



Universidad Nacional Autónoma de México  
Facultad de Ciencias



Defensa de tesis de apoyo a la docencia

---

# Introducción a la teoría de juegos no cooperativos rectangulares finitos con $R$

---

Asesora:  
Act. Claudia Villegas Azcorra

Presenta:  
Portilla Ramírez Hugo Iván

Octubre de 2019

# Teoría de juegos

La *teoría de juegos* es la rama matemática que estudia los juegos surgida en la década de los 40.

Un *juego* es un modelo simplificado de un conflicto en donde cada participante debe tomar decisiones tratando de conseguir la mayor utilidad posible.



# Elementos de un juego

Definición 1.1.2.1. Sean:

- $N$  el conjunto de jugadores,
- $D_j$  el conjunto de estrategias puras del jugador  $j$ ,
- y  $\varphi_j$  la función de pago del jugador  $j$ , entonces

Estrategias

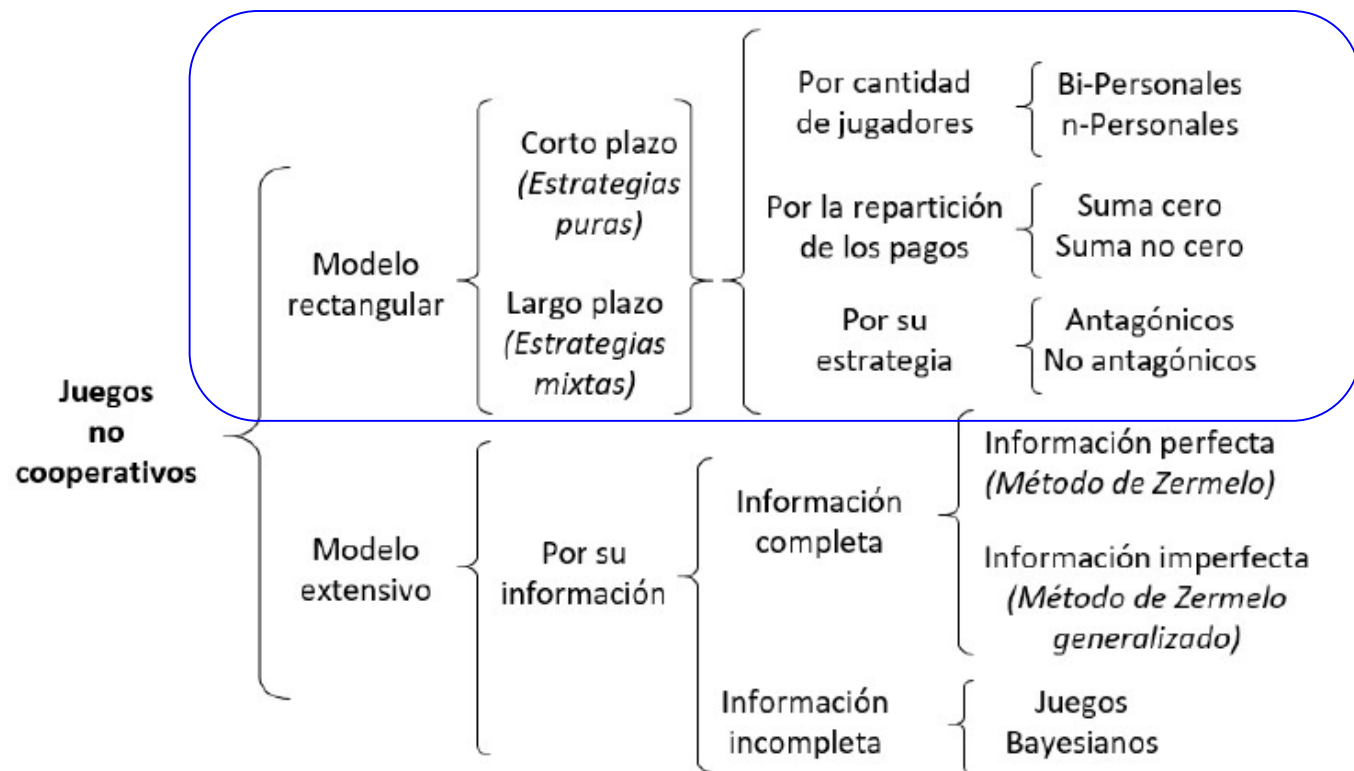
$$(N, \{D_j\}_{j \in N}, \{\varphi_j\}_{j \in N})$$

