Supplementary Materials for

Effects of network modularity on the spread of perturbation impact in experimental metapopulations

Luis J. Gilarranz, Bronwyn Rayfield, Gustavo Liñán-Cembrano, Jordi Bascompte, Andrew Gonzalez*

*Corresponding author. Email: andrew.gonzalez@mcgill.ca

Published 14 July 2017, *Science* **357**, 199 (2017)
DOI: 10.1126/science.aal4122

This PDF file includes:

Other Supplementary Material for this manuscript includes the following:
(available at www.sciencemag.org/content/357/6347/199/suppl/DC1)

Materials and Methods

Network design and construction

The microcosms were maintained in a controlled laboratory setting and consisted of a single microarthropod species, *Folsomia candida* Willem, living in networks of artificial habitat.

The networks were composed of habitat patches (nodes) where *F. candida* could live and links allowed the movement of individuals between habitat patches. Habitat patches were constructed by filling polypropylene vials (85 mm height, 30 mm inner diameter) 2/3 full with a mixture of de-ionized water and activated charcoal:plaster of Paris in a 1:8 ratio by weight. This created a living surface of 706.9 mm^2 per habitat patch (Figure S1a). Each vial had a 1 mm hole pierced into its lid to allow for air exchange. The lid of each vial was closed at all times other than when the habitat patch was being photographed. Holes were burned into the walls of the vials at the level of the living surface using a heated metal rod to provide access to the tubing that served as links among habitat patches. Careful consideration of the angle at which the holes had to be positioned was vital to prevent the tubes from being kinked. This was as important as the spatial positions of the vials themselves. Links were made of flexible Tygon tubing (70 mm long, 6 mm outside diameter, and 0.8 mm wall thickness). It is important to note that the links themselves were not habitable and only served as movement conduits.

To create connectors that would allow the links to be attached to the habitat patches, the wide ends of pipette tips (Eppendorf epT.I.P.S) were cut (15 mm long) and glued into the holes that had been burned into the walls of the vials. These connectors were glued in place prior to filling the vials with the plaster of Paris mixture, which provided the habitat surface. The final step was to fit the tubing over the connectors to create links between the habitat patches (Figure S1b).

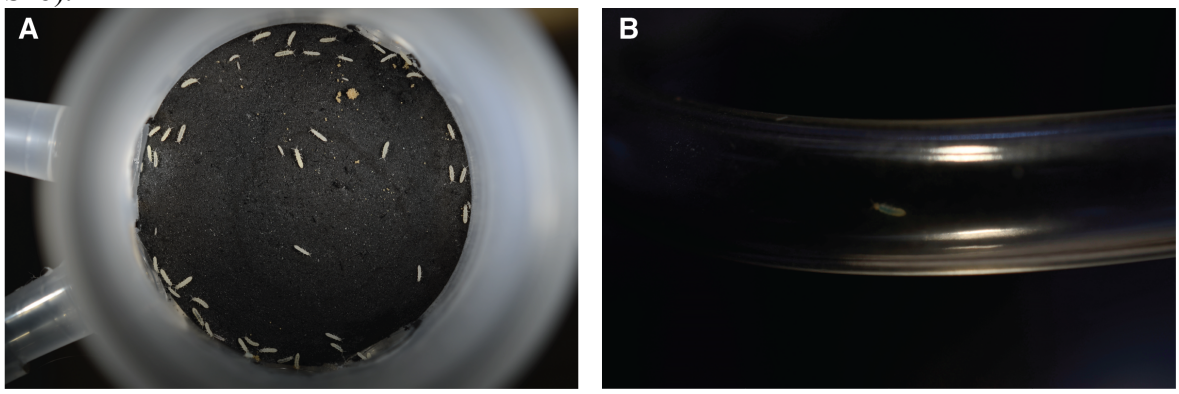

Figure S1: Nodes and links. **(A)** Surface of a node. The collembola and eggs contrast sharply against the black substrate. **(B)** An individual travelling through a link.

The pattern of links between habitat patches was designed to control for network modularity. Some elements are worth mentioning:

1) A single network structural template served as the basis for all replicates. It was designed creating four dense networks with an Erdös-Renyi degree distribution. Then we randomly assigned links between those small networks that became the modules of the larger network. Following this procedure, the final network consisted of 20 nodes with an average degree of 4, and an Erdös-Rényi degree distribution with a highly modular configuration.

Note that no coordinates are taken into account in the algorithm that assigns the links between nodes. Modularity was calculated following ref. (*11*). The final network configuration had an absolute modularity value of 0.56.

2) The template defined not only the connectivity pattern but also the spatial layout of the habitat patches and links. Habitat patches were glued onto corrugated cardboard to fix their spatial locations. Each network covered a total area of 30 x 30 cm.
3) Habitat patch numbering was preserved across replicates such that, for example, habitat patch 7 was always in the same position in all replicates (with the same neighbors and links).
4) All links had the same length but differed in their shape depending on the relative spatial locations of the connected nodes. This was achieved due to the flexibility of the tubing and that allowed us to create a spatial network structure controlling for effects of link length. Hence, nodes only varied in their degree, while distances among nodes separated by a given number of links were constant.

Once habitat networks were constructed, they were populated with *F. candida* individuals by tapping a culture container over each habitat patch vial. This method was preferred over aspiration as it resulted in fewer accidental injuries or deaths during transfer. All habitat patches received a similar number of individuals per habitat patch (between 25 and 47 with a median of 33). For the duration of the experiment, microcosms were stored in polyethylene bins to minimize light exposure.

Population maintenance and perturbation

F. candida has been used in ecotoxicology as a standard test organism for over 40 years—reviewed in Ref. (17). For this study, we obtained a taxonomically confirmed laboratory culture of *F. candida* from Environment Canada. The culture was acclimated to a daily average temperature of 20 +/- 2°C and this temperature was maintained in the laboratory for the duration of the experiment. Synchronized cultures were induced by transferring adult collembola to fresh plaster of Paris substrate, removing them after 3 days, and leaving behind only their eggs. Once the eggs had hatched (10 days) and the juveniles were 9-12 days old, these collembola formed the base of the synchronized culture. The substrate was watered with de-ionized water once per week to the point of being moist but not waterlogged. Each habitat patch received 2 mg of granulated dry Baker's yeast per week. Any uneaten yeast was removed prior to adding new yeast. The age-structure of the culture was heterogeneous and the culture population was at carrying capacity when the culture was used to populate the experimental microcosms. The weekly watering regime established in culture maintenance was continued during the course of the experiment. For the duration of the experiment, collembola in all habitat patches were fed *ad libitum*. Each habitat patch received 2 mg of granulated dry Baker's yeast per week. Again, any uneaten yeast was removed prior to adding new yeast. Therefore, the carrying capacity of the nodes was limited by habitat size and not by food availability.

To sample the populations, a photograph was taken of every node in each experimental habitat network with a frequency of one or two days. The camera was on a fixed mount. We used a microscopy illuminator as a light source (Figure S2). See section below for details about the camera settings.

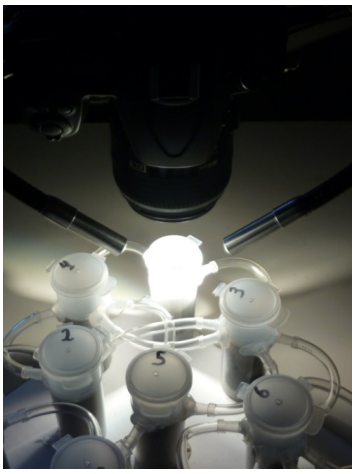

Figure S2: Illumination and camera position. Abundant light levels were needed because of the fast camera shutter speed. A fast shutter speed and large aperture were used to avoid blurred images of the highly mobile collembola individuals.

We began the experimental perturbation on day 50—once all patches had reached their carrying capacity—and we stopped it on day 105. The perturbation occurred only in node #5. Perturbations were carried out every day, and were accomplished by removing all individuals and eggs from the perturbed habitat patch via aspiration. Only individuals on the perturbed habitat patch itself were removed and any collembola in adjacent connectors or links were left undisturbed. Half of the replicate networks were perturbed and the other half was treated as a control. The watering and feeding schedule was not affected during the perturbation, even in the perturbed node.

Image recognition analysis

The white bodies of the collembola are in sharp contrast with the living substrate. We developed an algorithm to count the number of individuals in each of the vials for the duration of the experiment. Over the course of the experiment we took 14,000 images where each image represents one of the local populations (nodes) at a single point in time. The results of the algorithm were validated against a manually-counted time series and then used to process the entire database of images. Only collembola larger than 0.8mm in length were counted by the algorithm.

The Database of Images

The images we used to estimate population size are available online:
http://www2.imse-cnm.csic.es/folsomiacandida

An automatized procedure was developed to process the data we collected in the form of digital images of the network nodes. This was essential given the quantity of data collected. We had twenty nodes per experimental network; with ten identical network replicates and seventy 16.2Mpix (4928x3264) images were taken per node over the course of 130 days. The experiment database contained 14,000 images and occupied 76.87GB in .jpg format. As way of comparison, we estimate that a person could open an image, manually count the number of

collembola, measure their length and size (area), record the results, and close the file in 10-15 minutes on average using a standard tool like Adobe Photoshop®. We did some manual measurements on a set of test samples that were used to tune the image processing algorithm and its parameters. On average, the images contain some 120 collembola individuals, and it took some 5 to 10 seconds per collembolan to measure length, area and radius. Therefore we estimated that, the time required to manually process the entire database would amount to some 3500 hours. This task would not be and would compromise the repeatability of the experiment. To avoid this error-prone effort we automated the process of measuring the population on each node.

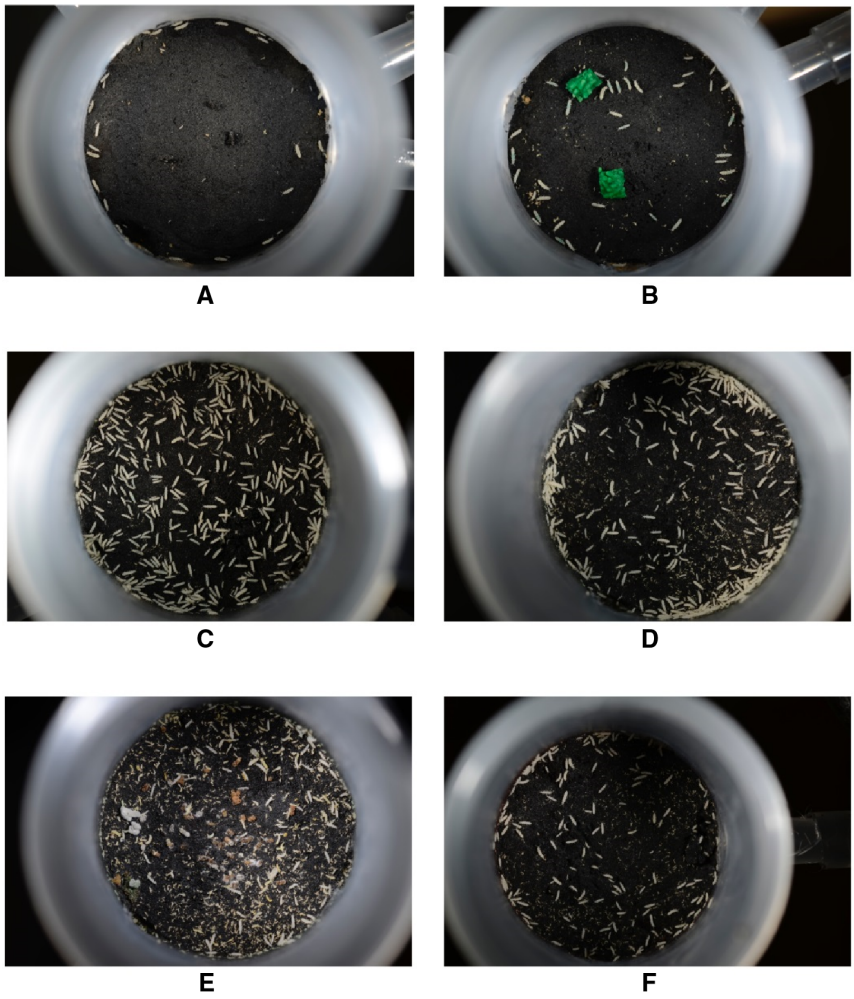

Figure S3: Typical examples found in the database: **(A)** Image at the beginning of the experiment **(B)** Image at the beginning of the experiment showing colored food (yeast grains) **(C)** Image at day 35 showing a large population **(D)** Typical concentration of population at the borders of a node **(E)** Typical image at the end of the experiment containing a mixture of living and dead collembola and lot of debris **(F)** Image which shows the node positioned off-center and with a smaller radius.

