Cálculos

Roberto Álvarez

Table of contents

1 Ecuación Lotka-Volterra

$$\frac{dx_i}{dt} = r_i x_i + \sum_{j=1}^{n} A_{ij} x_i x_j$$

$$\forall i = 1, 2, 3, \dots n$$

$$\frac{dx_i}{dt} = x_i (r_i + \sum_{j=1}^{n} A_{ij} x_j)$$

Lotka-Volterra en forma vectorial

$$\frac{d\mathbf{x}}{dt} = D(\mathbf{x})(\mathbf{r} + \mathcal{A}\mathbf{x})$$

con el vector de abundancias de las poblaciones

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

$$D(\mathbf{x}) = \begin{pmatrix} x_1(t) & 0 & 0 & ... & 0 \\ 0 & x_2(t) & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & x_n(t) \end{pmatrix}$$

El vector de tasas de crecimiento

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix}$$

La matriz de interacciones

$$\mathcal{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & ... & a_{nn} \end{pmatrix}$$

1.1 Demostración de ambas ecuaciones

Entonces la parte $\mathbf{r} + \mathcal{A}\mathbf{x}$ es un vector columna

$$\mathbf{r} + \mathcal{A}\mathbf{x} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} + \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & ... & a_{nn} \end{pmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

Si efectuamos la operación del producto de la matriz de inetracciones $\mathbf{A}$ con el vector columna de abundancias $\mathbf{x}$

$$\mathbf{r} + \mathcal{A}\mathbf{x} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} + \begin{bmatrix} a_{11}x_1(t) + a_{12}x_2(t) + a_{13}x_3(t) + ... + a_{1n}x_n(t) \\ a_{21}x_1(t) + a_{22}x_2(t) + a_{23}x_3(t) + ... + a_{2n}x_n(t) \\ a_{31}x_1(t) + a_{32}x_2(t) + a_{33}x_3(t) + ... + a_{3n}x_n(t) \\ \vdots \\ a_{n1}x_1(t) + a_{n2}x_2(t) + a_{n3}x_3(t) + ... + a_{nn}x_n(t) \end{bmatrix}$$

Finalmente sumamos ambos vectores columnas $\mathbf{r}$ y $\mathcal{A}\mathbf{x}$ tenemos:

$$
\mathbf{r} + \mathcal{A}\mathbf{x} = \begin{bmatrix} r_1 + a_{11}x_1(t) + a_{12}x_2(t) + a_{13}x_3(t) + ... + a_{1n}x_n(t) \\ r_2 + a_{21}x_1(t) + a_{22}x_2(t) + a_{23}x_3(t) + ... + a_{2n}x_n(t) \\ r_3 + a_{31}x_1(t) + a_{32}x_2(t) + a_{33}x_3(t) + ... + a_{3n}x_n(t) \\ \vdots \\ r_n + a_{n1}x_1(t) + a_{n2}x_2(t) + a_{n3}x_3(t) + ... + a_{nn}x_n(t) \end{bmatrix}
$$

Finalmente hacemos el producto de la matriz $\mathcal{D}(\mathbf{x})$ con el vector $\mathbf{r} + \mathcal{A}\mathbf{x}$

$$
\mathcal{D}(\mathbf{x})(\mathbf{r}+\mathcal{A}\mathbf{x}) = \begin{pmatrix} x_1(t) & 0 & 0 & ... & 0 \\ 0 & x_2(t) & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & x_n(t) \end{pmatrix} \begin{bmatrix} r_1 + a_{11}x_1(t) + a_{12}x_2(t) + a_{13}x_3(t) + ... + a_{1n}x_n(t) \\ r_2 + a_{21}x_1(t) + a_{22}x_2(t) + a_{23}x_3(t) + ... + a_{2n}x_n(t) \\ r_3 + a_{31}x_1(t) + a_{32}x_2(t) + a_{33}x_3(t) + ... + a_{3n}x_n(t) \\ \vdots \\ r_n + a_{n1}x_1(t) + a_{n2}x_2(t) + a_{n3}x_3(t) + ... + a_{nn}x_n(t) \end{bmatrix}
$$

