Modelos dinámicos y computacionales en Economía

Introducción a R

Licenciatura en Economía, FCEA, UDELAR

14 de septiembre de 2021

Contenido de la clase:

- Presentación del software R
- Pros y contras
- R y RStudio
- Expresiones, asignaciones, funciones
- Vectores y matrices
- Estructuras de control
- Ejemplo: Ecuación logística

¿Qué es R?

- Es un ambiente de computación estadística y gráficos.
- Conjunto de herramientas muy flexibles, ampliables a partir de paquetes o generando nuestras propias funciones.
- Software libre: implica que R es gratis y además de código abierto, es decir, cualquiera puede ver "qué hay adentro". Cualquier usuario puede descargar y crear código de manera gratuita.
- Es un software construido en colaboración por una lista siempre creciente de contribuyentes: constantemente actualizado.
- R se ha convertido en una de las herramientas más utilizadas en estadística y análisis de datos, siendo particularmente popular en data mining

Pros y contras

- A favor: software versátil, que permite estudiar multiplicidad de problemas y utilizar diferentes técnicas.
  - es el software estadístico y econométrico más completo
  - manejo de grandes bases de datos
  - permite generar nuevos paquetes y funciones, para estudiar nuevos problemas
  - Programación Orientada a Objetos
- En contra: hace exactamente lo que le pedimos $\rightarrow$ curva de aprendizaje

Solución: utilizar la documentación de ayuda

- ?median : consultamos acerca de esta función
- ??median: consultamos todos los paquetes que se relacionan con la palabra buscada

RStudio

- Ambiente de Desarrollo Integrado (IDE) específico para R
- Existe una edición libre y una edición comercial
- Incluye:
  - Consola (para ejecutar los comandos)
  - Ambiente de trabajo (donde figuran los distintos objetos creados)
  - Gráficos
  - Paquetes (instalados y disponibles para instalar)
  - Ayuda
  - Herramientas para resaltar sintaxis
- Disponible para Windows, Linux y Mac, en versiones de Escritorio, Server y *Cloud*.

R es mucho más que software estadístico

- En los últimos años se han generado muchas herramientas para analizar, explorar, modelizar, comunicar y conectar los resultados del trabajo en R.
- De esta manera, podemos integrar el análisis de datos al flujo de trabajo mediante aplicaciones web, documentos (markdown o LaTeX) o interfases con otros programas.

Links:

- R: https://cran.r-project.org/
- RStudio: https://www.rstudio.com/products/rstudio/download/
- RStudio Cloud: `https://rstudio.cloud/`

Expresiones y asignaciones

- Expresión: 7 * 8 $\rightarrow$ R devuelve el resultado
- Asignación: X = 10 * 4
  - R no devuelve el resultado
  - el resultado queda guardado: ver en "Environment", arriba a la derecha
  - si en la siguiente línea escribimos *X*, el programa nos devuelve el resultado
  - prestar atención a las mayúsculas y minúsculas
  - X = X + 10 ?

Expresiones y asignaciones

- Expresión: 7 * 8 $\rightarrow$ R devuelve el resultado
- Asignación: X = 10 * 4
  - R no devuelve el resultado
  - el resultado queda guardado: ver en "Environment", arriba a la derecha
  - si en la siguiente línea escribimos *X*, el programa nos devuelve el resultado
  - prestar atención a las mayúsculas y minúsculas
  - X = X + 10 ?
- Asignaciones no son ecuaciones!

Expresiones y asignaciones

- Expresión: 7 * 8 $\rightarrow$ R devuelve el resultado
- Asignación: X = 10 * 4
  - R no devuelve el resultado
  - el resultado queda guardado: ver en "Environment", arriba a la derecha
  - si en la siguiente línea escribimos *X*, el programa nos devuelve el resultado
  - prestar atención a las mayúsculas y minúsculas
  - X = X + 10 ?
- Asignaciones no son ecuaciones!
- Comparadores: <, >, <=, >=, ==, !=

Expresiones y asignaciones

- Expresión: 7 * 8 $\rightarrow$ R devuelve el resultado
- Asignación: X = 10 * 4
  - R no devuelve el resultado
  - el resultado queda guardado: ver en "Environment", arriba a la derecha
  - si en la siguiente línea escribimos *X*, el programa nos devuelve el resultado
  - prestar atención a las mayúsculas y minúsculas
  - X = X + 10 ?
- Asignaciones no son ecuaciones!
- Comparadores: <, >, <=, >=, ==, !=
- Operadores lógicos: AND (&), OR (|), XOR (xor), NOT(!)