A detailed examination of the image database reveals different aspects of the image acquisition process that had to be taken into account in the design of the image processing

algorithm (Fig. S3). All images in the experiment correspond to a view from above of one node and were taken with a 16.2Mpix Nikon D7000 camera. Exposure was manually controlled, and focal length was set to 40mm. The position of the center of the node did not necessarily match the center of the image. The orientation of the node was random and the image did not contain an orientation marker. Due to the picture-taking process, node positioning, and the radius of the nodes in the focal plane varied from image to image (for instance, compare figure S3f and figure S3d, where differences are approximately 15%). The range of sizes of individuals was not consistent among images. The true radius of the node was known to be 15mm and served as a reference (see below). Finally, illumination was not uniform among images (see Fig. S3), which produced shadows and non-uniformities in the degree of contrast.

Figure S3a shows a typical case at the beginning of the experiment with only a few collembola in the scene. Figure S3b shows a case at the beginning of the experiment where collembola are fed with green-colored yeast –though in other cases the yeast was non-colored, red-colored, or blue-colored. Notice that some of the collembola are also partially colored due to the colored yeast. Figure S3c shows a typical picture after few weeks (day 35 in the picture). No perturbation had yet occurred and the population had grown significantly. At this density, individuals are touching or overlapping. It can also be appreciated that the collembola show a preference for the periphery of the node, making the detection of individuals extremely difficult or, in many cases, impossible due to partial or even total occlusions. This aspect is clearly visible in figure S3d. In figure S3e a typical image at the very end of the experiment is shown. The image contains lots of debris, remains of molts, both colored and uncolored yeast, etc. An important part of the operation of the algorithm is intended to minimize the false positives in these images. Finally, figure S3f shows a typical example where the node is clearly not centered on the image and its apparent size –in terms of pixels-- is smaller than in normal images.

Adequate image recognition analysis required that:

(1) If collembola are to be counted and measured, there is a need to find the circle that defined the position of the node and combine the information about the measured radius (in pixel units) with the known fact that the real radius is 15mm. This allowed us to define a scaling constant to transform pixel measurements into real world measurements.

(2) Some color processing was needed to partially eliminate colored food, debris, and dead collembola from the counting process. However, excessive color filtering might result in discarding living collembola which had colored guts from ingesting colored food.

(3) Clearly, converting the original image to a binary image would not separate collembola from the background as the background is heterogeneous in both texture and color (plenty of small items with the same intensity or color as the collembola). A more sophisticated process to separate the foreground from the background was required.

(4) Finally, once the image was transformed into a black and white image with black marking background and white denoting foreground elements, we needed a strategy to analyze each white block and determine whether it corresponded to a collembola depending on its morphological features. Special treatment is required to identify collembola at the periphery of the nodes.

The next section presents the algorithm starting with a high-level overview, followed by detailed descriptions of specific routines.

The Image-processing Algorithm Overview

The image-processing algorithm involves 6 steps: 1) retrieving images from the database; 2) running the FindNode routine; 3) running the ProcessColor routine; 4) separating foreground and background objects; 5) morphological analysis to identify *F. candida* individuals; and 6) post-processing (Fig. S4.). During the image retrieval, we extract information about date, replicate, and node number. The image then goes through the FindNodeArea routine, which identifies the position of the node within the image and measures its radius. This routine also sets all pixels outside the node area to black, and creates a black and white mask of the ROI (Region Of Interest) in the current image (Section S.1.3.3). Once the node area is detected and the circle radius is known, the ProcessColour routine is enabled. Collembola tend to have no color in general except in when they ingested colored yeast. The ProcessColor routine finds excessively colored areas in the image and turns these pixels to black (Section S.1.3.4.). The resulting image is then processed to separate the foreground from the background. White areas in the final black and white image represent possible *F. candida* individuals, and information about the image background (primarily the plaster of paris substrate) is discarded. Finally, the binary image containing the *F. candida* candidates goes through a morphological filter function. This function examines every blob (isolated set of white pixels) and measures its spatial properties: length, area, solidity, number of points in the skeleton (*29*). For metric measurements, pixel units are transformed into microns or millimeters using the true and image node radii to define the scaling factor. This function also checks the numbers of collembola counted, and, if this number is above a threshold, generates a second count that estimates the number of collembola in a peripheral ring next to the node borders (Section S.1.3.5). All routines have been developed using MATLAB®.

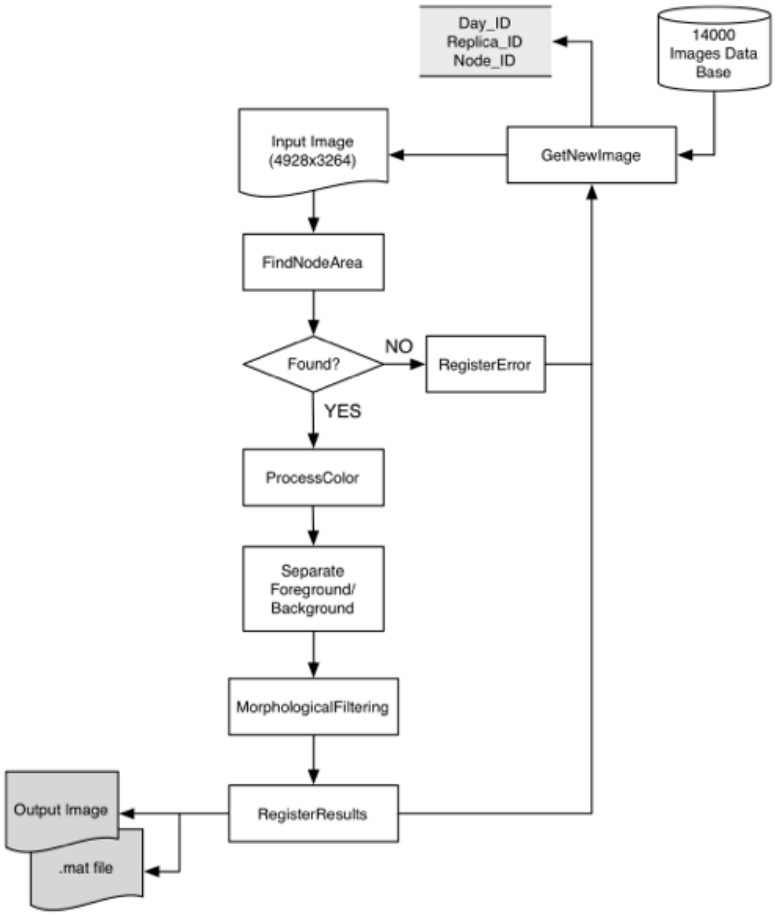

Figure S4: High-Level Schematic of the Image processing Algorithm

FindNode Routine

The FindNode routine is responsible for finding the location, area, and radius of the node in the image (Fig. S5). Node radius will establish the constant pix2mm that defines the scaling factor from pixel measurements to millimeters. Color information is not relevant to the task of finding the position of the node in the scene. Therefore, the image is first transformed to its gray-scale version (RGB2GRAY) and contrast is enhanced via histogram equalization (HistogramEq; Fig. S5).

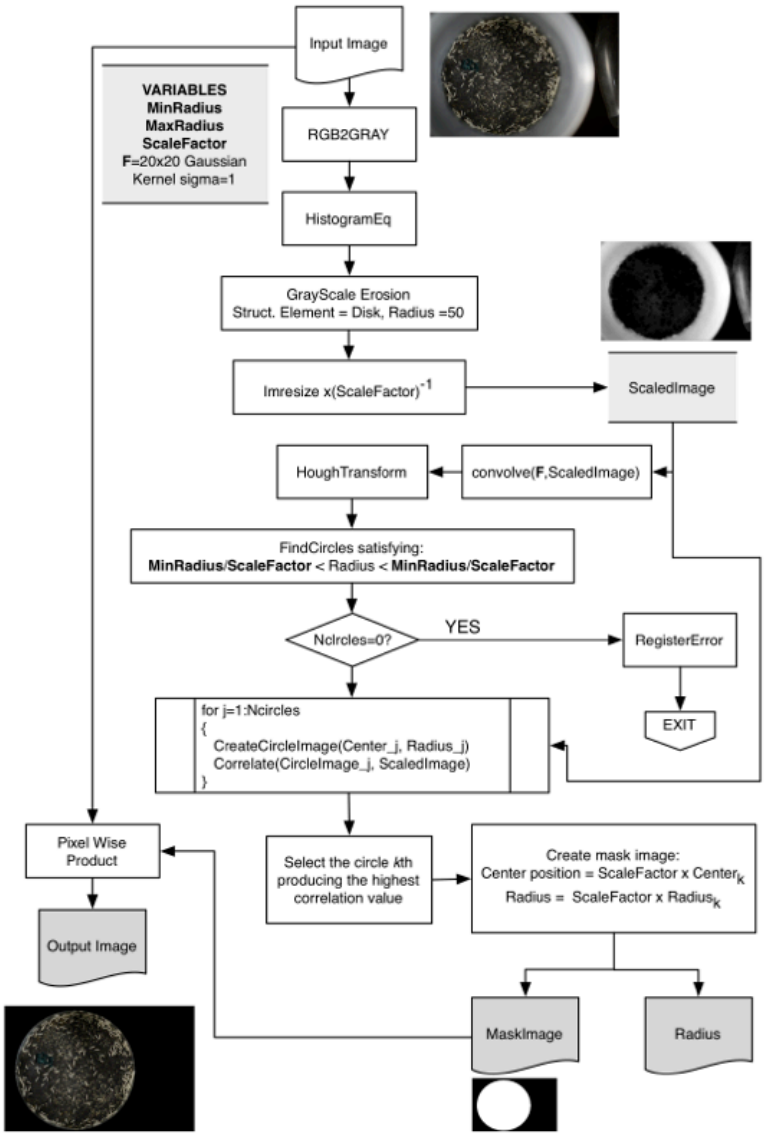

Figure S5: High-Level Scheme of the Image Processing Algorithm

Then, the image undergoes a gray scale erosion (*30*) process using a circular structuring element with a radius of 50px. This operation removes most of the small features in the image, including the *F. candida* individuals in order to correctly detect the circle. Figure S6 shows a visualization of these first steps in the FindNode routine. The circle defining the node area is more easily recognized in Figure S6b as most of the fine grain details in the image have been eroded. At this point, since the node area is a large structure in the image, it is not necessary to keep the 16.2Mpix resolution. Moreover, since we used the Hough transform (*31*), a computationally intensive technique, to find the circle area, we downsample the input image by ScaleFactor=20. Figure S6c shows the accumulator for the resulting Hough transformation (*31*). In order to save execution time, which is very important when we are dealing with 14,000 images, the circle identification is restricted to circles whose radii (in pixels) are within the limits 1300/ScaleFactor≤Radius≤1632/ScaleFactor. However, the node area is not the only large circular structure in the image. The outer part of the node vial is also a circular structure; due to

the out of focus effect, shows also partial circular structures insides. The Hough transform will identify all circles whose radii are within the limits established above. In order to identify the circle corresponding to the node area, we use our knowledge of what the node area should look like from the downsampled image. Thus, for every candidate circle found, we create a black and white image of it (drawing the circle in black and the background in white) and correlate this image with the downsampled image (denoted as ScaledImage in Fig. S5). We store the correlation values for all circles and select the circle which produces the maximum correlation; this indicates that this image is more similar to the original one. By doing so, we have been able to successfully detect 100% of the node's active area through our 14,000 images. Figure S6d shows the detected circle superimposed on the original image (line width in the drawing is set to 5px for visualization purposes).

