Component response rate variation drives stability in large complex systems

Supporting information

Brad Duthie

Contents

- Code and simulations underlying Fig. 1
- Code and simulations underlying Fig. 2
- Stability across increasing S
- Stability of ecological networks
  - Competitor networks
  - Mutualist networks
  - Predator-prey networks
- Feasilbility of complex systems
- Genetic algorithm

Code and simulations underlying Fig. 1

The sample M used for the eigenvalue distributions in Fig. 1 of the text is available on GitHub, and was produced with by running the following function.

```
find_bgamma <- function(S = 200, C = 0.05, Osd = 0.4, iters = 10000){
    while(iters > 0){
        A_dat  <- rnorm(n = S * S, mean = 0, sd = Osd);
        A_mat  <- matrix(data = A_dat, nrow = S);
        C_dat  <- rbinom(n = S * S, size = 1, prob = C);
        C_mat  <- matrix(data = C_dat, nrow = S, ncol = S);
        A_mat  <- A_mat * C_mat;
        gammas <- c(rep(1.95, S/2), rep(0.05, S/2))
        mu_gam <- mean(gammas);
        diag(A_mat) <- -1;
        A1     <- gammas * A_mat;
        A0     <- mu_gam * A_mat;
        A0_e   <- eigen(A0)$values;
        A0_r   <- Re(A0_e);
        A0_i   <- Im(A0_e);
        A1_e   <- eigen(A1)$values;
        A1_r   <- Re(A1_e);
        A1_i   <- Im(A1_e);
        if(max(A0_r) >= 0 & max(A1_r) < 0){
            return(list(A0 = A0, A1 = A1));
            break;
        }
        print(iters);
        iters <- iters - 1;
    }
}
```

The above function terminates when a matrix M is found that is not stable when all component response rates are set to $\gamma = 1$, but is stable when half of component response rates are 1.95 and half are 0.05. The function is used to illustrate the concept of how a fast versus slow component responses can cause a system to become stable. Simulations were run for `iter = 1000000`, but terminated once an acceptable `A0` and `A1` were found. The code below plots the eigenvalue distributions of `A0` and `A1` in panels **a** and **b**, respectively.

The plot itself can be recreated with the code below.