Funciones

- funciones simples:
  - sin(pi), sin(pi/2)
  - exp(10)
  - 100 ** 2
  - log(1)
  - sec = 1:14
  - seq(0,1,0.01)
  - rnorm(100)
- Composición de funciones:
  - mean(rnorm(100))
  - exp(log(pi))
  - plot(rnorm(100),type="l")
  - plot(density(rnorm(10000)))

Vectores y matrices

- Vectores:
  - vec = vector("numeric",10)
  - vec[i]: el i-ésimo número en el vector
  - vec[4] = 5
- Matrices:
  - mat = matrix(0,nrow=3,ncol=3)
  - mat[i,j] = valor correspondiente a la asignación de la fila "i" y la columna "j".
  - mat [1,2] = 3

Loops - Estructuras de control

- For: utilizado para ejecutar una serie de comandos *n* veces.
  - Sintaxis: **for (condición) { acción }**

Loops - Estructuras de control

- For: utilizado para ejecutar una serie de comandos *n* veces.
  - Sintaxis: **for (condición) { acción }**
  - Ejemplo: for (i in 1:10) { x = x + 1 }
  - Ejemplo: for (i in 1: length(vect)) { x = x - 1 }

Loops - Estructuras de control

- For: utilizado para ejecutar una serie de comandos *n* veces.
  - Sintaxis: **for (condición) { acción }**
  - Ejemplo: for (i in 1:10) { x = x + 1 }
  - Ejemplo: for (i in 1: length(vect)) { x = x - 1 }
- While: utilizado para ejecutar una serie de comandos, mientras la condición sea cierta.
  - Sintaxis: **while (condición) { acción }**

Loops - Estructuras de control

- For: utilizado para ejecutar una serie de comandos *n* veces.
  - Sintaxis: **for (condición) { acción }**
  - Ejemplo: for (i in 1:10) { x = x + 1 }
  - Ejemplo: for (i in 1: length(vect)) { x = x - 1 }
- While: utilizado para ejecutar una serie de comandos, mientras la condición sea cierta.
  - Sintaxis: **while (condición) { acción }**
  - Ejemplo: while (i < 10) { x = x*2 ; i= i + 1 }

Loops - Estructuras de control

- For: utilizado para ejecutar una serie de comandos *n* veces.
  - Sintaxis: **for (condición) { acción }**
  - Ejemplo: for (i in 1:10) { x = x + 1 }
  - Ejemplo: for (i in 1: length(vect)) { x = x - 1 }
- While: utilizado para ejecutar una serie de comandos, mientras la condición sea cierta.
  - Sintaxis: **while (condición) { acción }**
  - Ejemplo: while (i < 10) { x = x*2 ; i= i + 1 }
- if: utilizado para ejecutar una serie de comandos, si la condición es cierta.
  - Sintaxis: **if (condición1) { acción1 } else { acción2 }**

Loops - Estructuras de control

- For: utilizado para ejecutar una serie de comandos *n* veces.
  - Sintaxis: **for (condición) { acción }**
  - Ejemplo: for (i in 1:10) { x = x + 1 }
  - Ejemplo: for (i in 1: length(vect)) { x = x - 1 }
- While: utilizado para ejecutar una serie de comandos, mientras la condición sea cierta.
  - Sintaxis: **while (condición) { acción }**
  - Ejemplo: while (i < 10) { x = x*2 ; i= i + 1 }
- if: utilizado para ejecutar una serie de comandos, si la condición es cierta.
  - Sintaxis: **if (condición1) { acción1 } else { acción2 }**
  - Ejemplo: if (x < 1500) { x = x*2 }
    else { x = x * 2 /3 }

Escribir un modelo en R

¿Qué datos necesitamos?