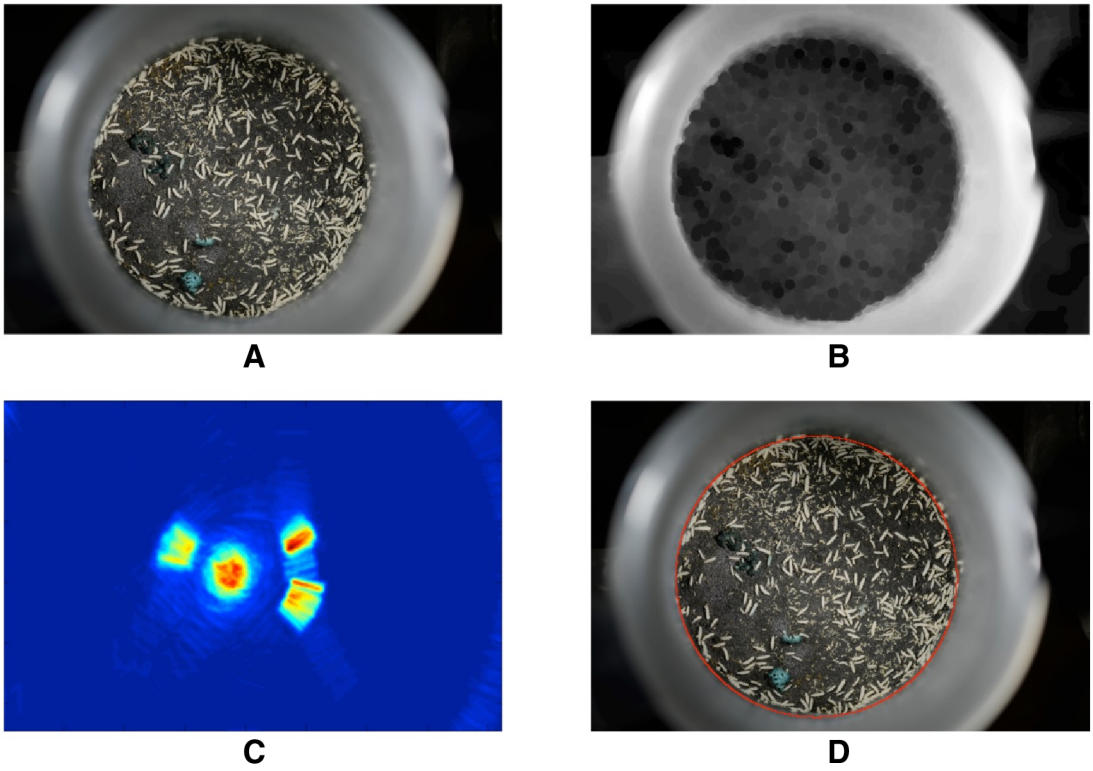

Figure S6: (A) Input image to the FindNode block **(B)** Intermediate result after gray-scale erosion **(C)** Hough Transform result **(D)** Final image with superimposed detected circle.

ProcessColor Routine

Color-processing of the original RGB input image was performed to eliminate colored objects from the analysis (Fig. S7). Generally speaking, the collembola have no predominant color, showing quite balanced R G B components, whereas objects like colored yeast, eggs, debris, or dead individuals tend to show a (sometimes very weak) prevalence of a color in their pixel values---dead insects tend to show a yellowish-grey coloration. Consequently, eliminating image areas that are excessively colored would result in a higher performance of the algorithm mostly in terms of false positives in images taken in the final stages of the experiment (plenty of debris and dead individuals). Moreover, in some special cases, some textures found in the yeast

may resemble the shape and size of collembola and are counted as such by the morphological filter.

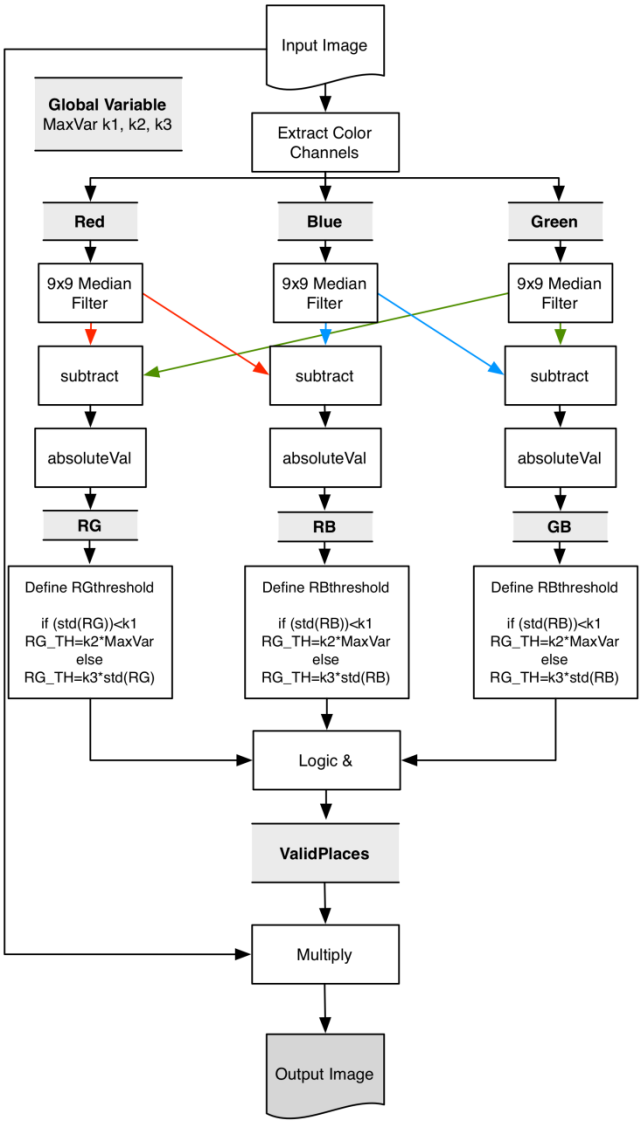

Figure S7: Color-processing Routine.

Figure S8 shows an example of highly texturized blue-colored yeast that would produce false positives because several shapes within the yeast meet the requirements of the morphological filter regarding size, length, aspect ratio, etc.

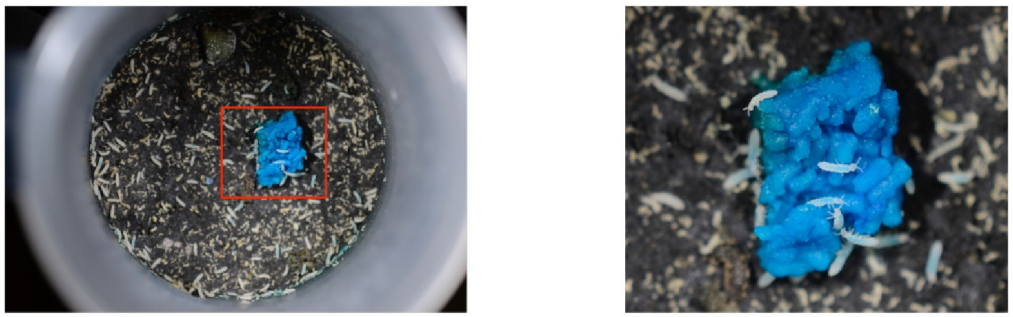

Figure S8: False Positives in highly texturized colored food: **(A)** Input Image **(B)** Zoomed image showing superimposed false positives.

The color processing simply consists of generating differential color channels (after some median filtering to reduce noise) and measuring the disparity among them. The routine creates three new images RG, RB, and GB, containing absolute value information about Red channel minus Green channel, Red channel minus Blue channel, and Green channel minus Blue channel respectively. Figure S9b shows the RB channel for the input image in Figure S9a (others channels are omitted for visualization purposes).

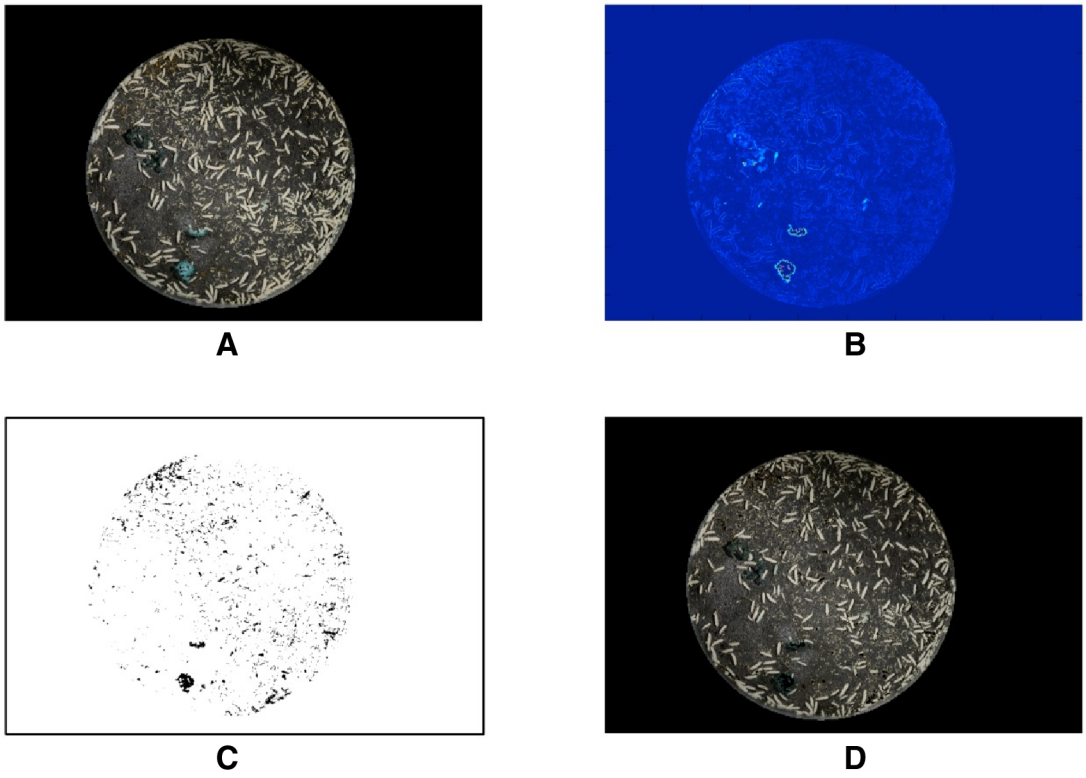

Figure S9: SuppressColor routine: **(A)** Input image **(B)** RB Channel **(C)** Valid Positions (white) **(D)** Final image after color areas suppression.

For these three channels, we compute their standard deviation and define a channel threshold depending on whether it is larger or smaller than a constant k1. If it is smaller than k1,

then this channel threshold is defined as k2*MaxVar, otherwise it is defined as k3. The values for k1, k2, k3, and MaxVar were determined by a heuristic method based on 100 randomly selected images. Final tuning produced the values k1=6, k2=1.3, k3=6.2, and MaxVar=25.

Background-Foreground Separation

At this stage in the image-processing, we still need to separate the background (basically the plaster of paris substrate and debris) from the foreground (the collembola to be counted). However, the images are still too texturized to simply convert the grayscale images to binary images based on a global threshold. Therefore, the main purpose of this background-foreground processing routine will be to create a local varying threshold that allows us to separate collembola from the background in a way such that no individual is marked as a background area, (that is, we accept false positives but penalize false negatives).

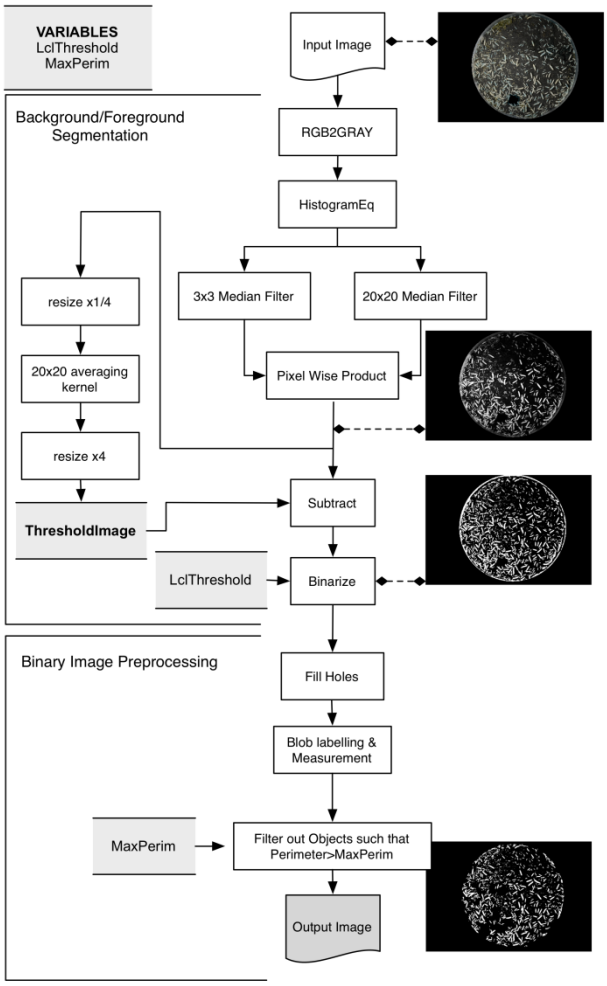

Figure S10: Background / Foreground Separation

The algorithm for background-foreground separation relies on producing two images, which after being subtracted provide clean background suppression (Fig. S10). First, as color information has already been used, we convert the image to gray scale and apply histogram equalization to maximize the global contrast. The resulting image goes, in parallel, through a

3x3 and a 20x20 median filter, and both resulting images are correlated. From this correlation image, a threshold image is produced by means of the application of a 20px radius disk-shaped averaging filter. Finally both images are subtracted and the difference is thresholded to generate a black and white image with white pixels marking foreground information. Figure S11 shows some exemplar images processed with the background-foreground separation algorithm.