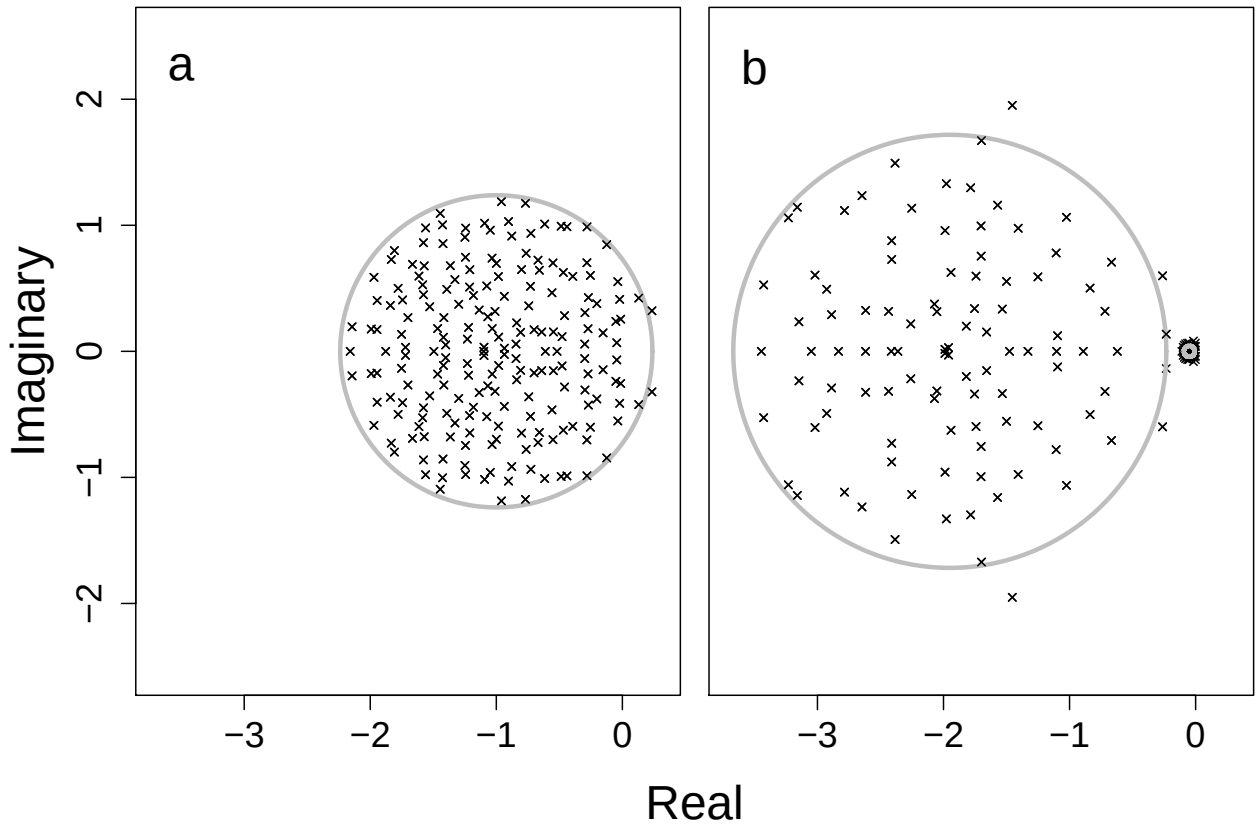
```
par(mfrow = c(1, 2), mar = c(0.5, 0.5, 0.5, 0.5), oma = c(5, 5, 0, 0));
plot(A0_r, A0_i, xlim = c(-3.7, 0.3), ylim = c(-2, 2), pch = 4, cex = 0.7,
     xlab = "", ylab = "", cex.lab = 1.5, cex.axis = 1.5, asp = 1);
vl <- seq(from = 0, to = 2*pi, by = 0.001);
A0x0 <- sqrt(200) * sd(A0vec) * cos(vl) + mean(diag(A0));
A0y0 <- sqrt(200) * sd(A0vec) * sin(vl);
text(x = -3.5, y = 2.25, labels = "a", cex = 2);
points(x = A0x0, y = A0y0, type = "l", lwd = 3, col = "grey");
points(A0_r, A0_i, pch = 4, cex = 0.7);

plot(A1_r, A1_i, xlim = c(-3.7, 0.3), ylim = c(-2, 2), pch = 4, cex = 0.7,
     xlab = "", ylab = "", cex.lab = 1.5, cex.axis = 1.5, asp = 1,
     col = "black", yaxt = "n");

vl <- seq(from = 0, to = 2*pi, by = 0.001);
A0x1a <- sqrt(100) * sd(A1vec[fhalf]) * cos(vl) + mean(diag(A1)[1:100]);
A0y1a <- sqrt(100) * sd(A1vec[fhalf]) * sin(vl);
points(x = A0x1a, y = A0y1a, type = "l", lwd = 3, col = "grey");
A0x1b <- sqrt(100) * sd(A1vec[shalf]) * cos(vl) + mean(diag(A1)[101:200]);
A0y1b <- sqrt(100) * sd(A1vec[shalf]) * sin(vl);
points(x = A0x1b, y = A0y1b, type = "l", lwd = 3, col = "grey");

points(A1_r[1:100], A1_i[1:100],pch = 4, cex = 0.7);

text(x = -3.5, y = 2.25, labels = "b", cex = 2);
mtext(side = 1, "Real", outer = TRUE, line = 3, cex = 2);
mtext(side = 2, "Imaginary", outer = TRUE, line = 2.5, cex = 2);
```



To find out how frequently M was stable given that all $\gamma = 1$ versus $\gamma = \{1.95, 0.05\}$, the function below was created.

```
stab_bgamma <- function(S = 200, C = 0.05, Osd = 0.4, iters = 10000){
    ress     <- matrix(data = 0, nrow = iters, ncol = 2);
    A0_count <- 0;
    A1_count <- 0;
    while(iters > 0){
        A_dat  <- rnorm(n = S * S, mean = 0, sd = Osd);
        A_mat  <- matrix(data = A_dat, nrow = S);
        C_dat  <- rbinom(n = S * S, size = 1, prob = C);
        C_mat  <- matrix(data = C_dat, nrow = S, ncol = S);
        A_mat  <- A_mat * C_mat;
        gammas <- c(rep(1.95, S/2), rep(0.05, S/2))
        mu_gam <- mean(gammas);
        diag(A_mat) <- -1;
        A1     <- gammas * A_mat;
        A0     <- mu_gam * A_mat;
        A0_e   <- eigen(A0)$values;
        A0_r   <- Re(A0_e);
        A0_i   <- Im(A0_e);
        A1_e   <- eigen(A1)$values;
        A1_r   <- Re(A1_e);
        A1_i   <- Im(A1_e);
        if(max(A0_r) < 0){
            ress[iters, 1] <- 1;
            A0_count       <- A0_count + 1;
            }
```

```
        if(max(A1_r) < 0){
            ress[iters, 2] <- 1;
            A1_count       <- A1_count + 1;
        }
        print(c(iters, A0_count, A1_count));
        iters <- iters - 1;
    }
    return(ress);
}
```

The function above was run for `iters = 1000000`, and the resulting matrix `ress` was returned. Each row of `ress` represents a single M given $\gamma = 1$ (column 1) versus $\gamma = \{1.95, 0.05\}$ (column 2). Values of 0 indicate that M was found to be unstable (at least one real component of its eigenvalues greater than or equal to zero), whereas values of 1 indicate that M was found to be stable (all real components of eigenvalues are negative). The frequencies of stable M were 0 given $\gamma = 1$ and 0 given $\gamma = \{1.95, 0.05\}$, as reported in the main text and legend of Fig. 1.

