Electronic Supplementary Material

Kin Selection Model

Here we show how we derived the marginal fitness of helpers (equation 3) used in the kin selection approach. We use d to denote the probability of dispersal for a focal offspring, $\bar{d}$ to denote the average level of dispersal from the natal patch, and d' to denote the population average level of dispersal. To calculate the probability that an offspring becomes a breeder, we note that a helper will share a patch with $(1-\bar{d})K$ other helpers and $d'cK$ non-helpers who dispersed to the patch. Thus, if the breeder dies, a focal helper will become a breeder with probability $[(1-d)]/[K((1-\bar{d})+d'c)]$. A non-helper will be one of $d'cK$ dispersers and will share the contested patch with $(1-d')K$ helpers. Thus if the breeder dies, all else being equal, a non-helper will become a breeder with probability $dc/[K(1-d')+Kd'c]$. Therefore, the probability of a focal offspring becoming a breeder is given by:

$$w_{AJ} = \left(1 - S_b\left(\bar{d}\right)\right)\frac{1-d}{K\left(\left(1-\bar{d}\right)+cd'\right)} + \left(1 - S_b\left(d'\right)\right)\frac{cd}{K\left(\left(1-d'\right)+cd'\right)} \quad . \qquad \text{(A1)}$$

As the notation in (A1) suggests, w_{AJ} can also be interpreted as the breeding-adult (A) component of fitness belonging to focal juvenile (J) individual.

There is also a juvenile (J) component to the fitness of a focal juvenile (J) individual, $w_{JJ} = Kw_{AJ}$. Note that this expression weights the focal individual's genetic contribution to its K locally produced offspring by 0.5, and weights the focal individual's genetic contribution to its K offspring produced on other patches by the same amount.

Similar fitness measures for adult focal individuals include, $w_{AA} = S_b(\bar{d})$ (for the breeding-adult component of fitness) and $w_{JA} = KS_b(\bar{d})$ (for the juvenile component of fitness). These fitness components are combined using the neutral reproductive values and

relative age-class frequencies, calculated as the dominant left and right eigenvectors, respectively, of the matrix,

$$W = \left(\begin{array}{cc} w_{JJ} & w_{JA} \\ w_{AJ} & w_{AA} \end{array} \right)_{d=\bar{d}=d'} = \left(\begin{array}{cc} 1-S_b(d') & KS_b(d') \\ (1-S_b(d'))/K & S_b(d') \end{array} \right) . \qquad \text{(A2)}$$

It is easy to show that the dominant eigenvalue of W in (A2) is one. It follows that the reproductive values of juveniles (J) and adults (A) are the elements of

$$\mathbf{v} = [\, v_J \;\; v_A \,] = [(1 - S_b(d'))/KS_b(d'), 1],$$

and the relative frequencies of juveniles (J) and adults (A) are the elements of

$$\mathbf{u} = [\, u_J \;\; u_A \,]^T = [\, K, \;\; 1 \,]^T .$$

Following Taylor and Frank's approach [27] we weight the juvenile and adult components of fitness to construct the fitness of a focal juvenile (J), as

$$W_J = v_J w_{JJ} u_J + v_A w_{AJ} u_J = \left(\frac{1-S_b(d')}{S_b(d')} + 1 \right) K w_{AJ} ,$$

$$= \left(\frac{1}{S_b(d')} \right) \left[\left(1-S_b(\bar{d})\right) \frac{1-d}{\left((1-\bar{d})+cd'\right)} + \left(1-S_b(d')\right) \frac{cd}{\left((1-d')+cd'\right)} \right] , \qquad \text{(A3)}$$

and the fitness of a focal adult (A) as

$$W_A = v_J w_{JA} u_A + v_A w_{AA} u_A$$

$$= \frac{1-S_b(d')}{KS_b(d')} K w_{AA} + w_{AA} = \left(\frac{1-S_b(d')}{S_b(d')} + 1 \right) w_{AA} = \frac{S_b(\bar{d})}{S_b(d')} . \qquad \text{(A4)}$$

