

1 **High Redundancy as well as Complementary Prey Choice**
2 **Characterize Generalist Predator Food Webs in**
3 **Agroecosystems**

4

5 Eve Roubinet ^a, Tomas Jonsson ^{a, b}, Gerard Malsher ^a, Karin Staudacher ^c,
6 Michael Traugott ^c, Barbara Ekbom ^a, Mattias Jonsson ^{a*}

7 ^a Department of Ecology, Swedish University of Agricultural Sciences, Uppsala,
8 Sweden

9 ^b Ecological Modelling group, School of Biosciences, Skövde University, Skövde,
10 Sweden

11 ^c Mountain Agriculture Research Unit, Institute of Ecology, University of Innsbruck,
12 Innsbruck, Austria

13 * Corresponding author: mattias.jonsson@slu.se

14

15

16 **Supplementary information**

17 **Table S1:** Number of predator screened per field and per period during the cropping
18 season.

19 **Table S2:** Predator taxa caught in the pitfall traps.

20 **Table S3a-c:** Final concentration of primers when used in the diagnostic multiplex
21 PCRs.

22 **Fig. S1:** Proportion of predator individuals screened that were positive for a)
23 extraguild prey and b) intraguild prey in eight generalist predator species.

24 **Table S4:** ANOVA table showing the effects of period on the food-web metrics.

25 **Table S5:** For network-level specialization, p-value for each site as the proportion of

26 1000 null model replicates with values lower than the observed value, and p-value

27 from binomial tests as a measure of the overall deviance from null model expectations

28 across all sites.

29 **Table S6:** ANOVA table showing the effect of period on the deviation of the food-

30 web metrics to the mean expected from 1000 null models.

31 **Table S7:** For species-level specialization, p-value for each site as the proportion of

32 1000 null model replicates with values lower than the observed value. On the top row,

33 p-value from binomial tests as a measure of the overall deviance from null model

34 expectations across all sites.

35 **Table S8:** ANOVA table showing the effect of period on the absolute value of the

36 temperature of links.

37 **Fig. S2:** Extraguild prey density in the Early and Late period within the cropping

38 season.

39 **Method S1:** Protocol explaining the DNA-based MGCA of field-collected predators.

40 **Method S2:** Algorithm in R to generate null model that randomly redistributes the

41 observed feeding interactions of a prey among the allowable predators

Table S1: Number of predators (carabids: *Pterostichus melanarius*, *Poecilus cupreus*, *Harpalus rufipes*, *Bembidion lampros*, *Trechus secalis* and spiders: *Agyneta rurestris* [Linyphiidae], *Oedothorax apicatus* [Linyphiidae] and *Pardosa agrestis* [Lycosidae]) screened per field and per period (Early, Late) within the cropping season. Fields 1-5 (in grey) and 6-10 were managed conventionally and organically, respectively.

	Field	<i>P. mel</i>	<i>P. cup</i>	<i>H. ruf</i>	<i>B. lam</i>	<i>T. sec</i>	<i>A. rur</i>	<i>O. api</i>	<i>P. agr</i>
Early period	1	8	16	57	1	16	21	5	3
	2	0	45	36	21	42	3	52	36
	3	12	6	48	28	3	4	4	1
	4	10	4	2	5	0	31	18	3
	5	50	3	48	36	15	2	4	59
	6	50	48	32	11	33	31	13	56
	7	15	15	23	2	0	0	51	58
	8	32	42	23	5	1	12	14	7
	9	1	50	37	41	5	10	30	50
	10	23	11	14	16	4	11	12	52
Late period	1	50	28	31	37	8	22	8	2
	2	4	74	37	39	6	3	31	8
	3	21	1	67	36	3	8	5	0
	4	42	3	37	26	0	0	51	2
	5	117	0	75	56	7	0	5	17
	6	49	41	50	20	11	1	5	16
	7	28	14	24	27	1	2	20	51
	8	50	50	50	49	0	5	20	6
	9	17	29	49	41	5	1	32	8
	10	50	14	48	50	4	8	18	50
Total	-	629	494	788	547	164	175	398	485

50 **Table S2:** Predator taxa caught in the pitfall traps. In addition to these several
51 specimens of Staphylinidae were caught. However, they were not determined to
52 species.