Finalmente:

$$
\begin{bmatrix} r_1x_1(t) + a_{11}x_1^2(t) + a_{12}x_2(t)x_1(t) + a_{13}x_3(t)x_1(t) + ... + a_{1n}x_n(t)x_1(t) \\ r_2x_2(t) + a_{21}x_1(t)x_2(t) + a_{22}x_2^2(t) + a_{23}x_3(t)x_2(t) + ... + a_{2n}x_n(t)x_2(t) \\ r_3x_3(t) + a_{31}x_1(t)x_3(t) + a_{32}x_2(t)x_3(t) + a_{33}x_3^2(t) + ... + a_{3n}x_n(t)x_3(t) \\ \vdots \\ r_nx_n(t) + a_{n1}x_1(t)x_n(t) + a_{n2}x_2(t)x_n(t) + a_{n3}x_3(t) + ... + a_{nn}x_n^2(t) \end{bmatrix}
$$

1.2 Para 1-D

$$
\frac{dx}{dt} = x(t)(r + ax(t))
$$

La solución no trivial es:

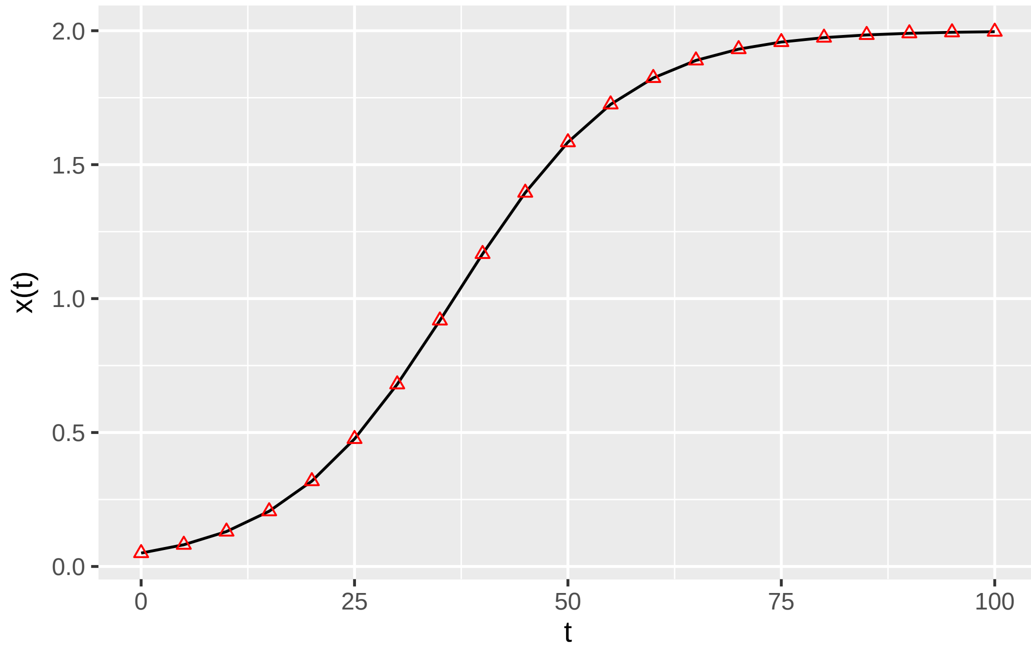
$$
\frac{dx}{dt} = x^*(r + ax^*) = 0
$$
$$
r + ax^* = 0
$$
$$
x = -\frac{r}{a}
$$