Para escribir un modelo, necesitamos conocer:

- Parámetros del modelo.
- Variables del modelo.
- Manera en que se relacionan los parámetros y las variables (ecuaciones).
- Valores iniciales de las variables.

ejemplo 1: Ecuación logística

Esta ecuación la podemos escribir de la siguiente manera:

$$q_{t+1} = Aq_t(1 - q_t)$$

Entonces, para graficar la curva:

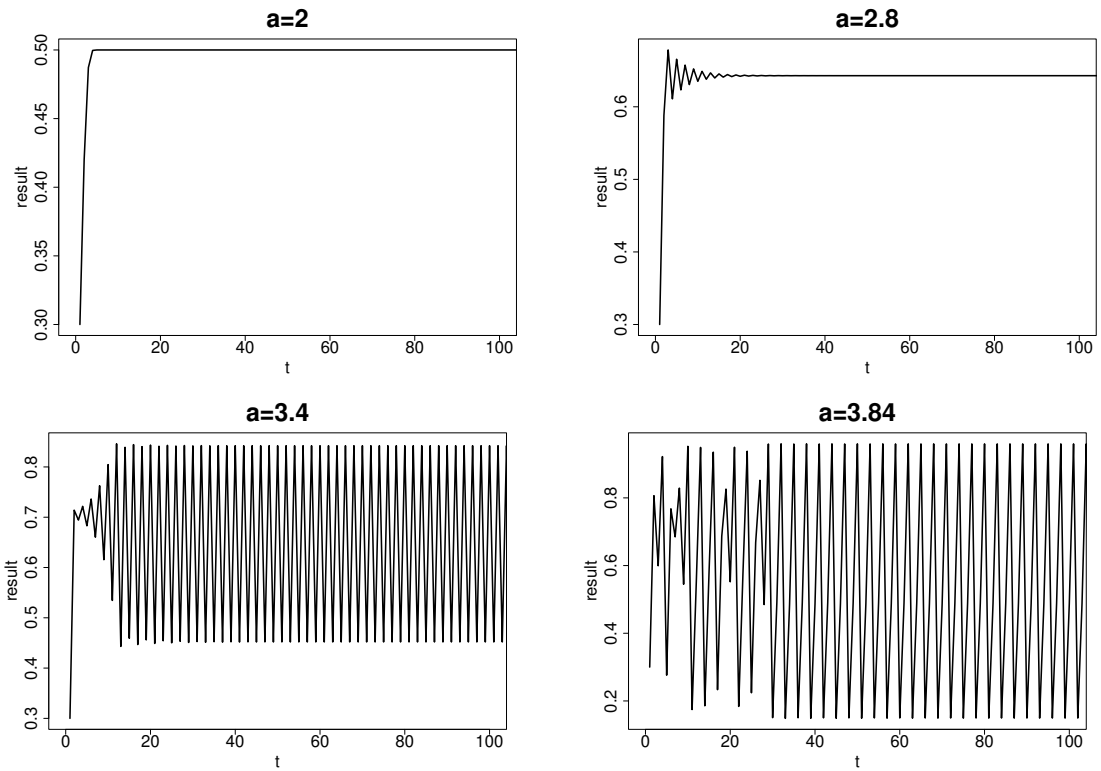
- dar un valor al parámetro A (comenzar con `A = 2`)
- generar una secuencia de valores posibles para q_t
  - sugerencia: `q=seq(0,1,0.001)` -
- generar un valor de la función logística para cada valor de q_t:
  `logist[i]=A*q[i](1-q[i])`

Ahora, para calcular el resultado a partir de un valor inicial:

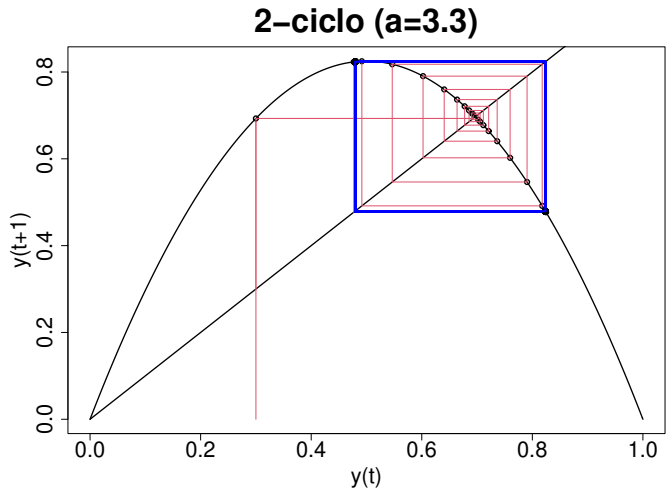
- creamos un vector de resultados, con 500 períodos:
  `result=vector("numeric",500)`
- para el primer período, le imputamos un valor inicial igual a 0.3
  `result[1]=0.3`
- escribimos el loop, partiendo de la condición inicial
  `for (i in 2:length(result))`
  `{result[i]=.....................................}`