53

Araneae

Pardosa agrestis
Oedothorax apicatus
Meioneta rurestris
Trochosa sp.
Pardosa palustris
Linyphiidae sp.
Erigone atra
Pardosa amentata
Pardosa prativaga
Trochosa ruricola
Porrhomma microphthalmum
Pardosa sp.
Erigone dentipalpis
Trochosa spinipalpis
Oedothorax retusus
Pardosa fulvipes
Pardosa pullata
Micrargus subaequalis
Porrhomma pygmaeum
Walckenaeria nudipalpis

Coleoptera

Harpalus rufipes
Pterostichus melanarius
Bembidion lampros
Poecilus cupreus
Trechus secalis
Bembidion quadrimaculatum
Trechus quadristriatus
Poecilus versicolor
Harpalus affinis
Pterostichus niger
Bembidion guttula
Harpalus seladon
Harpalus tardus
Trechus discus
Harpalus distinguendus
Coccinella septempunctata

Table S3a: Targets, primer names, amplicon length and final concentration (Conc.; * indicates the reduced primer concentration when screening the respective consumer, i.e., beetle or spider) of each group-specific primer (forward and reverse) when used in the “MPI” PCRs. This table was originally published in⁴².

Targets	Primer names (forward-reverse)	Amplicon length (bp)	Conc. (µM)
beetles/thrips	S405-A406	~208	0.2 (0.04*)
Spiders	S407-A408	~258	0.2 (0.04*)
Aphids	S423-A424	148	0.4
Earthworms	S408-A413	85	0.2
Springtails	S411-A415	289	0.2
Dipterans	S414-A416	341	0.4
Lacewings	S417-A420	390	0.2

Table S3b: Targets, primer names, amplicon length and final concentration (Conc.) of each primer (forward and reverse; if concentrations are different, both are listed) when used in the “MPII spiders” PCRs.

Targets	Primer names (forward-reverse)	Amplicon length (bp)	Conc. (µM)
Lycosidae	S486-A488	181	0.4 and 0.6
Linyphiidae	S487-A490	153	0.4
<i>Pachygnatha</i>	S488-A493	249	0.6 and 0.8
Springtails	S411-A415	289	0.1

Table S3c: Targets, primer names, amplicon length and final concentration (Conc.) of each primer (forward and reverse; if concentrations are different, both are listed) when used in the “MPII beetles/thrips” PCRs.

Targets	Primer names (forward-reverse)	Amplicon length (bp)	Conc. (µM)
<i>Poecilus</i>	S475-A486	112	0.15
<i>Pterostichus</i>	S467/S467.1-A467.1	166	0.4 and 0.2
<i>Bembidion</i>	S468-A470	~152	0.15 and 0.1
<i>Harpalus</i>	S473-A475	349	0.6 and 0.55
Thrips	S477-A481	~272	0.2 and 0.4
<i>Coccinella septempunctata</i>	S480-A485	238	0.15 and 0.1