The direction of selection acting upon the dispersal trait, d is given by the marginal fitness with respect to that trait. This is given by:

$$\frac{\mathrm{d}W}{\mathrm{d}d'} = \frac{\mathrm{d}W_J}{\mathrm{d}d'} + \frac{\mathrm{d}W_A}{\mathrm{d}d'} \qquad \text{(A5)}$$

These derivatives may be separated into partial derivatives, describing the effect of an individual's trait value, and the effect of the average trait value of its group, on the juvenile and adult class fitnesses, using the chain rule methodology of Taylor and Frank [27] (see also [28, 42]). This gives $\mathrm{d}W_J/\mathrm{d}d' = \partial W_J/\partial d + \bar{r}\,\partial W_J/\partial \bar{d}$ and $\mathrm{d}W_A/\mathrm{d}d' = \partial W_A/\partial d + \bar{R}\,\partial W_A/\partial \bar{d}$, where $\bar{r} = \mathrm{d}\bar{d}/\mathrm{d}d$ is the kin-selection coefficient of (whole-group) relatedness between a focal individual and other members of the juvenile class, and $\bar{R} = \mathrm{d}\bar{d}/\mathrm{d}d$ is the kin-selection coefficient of (whole-group) relatedness between a focal individual and members of the adult class [27, 43]. This yields an expression that is equal up to multiplication by a positive function of d' (namely $1/S_b(d')$) to the marginal fitness given in the main text (equation 3). Note that the main text uses the notation, $d' = d$.

Individual based simulation

We constructed an individual-based simulation of the model using the R2011a release of the MATLAB software package (MathWorks Inc., Boston, MA). The simulation was formulated on the same set of assumptions used in the kin-selection based model.

In our simulation, each breeder was initially assigned two random numbers that, respectively, corresponded to two dispersal phenotypes: one random number corresponded to the "maternally" inherited dispersal phenotype, and the other corresponded to the

68 "paternally" inherited dispersal phenotype. Each run of the simulation repeated the series of
69 events outlined in main text. However, in the simulation, dispersal and competition were
70 combined for the purpose of efficiency. Specifically, simulated competition for vacant
71 patches was hierarchical in nature, so that the winner of the patch was native (i.e. a helper)
72 with probability $\frac{1-Average\ d\ of\ Vacant\ Patch}{c(Population\ Average\ d)+1-Average\ d\ of\ Vacant\ Patch}$, and non-native (i.e. a non-
73 helper) from a specific foreign patch with probability
74 $\frac{Average\ d\ of\ Specific\ Foreign\ Patch}{c(Population\ Average\ d)+1-Average\ d\ of\ Vacant\ Patch}$. Once the natal patch of the winner was
75 established, the winner itself was identified by choosing randomly from among all the natal
76 offspring of that patch in such a way that the probability of an individual being chosen was
77 either $(1 - d)/(1 -$ winner's natal patch average $d)$, in the case that the winner was born on the
78 vacant patch, or d/winner's natal patch average d, in the case that the winner was not born on
79 the vacant patch.
80 Typical collections of 32 simulated evolutionary trajectories are presented as thin grey
81 lines in Figure A1. Trajectories were simulated for 3000 generations — long enough for an
82 equilibrium level of dispersal (say, d*) to become established. For each simulated trajectory
83 we fixed the number of patches at $P = 1000$, and we fixed the fecundity of each breeder at K
84 = 100 offspring. Smaller values of P and K tended to produce unacceptably large
85 fluctuations among runs; the larger values used provided us with a consistent estimate of d^*.
86 We investigated the relationship between the adaptive level of helping $(1 - d^*)$
87 predicted by simulation, and the extent of promiscuity in the model population (M). We
88 found that the predicted level of helping increased with increasing promiscuity (Figure 4).
89 Hence, the tendency to help juveniles born on the same patch increased with decreasing
90 relatedness (from increasing promiscuity).
91 Here we provide the Matlab script used to simulate the evolution of the helping trait, 1
92 – d. The particular script presented here generates 32 evolutionary trajectories, then plots