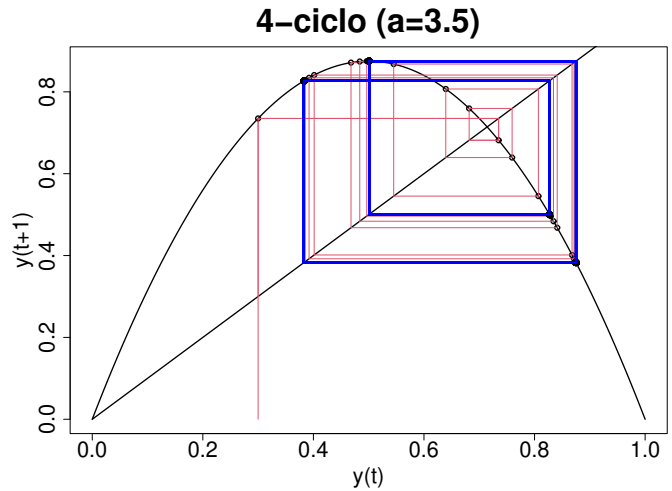
Para graficar

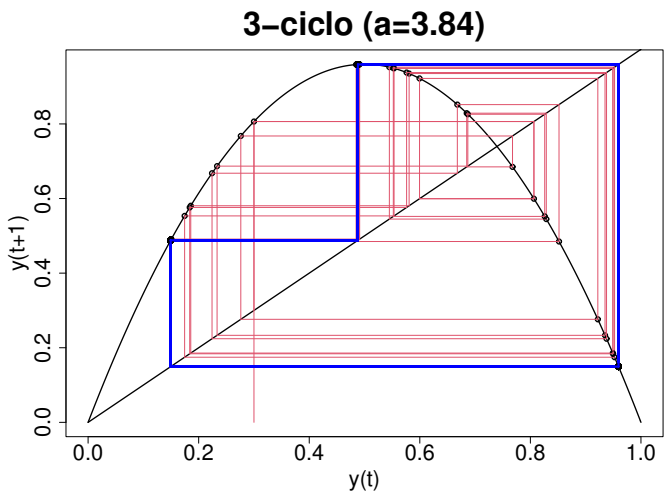
- Utilizamos funciones: plot, lines, segments, points
- Al graficar **logist** y **q** respecto a **q**, tenemos el gráfico habitual de la ecuación logística.
- Ahora, utilizamos las funciones **segments** y **points** para representar la dinámica, a partir del valor del parámetro y de un valor inicial.



ciclos


2–ciclo (a=3.3)
y(t+1)
y(t)
0.0
0.2
0.4
0.6
0.8
1.0


4–ciclo (a=3.5)
y(t+1)
y(t)
0.0
0.2
0.4
0.6
0.8
1.0


3–ciclo (a=3.84)
y(t+1)
y(t)
0.0
0.2
0.4
0.6
0.8
1.0

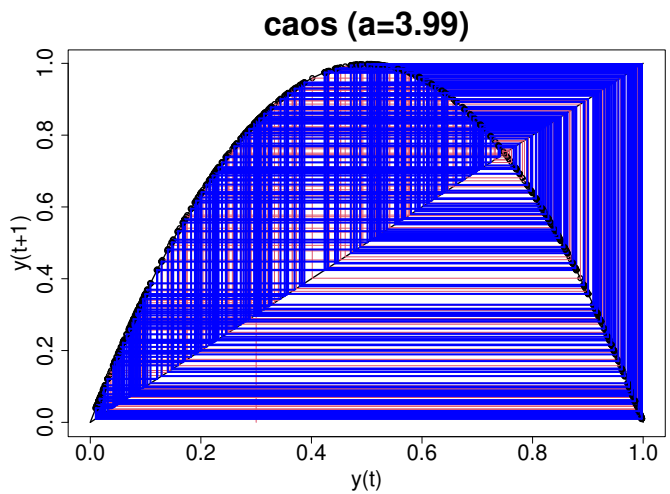

caos (a=3.99)
y(t+1)
y(t)
0.0
0.2
0.4
0.6
0.8
1.0

Diagrama de bifurcación