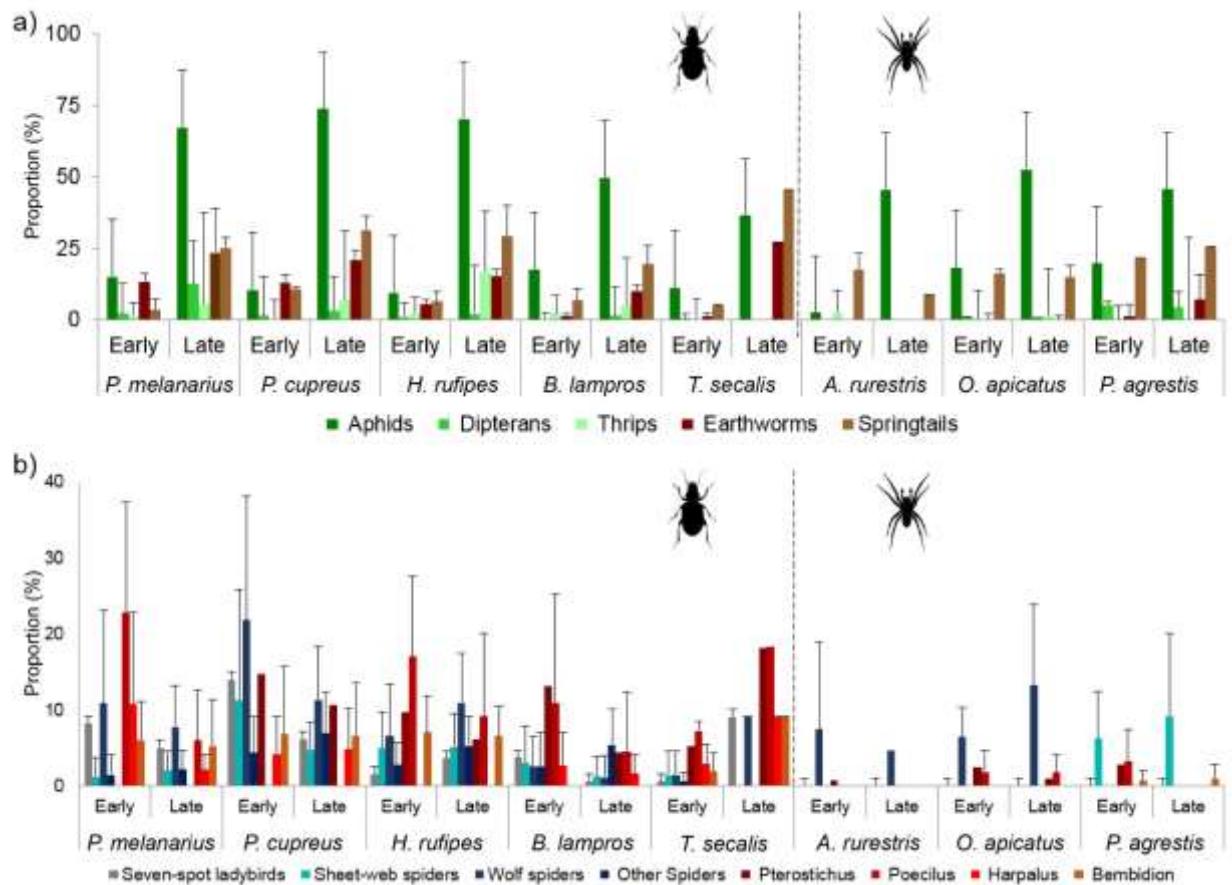


Fig. S1: Proportion of predator individuals screened that were positive for a) extraguild prey and b) intraguild prey in eight generalist predator species (carabids: *Pterostichus melanarius*, *Poecilus cupreus*, *Harpalus rufipes*, *Bembidion lampros*, *Trechus secalis* and spiders: *Agyneta rurestris* [Linyphiidae], *Oedothorax apicatus* [Linyphiidae] and *Pardosa agrestis* [Lycosidae] in the Early and Late period during the cropping season (mean $\pm$ SD detection frequencies in individual fields). Predators were included in the calculation if the number of screened individuals >9 /field/period. Prof Klaus Birkhofer is acknowledged for drawing the invertebrates.

Table S4: ANOVA table showing the effects of period within the cropping season on the food-web metrics. Bold letters highlight significant factors ($\alpha=0.05$). Italicized interactions were removed during model selection based on AIC.

	Factor	Mean Sq	NumDF	DenDF	F.value	Pr(>F)
H ₂	Period	0.14	1	9	53.43	<0.001
d'	Period	0.69	1	117.32	41.2	<0.001
	Predator	0.03	7	117.43	1.54	0.16
	Period:Predator	0.05	7	117.32	3.27	0.003

Table S5: For network-level specialization, p-value for each site (numbered 1 to 10) as the proportion of 1000 null model replicates with values lower than the observed value (Prop) in the Early and Late period within the cropping season (light and dark grey cases respectively). P-values lower than 0.025 or larger than 0.975 are highlighted in yellow. On the last column, p-value from binomial tests (in bold if lower than 0.05) as a measure of the overall deviance from null model expectations across all sites (see methods).

	Field										Binom . Test
	1	2	3	4	5	6	7	8	9	10	
Prop.	0.88	0.99	0.94	0.04	0.98	0.99	0.85	0.84	0.61	0.06	0.012
Early	2	6	6	6	7	8	3	6	6	9	
Prop.			0.97	0.57	0.91		0.64	0.99	0.98		0.001
Late	1	1	9	2	8	1	1	4	5	1	

Table S6: ANOVA table showing the effect of period within the cropping season on the deviation of the food-web metrics to the mean expected from 1000 null models. Bold letters highlight significant factors ($\alpha=0.05$). Italicized interactions were removed during model selection based on AIC.

	Factor	Mean Sq	NumDF	DenDF	F.value	Pr(>F)
$\Delta H_2'$	Period	0.01	1	9	3.53	0.092
$\Delta d'$	Period	0.08	1	124.11	3.38	0.068
	Predator	0.14	7	124.41	6.11	<0.001
	<i>Period:Predator</i>	<i>0.04</i>	<i>7</i>	<i>117.26</i>	<i>1.75</i>	<i>0.104</i>

Table S7: For species-level predator specialization, p-value for each site (numbered 1 to 10) as the proportion of 1000 null model replicates with values lower than the observed value in the Early and Late period (light and dark grey cases respectively). P-values lower than 0.025 or larger than 0.975 are highlighted in yellow. On the top row, p-value from binomial tests (in bold if lower than 0.05) as a measure of the overall deviance from null model expectations across all sites (see methods). The eight generalist predators are: carabids: *Pterostichus melanarius*, *Poecilus cupreus*, *Harpalus rufipes*, *Bembidion lampros*, *Trechus secalis* and spiders: *Agyneta rurestris* (Linyphiidae), *Oedothorax apicatus* (Linyphiidae) and *Pardosa agrestis* (Lycosidae). NA denotes field where no individuals of the particular predator species have been screened.