---

Jugadores

Función  
de pago

# Tipos de juegos



# Antecedentes

## Curso impartido de la facultad



Universidad Nacional Autónoma de México

Asignaturas de Actuaría (plan 2006)

Facultad de Ciencias

Teoría de Juegos en Economía

Optativas

Clave 1056, 10 créditos. Tipo Normal.

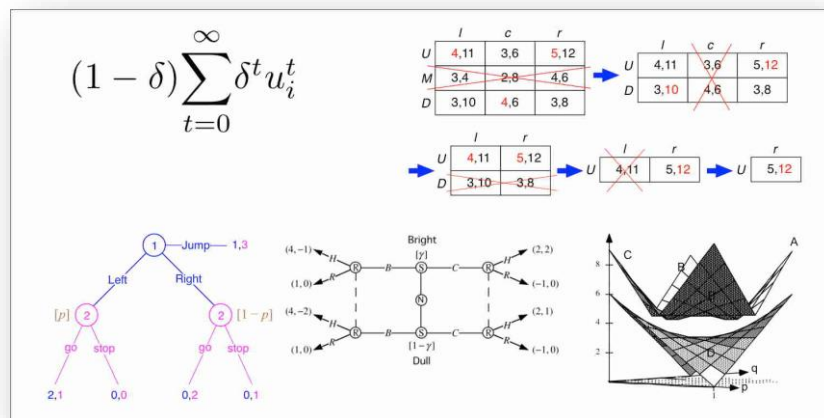
Descargar temario en formato PDF

Resumen

Comprenderá y será capaz de utilizar los conceptos básicos de la teoría de juegos como análisis en términos matemáticos de los conflictos sociales.