Code and simulations underlying Fig. 2

Figure 2 of the main text shows how eigenvalue distributions in a system where $S = 1000$, $C = 1$, and $\sigma = 0.4$. Eigenvalues can be reproduced using the code below for when $\gamma = 1$ (panel a) and $\gamma \sim \mathcal{U}(0, 2)$ (panel b).

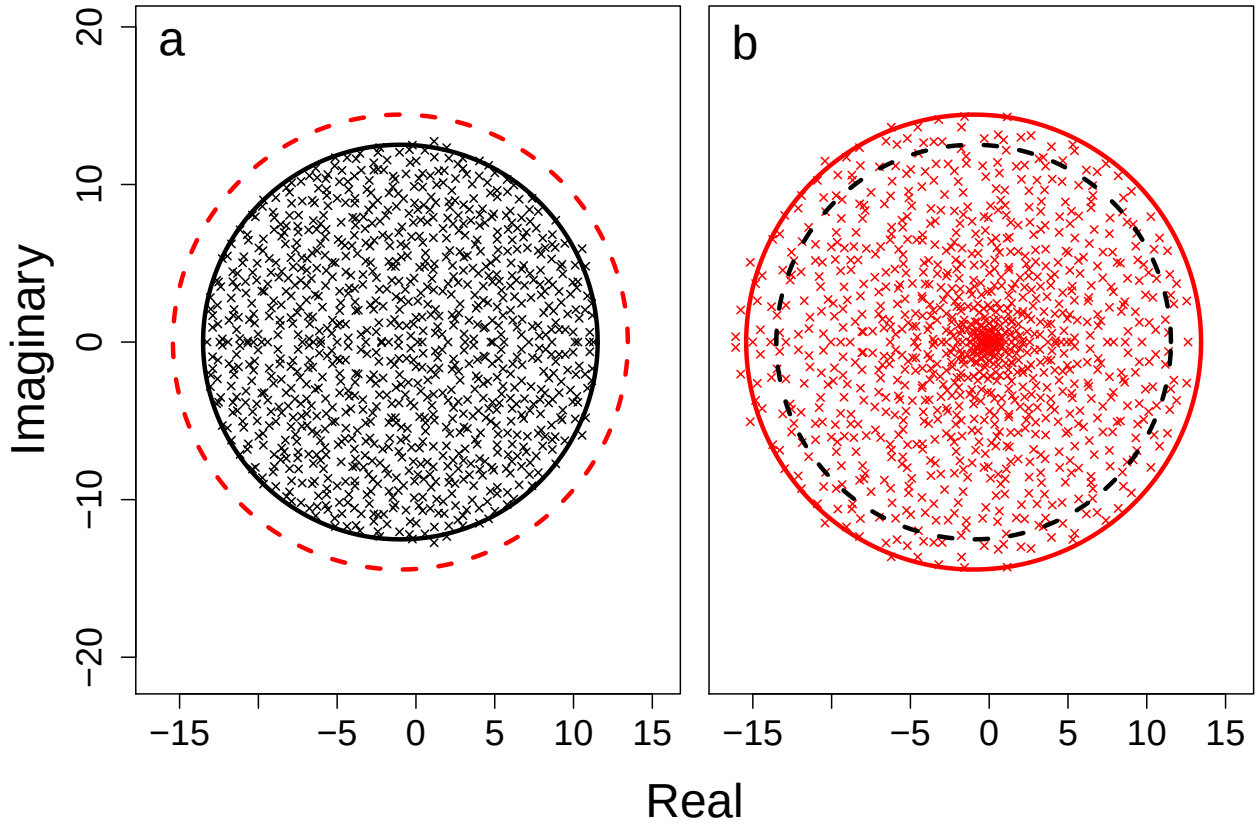
```
A_comp <- NULL;
A_dat  <- rnorm(n = 1000000, mean = 0, sd = 0.4);
A_mat  <- matrix(data = A_dat, nrow = 1000);
C_dat  <- rbinom(n = 1000 * 1000, size = 1, prob = 1);
C_mat  <- matrix(data = C_dat, nrow = 1000, ncol = 1000);
A_mat     <- A_mat * C_mat;
gammas <- runif(n = 1000, min = 0, max = 2);
mu_gam <- mean(gammas);
diag(A_mat) <- -1;
A1     <- gammas * A_mat;
A0     <- mu_gam * A_mat;
A0_e   <- eigen(A0)$values;
A0_r   <- Re(A0_e);
A0_i   <- Im(A0_e);
A1_e   <- eigen(A1)$values;
A1_r   <- Re(A1_e);
A1_i   <- Im(A1_e);

A0_vm       <- A0;
diag(A0_vm) <- NA;
A0vec       <- as.vector(A0_vm);
A0vec       <- A0vec[is.na(A0vec) == FALSE];
A1_vm       <- A1;
diag(A1_vm) <- NA;
A1vec       <- as.vector(A1_vm);
A1vec       <- A1vec[is.na(A1vec) == FALSE];
```