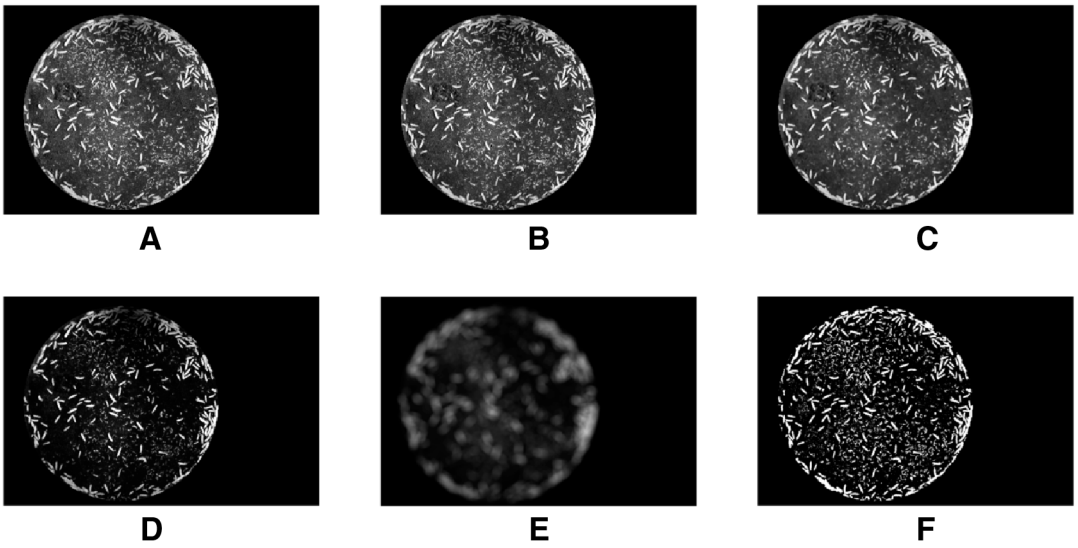

Figure S11: Background-Foreground Separation Images: **(A)** Original Image **(B)** Result after 3x3 Median Filter **(C)** Result after 20x20 Median Filter **(D)** Correlation Result **(E)** Threshold Image **(F)** B&W Output.

Morphological Filter

The morphological filter is the last stage in the image-processing chain. It receives a black and white image from the background-foreground routine that contains only potential *F. candida* individuals in the node area. The morphological filter analyzes the different pixel groups (blobs) in the image and applies different morphological rules to decide whether a particular group of pixels corresponds to a collembolan. Identified collembola are counted and their morphological properties are recorded (area, location, length, among others). Many of the parameters regarding the morphological properties of the individuals were set manually whilst thresholds and other parameters were fine-tuned using simulated-annealing techniques (*32*).

A time series of images for a single node is shown in Figure S12 to illustrate the range and variability of morphological characteristics of the objects in a node. During the first days of the experiment, no reproduction cycle had yet occurred therefore the number of collembola in the image remains small, and the substrate is relatively homogeneous (Fig. S12a). Starting on approximately day 20, a new generation is born and the images become populated with collembola gathering at the periphery of the node (Fig. S12b). By the end of the experiment, images contain a lot of debris from yeast, dead collembola, molts, etc. and have a very low signal to noise ratio (defined as the number of collembola divided by the number of candidate shapes on the order of 1:500; Figure S12f).

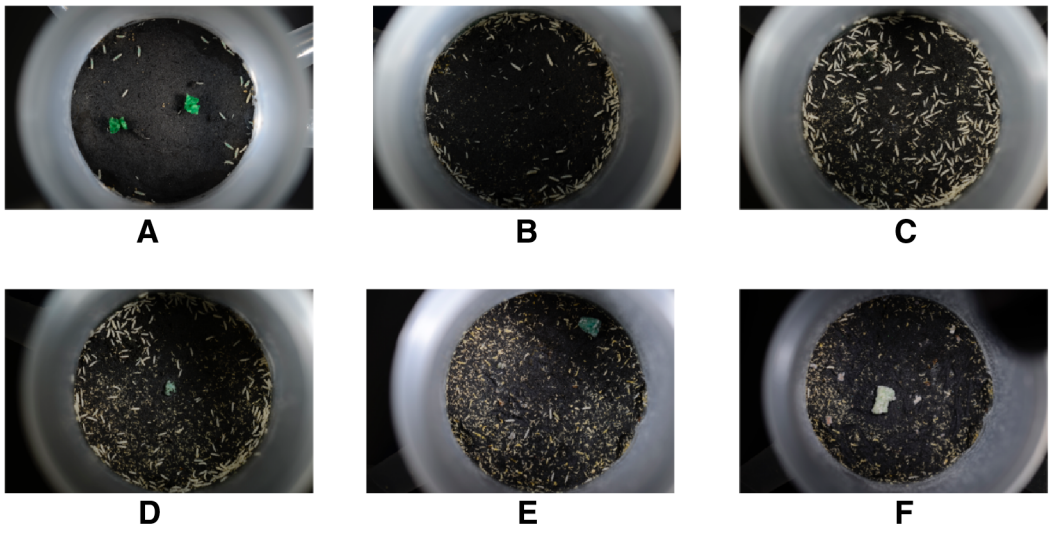

Figure S12: Rep 1 Node 10: **(A)** Day 2 **(B)** Day 24 **(C)** Day 44 **(D)** Day 68 **(E)** Day 95 **(F)** Day 128.

We identified three types of images that correspond to different times during the course of the experiment and required slightly different treatment by the morphological filter. Initially, almost 100% of the objects that are separated from the background were collembola. In the middle of the experiment, population density increases and the size structure of the population changes based on reproductive cycles. Many individuals crowd around the periphery of the node such that their body shapes are indistinguishable and require special treatment (see below). Towards the end of the experiment, the rate of false positives increases due to accumulated debris in the node and requires different filtering parameters (see below).

The Morphological Filter algorithm analyzes the shapes in the black and white input image and, using the information extracted from the circle finder about the relationship between pixel and millimeters, filters out the shapes which do not satisfy maximum or maximum constraints on length, area, aspect ratio (length over width), solidity of the shape, and number of end points in the skeletonized version of the shape (Figure S13). Shape solidity is a scalar magintude which specifies the proportion of the pixels in the convex hull image of the shape that are also part of the shape itself, it is computed as shapeArea/ConvexshapeArea. Minimum length threshold of an individual is not constant throughout the experiment but rather is determined by the biology of the study organism and the current day of the experiment.

Variability in the minimum length threshold is important because it allows us to filter out most of the debris if tuned properly. During the first days (until day 20) minimum length is set to 0.8mm based on data extracted from measurements of the individuals during the first days of the experiment across nodes. After day 20, the next generation is recruited into the population and minimum length is lowered to 0.3mm as this was found to be a good point to start counting juveniles. From days 50 to 70, length is again increased to 0.8mm as juveniles reach adult size. From day 70 onwards, the minimum length is increased to 1.1mm allowing us to filter out a large amount of debris from the final day's images.

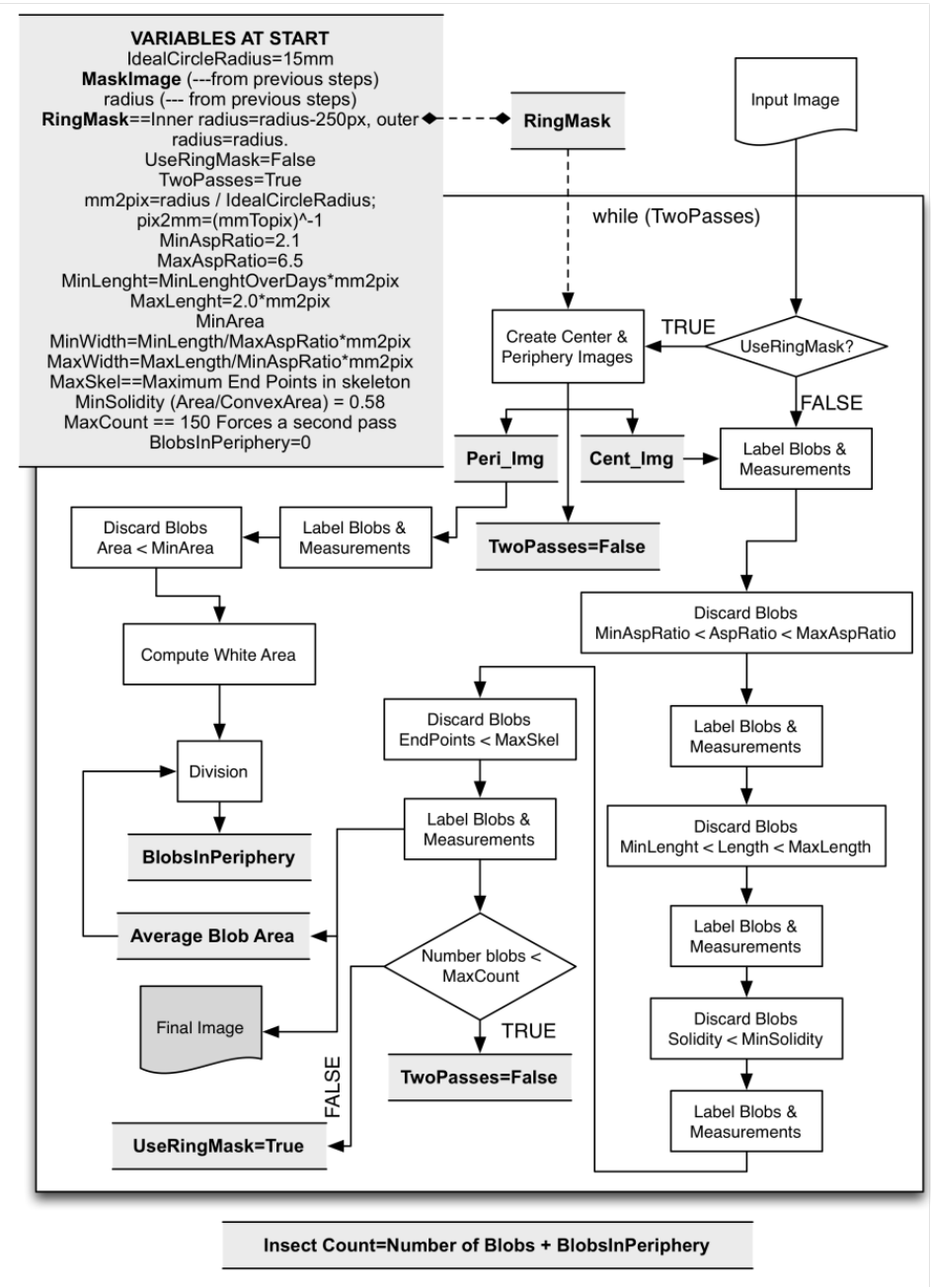

Figure S13: Morphological Filter Algorithm

Another important aspect of the algorithm is that it works on, conditionally, two runs. In the first run, we process the entirety of the node area and count the number of collembola. If the number of collembola counted within an internal area (node radius minus 250px) is above a given threshold (MaxCount) we assume that the population density is high, meaning that individuals at the node periphery will not be accurately counted. The following steps estimate the number and area of individuals at the periphery under high density conditions:

1) We obtain the morphological statistics for collembola in the central part of the image, where overlap between collembola is low.