\* Desde finales de los 80

## Métodos y algoritmos

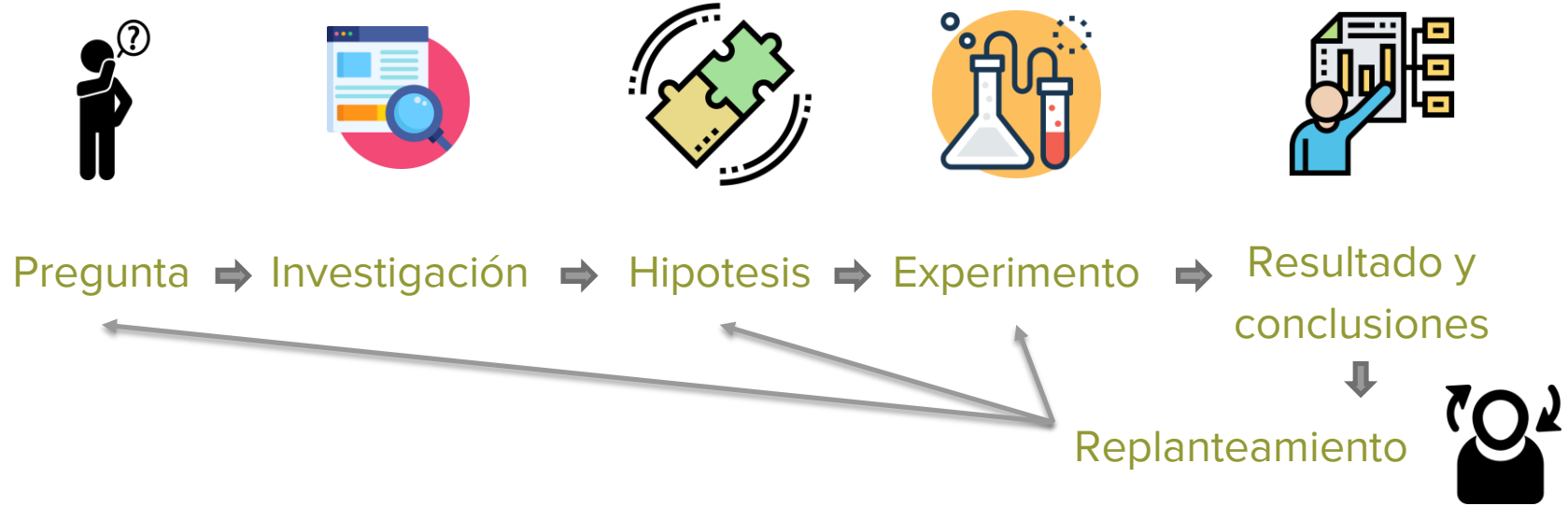


## Lenguaje de programación R



# ¿Cómo se resuelve un juego?

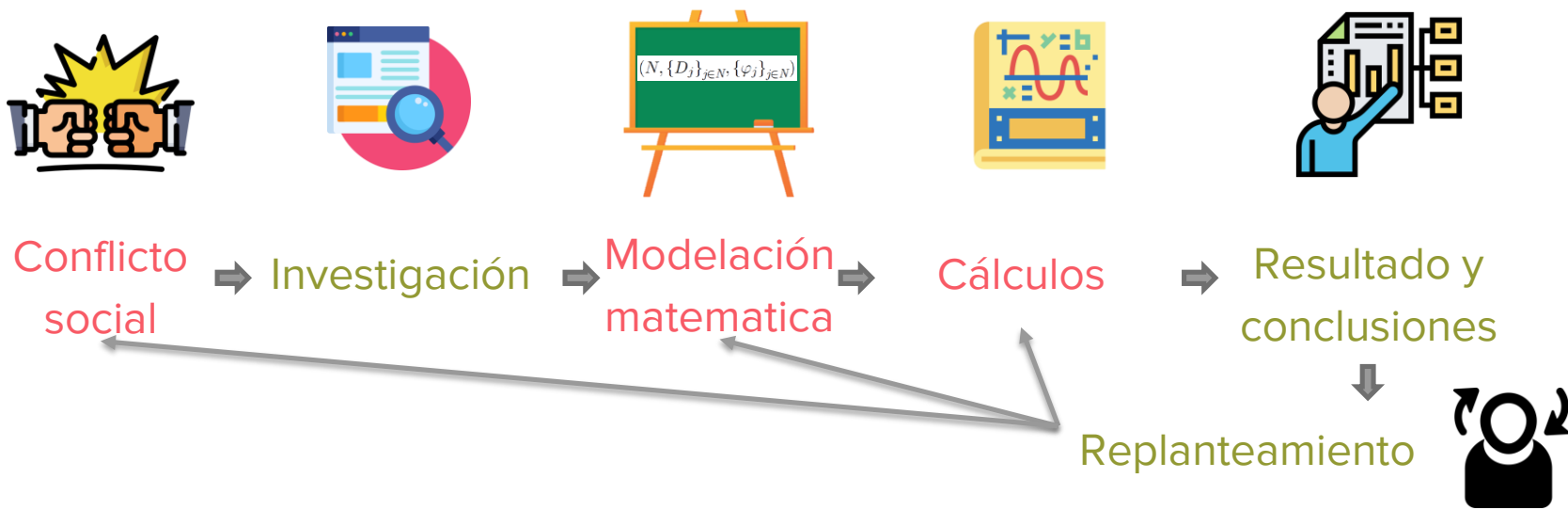
Basado en el método científico \*



\* Modelo simplificado del método científico

# ¿Cómo se resuelve un juego?

En particular



# ¿Cómo se resuelve un juego?

En particular



Cálculos



Resultado y conclusiones



Replanteamiento



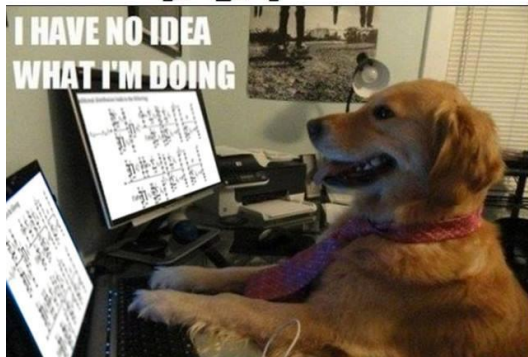


# ¿Cómo se resuelve un juego?

## En particular

Then, the unconditional distribution leads to the following:

$$p(y_{i1}, y_{i2}, \dots, y_{iJ}) = \frac{\prod_{j=1}^J \sum_{\lambda_{ij}} \frac{\Gamma(\theta) \sum_{\lambda_{ij}} \lambda_{ij}^{\sum_{j=1}^J y_{ij}}}{\Gamma(\theta) \sum_{\lambda_{ij}} \lambda_{ij} + \theta}}{\prod_{j=1}^J \sum_{\lambda_{ij}} \frac{\Gamma(\theta) \sum_{\lambda_{ij}} \lambda_{ij}^{\sum_{j=1}^J y_{ij}}}{\Gamma(\theta) \sum_{\lambda_{ij}} \lambda_{ij} + \theta}}$$



Cálculos

Resultado y conclusiones



Replanteamiento



# ¿Cómo se resuelve un juego?

## En particular

Then, the unconditional distribution leads to the following:

$$p(y_{i1}, y_{i2}, \dots, y_{iJ}) = \frac{\prod_{j=1}^J \sum_{\lambda_{ij}} \Gamma(\theta) \lambda_{ij}^{\sum_{i=1}^I y_{ij}}}{\prod_{j=1}^J \Gamma(\theta) \sum_{\lambda_{ij}} \lambda_{ij}^{\sum_{i=1}^I y_{ij}}}$$



Cálculos

Resultado y conclusiones



Replanteamiento



# Estado del arte

Oportunidad ➡ Motivación ➡ Medios ➡ ¿Solución?

- Escasez de estas
    - AxlRod (Knight, 2017). RGameTheory (Cano-Berlanga, 2017)
  - Cubren problemas particulares
  - Sustento avanzado
  - Sin semántica común
- ➡ No existe una solución suficiente

# Objetivo general

Aportar a la docencia didáctica de los principales conceptos y técnicas de los juegos no cooperativos rectangulares finitos con el uso de las tecnologías de la información y cómputo, a través del desarrollo y presentación de un conjunto de códigos que permitan experimentar y dar solución ágil a problemas asociados a la materia que por la vía tradicional, resultarían muy laboriosos de realizar.



# Objetivos particulares



- Crear un código para cada algoritmo
- Presentar texto que explique su uso
- Poner a disposición publica los códigos y el texto
- Motivar la experimentación con estas herramientas

# Disponibilidad de los materiales



@h hugoportillar22 Add files via upload	
1MarcoTeorico	Rename MarcoTeorico/FormaNormal.R
2Dominancias	Add files via upload
3EquilibrioDeNash	Add files via upload
4ECyMA	Add files via upload
5Antagonismos	Add files via upload
6PuntoSilla	Add files via upload
RecopilacionJuegos	Delete dummy.txt
.gitignore	Initial commit
Hugo_Portilla-Teoria_de_Juegos_C...	Add files via upload

Materiales (códigos y texto) disponibles en Github:

[github/hugoportillar22/GameTheory](https://github.com/hugoportillar22/GameTheory)