		<i>P. mel</i>	<i>P. cup</i>	<i>H. ruf</i>	<i>B. lam</i>	<i>T. sec</i>	<i>A. rur</i>	<i>O. api</i>	<i>P. agr</i>
Binom. test		P=0.063	P=1.000	P=0.074	P=1.000	P<0.001	P=0.063	P=0.063	P<0.001
Field	1	0.902	0.740	0.449	NA	0.992	0.017	0.837	0.696
	2	NA	0.947	0.679	0.626	1.000	0.736	0.325	0.982
	3	0.765	0.347	0.995	0.633	NA	0.022	0.669	0.591
	4	0.011	0.169	0.342	0.556	NA	0.505	0.293	0.000
	5	0.717	0.792	0.992	0.733	1.000	0.568	NA	0.633
	6	0.786	0.948	0.282	0.953	1.000	0.648	0.858	0.024
	7	0.992	0.107	0.386	0.875	NA	NA	0.973	0.008
	8	0.895	0.859	0.559	NA	NA	0.487	0.006	0.470
	9	0.628	0.091	0.954	0.121	1.000	0.272	0.800	0.414
	10	0.244	0.274	0.169	0.363	0.976	0.399	0.290	0.025
Binomial s test		P<0.001	P<0.001	P<0.001	P=0.310	P<0.001	P=0.257	P=1.000	P=0.289
Field	1	1.000	0.997	0.996	0.966	1.000	0.997	0.589	0.132
	2	0.997	0.999	0.998	0.551	1.000	0.824	0.944	0.668
	3	0.967	0.569	0.941	0.770	0.958	0.094	NA	NA
	4	0.259	0.688	0.144	0.319	NA	NA	0.727	0.434
	5	0.508	NA	0.106	0.971	0.653	NA	0.631	0.993
	6	1.000	0.999	0.999	0.936	1.000	NA	0.516	0.444
	7	0.492	0.964	0.842	0.270	0.372	0.963	0.054	0.255
	8	0.942	0.909	0.804	0.998	NA	0.703	0.954	0.642
	9	0.975	0.994	0.327	0.327	1.000	0.709	0.032	0.813
	10	1.000	1.000	1.000	0.866	NA	0.271	0.205	0.762

Table S8: ANOVA table showing the effect of period within the cropping season on the absolute value (i.e., the deviation from zero) of the temperature of links (arcsine square-root transformed). Bold letters highlight significant factors ($\alpha=0.05$). Italicized interactions were removed during model selection based on AIC.

	Sum Sq	Mean Sq	NumDF	DenDF	F.value	Pr(>F)
Period	0.06	0.06	1	1390.2	0.30	0.586
Predator	35.10	5.01	7	1390.2	24.46	<0.001
Prey	16.35	1.36	12	1389.2	6.65	<0.001
Predator:Prey	42.21	0.58	73	1388.2	2.82	<0.001
<i>Period:Prey</i>	<i>2.50</i>	<i>0.21</i>	<i>12</i>	<i>1297.6</i>	<i>1.02</i>	<i>0.427</i>
<i>Period: Predator</i>	<i>2.69</i>	<i>0.38</i>	<i>7</i>	<i>1299.0</i>	<i>1.88</i>	<i>0.069</i>
<i>Period: Predator: Prey</i>	<i>14.53</i>	<i>0.20</i>	<i>72</i>	<i>1297.1</i>	<i>0.99</i>	<i>0.505</i>