The code below reproduces the figure itself.

```
par(mfrow = c(1, 2), mar = c(0.5, 0.5, 0.5, 0.5), oma = c(5, 5, 0, 0));
plot(A0_r, A0_i, xlim = c(-16.5, 15.5), ylim = c(-16.5,15.5), pch = 4, cex = 0.7,
```

```
     xlab = "", ylab = "", cex.lab = 1.5, cex.axis = 1.5, asp = 1);
vl <- seq(from = 0, to = 2*pi, by = 0.001);
x0 <- sqrt(1000) * sd(A0vec) * cos(vl) + mean(diag(A0));
y0 <- sqrt(1000) * sd(A0vec) * sin(vl);
x1 <- sqrt(1000) * sd(A1vec) * cos(vl) + mean(diag(A1));
y1 <- sqrt(1000) * sd(A1vec) * sin(vl);
text(x = -15.5, y = 19, labels = "a", cex = 2);
points(x = x0, y = y0, type = "l", lwd = 3);
points(x = x1, y = y1, type = "l", col = "red", lwd = 3, lty = "dashed");
plot(A1_r, A1_i, xlim = c(-16.5, 15.5), ylim = c(-16.5,15.5), pch = 4, cex = 0.7,
     xlab = "", ylab = "", cex.lab = 1.5, cex.axis = 1.5, asp = 1, col = "red",
     yaxt = "n");
text(x = -15.5, y = 19, labels = "b", cex = 2);
points(x = x1, y = y1, type = "l", col = "red", lwd = 3)
points(x = x0, y = y0, type = "l", lwd = 3, lty = "dashed");
mtext(side = 1, "Real", outer = TRUE, line = 3, cex = 2);
mtext(side = 2, "Imaginary", outer = TRUE, line = 2.5, cex = 2);
```



Stability across increasing S

```
dat <- read.csv(file = "sim_results/C_1/random_all.csv");
dat <- dat[,-1];
```

The table below shows the results for all simulations of random M matrices at $\sigma = 0.4$ and $C = 1$ given a range of S from 2 to 32. In this table, the `A0` refers to matrices when $\gamma = 1$, while `A1` refers to matrices after

$Var(\gamma)$ is added and $\gamma \sim \mathcal{U}(0, 2)$. Each row summarises data for a given S over 1 million randomly simulated M (`A0` and `A1`). The column `A0_unstable` shows the number of `A0` matrices that are unstable, and the column `A0_stable` shows the number of `A0` matrices that are stable (these two columns sum to 1 million). Similarly, the column `A1_unstable` shows the number of `A1` matrices that are unstable and `A1_stable` shows the number that are stable. The columns `A1_stabilised` and `A1_destabilised` show how many `A0` matrices were stabilised or destabilised, respectively, by $Var(\gamma)$.

S	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
2	293	999707	293	999707	0	0
3	3602	996398	3609	996391	0	7
4	14937	985063	15008	984992	0	71
5	39289	960711	39783	960217	36	530
6	78845	921155	80207	919793	389	1751
7	133764	866236	136904	863096	1679	4819
8	204112	795888	208241	791759	5391	9520
9	288041	711959	291775	708225	12619	16353
10	384024	615976	384931	615069	23153	24060
11	485975	514025	481019	518981	35681	30725
12	590453	409547	577439	422561	48302	35288
13	689643	310357	669440	330560	57194	36991
14	777496	222504	751433	248567	60959	34896
15	850159	149841	821613	178387	58567	30021
16	905057	94943	877481	122519	51255	23679
17	943192	56808	919536	80464	40854	17198
18	969018	30982	949944	50056	30102	11028
19	984301	15699	970703	29297	20065	6467
20	992601	7399	983507	16493	12587	3493
21	996765	3235	991532	8468	7030	1797
22	998693	1307	995567	4433	3884	758
23	999503	497	997941	2059	1883	321
24	999861	139	999059	941	899	97
25	999964	36	999617	383	380	33
26	999993	7	999878	122	121	6
27	999995	5	999946	54	53	4
28	1000000	0	999975	25	25	0
29	1000000	0	999997	3	3	0
30	1000000	0	999999	1	1	0
31	1000000	0	999999	1	1	0
32	1000000	0	1000000	0	0	0
33	1000000	0	1000000	0	0	0
34	1000000	0	1000000	0	0	0
35	1000000	0	1000000	0	0	0
36	1000000	0	1000000	0	0	0
37	1000000	0	1000000	0	0	0
38	1000000	0	1000000	0	0	0
39	1000000	0	1000000	0	0	0
40	1000000	0	1000000	0	0	0
41	1000000	0	1000000	0	0	0
42	1000000	0	1000000	0	0	0
43	1000000	0	1000000	0	0	0
44	1000000	0	1000000	0	0	0
45	1000000	0	1000000	0	0	0
46	1000000	0	1000000	0	0	0
47	1000000	0	1000000	0	0	0