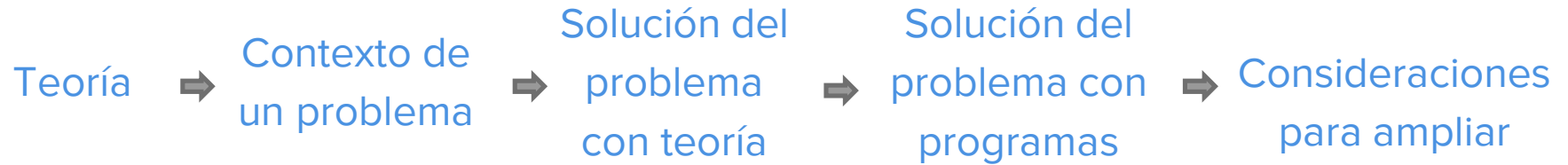
## ESTRATEGIAS PURAS

- Equilibrio de Nash
- ECyMA
- Antagonismo
- Dominancias
- Punto Silla

## ESTRATEGIAS MIXTAS

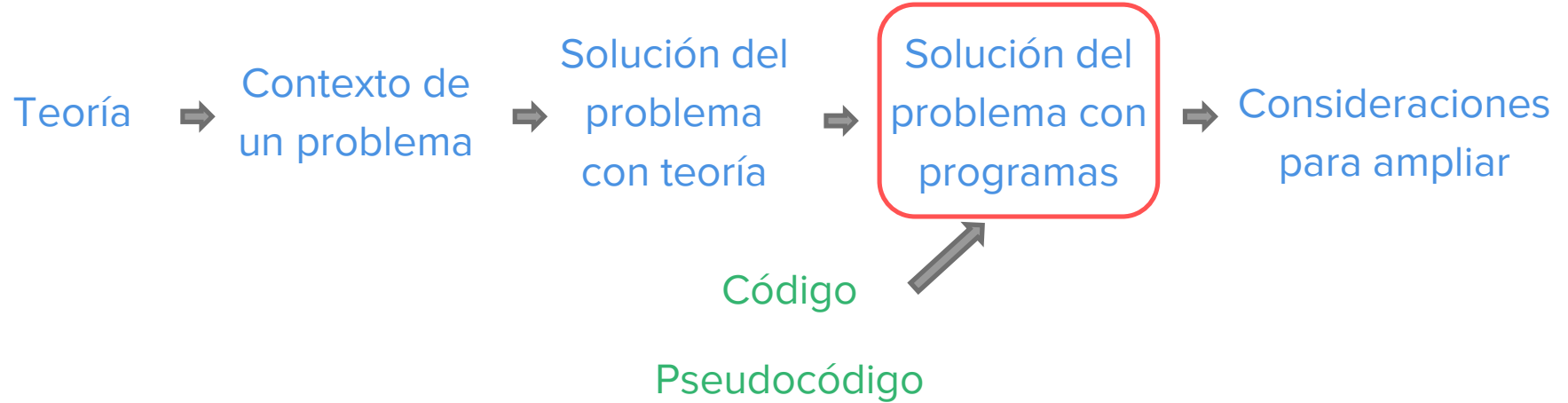
- Equilibrio de Nash
  - Juego Ficticio
  - Reajuste de Nash
- ECyMA
  - Método algebraico
- Antagonismo