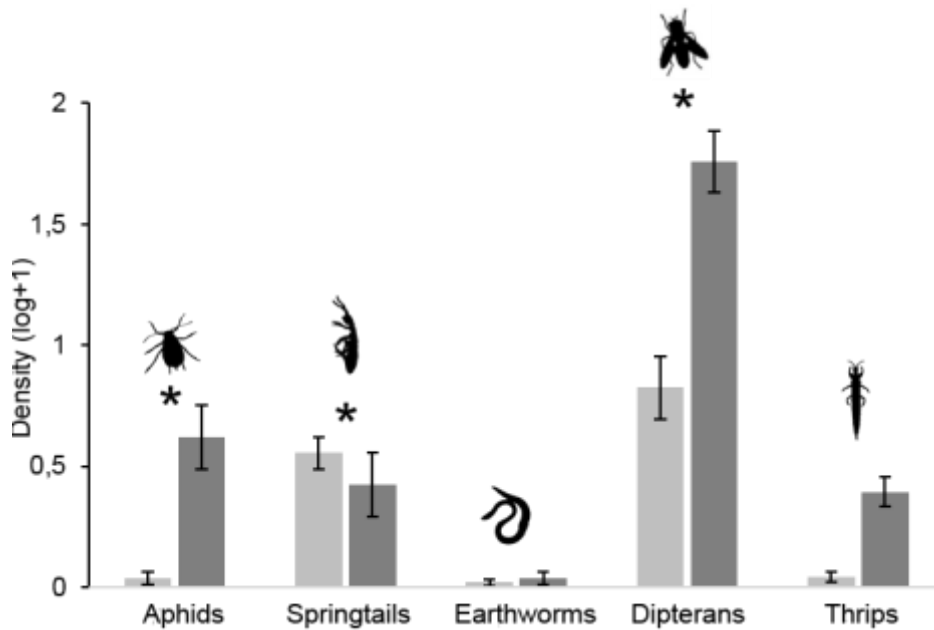


Fig. S2: Extraguild prey density (mean per sample unit (log x+1 transformed)/per field $\pm$ SEM) in the Early (light grey) and Late period (dark grey) of the cropping season. * denotes significant within seasonal variations ($\alpha=0.05$) using permutational analyses of variance for a hierarchical mixed effects model with fixed factors farming system and period within the cropping season for extraguild prey abundances (log x+1 transformed) (Roubinet *et al.* 2017). Prof Klaus Birkhofer is acknowledged for drawing the invertebrates.

Reference:

Roubinet, E., K. Birkhofer, G. Malsher, K. Staudacher, B. Ekbom, M. Traugott, and M. Jonsson. 2017. Diet of generalist predators reflects effects of cropping period and farming system on extra- and intraguild prey. *Ecological Applications* 27:1167–1177.

Method S1: After morphological species identification each predator sample was subjected to whole-body DNA extraction and screened with diagnostic PCR assays largely following the protocols described in⁴⁵ with some modifications (see below and for a detailed description⁴²). A first group-specific multiplex PCR assay (“MPI”) allowed the examination of a predator’s prey choice on a more general level (targeting extraguild: aphids, dipterans, springtails, and earthworms; and intraguild prey groups: lacewing, spiders, beetles/thrips) followed by two additional assays enabling the assessment of IGP at a higher taxonomic resolution (“MPII spiders”: Lycosidae, Linyphiidae, *Pachygnatha*, and “MPII beetles/thrips”: thrips, *Pterostichus*, *Poecilus*, *Harpalus*, *Bembidion*, *Coccinella septempunctata*).

The “MPI” assay was performed in a total volume of 10 µl containing 1.5 µl of DNA extract, 1× QIAGEN Multiplex PCR Master Mix (Qiagen), each primer at its corresponding concentration (Table S2a), 0.5× Q-solution (Qiagen), 5 µg BSA, 30 mM TMAC (Sigma-Aldrich), and PCR-grade water to adjust the volume. Amplifications were carried out under the following thermocycling conditions: 15 min at 95°C, 35 cycles of 30 s at 94°C, 90 s at 60°C and 90 s at 72°C, and 10 min at 72°C.

The “MPII spiders” assay was performed in 10 µl PCRs containing 1.5 µl of DNA extract, 1× Type-it Multiplex PCR Master Mix (Qiagen), each primer at its corresponding concentration (Table S2b), and 5 µg BSA. The thermocycling protocol included an initial activation step of 5 min at 95°C, followed by 35 cycles of 30 s at 95°C, 90 s at 63.5°C and 30 s at 72°C, and 10 min at 68°C. Not all beetles testing positive for spider-prey in “MPI” could be assigned to a specific spider taxon (i.e.,

lower taxonomic level) in "MPII spiders". These predator DNA samples were subsequently subjected to DNA barcoding to resolve feeding interactions that occurred with other spider taxa than the targeted Lycosidae, Linyphiidae, and *Pachygnatha*. All barcoded spider-prey samples were assigned to family level to make results comparable to the ones of the "MPII spiders" assay.

The PCR protocol of the "MPII beetles/thrips" assay differed only slightly from the "MPII spiders": 1.5 µl of DNA extract and 30 mM TMAAC (Sigma-Aldrich) were used in the total volume of 10 µl (plus PCR-grade water to adjust the volume); for primer concentrations see Table S2c. Thermocycling conditions were as described above, but with an annealing temperature of 63.5°C. A detailed description of primer sequences and assay specificity can be found in⁴⁵. All PCR products were separated and visualized using the QIAxcel electrophoresis system (Qiagen) following the protocol described in⁴⁵.

174 **Method S2:** Script to generate random null-model versions of an observed matrix
175 with integer frequencies. The random matrices are filled row by row, assuming that
176 the observed frequencies are multinomially redistributed among the allowed entries,
177 making sure that the row sums and column sums stay the same as in the observed
178 matrix. Forbidden entries (e.g. impossible links between predator and prey) should be
179 coded by -1 in the observed matrix.