S	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
48	1000000	0	1000000	0	0	0
49	1000000	0	1000000	0	0	0
50	1000000	0	1000000	0	0	0

The results underlying this table were produced with the `rand_gen_var` function below.

```
rand_gen_var <- function(max_sp, iters, int_type = 0, rmx = 0.4, C = 1){
    tot_res <- NULL;
    fea_res <- NULL;
    for(i in 2:max_sp){
        iter           <- iters;
        tot_res[[i-1]] <- matrix(data = 0, nrow = iter, ncol = 7);
        fea_res[[i-1]] <- matrix(data = 0, nrow = iter, ncol = 7);
        while(iter > 0){
            r_vec    <- rnorm(n = i, mean = 0, sd = rmx);
            A0_dat   <- rnorm(n = i * i, mean = 0, sd = 0.4);
            A0       <- matrix(data = A0_dat, nrow = i, ncol = i);
            A0       <- species_interactions(mat = A0, type = int_type);
            C_dat    <- rbinom(n = i * i, size = 1, prob = C);
            C_mat    <- matrix(data = C_dat, nrow = i, ncol = i);
            A0       <- A0 * C_mat;
            diag(A0) <- -1;
            gam1     <- runif(n = i, min = 0, max = 2);
            A1       <- A0 * gam1;
            A0       <- A0 * mean(gam1);
            A0_stb   <- max(Re(eigen(A0)$values)) < 0;
            A1_stb   <- max(Re(eigen(A1)$values)) < 0;
            A0_fea   <- min(-1*solve(A0) %*% r_vec) > 0;
            A1_fea   <- min(-1*solve(A1) %*% r_vec) > 0;
            if(A0_stb == TRUE){
                tot_res[[i-1]][iter, 1] <- 1;
            }
            if(A1_stb == TRUE){
                tot_res[[i-1]][iter, 2] <- 1;
            }
            if(A0_fea == TRUE){
                fea_res[[i-1]][iter, 1] <- 1;
            }
            if(A1_fea == TRUE){
                fea_res[[i-1]][iter, 2] <- 1;
            }
            iter    <- iter - 1;
        }
        print(i);
    }
    all_res <- summarise_randmat(tot_res = tot_res, fea_res = fea_res);
    return(all_res);
}
```

The above function calls the two functions `species_interactions` and `summarise_randmat`, which are provided below.

```
species_interactions <- function(mat, type = 0){
    if(type == 1){
        mat[mat > 0] <- -1*mat[mat > 0];
    }
    if(type == 2){
        mat[mat < 0] <- -1*mat[mat < 0];
    }
    if(type == 3){
        for(i in 1:dim(mat)[1]){
            for(j in 1:dim(mat)[2]){
                if(mat[i, j] * mat[j, i] > 0){
                    mat[j, i] <- -1 * mat[j, i];
                }
            }
        }
    }
    return(mat);
}

summarise_randmat <- function(tot_res, fea_res){
    sims    <- length(tot_res);
    all_res <- matrix(data = 0, nrow = sims, ncol = 13);
    for(i in 1:sims){
        all_res[i, 1]  <- i + 1;
        # Stable and unstable
        all_res[i, 2]  <- sum(tot_res[[i]][,1] == FALSE);
        all_res[i, 3]  <- sum(tot_res[[i]][,1] == TRUE);
        all_res[i, 4]  <- sum(tot_res[[i]][,2] == FALSE);
        all_res[i, 5]  <- sum(tot_res[[i]][,2] == TRUE);
        # Stabilised and destabilised
        all_res[i, 6] <- sum(tot_res[[i]][,1] == FALSE &
                             tot_res[[i]][,2] == TRUE);
        all_res[i, 7] <- sum(tot_res[[i]][,1] == TRUE &
                             tot_res[[i]][,2] == FALSE);
        # Feasible and infeasible
        all_res[i, 8]  <- sum(fea_res[[i]][,1] == FALSE);
        all_res[i, 9]  <- sum(fea_res[[i]][,1] == TRUE);
        all_res[i, 10]  <- sum(fea_res[[i]][,2] == FALSE);
        all_res[i, 11]  <- sum(fea_res[[i]][,2] == TRUE);
        # Feased and defeased
        all_res[i, 12] <- sum(fea_res[[i]][,1] == FALSE &
                             fea_res[[i]][,2] == TRUE);
        all_res[i, 13] <- sum(fea_res[[i]][,1] == TRUE &
                             fea_res[[i]][,2] == FALSE);
    }
    cnames <- c("N", "A0_unstable", "A0_stable", "A1_unstable", "A1_stable",
                "A1_stabilised", "A1_destabilised", "A0_infeasible",
                "A0_feasible", "A1_infeasible", "A1_feasible",
                "A1_made_feasible", "A1_made_infeasible");
    colnames(all_res) <- cnames;
    return(all_res);
}
```

Note that feasibility results were ommited for the table above, but are shown below.