con $a < 0$ la solución es positiva

```r
library(deSolve) # integrate ODEs
library(tidyverse) # plotting and wrangling
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.0     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```r
# define the differential equation
logistic_growth <- function(t, x, parameters){
  with(as.list(c(x, parameters)), {
    dxdt <- x * (r + a * x)
    list(dxdt)
  })
}
# define parameters, integration time, initial conditions
times <- seq(0, 100, by = 5)
x0 <- 0.05
r <- 0.1
a <- -0.05
parameters <- list(r = r, a = a)
# solve numerically
out <- ode(y = x0, times = times,
           func = logistic_growth, parms = parameters,
           method = "ode45")
# now compute analytically
solution <- r * x0 * exp(r * times) / (r - a * x0 * (exp(r * times) - 1))
# use ggplot to plot
res <- tibble(time = out[,1], x_t = out[,2], x_sol = solution)
ggplot(data = res) + aes(x = time, y = x_t) +
  geom_line() +
  geom_point(aes(x = time, y = x_sol), colour = "red", shape = 2) +
  ylab(expression("x(t)")) + xlab(expression("t"))
```



1.3 Para 2D

La solución de co-existencia para el modelo Lotka-Volterra generalizado es la siguiente:

$$\frac{dx_1}{dt} = x_1(r_1 + ax_1 + bx_2) = 0 \quad (1)$$
$$\frac{dx_2}{dt} = x_2(r_2 + cx_1 + dx_2) = 0 \quad (2)$$

Tendremos que encontrar las soluciones para x_1 y x_2 en términos de el vector de tasas de crecimiento $\mathbf{r}$ y de la matriz de interacciones $\mathcal{A}$

Tarea

Es decir debemos demostrar que las soluciones de co-existencia:

$$r_1 + ax_1 + bx_2 = 0 \quad (3)$$
$$r_2 + cx_1 + dx_2 = 0 \quad (4)$$

son iguales a las soluciones vectoriales.

$$\mathbf{x}* = \begin{bmatrix} x_1* \\ x_2* \end{bmatrix} = \mathcal{A}^{-1} \begin{bmatrix} -r_1 \\ -r_2 \end{bmatrix}$$

en dos dimensiones si una matriz $\mathcal{A}$

$$\mathcal{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

su inversa se calcula como:

$$\mathcal{A}^{-1} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \frac{1}{det(\mathcal{A})}$$

1.4 Dinámicas Lotka-Volterra

```
# Generalized Lotka-Volterra model
GLV <- function(t, x, parameters){
  with(as.list(c(x, parameters)), {
    x[x < 10^-8] <- 0 # prevent numerical problems
    dxdt <- x * (r + A %*% x)
    list(dxdt)
  })
}
# function to plot output
plot_ODE_output <- function(out){
  out <- as.data.frame(out)
  colnames(out) <- c("time", paste("sp", 1:(ncol(out) -1), sep = "_"))
  out <- as_tibble(out) %>% gather(species, density, -time)
  pl <- ggplot(data = out) +
    aes(x = time, y = density, colour = species) +
    geom_line()
  show(pl)
  return(out)
}
# general function to integrate GLV
integrate_GLV <- function(r, A, x0, maxtime = 100, steptime = 0.5){
  times <- seq(0, maxtime, by = steptime)
  parameters <- list(r = r, A = A)
  # solve numerically
  out <- ode(y = x0, times = times,
             func = GLV, parms = parameters,
```