180 The necessary inputs are:

- 181 - n_repl: number of null-model replicates to generate
- 182 - InMat: observed matrix with integer frequencies
- 183 - attempt_max: maximum number of iterations to try for obtaining the desired
184 number of replicates. Obviously, attempt_max > n_repl (e.g. set attempt_max
185 to 10 or 100 times n_repl)

186 The output is a list of random matrices (Rand_mats).

187 The script has been written by Tomas Jonsson (25 January 2016).

188 _____
189 randmats_multinomial2 = function(n_repl, InMat, pred_num, attempt_max) {
190 #-----

191 **#INITIALIZATION:**

192 **#Check that desired number of replicates does not exceed number of iteration**

193 **attempts:**

```
194 if (attempt_max < n_repl) {  
195     stop('Number of iteration attempts can not be smaller than the desired number  
196 of replicates!')  
197 }
```

198

```

199  #Create working matrix with coding for forbidden links removed:
200  InMat2 = InMat;
201  InMat2[InMat==-1] = 0; #remove -1 from observed matrix and replace with zeros:
202
203  #Obtain web statistics:
204  row_sums = rowSums(InMat2); #row sums
205  col_sums = colSums(InMat2); #column sums
206  n_obs = sum(InMat2); #total number of observations
207  n_rows = length(row_sums); #number of rows
208  n_cols = length(col_sums); #number of columns
209
210  #Initialize list for storing random matrices:
211  Rand_mats = vector('list',n_repl);
212
213  #By default, the random number generator (RNG) is seeded based on the current time
214  so that rand produces a different sequence of numbers after each time rng is called. If
215  the same random number sequence is wanted, use set.seed to set RNG to a known
216  state, e.g.: set.seed(12)
217
218  repl = 0; #set counter for number of successful replicates to zero
219  attempts = 0; #set counter for number of iteration attempts to zero
220
221  # Obtain loop order for rows (first all-zero-rows, then rows with forbidden links
222  sorted in descending order of number of forbidden links, followed by rows with
223  nonzero elements and no forbidden links):

```



```

224 all_zeros_ri = which(row_sums==0) #rows with all zero elements (no non-zero
225 entries)
226 A_temp = matrix(0,nrow=n_rows,ncol=n_cols); A_temp[InMat==1]=1 #temporary
227 matrix with position of forbidden links coded by one...
228 row_sums_A_temp = rowSums(A_temp); #Number of forbidden links per row
229 col_sums_A_temp = colSums(A_temp); #Number of forbidden links per column
230 sort_res = sort(row_sums_A_temp,decreasing = TRUE,index.return = 1) #obtain sort
231 order for rows in descending order after number of forbidden links
232 r = sort_res$x; fl_sort_i = sort_res$ix
233 nonzero_ri = setdiff(fl_sort_i,all_zeros_ri);
234 row_order= c(all_zeros_ri, nonzero_ri); #loop through rows in this order
235
236 #-----
237 #MAIN ALGORITHM:
238 while (repl < n_repl) { #Outer loop for generating a set of random matrices
239     next_attempt_flag = 0; #Flag used to jump to next iteration attempt
240     attempts = (attempts + 1); #update number of iteration attempts
241
242     #Initialize random matrix and necessary parameters:
243     rnd_mat = matrix(0,nrow=n_rows,ncol=n_cols);
244     rem_colfreq = col_sums; #Remaining column frequencies (i.e. number of
245 observations remaining to distribute in columns)
246     rem_rowfreq = row_sums #Remaining row frequencies (i.e. number of
247 observations remaining to distribute on rows)

```

```

248         rem_obs = n_obs; #Total number of observations remaining to distribute in
249 matrix
250
251         #Main loop for allocating random numbers to matrix:
252         for (r_ind in 1:(n_rows-1)) { #loop through all but "last" row:
253             if (row_sums[row_order[r_ind]] !=0) { #ignore rows with no non-zeros
254 entries!
255
256                 #Obtain loop order for elements (columns) on row in question
257 (i.e. row_order[r_ind]):
258
259                 all_zeros_ci = which(col_sums==0) #columns with no non-
260 zeros entries
261
262                 fl_ind1 = which(InMat[row_order[r_ind],] == -1); #elements
263 with forbidden links on row in question
264
265                 fl_ind2 = which(InMat[row_order[r_ind+1],] == -1); #elements
266 with forbidden links on next row
267
268                 c_ind_temp1 = sort(union(all_zeros_ci,fl_ind1)); #union of
269 elements with column sum equal to zero and elements with forbidden links on row in
270 question...
271
272                 c_ind_temp2 = sort(union(c_ind_temp1,fl_ind2)); #union of
273 elements with column sum equal to zero and elements with forbidden links on this and
274 next row...
275
276                 c_ind_temp3 = setdiff((1:n_cols),c_ind_temp2); #Extract
277 indices in 1:n_cols that are not in c_ind_temp2
278
279                 col_order = c(c_ind_temp2, c_ind_temp3); #loop through
280 elements on row l in this order