2) For the collembola in the outer ring (250px wide), we delete the shapes with an area below the minimum area limit of an collembolan; we consider that all remaining shapes correspond to area occupied by real collembola, and we extract the total area that is occupied.

3) We assume that the collembola in the outer ring have the same properties (statistical distribution of sizes) as those in the central part of the image.

4) From the total area computed in step (2) and the statistics in (1), we estimate the number of collembola in the outer ring.

5) Finally, the number of collembola identified in the center of the node (1) and the number of collembola estimated in the periphery of the node (4) are added to provide the count for this image.

Results, Validation, and Post-processing

The image-processing algorithm was trained on two sets of manually counted images. The first training set was randomly chosen among the nodes, replicates, and days, whereas the second training set was chosen to represent particularly difficult images to analyze, mainly emphasizing days when the population density was high after the end of a reproductive cycle or when there was a lot of debris and only a few collembola.

The algorithm was validated against a manually-counted validation set corresponding to all of the images (70) of Node 11 - Replica 9. In general, both manual and automatic counts match reasonably well (Fig. S14a, S14b). Note that the manual counts did not filter individuals depending on their length, which largely accounts for the disparities (that can be considered as delays) in the leading edge of the curves. The automated count excludes collembola below 0.8mm length until day 20 and beyond day 50. However, when counting manually we also counted very collembola. Despite these differences, the mean relative error across all the images of this particular node is 16.6%. Interpolation was used when counts on previous and successive days differed by more than 30%. In these instances, a moving average was used based on six days preceding each count. This post-processing improved the validation results (Fig. S14c, S14d). The match between both curves was better during the perturbation period.

The Matlab source code was executed on the validation set using a 4GB RAM iMAC with a 3.06GHz Intel Core 2 Duo Processor running MATLAB R2010B. Average processing time was 51.4s per image, with the following distribution: color processing (7.4%), circle finder (19.78%), background-foreground detection (43.2%), morphological filter (28.7%), file reading and result saving (0.92%). The execution of the algorithm over the 14000 images in the database was executed on 10 identical iMACs running in parallel. It took nearly 20 hours to analyze all images. This execution time is reasonable for our purposes. New ideas could be tested on the validation data set in less than an hour and the complete data set could be analyzed in less than a day.

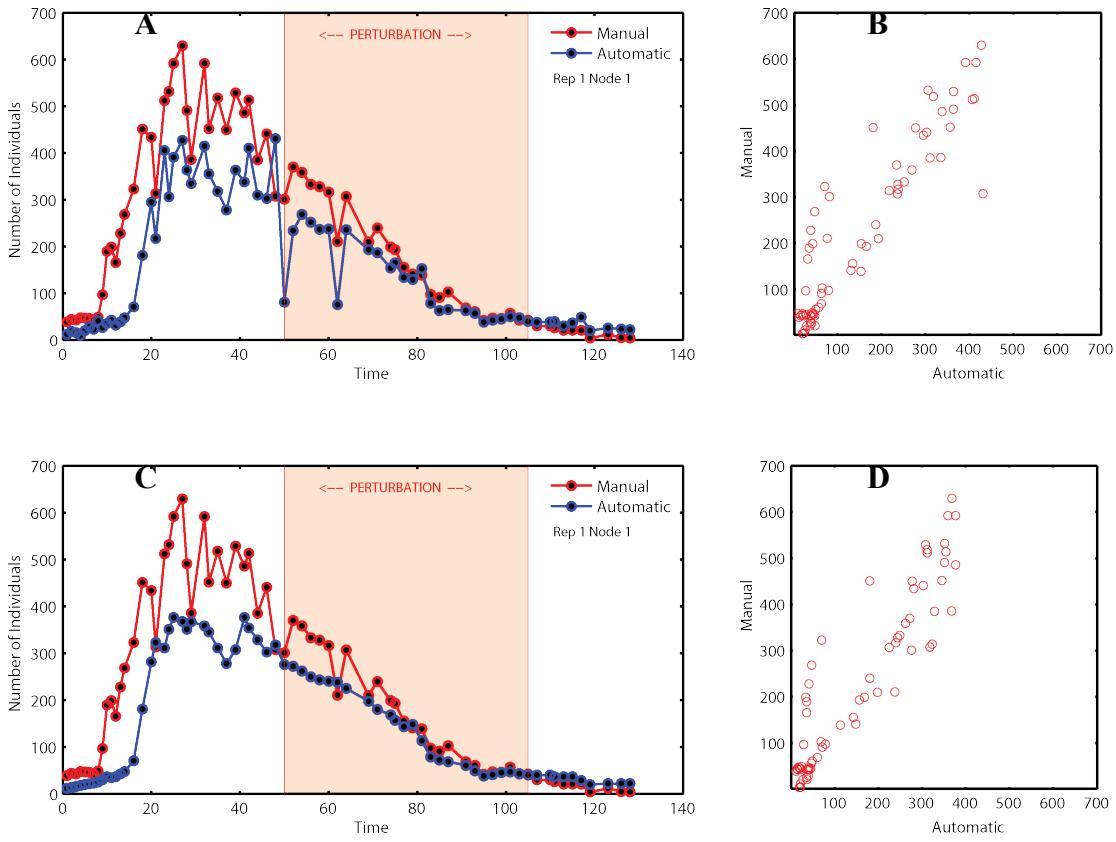

Figure S14: Manual vs Automated Counts from node 1 from the first network replicate: **(A)** Time series **(B)** Scatter plot. Panels **(C)** and **(D)** are the corrected versions after post processing.

Results

Quantifying the buffering effect of modularity

In order to quantify how the perturbation affects the local population size ($I_{v,r}$) of node v in network replicate r, we first calculated the area under the time series delimited by the beginning and the end of the perturbation period as follows:

$$I_{v,r} = \int_{t_0}^{t_e} N_{v,r}(t) \qquad \text{(eq. S1)}$$

$$I_{v',r'} = \int_{t_0}^{t_e} N_{v',r'}(t)$$

where $N_{v,r}(t)$ is the number of individuals over time in the node "v" and replicate "r". t_0 and t_e are the time when the perturbation begins and ends respectively. Finally, $I_{v,r}$ is the total number of individuals integrated over time in a node "v" and replicate "r" for perturbed microcosms. The total number of individuals integrated over the same time period for the microcosms that were not perturbed is notated as $I_{v',r'}$.

Quantifying the area below the curve has the advantage that we do not need to decide on a specific time step at which to compare the number of individuals. We considered the entire perturbation period.

We expect that areas under the time series—for the same node—in the perturbed networks will be smaller than those in the non-perturbed networks. These comparisons only apply to nodes that are equidistant from the origin of the perturbation (Figure S15).

To characterize the effects of the perturbation at the node-level, we compared the perturbation size in nodes that were inside with nodes that were outside the module where the perturbation occurred (Figure S15a). This comparison was performed for each topological distance from the origin of the perturbation (Figure S15b). For example, we compared nodes inside the module that were one link away from the perturbed node to nodes outside the module also 1 link away from the perturbed node. The same comparison was performed for nodes that were two links away from the perturbed node (Figure 15b). We expected to find a larger perturbation in the nodes that were inside the module, which would reveal the buffering effect provided by the modular structure.

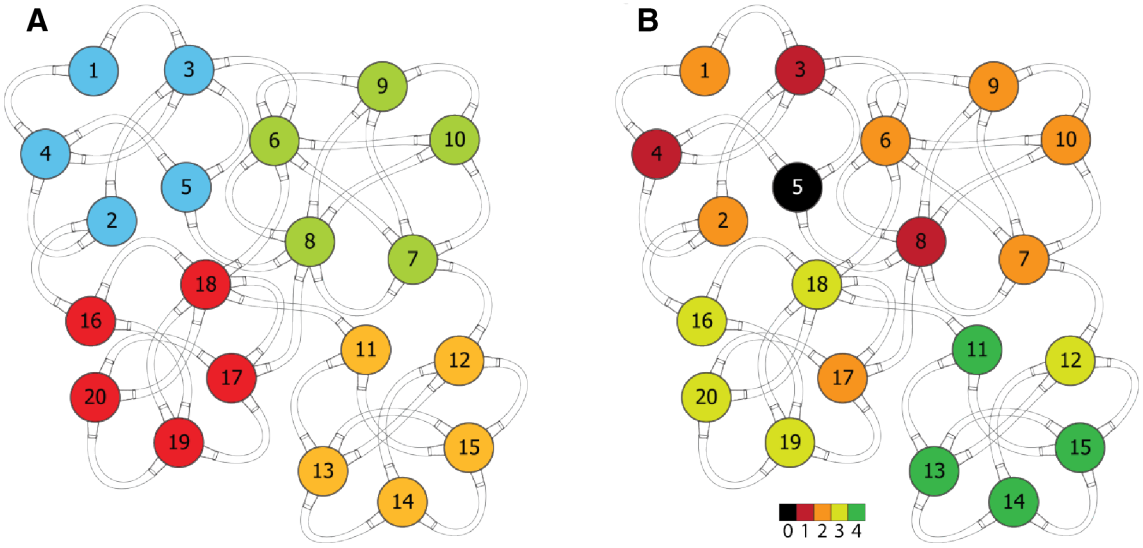

Figure S15: Topological details of the nodes. **(A)** Color code as a function of the module partition. **(B)** Color code as a function of the topological distance of a node to the perturbed

node (node 5). In our experimental networks, link lengths were all equal to control for geographical distance.

We define K_d as the set that contains all nodes inside the module where the perturbation beings, at a topological distance d from the origin of the perturbation. For example in the network depicted in figure S15, $K_2 = \{1,2\}$. In the same way, Q_d is defined as the set that contains all nodes outside the module where the perturbation begins, at a topological distance d from the origin of the perturbation. For example in the network depicted in figure S15, $Q_2 = \{6,7,9,10\}$. We also define the set of perturbed network replicates as R, while the set of unperturbed network replicates is defined as R'.

Next we calculate the magnitude of the perturbation for each combination pair of perturbed and unperturbed network replicates. The set of perturbation values inside the module where the perturbation originates is defined as W_d, while the set of perturbation values outside that module is defined as O_d. Each element of those sets (w for W_d and o for O_d) is defined following equation S2:

$$w = I_{n,r} - I_{n,r'},$$
$$o = I_{m,r} - I_{m,r'}, \qquad \text{(eq. S2)}$$

where nodes $n \in K_d$, $m \in Q_d$, and replicates $r \in R$, $r' \in R'$.

Buffering effect measures how many times there are more individuals outside the module where the perturbation occurs than inside that module. We calculate the average buffering over all possible combinations of the ratio between the perturbation values in the nodes inside the module containing the perturbed node (w) and the nodes outside that module at a distance d from the perturbed node (o). The average buffering ($\overline{B_d}$) at a distance d from the perturbed node is therefore calculated following equation S3:

$$\overline{B_d} = \frac{\sum_{w \in W_d} \sum_{o \in O_d} (w/o)}{|W_d| \cdot |O_d|}, \qquad \text{(eq. S3)}$$

where $|W_d|$ is the number of elements in W_d and $|O_d|$ is the number of elements in O_d.

To test whether there is a significant buffering effect we performed a one-sample t-test on the distribution of buffering values. The null hypothesis was rejected when the buffering effect was larger than one. If, for example, average buffering at a certain topological distance is 1.46 this

means that there are 46% more individuals outside than inside the module where the node in which the perturbation begins resides.

Randomizing module assignment

The community-finding algorithm (*11*) tries to find the module partition that maximizes modularity (*10*). A partition is just a method of assigning nodes into modules. We wanted to know if the partition that maximizes modularity is the same partition that also maximizes perturbation buffering.

We randomized node-module assignment. In particular, we were interested in knowing which nodes were inside the module where the perturbation begins—node 5—and which were outside. We first calculated all combinations wherein each of the 20 nodes in the network could be either inside or outside the module containing the perturbed node. To create fair comparisons, we also filtered combinations to keep only those combinations where there were 5 nodes inside the module with the perturbed node—as in the partition produced by the community-finding algorithm, and keeping the same number of nodes inside and outside the module at every distance as in the empirical case. Then we calculated the buffering effect for each of those combinations.

Data analysis

The result of the image recognition analysis and post-processing is a time series of the number of individuals in each of the nodes (i.e. local population size) and each of the network replicates (i.e. metapopulation size). To analyze these time series we first calculate a moving average across the time series in order to have the same number of data points in all of the time series. Then we normalize the resulting time series by the number of individuals in each node at time zero to remove the effect of slightly different initial population sizes in each of the nodes.