Stability of ecological networks

While the foundational work of May[1] applies broadly to complex networks, much attention has been given specifically to ecological networks of interacting species. In these networks, the matrix M is interpreted as a community matrix and each row and column is interpreted as a single species. The effect that the density of any species i has on the population dynamics of species j is found in M_{ij}, meaning that M holds the effects of pair-wise interactions between S species[2–4]. While May's original work[1] considered only randomly assembled communities, recent work has specifically looked at more restricted ecological communities including competitive networks (all off-diagonal elements of M are negative), mutualist networks (all off-diagonal elements of M are positive), and predator-prey networks (for any pair of i and j, the effect of i on j is negative and j on i is positive, or vice versa)[2–5]. In general, competitor and mutualist networks tend to be unstable, while predator-prey networks tend to be highly stabilising.

I investigate competitor, mutualist, and predator-prey networks following Allesina et al.[2]. To create these networks, I first generated a random matrix M, then changed the elements of M accordingly. If M was a competitive network, then the sign of any positive off-diagonal elements was reversed to be negative. If M was a mutualist network, then the sign of any positive off-diagonal elements was reversed to be positive. And if M was a predator-prey network, then all i and j pairs of elements were checked; any pairs of the same sign were changed so that one was negative and the other was positive. The `species_interaction` function used to do this is below.

```
species_interactions <- function(mat, type = 0){
    if(type == 1){
        mat[mat > 0] <- -1*mat[mat > 0];
    }
    if(type == 2){
        mat[mat < 0] <- -1*mat[mat < 0];
    }
    if(type == 3){
        for(i in 1:dim(mat)[1]){
            for(j in 1:dim(mat)[2]){
                if(mat[i, j] * mat[j, i] > 0){
                    mat[j, i] <- -1 * mat[j, i];
                }
            }
        }
    }
    return(mat);
}
```

This function was applied to all created matrices M, then the number of stable M matrices was estimated exactly as it was in the main text for random matrices for values of S from 2 to 50 (100 in the case of the relatively more stable predator-prey interactions), except that only 100000 random M were generated instead of 1 million. The following tables for restricted ecological communities can therefore be compared with the random M results above.

Competition {#competition}

Results for competitor interaction networks are shown below

```
cdat   <- read.csv(file = "sim_results/ecology/competition_C_1.csv");
cdat   <- cdat[,-1];
cd_tab <- cdat[,1:7];
kable(cd_tab);
```

N	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
2	48	99952	48	99952	0	0
3	229	99771	231	99769	0	2
4	701	99299	704	99296	0	3
5	1579	98421	1587	98413	0	8
6	3218	96782	3253	96747	6	41
7	5519	94481	5619	94381	23	123
8	9062	90938	9237	90763	77	252
9	13436	86564	13729	86271	230	523
10	18911	81089	19303	80697	505	897
11	25594	74406	25961	74039	1011	1378
12	33207	66793	33382	66618	1724	1899
13	41160	58840	41089	58911	2655	2584
14	50575	49425	49894	50106	3777	3096
15	59250	40750	57892	42108	4824	3466
16	67811	32189	65740	34260	5634	3563
17	75483	24517	73056	26944	5943	3516
18	82551	17449	79878	20122	5780	3107
19	88030	11970	85204	14796	5417	2591
20	92254	7746	89766	10234	4544	2056
21	95233	4767	93002	6998	3695	1464
22	97317	2683	95451	4549	2803	937
23	98508	1492	97122	2878	1991	605
24	99240	760	98407	1593	1216	383
25	99669	331	99082	918	739	152
26	99871	129	99490	510	452	71
27	99938	62	99732	268	240	34
28	99985	15	99888	112	108	11
29	99990	10	99951	49	46	7
30	100000	0	99981	19	19	0
31	100000	0	99993	7	7	0
32	100000	0	99996	4	4	0
33	100000	0	99998	2	2	0
34	100000	0	100000	0	0	0
35	100000	0	100000	0	0	0
36	100000	0	100000	0	0	0
37	100000	0	100000	0	0	0
38	100000	0	100000	0	0	0
39	100000	0	100000	0	0	0
40	100000	0	100000	0	0	0
41	100000	0	100000	0	0	0
42	100000	0	100000	0	0	0
43	100000	0	100000	0	0	0
44	100000	0	100000	0	0	0
45	100000	0	100000	0	0	0
46	100000	0	100000	0	0	0
47	100000	0	100000	0	0	0
48	100000	0	100000	0	0	0
49	100000	0	100000	0	0	0
50	100000	0	100000	0	0	0