```

```

273          #Generate multinomially distributed random numbers:
274          multinom_nn = length(c_ind_temp3) #number of allowed
275 entries on row l (i.e. number of potentially non-zero elements)
276          multinom_size = row_sums[row_order[r_ind]] #total number
277 of frequencies to distribute on row l
278          #multinom_prob = rep(1,multinom_nn) #relative probabilities
279 for elements
280          multinom_prob = pred_num/sum(pred_num)
281          multinom_freq = rmultinom(1:multinom_nn , multinom_size,
282 multinom_prob) #multinomially distributed random numbers!
283          for (c_ind in 1:(n_cols-1)) { #loop through all but "last"
284 element (column) of row l
285          #Check that element (i) is not in a column with initial
286 or (ii) does not have remaining column sum equal to zero and (iii) that element does
287 not forbidd non-zero entry:
288          if (col_sums[col_order[c_ind]] != 0) { #ignore elements
289 with column sum equal to zero!
290          if (rem_colfreq[col_order[c_ind]] != 0) {#ignore
291 elements with remaining column sum equal to zero
292          if
293 (InMat[row_order[r_ind],col_order[c_ind]] != -1) { #Ignore elements that forbidd
294 non-zero entry
295          #Check whether all forthcoming
296 elements in this column are forbidden, if so put all remaining observations for this
297 column here!

```

```

298                                     if
299 (all(InMat[row_order[(r_ind+1):n_rows],col_order[c_ind]] == -1)) {
300     rnd_mat[row_order[r_ind],col_order[c_ind]] = rem_colfreq[col_order[c_ind]];
301     rem_rowfreq[row_order[r_ind]] = rem_rowfreq[row_order[r_ind]] -
302 rem_colfreq[col_order[c_ind]] #Update remaining number of obs to distribute on row
303 l
304                                     rem_obs = rem_obs -
305 rem_colfreq[col_order[c_ind]]
306     rem_colfreq[col_order[c_ind]] = 0
307                                     } else {
308                                     if
309 (rem_rowfreq[row_order[r_ind]] == 0) {#row is full
310                                     break #terminate
311 (break out of) c_ind loop!
312                                     }
313                                     #Allocate randomly
314 picked multinomially distributed number within interval set by constraints to (i) not
315 exceed remaining column or rowsum for element in question, and (ii) be able to
316 reach rowsum with remainig columnsums:
317                                     rnd_lmn =
318 multinom_freq[col_order[c_ind]] #
319                                     rnd_max =
320 min(rem_colfreq[col_order[c_ind]],rem_rowfreq[row_order[r_ind]]); #maximum
321 value for random number that does not exceed remaining column or rowsum
322 frequencies

```

```

323                                     rnd_min =
324 rem_rowfreq[row_order[r_ind]] - sum(rem_colfreq[col_order[(c_ind+1):n_cols]]);
325 #minimum value needed for random number in order for remaining rowsum to be
326 reached with remaining columnsums
327                                     if (rnd_min < 0) {
328                                     rnd_min = 0;
329                                     }
330                                     if (rnd_min > rnd_max)
331 {#
332                                     warning('Negative
333 interval attempted for random number generator!')
334                                     next_attempt_flag
335 = 1;
336                                     break #terminate
337 (break out of column loop: for c_ind = 1:(n_cols-1))
338                                     } else {#
339                                     if (rnd_min ==
340 rnd_max) {
341                                     rnd_lmn =
342 rnd_min; #frequency can only take on this value!
343                                     } else {#rnd_min
344 < rnd_max
345                                     while
346 (rnd_lmn>rnd_max) {#
347         rnd_lmn = rnd_lmn-1

```

```

348                                     }
349                                     while
350 (rnd_lmn<rnd_min) {#
351     rnd_lmn = rnd_lmn+1
352                                     }
353                                     }
354     }
355
356                                     #Check whether
357 suggested random element fulfils remaining criteria, if not make corrections...
358                                     z = 1;
359                                     while (z == 1) {#
360                                     w1 = rem_obs -
361 rnd_lmn; #total number of observations remaining to distribute in matrix if rnd_lmn
362 is accepted
363                                     w2 = rem_colfreq;
364     w2[col_order[c_ind]] = w2[col_order[c_ind]] - rnd_lmn; #remaining column
365 frequencies if rnd_lmn is accepted
366                                     w3 = sum(w2);
367 #total remaining observations that can be distributed in "available" elements based
368 on remaining column frequencies
369                                     if (rnd_lmn >
370 rnd_max) {
371     warning('Algorithm failure, next attempt...')
372     next_attempt_flag = 1;