Population dynamics model

To extend our experimental results, we simulated network dynamics using a matrix population model for networks with different levels of modularity, parameterized for *F. candida*. We implemented a metapopulation version of the model described by ref. (*19*) with minor modifications. The model is a spatial, stage-structured population model that tracks four stages of the *F. candida* life cycle: eggs, juveniles, and two adult phases (Figure S16).

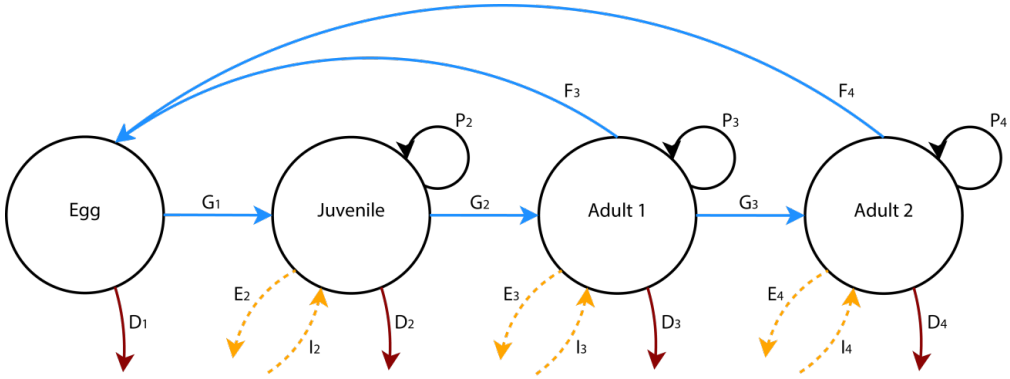

Figure S16: Weekly life-cycle of Folsomia candida. E = emigration to another patch, I = immigration from another patch, D = death, G =growth to the next stage, P = remains in the same stage, F = fecundity.

We made some biologically reasonable assumptions. Dispersal operates at a faster time scale than demography, so when an individual moved from one patch to another it remained in the same developmental stage. The number of individuals that moved from patch *i* to patch *j* was a function of the number of individuals in patch *i*. The fraction of individuals that moved between patches increased linearly in every stage because larger individuals are more mobile. The duration of each developmental stage was parameterized as follows: eggs hatch after one week, the juvenile stage lasts 3 weeks, the first adult stage lasts 5 weeks, and the second and last adult stage lasts 48 days. This makes a life span of 111 days at 25 ºC (*19*).

Density dependent fecundity (F) and survival (S) were parameterized according to ref. (*19*):

$$F(N) = a\ln(N) + b\ln(N)^2 + c \qquad \text{(eq. S4)}$$

$$S(N) = d - e\ln(N) \qquad \text{(eq. S5)}$$

Where a=18.53, b=-1.74, c=-44.04, d=1.35, e=0.14. The lower-limit of fecundity was set to zero. The upper-limit of survival was set to one. We ran the model 400 time steps where a time step represents one day. We perturbed half of the replicates in the simulations, as we did experimentally. The perturbation occurred at time step 54 and removed all individuals in patch 5. No replication was required because the model is deterministic.

At $t=0$ we assumed an initial number of eggs, juveniles, and adults of the two adult classes. As a function of those initial values, the system could exhibit quite different behaviors. For example, if we started with 60 eggs, 10 juveniles, 10 and 22 adults, we obtained the behavior shown in figure S17.

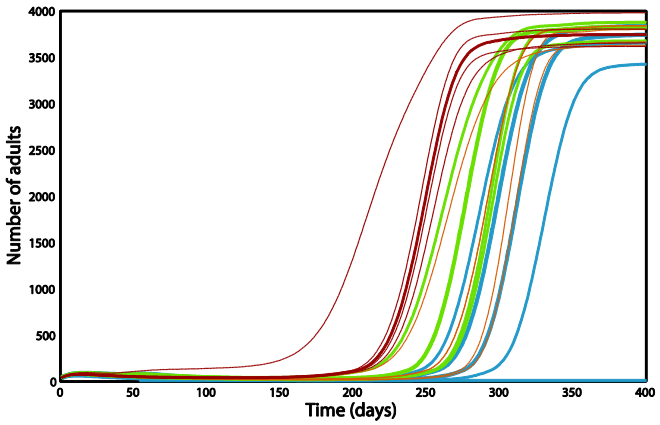

Figure S17: population dynamics for the initial values of 60 eggs, 10 juveniles, 10 and 22 adults. Every line represents a patch in the habitat network. Colors correspond to the module id of the patch.

However, if we started with just one adult less, i.e, 60 eggs, 10 juveniles, 10 and 21 adults, we obtained the behavior shown in figure S18.

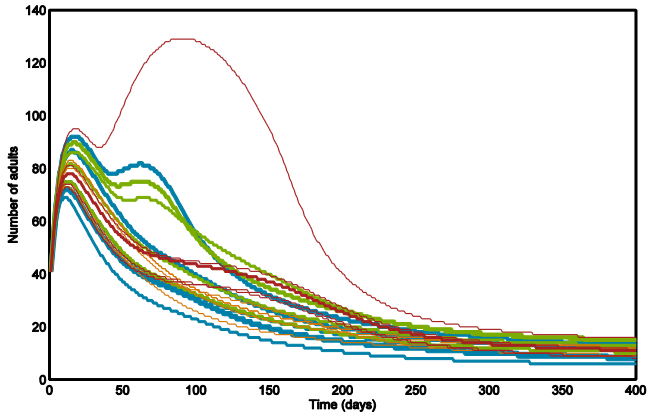

Figure S18: population dynamics for the initial values of 60 eggs, 10 juveniles, 10 and 21 adults. Every line represents a node. Color code deepens on the module id to which the node belongs.

The dynamics in Fig. S18 resemble qualitatively what we observed during the experiment. We took these dynamics as a benchmark to further explore the effect of topology on population stability.

To test whether our results are robust to the choice of the target patch we use a metapopulation model of *F. candida* as described in the Section S2.1. Using the exact same network configuration as in the experiments, we show that we obtain the same results independently from the node where the perturbation occurs (Figure S19).

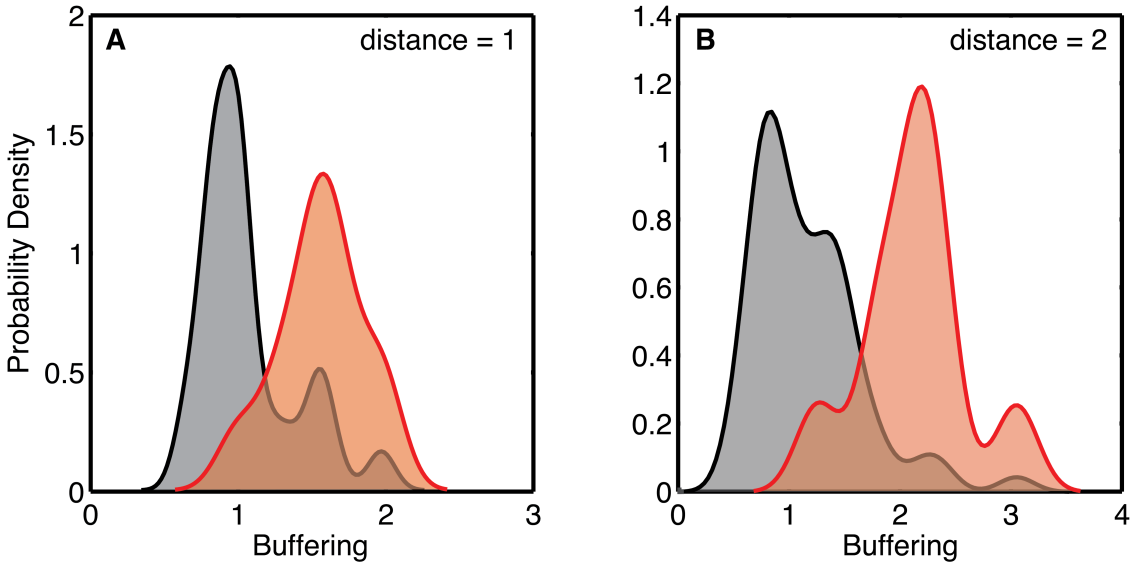

Figure S19: Modular networks buffer the spread of perturbations regardless of the selected target patch. This figure resembles Figure 3 in the main text, but in this case we used time series obtained from the theoretical model. We systematically iterated the target patch among all patches. In both panels, the black area is the probability density distribution of the buffering effect when the node-module assignment is randomized. The red area corresponds to the probability density distribution of buffering when the node-module assignment is the one that maximizes modularity (this is the empirical partition returned by the community-finding algorithm). For each target patch, the buffering calculated from the node-module partition that maximizes modularity is always larger than the randomized version. There is some overlap between distributions. This is explained because in some cases the randomized version for a certain target patch can be higher than the empirical from another target patch. Panels **(A)** and **(B)** correspond to distances from the perturbed node of one and two links respectively.

A concern that the readers may have is that the reported p-value for buffering effect at a distance of 1 link is non-significant. There is no typo in the p-value. There is a buffering effect of modularity both at the distance of one and two links, but this effect is only statistically significant at a distance of two links. As we now explain, this is expected.

Since buffering at a distance of one is closer to being 1 (a value of 1 would mean that there is no buffering) there are two reasons for which it is more difficult to obtain a statistically significant value at a distance of one link apart. First because in the experimental setting there are less

nodes at a distance 1—only two nodes outside and one node inside at a distance one as can be seen in figure S15—and therefore there is less statistical power. Second because this is an experiment and there is random noise and observation error. Assuming that this error/noise is the same for all nodes independently of their distance to the origin of the perturbation then we expect a lower likelihood of obtaining a significant value for buffering at a distance of 1.

This reasoning about the noise observed in the experiment (stochastic noise + measurement error) motivated the following simulations:

In the theoretical results presented in the manuscript, the population dynamics model is deterministic. There is no added noise. If our reasoning behind the fact that buffering at a distance 1 is less likely to be significant because of the noise, then at a certain level of noise added to the model the p-values of buffering at a distance 1 and at a distance 2 would be different. Indeed, figure S20 shows that for the orange-shaded area, the p-value of the one-sample t-test goes above the significant level of 0.05 for buffering at a distance 1 while for buffering at a distance 2 it remains significant. The range of noise delimited by the shaded area is compatible with the experimental findings. Actually, the validation of the image recognition analysis protocol shows a mean relative error across all the images of 16.6%. This value is roughly in the middle of the noise range delimited by the shaded orange area on the plot.

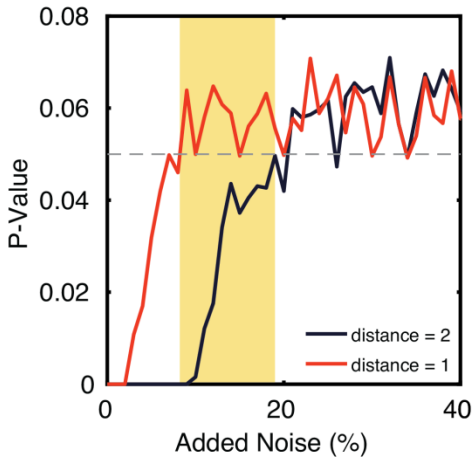

Figure S20 Average change in p-value of the one-sample t-test to tell whether buffering effect is significant as a function of the percentage of noise added to the time series obtained by the model. The red line represents the average p-value for the buffering effect at a distance of 1 over a 1000 replicate simulations, while the blue area represents the buffering effect at a distance of 2. The dashed gray line is the 0.05 significance level. The orange area is the range of noise in which the buffering effect at a topological distance of 1 to the origin of the perturbation is no significant while the buffering effect at a topological distance of 2 is significant. This is in agreement with what was found in the experiments.

To obtain this figure we first run the model as described in section S2 of the Supplementary Materials. This model returns the number of individuals in each node of the network over time $N_n(t)$. Then, for each level of noise we create a new time series where the new number of individuals for each point in time is a random value chosen within the interval defined by $(N_{n,t} - \varepsilon N_{n,t}, \ldots, N_{n,t} + \varepsilon N_{n,t})$, where ε is the level of noise and $N_{n,t}$ is the number of individuals in node n at time t. For each level of noise we created five perturbed and five unperturbed replicates, perturbing node 5 in the perturbed replicates. This exactly mimics the experimental design. The blue and red lines in figure S20 represent the average p-values over a thousand replicates.