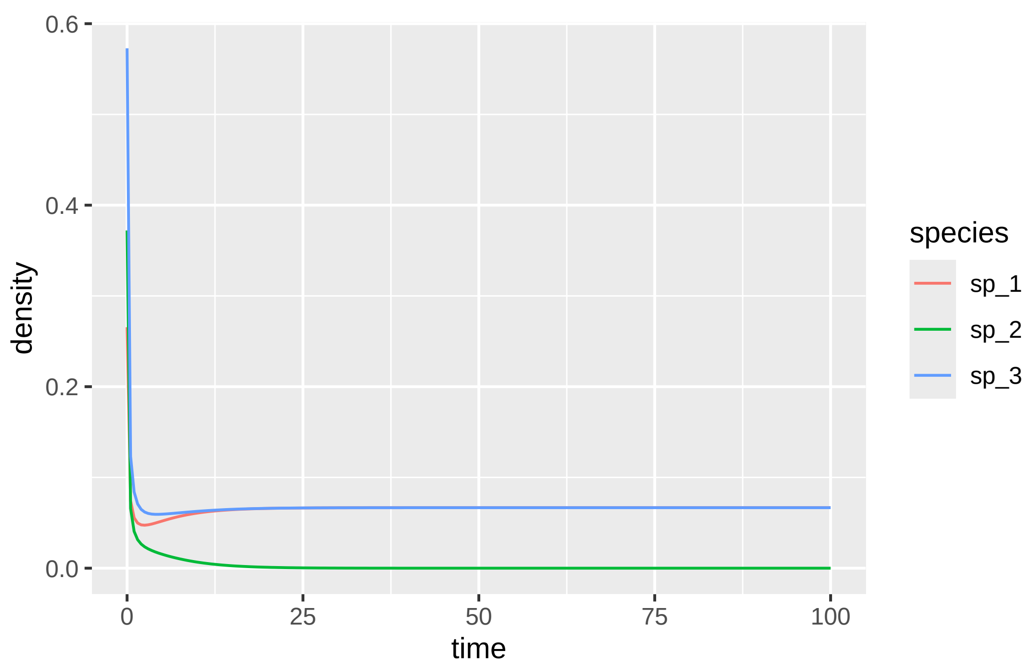
```
              method = "ode45")
  # plot and make into tidy form
  out <- plot_ODE_output(out)
  return(out)
}
```

1.4.1 Competencia que conduce a la extinción de especies

```
set.seed(1) # for reproducibility
r_1 <- rep(1, 3)
A_1 <- -matrix(c(10, 9, 5,
                 9, 10, 9,
                 5, 9, 10), 3, 3, byrow = TRUE)
# check the existence of feasible equilibrium
print(solve(A_1, -r_1)) # not feasible
```

```
[1] -0.08333333  0.25000000 -0.08333333
```

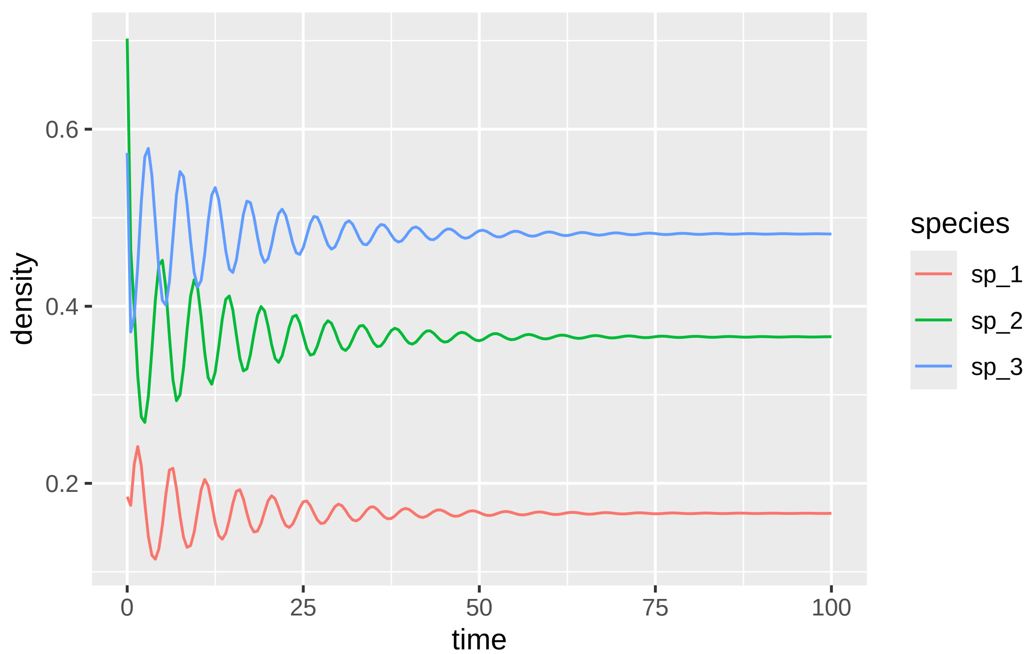
```
x0_1 <- runif(3)
res_1 <- integrate_GLV(r_1, A_1, x0_1)
```



```r
set.seed(2) # for reproducibility
r_2 <- rep(10, 3)
A_2 <- -matrix(c(10, 7, 12,
                 15, 10, 8,
                 7, 11, 10), 3, 3, byrow = TRUE)
# check the existence of feasible equilibrium
print(solve(A_2, -r_2)) # feasible
```

```
[1] 0.1661130 0.3654485 0.4817276
```