```

```

373                                     break
374  #terminate (break out of) while loop
375                                     }
376                                     if (rnd_lmn <
377  rnd_min) {
378         warning('Algorithm failure, next attempt...')
379         next_attempt_flag = 1;
380                                     break
381  #terminate (break out of) while loop
382                                     }
383
384                                     #Is suggested
385  random number too large or too small?
386                                     if (w1 < w3) {
387                                             rnd_lmn =
388  rnd_lmn - 1;
389                                     } else {
390                                             if (w1 >
391  w3) {
392             rnd_lmn = rnd_lmn + 1;
393                                     } else
394  {#w1 == w3
395             break
396                                     }
397                                     }

```

```

398                                     } #end of while loop
399 (z==1)
400                                     if (next_attempt_flag ==
401 0) {
402         rnd_mat[row_order[r_ind],col_order[c_ind]] = rnd_lmn; #element m of row l
403         rem_rowfreq[row_order[r_ind]] = rem_rowfreq[row_order[r_ind]] - rnd_lmn;
404         #Update remaining number of obs to distribute on row l
405         rem_colfreq[col_order[c_ind]] = rem_colfreq[col_order[c_ind]] - rnd_lmn;
406         #Update remaining number of obs to distribute in column m
407                                     rem_obs =
408 rem_obs - rnd_lmn; #Update total number of remaining observations to distribute
409                                     }
410                                     }
411                                     }
412         }
413     }
414     } #end of column loop (for c_ind in 1:(n_cols-1))
415     if (next_attempt_flag == 1) {
416         break
417     } else {
418         if (rem_rowfreq[row_order[r_ind]] != 0) {
419             rnd_mat[row_order[r_ind],col_order[n_cols]] =
420 rem_rowfreq[row_order[r_ind]]; #last element (column) on row in question

```



```

421 rem_colfreq[col_order[n_cols]] =
422 rem_colfreq[col_order[n_cols]] - rem_rowfreq[row_order[r_ind]]; #Update
423 remaining number of obs to distribute in column in question
424 rem_obs = rem_obs -
425 rem_rowfreq[row_order[r_ind]]; #Update total number of remaining observations to
426 distribute
427 rem_rowfreq[row_order[r_ind]] = 0; #Update
428 remaining number of obs to distribute on row in question
429 }
430 }
431 }
432 } #end of row loop (for r_ind in 1:(n_rows-1))
433 if (next_attempt_flag == 0) {#
434     #Compute entries of "last" row of rnd_mat:
435     ind_lr = which(InMat[row_order[n_rows],] != -1); #loop through
436     elements on last row in this order
437     #ind_lr = sample(ind_lr) #randomly permute order of elements
438     for (c_ind in 1:(length(ind_lr)-1)) {
439         if (InMat[row_order[n_rows],ind_lr[c_ind]] != -1) { #Check
440             that element does not forbidd non-zero entry
441             rnd_mat[row_order[n_rows],ind_lr[c_ind]] =
442             rem_colfreq[ind_lr[c_ind]];
443             rem_rowfreq[row_order[n_rows]] =
444             rem_rowfreq[row_order[n_rows]] - rem_colfreq[ind_lr[c_ind]];

```

```

445             rem_obs = rem_obs - rem_colfreq[ind_lr[c_ind]];
446     #Update total number of remaining observations to distribute
447             rem_colfreq[ind_lr[c_ind]] = 0;
448     }
449 }
450     #last element of last row:
451     rnd_mat[row_order[n_rows],ind_lr[length(ind_lr)]] =
452     rem_rowfreq[row_order[n_rows]];
453     #Make final check that row and column sums match observed
454     matrix:
455     row_sums_rnd = rowSums(abs(rnd_mat)); #row sums of absolute
456     element values in random matrix
457     col_sums_rnd = colSums(abs(rnd_mat)); #column sums of absolute
458     element values in random matrix
459     if (all(all(row_sums_rnd == row_sums) && all(col_sums_rnd ==
460     col_sums))) {
461         repl = (repl + 1);
462         Rand_mats[[repl]] = rnd_mat;
463     }
464 }
465     if (attempts > attempt_max) {
466         message('Maximum number of attempts (',attempt_max,') exceeded!')
467         break
468     }
469

```

```
470 }  
471 message('Number of attempts = ',attempts)  
472  
473 #Collect Output (random matrices...):  
474 return(list(Rand_mats))  
475 }  
476
```