each trajectory alongside the average trajectory and an approximate 95 % confidence interval.

```
Paths=32;          % number of Paths to simulate
T=3000;            % number of generations to simulate
P=1000;            % number of patches
MutStep=0.001;     % mutation step
Data=NaN(Paths,T);

M=2;     % number of mates
K=100;   % number of offspring
c=1.5;   % parameter
k=1;   % parameter

Sb=@(d) 1-exp(-k*(1-d));

cede=632;
myStream = RandStream.create('mt19937ar','seed',cede);
          RandStream.setDefaultStream(myStream);

tic
parfor path=1:Paths
% use “for” in place of “parfor” here if missing Parallel Comp Toolbox

   % Initialize Population

    Popn=rand(2,P); % one individual per patch
                    % two chromosomes per individual

    for t=1:T
        mates=NaN(M,P); % stores index of each mate of each individual
        for i=1:P
            mates(:,i)=randsample(P,M,true); % "true"=sample w replacement
        end

        Offspring=NaN(K,2,P); % stores the genotypes of each offspring
        for i=1:P

            Fathers=randsample(M,K,true);
            PatContribs=randsample(2,K,true);
            MatContribs=randsample(2,K,true);

            for j=1:K
                Offspring(j,1,i)=Popn(PatContribs(j), mates(Fathers(j),
i));
                Offspring(j,2,i)=Popn(MatContribs(j), i);
            end

        end

        OffspringPhenotype=reshape(mean(Offspring,2),K,P);
        AvgPhenotype=mean(OffspringPhenotype);
        GlobalAvgPhenotype=mean(AvgPhenotype');

        ProbLocal=(1-AvgPhenotype)./(c*GlobalAvgPhenotype + (1-
AvgPhenotype));
        BreederSurvival=Sb(AvgPhenotype);
```

```
150          PopnNext=NaN(2,P);
151          for i=1:P
152              X=rand();
153              if (X > BreederSurvival(i))
154                  % Breeder Dies
155                  Y=rand();
156                  if (Y < ProbLocal(i))
157                      % Winner Comes from Local Patch
158                      PatchWinner=i;
159
160                      % Local Offspring Compete
161                      Winner=...
162                          randsample(1:K, 1, true, 1-
163  OffspringPhenotype(:,PatchWinner)');
164                  else
165                      % Patches Compete
166                      PatchWinner=randsample(1:P, 1, true, AvgPhenotype);
167
168                      % Offspring On Winning Patch Compete
169                      Winner=...
170                          randsample(1:K, 1, true,
171  OffspringPhenotype(:,PatchWinner)');
172                  end
173
174                  WinnersGenes=reshape(Offspring(Winner,:,PatchWinner),1,2);
175
176                  % Mutate Winner's Genes
177                  WinnersGenes=WinnersGenes + MutStep*randn(1,2);
178                  WinnersGenes(WinnersGenes <= 0) = MutStep;
179                  WinnersGenes(WinnersGenes >= 1) = 1-MutStep;
180
181                  % Update Next Gen Population Array
182                  PopnNext(:,i) = WinnersGenes';
183              else
184                  % Breeder Survives
185                  PopnNext(:,i)=Popn(:,i);
186              end
187          end
188
189      Popn=PopnNext;
190
191      % Record Data
192      Data(path,t)=mean(mean(Popn,2));
193
194      end
195  end
196  toc
197
198  MeanPath=mean(Data);
199  SE=sqrt(var(Data)/Paths);
200
201  % --- Plot Data ----
202
203  figure
204  hold on
205
206  for path=1:Paths
207      plot(1:T, 1-Data(path,:), 'Color', [0.8,0.8,0.8], 'LineWidth',0.5);
208  end
209
```

```
plot(1:T, 1-MeanPath, '-r', 'LineWidth',2);
plot(1:T, 1-MeanPath+1.96*SE, '--r');
plot(1:T, 1-MeanPath-1.96*SE, '--r');
%plot([0,T],[NumPred, NumPred], '--k', 'LineWidth',2);

axis([0,T,0,1])
xlabel('time', 'FontSize', 16, 'FontName', 'Arial')
ylabel('average level of helping (1-d)', 'FontSize', 16, 'FontName',
'Arial')
 title(['M = ', num2str(M), 'c = ', num2str(c), ' k = ', num2str(k)], ...
            'FontSize', 12)
box on
```