Mutualism {#mutualism}

77 Results for mutualist interaction networks are shown below

```
mdat   <- read.csv(file = "sim_results/ecology/mutualism_C_1.csv");
mdat   <- mdat[,-1];
md_tab <- mdat[,1:7];
kable(md_tab);
```

N	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
2	56	99944	56	99944	0	0
3	3301	96699	3301	96699	0	0
4	34446	65554	34446	65554	0	0
5	86520	13480	86520	13480	0	0
6	99683	317	99683	317	0	0
7	99998	2	99998	2	0	0
8	100000	0	100000	0	0	0
9	100000	0	100000	0	0	0
10	100000	0	100000	0	0	0
11	100000	0	100000	0	0	0
12	100000	0	100000	0	0	0
13	100000	0	100000	0	0	0
14	100000	0	100000	0	0	0
15	100000	0	100000	0	0	0
16	100000	0	100000	0	0	0
17	100000	0	100000	0	0	0
18	100000	0	100000	0	0	0
19	100000	0	100000	0	0	0
20	100000	0	100000	0	0	0
21	100000	0	100000	0	0	0
22	100000	0	100000	0	0	0
23	100000	0	100000	0	0	0
24	100000	0	100000	0	0	0
25	100000	0	100000	0	0	0
26	100000	0	100000	0	0	0
27	100000	0	100000	0	0	0
28	100000	0	100000	0	0	0
29	100000	0	100000	0	0	0
30	100000	0	100000	0	0	0
31	100000	0	100000	0	0	0
32	100000	0	100000	0	0	0
33	100000	0	100000	0	0	0
34	100000	0	100000	0	0	0
35	100000	0	100000	0	0	0
36	100000	0	100000	0	0	0
37	100000	0	100000	0	0	0
38	100000	0	100000	0	0	0
39	100000	0	100000	0	0	0
40	100000	0	100000	0	0	0
41	100000	0	100000	0	0	0
42	100000	0	100000	0	0	0
43	100000	0	100000	0	0	0
44	100000	0	100000	0	0	0
45	100000	0	100000	0	0	0
46	100000	0	100000	0	0	0
47	100000	0	100000	0	0	0

N	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
48	100000	0	100000	0	0	0
49	100000	0	100000	0	0	0
50	100000	0	100000	0	0	0

78 **Predator-prey** {#pred-prey}

79 Results for predator-prey interaction networks are shown below

```
pdat   <- read.csv(file = "sim_results/ecology/pred-prey_C_1.csv");
pdat   <- pdat[,-1];
pd_tab <- pdat[,1:7];
kable(pd_tab);
```

N	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
2	0	100000	0	100000	0	0
3	0	100000	0	100000	0	0
4	0	100000	0	100000	0	0
5	1	99999	1	99999	0	0
6	4	99996	4	99996	0	0
7	2	99998	2	99998	0	0
8	5	99995	5	99995	0	0
9	20	99980	21	99979	0	1
10	20	99980	22	99978	0	2
11	38	99962	39	99961	0	1
12	64	99936	66	99934	0	2
13	87	99913	91	99909	0	4
14	157	99843	159	99841	0	2
15	215	99785	227	99773	0	12
16	293	99707	310	99690	0	17
17	383	99617	408	99592	0	25
18	443	99557	473	99527	3	33
19	642	99358	675	99325	4	37
20	836	99164	887	99113	7	58
21	1006	98994	1058	98942	10	62
22	1153	98847	1228	98772	20	95
23	1501	98499	1593	98407	30	122
24	1841	98159	1996	98004	40	195
25	2146	97854	2316	97684	58	228
26	2643	97357	2809	97191	119	285
27	3034	96966	3258	96742	158	382
28	3690	96310	3928	96072	201	439
29	4257	95743	4532	95468	290	565
30	4964	95036	5221	94779	424	681
31	5627	94373	5978	94022	452	803
32	6543	93457	6891	93109	666	1014
33	7425	92575	7777	92223	818	1170
34	8540	91460	8841	91159	1071	1372
35	9526	90474	9842	90158	1337	1653
36	10617	89383	10891	89109	1624	1898
37	12344	87656	12508	87492	2021	2185
38	13675	86325	13877	86123	2442	2644
39	15264	84736	15349	84651	2870	2955