A thousand node networks

One of the limitations of our experimental design is the relatively small size of the networks. A direct implication of using small networks is the limited range of distances to the origin of the perturbation that we can explore. To overcome this limitation, we create networks with the same degree distribution as the experimental microcosms but with a thousand nodes and varying modularity levels (Fig. S21). This allows us to better explore the relationship between the buffering effect of modularity and the topological distance to the perturbed node and how the level of modularity affects this relationship.

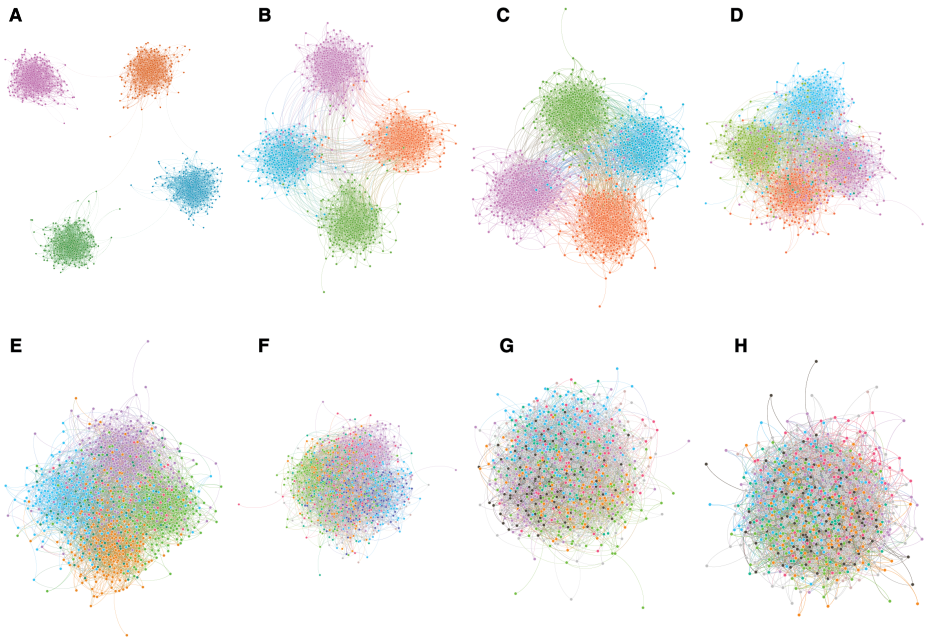

Figure S21: Eight different networks of decreasing modularity values with 1000 nodes, 4000 undirected links, and Erdös-Rényi random degree distribution. Different colors represent the different modules found by the community-detection algorithm (*11*). Network A is extremely modular, while network H is completely random.

For the networks in Fig. S21 we calculated the buffering effect (eq. S2) at every possible topological distance to the perturbed node (Fig. S22). The simulations also gave us the opportunity to repeat the calculations iterating over all the nodes in the networks that were sequentially selected as the target node.

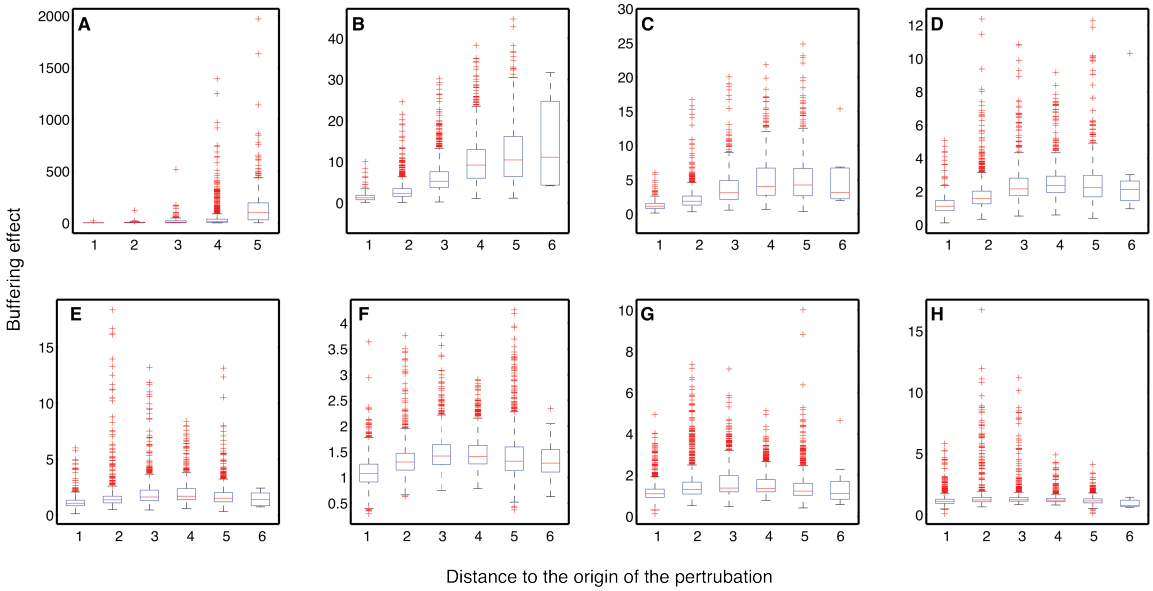

Figure S22: Buffering effect as a function of the distance to the perturbed node. Panels A-H correspond to the networks A-H in Fig. S21.

As Fig. S22 shows, the buffering effect increases with distance to the perturbed node. This is consistent with our experimental findings. For extreme modularity values (a), buffering increases as distance increases. However, as modularity decreases the buffering effect saturates with distance. The lower the value of modularity, the shortest the distance at which the buffering effect saturates.

In figure S23 we show this relationship for every one of the topological distances to the perturbed nodes--from 1 to 6. These simulations can also be used to show how the buffering effect changes as a function of modularity.

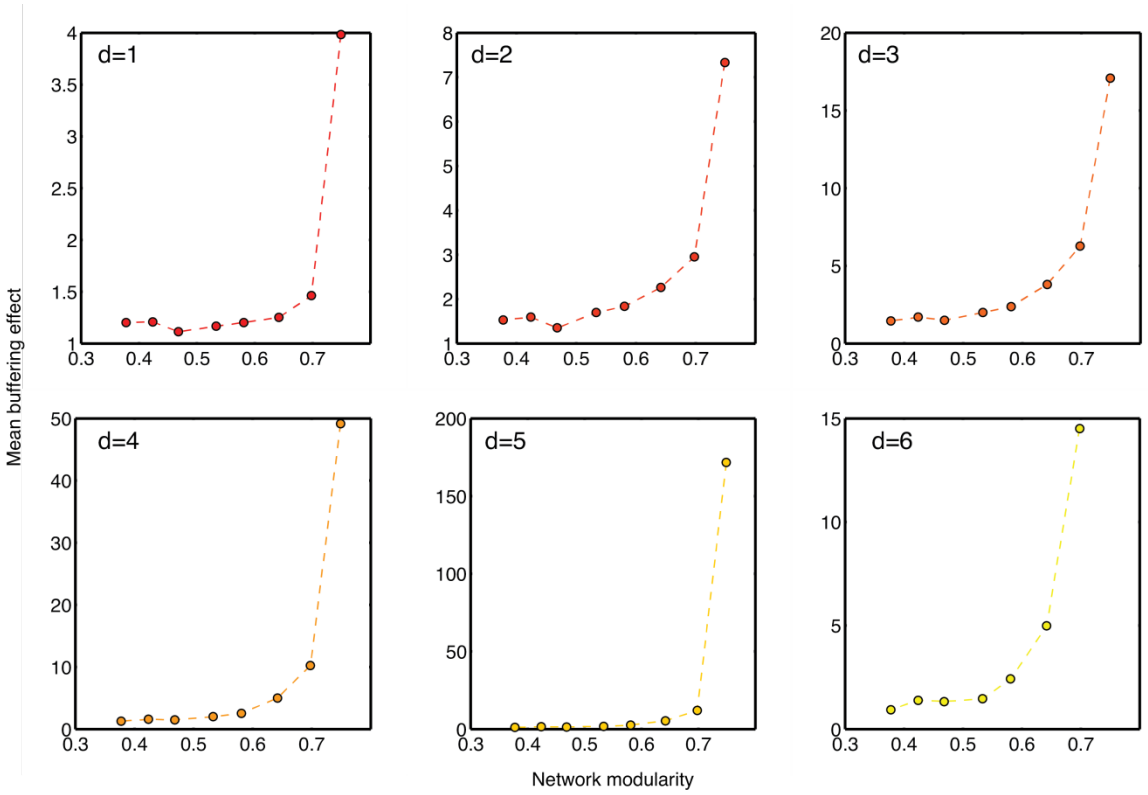

Figure S23: Mean buffering effect as a function of network modularity. Each panel shows the relationship for the nodes at a certain distance from the perturbed node (distance = 1, distance = 2 … distance = 6). There were no nodes at a distance of 7 or more. The mean buffering effect is calculated from the buffering effects that are found when iterating the target node--the node where the perturbation originates--across all the nodes in the network.

For every topological distance to the origin of the perturbation Fig. S23 shows that there is an exponential relationship between network modularity and the average buffering effect.

Scale-free networks

Throughout the manuscript we have used Erdös-Rényi random networks to be consistent with the approach by Robert May when he first hypothesized about the potential effects of compartmentalization for network stability. Most real-world networks, however, show heterogeneous degree distributions. Following a reviewer's suggestion, we assessed the robustness of our results with more heterogeneous scale-free networks. We began by using the Barabasi-Albert model to generate 4 scale-free networks. Then, we swapped interactions to link those networks so that they each become a module of a larger network. This created a modular network with heterogeneous degree distribution. The amount of swapped links allowed us to control the level of modularity of the resulting network. This is the same approach we used to create the Erdös-Rènyi networks used in both the experiment and Fig. S21. Similar results could have been obtained with the algorithm by (*32*) that provides a realistic benchmark for

modularity that not only accounts for degree heterogeneity, but also for variable module size. Here, however, we decided to maintain similar module size to make this new set of results comparable with both the experimental results and the simulations using Erdös-Rényi random networks.

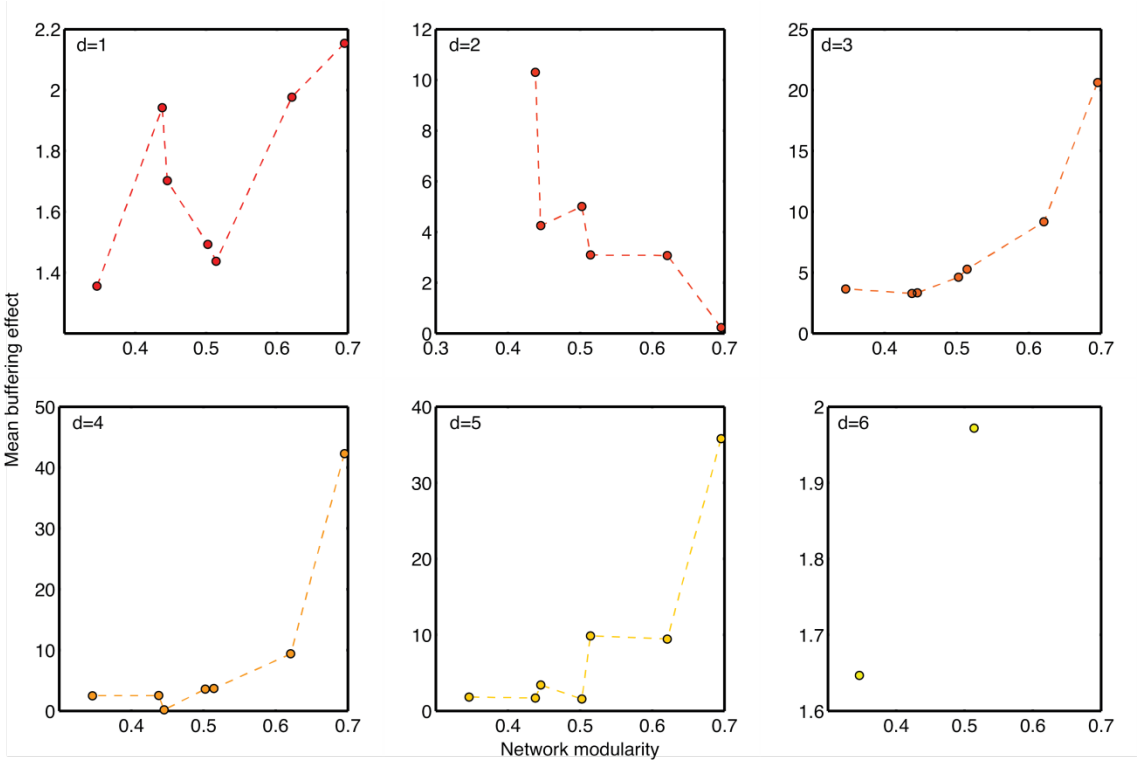

Figure S24 from Supplementary Materials, but using a scale-free network instead. Each panel shows buffering effect as a function of network modularity disaggregated by the distance at which nodes are from the origin of the perturbation (distance = 1, distance = 2 … distance = 6). There were no nodes at a distance of 7 or more. The mean buffering effect is calculated from the buffering effects that are found when iterating the target node—the node where the perturbation originates—across all the nodes in the network.