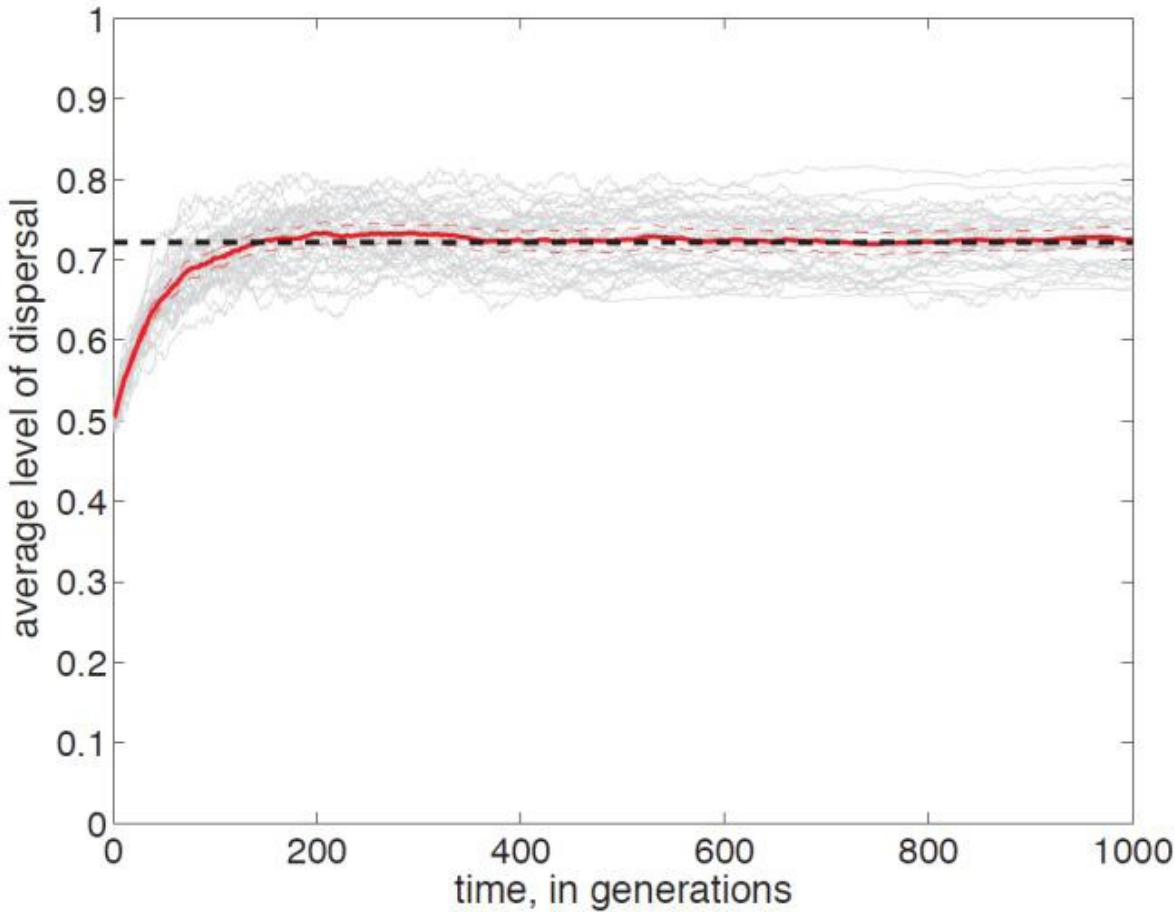

Figure A1: Level of helping over time. Typical collections of 32 simulated evolutionary trajectories are presented as thin grey lines. Red line indicates the mean of the runs. Dashed line indicates the equilibrium level of dispersal (d^*).