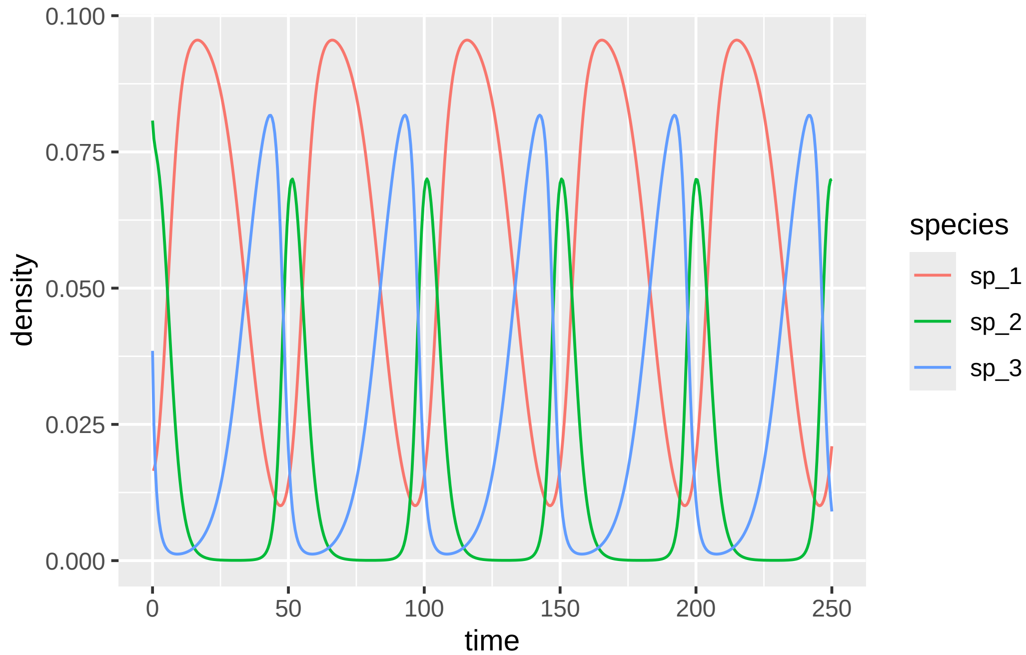
```r
x0_2 <- runif(3)
res_2 <- integrate_GLV(r_2, A_2, x0_2)
```




```r
set.seed(3) # for reproducibility
r_3 <- rep(1, 3)
A_3 <- -matrix(c(10, 6, 12,
                 14, 10, 2,
                 8, 18, 10), 3, 3, byrow = TRUE)
# check the existence of feasible equilibrium
print(solve(A_3, -r_3)) # feasible
```

```
[1] 0.05714286 0.01428571 0.02857143
```