In Fig. S24 we see that the pattern is similar to that observed in Fig. S23 for distances 3 to 6, while for shorter distances we find no apparent trend. The R^2 values of the correlation between buffering effect and network modularity (using Scale-Free networks) are: R^2= 0.01 for distance = 1, R^2= -0.05 for distance =2, R^2 = 0.42 for distance = 3, R^2= 0.60 for distance = 4, R^2= 0.41 for distance = 5, R^2= 0.06 for distance = 6). Keep in mind that the average and the maximum distance in Scale-Free networks is smaller than in Erdos-Renyi networks, and therefore at a distance 6 there are very few data.

That very same shorter average distance in Scale-Free networks is what prevents us from observing a clearer relationship between buffering and modularity at shorter distances from the origin of the perturbation. This was expected. Nevertheless, as we move away from the origin of the perturbation, the differences between inside and outside the module are very clear. This is shown in the robustness of the pattern at larger distances.

Modularity gradient and buffering effect

All experimental networks had identical structure with very high modularity. We demonstrated experimentally that highly modular network structure can buffer perturbations. However, we did not determine the minimum level of modularity required to observe this buffering effect. Furthermore, a modular structure may also have negative effects by limiting dispersal between modules which could decrease population size.

We used simulations to quantify the positive and negative effects of modularity, and consequently explore if there is an optimal value of modularity. To this end we generated a set of 426 networks. Each of them had a different value of modularity ranging from 0.2841 to 0.5691 (The network structure used in the experiments had a modularity of 0.5587). Modular structures were generated by brute force randomization of network (rewiring) preserving the total number of nodes, the total number of links, and the degree of every node (*11*).

We focused initially on the positive buffering-effect of modularity. We ran the stage-structured population model in each network across the modularity gradient, with and without perturbation. We calculated the buffering effect controlling for the topological distance to the perturbed node. Our results show that the more modular the network, the larger the buffering effect (Fig. S25). This holds independently of the distance over which the perturbation is measured.

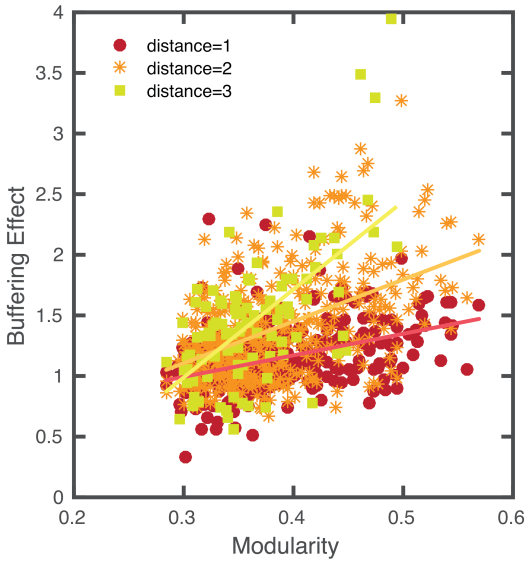

Figure S25: Buffering effect as a function of network modularity. The buffering effect in nodes 1, 2, or 3 links away from the origin of the perturbation is shown in red circles, orange stars, and green squares respectively. The Spearman correlation coefficients for distance = 1, 2 and 3 are 0.41, 0.43, and 0.52 respectively. All p-values are < 0.001. The red, orange, and lime lines are visual guides to the correlations for topological distances of 1, 2, and 3 links.

We note a minor methodological issue that may help to explain the variability we observe in Fig. S25. Through the rewiring process the distance of the nodes to the perturbed node might be different from what was observed in the empirical network. Therefore, the identity of the nodes inside or outside the module at a certain distance might be different from those observed in the original network. If we want to control by module partition and by distance to the perturbed node, it is impossible to control also by the identity of the node, and therefore by degree.

Additional Data table S1 (separate file)

Time series of population counts for each replicate network used in the analyses reported in the main text. The time series for each network are in separate worksheets. The perturbed node (node 5) is indicated in red in column 1, and the period of perturbation is indicated in orange in the top row.

References and Notes

1. D. Helbing, Globally networked risks and how to respond. *Nature* **497**, 51–59 (2013). doi:10.1038/nature12047 Medline

2. P. Holme, B. J. Kim, C. N. Yoon, S. K. Han, Attack vulnerability of complex networks. *Phys. Rev. E* **65**, 056109 (2002). doi:10.1103/PhysRevE.65.056109 Medline

3. J. M. Montoya, S. L. Pimm, R. V. Solé, Ecological networks and their fragility. *Nature* **442**, 259–264 (2006). doi:10.1038/nature04927 Medline

4. V. Colizza, A. Barrat, M. Barthelemy, A.-J. Valleron, A. Vespignani, Modeling the worldwide spread of pandemic influenza: Baseline case and containment interventions. *PLOS Med.* **4**, e13 (2007). doi:10.1371/journal.pmed.0040013 Medline

5. N. Bharti, A. Djibo, M. J. Ferrari, R. F. Grais, A. J. Tatem, C. A. McCabe, O. N. Bjornstad, B. T. Grenfell, Measles hotspots and epidemiological connectivity. *Epidemiol. Infect.* **138**, 1308–1316 (2010). doi:10.1017/S0950268809991385 Medline

6. M. R. Gardner, W. R. Ashby, Connectance of large dynamic (cybernetic) systems: Critical values for stability. *Nature* **228**, 784 (1970). doi:10.1038/228784a0 Medline

7. R. M. May, Will a large complex system be stable? *Nature* **238**, 413–414 (1972). doi:10.1038/238413a0 Medline

8. S. L. Pimm, J. H. Lawton, Are food webs divided into compartments? *J. Anim. Ecol.* **49**, 879–898 (1980). doi:10.2307/4233

9. D. Raffaelli, S. J. Hall, Compartments and predation in an estuarine food web. *J. Anim. Ecol.* **61**, 551–560 (1992). doi:10.2307/5610

10. M. E. Newman, M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004). doi:10.1103/PhysRevE.69.026113 Medline

11. R. Guimerà, L. A. N. Amaral, Functional cartography of complex metabolic networks. *Nature* **433**, 895–900 (2005). doi:10.1038/nature03288 Medline

12. M. A. Fortuna, J. A. Bonachela, S. A. Levin, Evolution of a modular software network. *Proc. Natl. Acad. Sci. U.S.A.* **108**, 19985–19989 (2011). doi:10.1073/pnas.1115960108 Medline

13. R. J. Fletcher Jr., A. Revell, B. E. Reichert, W. M. Kitchens, J. D. Dixon, J. D. Austin, Network modularity reveals critical scales for connectivity in ecology and evolution. *Nat. Commun.* **4**, 2572 (2013). doi:10.1038/ncomms3572 Medline

14. E. M. Albert, M. A. Fortuna, J. A. Godoy, J. Bascompte, Assessing the robustness of networks of spatial genetic variation. *Ecol. Lett.* **16** (suppl. 1), 86–93 (2013). doi:10.1111/ele.12061 Medline

15. R. Guimerà, D. B. Stouffer, M. Sales-Pardo, E. A. Leicht, M. E. J. Newman, L. A. N. Amaral, Origin of compartmentalization in food webs. *Ecology* **91**, 2941–2951 (2010). doi:10.1890/09-1175.1 Medline

16. M. A. Fortuna, A. G. Popa-Lisseanu, C. Ibáñez, J. Bascompte, The roosting spatial network of a bird-predator bat. *Ecology* **90**, 934–944 (2009). doi:10.1890/08-0174.1 Medline

17. D. B. Stouffer, J. Bascompte, Compartmentalization increases food-web persistence. *Proc. Natl. Acad. Sci. U.S.A.* **108**, 3648–3652 (2011). doi:10.1073/pnas.1014353108 Medline

18. J. Grilli, T. Rogers, S. Allesina, Modularity and stability in ecological communities. *Nat. Commun.* **7**, 12031 (2016). doi:10.1038/ncomms12031 Medline

19. M. T. Fountain, S. P. Hopkin, *Folsomia candida* (Collembola): A "standard" soil arthropod. *Annu. Rev. Entomol.* **50**, 201–222 (2005). doi:10.1146/annurev.ento.50.071803.130331 Medline

20. J. R. Beddington, Age distribution and the stability of simple discrete time population models. *J. Theor. Biol.* **47**, 65–74 (1974). doi:10.1016/0022-5193(74)90099-X Medline

21. Materials and methods are available as supplementary materials.

22. R. Lande, Risks of population extinction from demographic and environmental stochasticity and random catastrophes. *Am. Nat.* **142**, 911–927 (1993). doi:10.1086/285580

23. I. Hanski, *Metapopulation Ecology* (Oxford Univ. Press, 1999).

24. S. Maslov, K. Sneppen, Specificity and stability in topology of protein networks. *Science* **296**, 910–913 (2002). doi:10.1126/science.1065103 Medline

25. N. M. Haddad, L. A. Brudvig, J. Clobert, K. F. Davies, A. Gonzalez, R. D. Holt, T. E. Lovejoy, J. O. Sexton, M. P. Austin, C. D. Collins, W. M. Cook, E. I. Damschen, R. M. Ewers, B. L. Foster, C. N. Jenkins, A. J. King, W. F. Laurance, D. J. Levey, C. R. Margules, B. A. Melbourne, A. O. Nicholls, J. L. Orrock, D.-X. Song, J. R. Townshend, Habitat fragmentation and its lasting impact on Earth's ecosystems. *Sci. Adv.* **1**, e1500052 (2015). doi:10.1126/sciadv.1500052 Medline

26. C. C. Vos, P. Berry, P. Opdam, H. Baveco, B. Nijhof, J. O'Hanley, C. Bell, H. Kuipers, Adapting landscapes to climate change: Examples of climate-proof ecosystem networks and priority adaptation zones. *J. Appl. Ecol.* **45**, 1722–1731 (2008). doi:10.1111/j.1365-2664.2008.01569.x

27. C. H. Albert, B. Rayfield, M. Dumitru, A. Gonzalez, Applying network theory to prioritize multi-species habitat networks that are robust to climate and land-use change. *Conserv. Biol.* 10.1111/cobi.12943 (2017). doi:10.1111/cobi.12943 Medline

28. L. R. Gerber, L. W. Botsford, A. Hastings, H. P. Possingham, S. D. Gaines, S. R. Palumbi, S. Andelman, Population models for marine reserve design: A retrospective and prospective synthesis. *Ecol. Appl.* **13**, 47–64 (2003). doi:10.1890/1051-0761(2003)013[0047:PMFMRD]2.0.CO;2

29. M. J. Ferrari, S. E. Perkins, L. W. Pomeroy, O. N. Bjørnstad, Pathogens, social networks, and the paradox of transmission scaling. *Interdiscip. Perspect. Infect. Dis.* **2011**, 267049 (2011). doi:10.1155/2011/267049 Medline

30. R. M. May, S. A. Levin, G. Sugihara, Complex systems: Ecology for bankers. *Nature* **451**, 893–895 (2008). doi:10.1038/451893a Medline

31. A. G. Haldane, R. M. May, Systemic risk in banking ecosystems. *Nature* **469**, 351–355 (2011). doi:10.1038/nature09659 Medline

32. A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms. *Phys. Rev. E* **78**, 046110 (2008). doi:10.1103/PhysRevE.78.046110 Medline