N	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
40	17026	82974	17053	82947	3363	3390
41	18768	81232	18614	81386	3905	3751
42	20791	79209	20470	79530	4579	4258
43	23150	76850	22754	77246	5217	4821
44	25449	74551	24184	75816	6285	5020
45	27702	72298	26464	73536	6754	5516
46	30525	69475	28966	71034	7646	6087
47	32832	67168	31125	68875	8487	6780
48	36152	63848	33865	66135	9479	7192
49	38714	61286	36242	63758	10125	7653
50	41628	58372	38508	61492	11036	7916
51	44483	55517	41023	58977	11704	8244
52	48134	51866	44287	55713	12573	8726
53	51138	48862	46721	53279	13223	8806
54	54261	45739	49559	50441	13757	9055
55	57647	42353	52403	47597	14324	9080
56	60630	39370	55293	44707	14669	9332
57	63647	36353	57787	42213	15103	9243
58	66961	33039	60439	39561	15450	8928
59	69968	30032	63708	36292	15246	8986
60	72838	27162	66270	33730	15177	8609
61	75609	24391	68873	31127	15006	8270
62	77999	22001	71318	28682	14538	7857
63	80616	19384	73517	26483	14510	7411
64	83089	16911	76209	23791	13784	6904
65	85150	14850	78086	21914	13412	6348
66	86908	13092	80437	19563	12477	6006
67	88671	11329	82379	17621	11718	5426
68	90537	9463	84483	15517	10878	4824
69	91969	8031	86233	13767	10033	4297
70	93181	6819	87914	12086	9070	3803
71	94330	5670	89200	10800	8401	3271
72	95324	4676	90833	9167	7359	2868
73	96143	3857	91805	8195	6726	2388
74	96959	3041	93065	6935	5900	2006
75	97543	2457	93987	6013	5222	1666
76	97969	2031	94900	5100	4481	1412
77	98497	1503	95756	4244	3809	1068
78	98744	1256	96442	3558	3269	967
79	99045	955	96942	3058	2837	734
80	99276	724	97528	2472	2329	581
81	99481	519	97996	2004	1894	409
82	99556	444	98321	1679	1597	362
83	99691	309	98722	1278	1227	258
84	99752	248	98943	1057	1015	206
85	99833	167	99144	856	837	148
86	99895	105	99346	654	642	93
87	99925	75	99461	539	530	66
88	99945	55	99566	434	428	49
89	99976	24	99675	325	324	23
90	99977	23	99756	244	243	22
91	99982	18	99839	161	155	12

N	A0_unstable	A0_stable	A1_unstable	A1_stable	A1_stabilised	A1_destabilised
92	99988	12	99865	135	135	12
93	99994	6	99885	115	115	6
94	99993	7	99911	89	88	6
95	99998	2	99953	47	47	2
96	99999	1	99965	35	35	1
97	99999	1	99979	21	21	1
98	100000	0	99973	27	27	0
99	100000	0	99984	16	16	0
100	100000	0	99989	11	11	0

Feasilbility of complex systems

It also would be useful to look at feasibility criteria established by 6, who very recently made the point that some of May and Allesina's criteria allows for negative species densities when stable. Feasibility criteria are as follows,

$$x^* = -\left(\theta I + (CS)^{-\delta} A\right)^{-1} r.$$

The above is not nearly as nasty as it looks, especially because it is entirely reasonable to simply use convenient values to parameterise it. The variable x^* is just the vector of species abundances at equilibrium (we need all of them to be positive). The matrix I is just the identity matrix (1s on the diagonal, 0s on the off-diagonal elements), and the value θ is just the strength of intraspecific competition – we can just set this to $\theta = -1$ (as others have). The variable C is just the connectance of the community, which we will also set to 1 for convenience (lower values would mean that some species don't interact with one another, corresponding to an off-diagonal matrix element of 0). And δ just affects the strength of interactions – which we can set to $\delta = 0$ for strong interactions. Hence, the whole $(CS)^{-\delta} = 1$, so we're just adding the diagonal matrix of -1s (θI) to A, which has a diagonal of all zeros and an off diagonal effecting species interactions,

$$x^* = -\left(\theta I + A\right)^{-1} r.$$

To check the feasibility criteria, all that needs to be done is to invert $(\theta I + A)$ and multiply the matrix by the vector of growth rates r, which we can also just set to 1. So really, we're just multiplying $-(\theta I + A)$ by a vector of ones and checking to make sure that all the values are positive. This isn't much extra work, but it will probably go a long way toward satisfying any reviewers familiar with 6.

Genetic algorithm

References

1. May, R. M. Will a large complex system be stable? *Nature* **238,** 413–414 (1972).

2. Allesina, S. & Tang, S. Stability criteria for complex ecosystems. *Nature* **483,** 205–208 (2012).

3. Allesina, S. & Tang, S. The stability–complexity relationship at age 40: a random matrix perspective. *Population Ecology* 63–75 (2015). doi:10.1007/s10144-014-0471-0

4. Tang, S. & Allesina, S. Reactivity and stability of large ecosystems. *Frontiers in Ecology and Evolution* **2,**

1–8 (2014).

5. Allesina, S. & Levine, J. M. A competitive network theory of species diversity. *Proceedings of the National Academy of Sciences of the United States of America* **108,** 5638–5642 (2011).

6. Dougoud, M., Vinckenbosch, L., Rohr, R., Bersier, L.-F. & Mazza, C. The feasibility of equilibria in large ecosystems: a primary but neglected concept in the complexity-stability debate. *PLOS Computational Biology* **14,** e1005988 (2018).