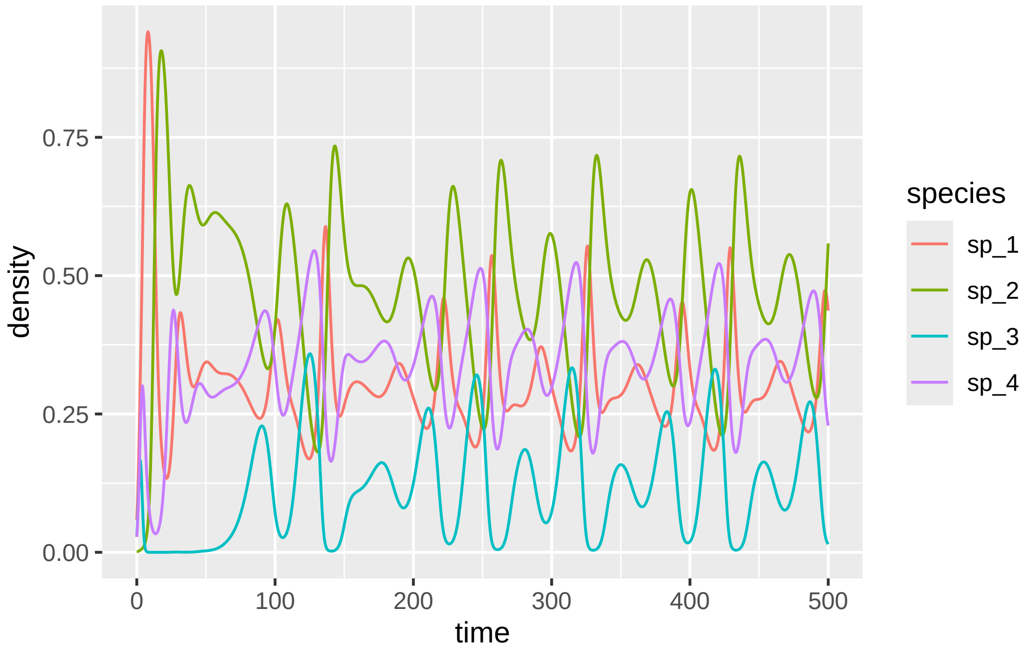
```
x0_3 <- 0.1 * runif(3)
res_3 <- integrate_GLV(r_3, A_3, x0_3, maxtime = 250)
```




```
set.seed(4) # for reproducibility
r_4 <- c(1, 0.72, 1.53, 1.27)
A_4 <- -matrix(c(1, 1.09, 1.52, 0,
                 0, 0.72, 0.3168, 0.9792,
                 3.5649, 0, 1.53, 0.7191,
                 1.5367, 0.6477, 0.4445, 1.27), 4, 4, byrow = TRUE)
# check the existence of feasible equilibrium
print(solve(A_4, -r_4)) # feasible
```

```
[1] 0.3013030 0.4586546 0.1307655 0.3557416
```

```
x0_4 <- 0.1 * runif(4)
res_4 <- integrate_GLV(r_4, A_4, x0_4, maxtime = 500)
```



