sptot + wgt * iocc(i)
696           jtot = jtot + iocc(i)
697           'esttot is the estimated total for species j at time iit
698       Next i
699       If Abs(sptot - esttot) < 0.0005 Then GoTo Found1
700       tf = tf * sptot / (esttot + 0.0000001)
701   Next k
702
703   Found1:
704   sptot1 = sptot + Sqr(estvar)
705   ' sptot1 is precisely 1 standard deviation bigger than sptot
706   ' recalculate with this target value to obtain standard error of tf
707
708   sd = 0#
709
710   tf1 = tf
711   For k = 1 To kmax
712       esttt1 = 0
713       For i = 1 To m
714           wgt = 1
715           If (smpint(i) < 0.0995) Then wgt = 10# * smpint(i) + 0.005
716           pfac = smpint(i) * fff(i)
717           'the probability of finding species i is multiplied by the sampling intensity
718           If (pfac > 0.98) Then pfac = 0.98
719           'this is a fudge to allow the next step of taking logs;
720           ' otherwise there is a danger that pfac=1
721           plog = -Log(1# - pfac)
722           estval = 1# - Exp(-plog * tf1)
723           esttt1 = esttt1 + wgt * estval
724           'esttt1 is the estimated total targeted at 1 sd bigger
725       Next i
726       If Abs(sptot1 - esttt1) < 0.0005 Then GoTo Found2
727       tf1 = tf1 * sptot1 / (esttt1 + 0.0000001)
728   Next k
729
730   Found2:
731   sd = tf1 - tf
732
733   End Sub
734
```

Appendix 3. VBA source code modNjtExpObs

```
'-----------------------------------------------------------------------------------------------------------
'   NjtExpObs: a program to calculate expected numbers of quadrants and add
'   observed numbers of quadrants for species in time periods
'   written by Rienk-Jan Bijlsma march 2012
'-----------------------------------------------------------------------------------------------------------
'** Input:
'-- tblFrescaFreq: output from Frescalo; see modFrescalo
'-- tblTrend: output from Frescalo; see modFrescalo
'
'** Output:
'-- tblSpecTimeNjt: number of expected and observed number of quadrants for
'   species per period; name of species (field spec; string); species rank in
'   tblSpec derived from tblSampSpecTime (field jrank; integer), time period,
'   specified as a class (e.g. 1970)(field time; long integer), rank of time
'   period in tblTime derived from tblSampSpecTime (field trank; integer),
'   expected (corrected) number of quadrants (field njtexp; single), observed
'   number of quadrants (field njtobs; integer).
'-----------------------------------------------------------------------------------------------------------
Option Compare Database
Option Explicit
Option Base 1

'** Const section
Const nn As Integer = 2000          'maximum number of species
Const nnt As Integer = 100          'maximum number of time periods

Sub NjtExpObs()

'** Declarations
Dim dbCurrent As Database           ‘database object
Dim iit As Integer                  'time period index
Dim j As Integer                    'species index
Dim njt(nn, nnt) As Single          'expected number of quadrants for j in iit
Dim nspec As Integer                'number of species
Dim ntime As Integer                'number of time periods
Dim qry As String                   'any SQL-query
Dim rstIn As Recordset              'recordset for reading
Dim rstOut As Recordset             'recordset for writing

'** Initialisations
Set dbCurrent = CurrentDb()
For j = 1 To nn
   For iit = 1 To nnt
      njt(j, iit) = 0#
   Next iit
Next j
nspec = 1: ntime = 1

‘** Calculations
```

```
52    qry = "SELECT tblFrescaFreq.spec, tblFrescaFreq.jrank, tblTrend.Time, " & _
53    "tblTrend.trank, tblFrescaFreq.samp, tblFrescaFreq.qijt, tblTrend.tfactor " & _
54    "FROM tblFrescaFreq INNER JOIN tblTrend ON tblFrescaFreq.jrank = tblTrend.jrank"
55    Set rstIn = dbCurrent.OpenRecordset(qry, dbOpenSnapshot)
56    With rstIn
57       Do Until .EOF
58          j = !jrank
59          iit = !trank
60          If CStr(!tfactor) <> "1.#INF" Then
61             njt(j, iit) = njt(j, iit) + (1# - Exp(-!qijt * !tfactor))
62          End If
63          If j > nspec Then nspec = j
64          If iit > ntime Then ntime = iit
65          .MoveNext
66       Loop
67    End With
68    rstIn.Close
69
70    On Error Resume Next
71    dbCurrent.TableDefs.Delete "tblSpecTimeNjt"
72    qry = "CREATE TABLE tblSpecTimeNjt (spec string,jrank short,[time] long, " & _
73    "trank short, njtexp single, njtobs short)"
74    dbCurrent.Execute qry, dbFailOnError
75    Set rstOut = dbCurrent.OpenRecordset("tblSpecTimeNjt", dbOpenDynaset)
76    With rstOut
77       For j = 1 To nspec
78          For iit = 1 To ntime
79             .AddNew
80             !jrank = j
81             !trank = iit
82             !njtexp = njt(j, iit)
83             .Update
84          Next iit
85       Next j
86    End With
87    rstOut.Close
88
89    qry = "UPDATE tblSpecTimeNjt INNER JOIN tblTrend ON (tblSpecTimeNjt.trank = " & _
90    "tblTrend.trank) AND (tblSpecTimeNjt.jrank = tblTrend.jrank) SET  " & _
91    "tblSpecTimeNjt.spec=tblTrend.spec,tblSpecTimeNjt.[time]=tblTrend.[time]," & _
92    "tblSpecTimeNjt.Njtobs=tblTrend.count"
93    dbCurrent.Execute qry, dbFailOnError
94
95    End Sub
```