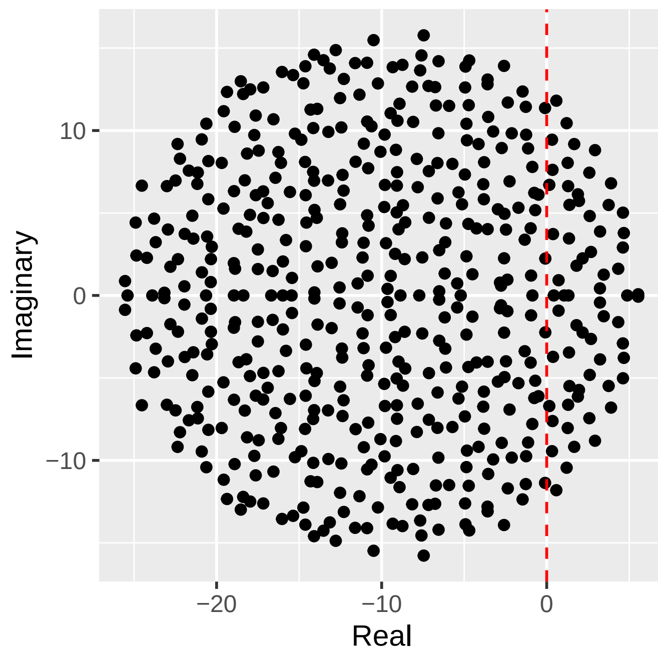
1.5 Estabilidad de comunidades grandes

Robert May (1973) propuso una forma de generar una matriz aleatoria de interacciones

```
build_May_normal <- function(n, C, d, sigma){
  # fill the whole matrix
  M <- matrix(rnorm(n * n, mean = 0, sd = sigma), n, n)
  # remove connections
  M <- M * matrix(runif(n * n) <= C, n, n)
  # set diagonals
  diag(M) <- -d
  return(M)
}
```

```
plot_eigenvalues <- function(M, prediction = NULL){
  eig <- eigen(M, only.values = TRUE)$values
  dt <- tibble(Real = Re(eig), Imaginary = Im(eig))
  pl <- ggplot(dt) + aes(x = Real, y = Imaginary) +
    geom_point() +
    coord_equal() +
    geom_vline(xintercept = 0, colour = "red", linetype = 2)
```

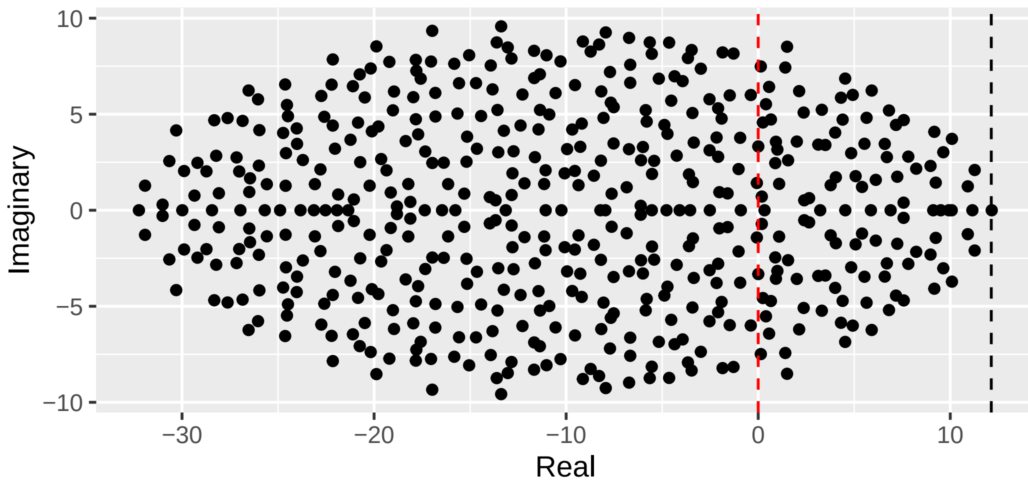
```
  if (is.null(prediction) == FALSE) {
    pl <- pl + geom_vline(xintercept = prediction, colour = "black", linetype = 2)
  }
  show(pl)
}
set.seed(100) # for reproducibility
# parameters
n <- 500
C <- 0.5
d <- 10
sigma <- 1
M <- build_May_normal(n, C, d, sigma)
#library(tidyr)
plot_eigenvalues(M)
```




```
build_Allesina_Tang_normal <- function(n, C, d, sigma, rho){
  # sample coefficients in pairs
  pairs <- MASS::mvrnorm(n = n * (n-1) / 2,
                         mu = c(0, 0),
                         Sigma = sigma^2 * matrix(c(1, rho, rho, 1), 2, 2))
  # build a completely filled matrix
```

```r
  M <- matrix(0, n, n)
  M[upper.tri(M)] <- pairs[,1]
  M <- t(M)
  M[upper.tri(M)] <- pairs[,2]
  # determine which connections to retain
  Connections <- (matrix(runif(n * n), n, n) <= C) * 1
  Connections[lower.tri(Connections)] <- 0
  diag(Connections) <- 0
  Connections <- Connections + t(Connections)
  M <- M * Connections
  diag(M) <- -d
  return(M)
}
```

```r
# parameters
n <- 500
C <- 0.5
d <- 10
sigma <- 1
rho <- 0.4
M <- build_Allesina_Tang_normal(n, C, d, sigma, rho)
prediction <- sqrt(n * C * sigma^2) * (1 + rho) - d
plot_eigenvalues(M, prediction)
```


Imaginary
10
5
0
−5
−10
−30
−20
−10
0
10
Real