# ¿Cómo se abordan los temas?



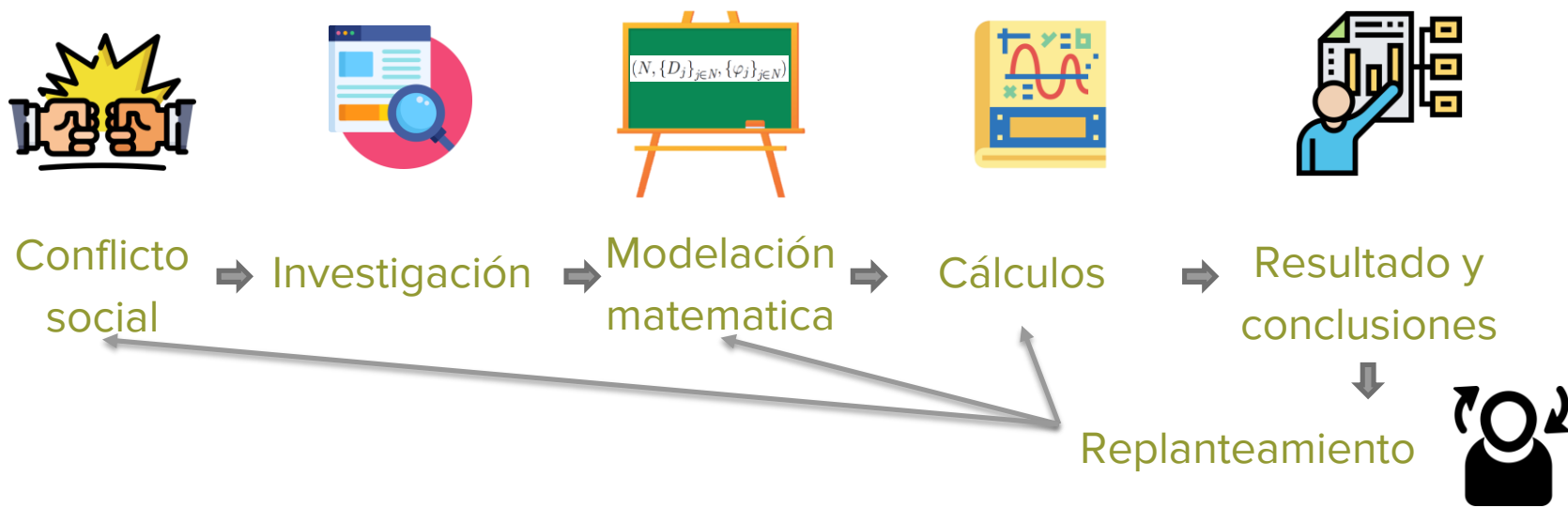


# ¿Cómo se abordan los temas?



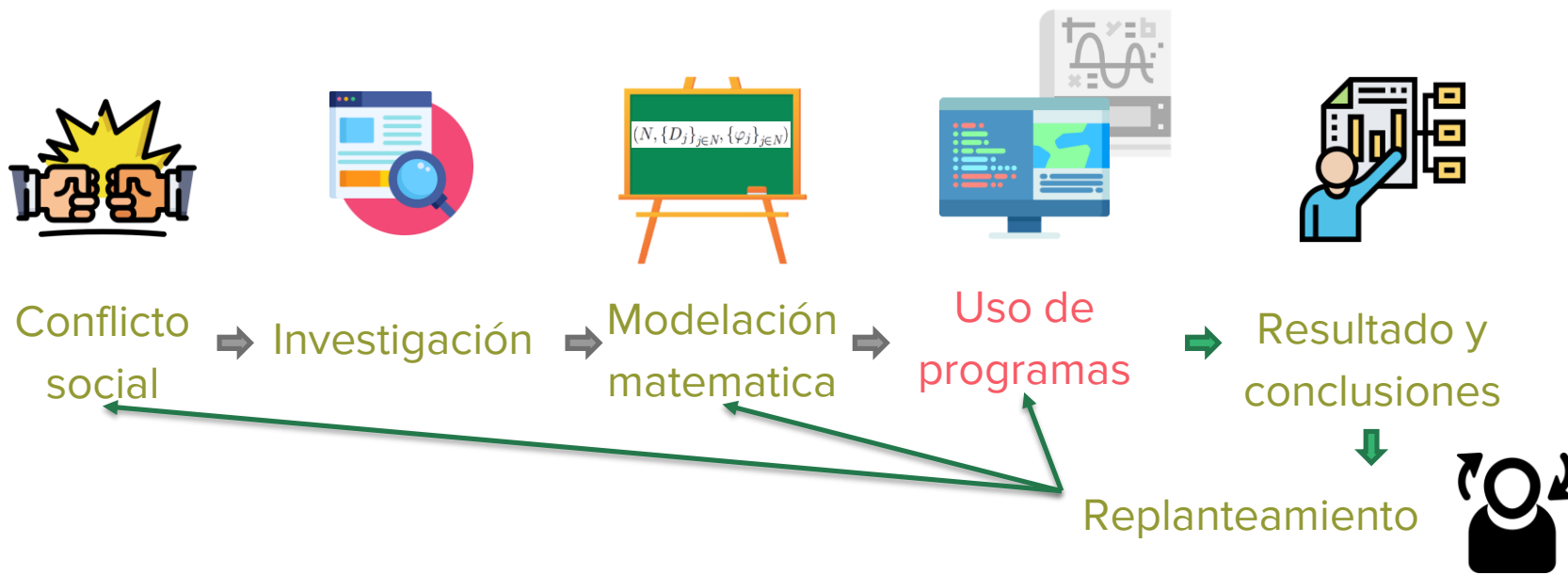
# Uso de los programas

## Proceso clásico

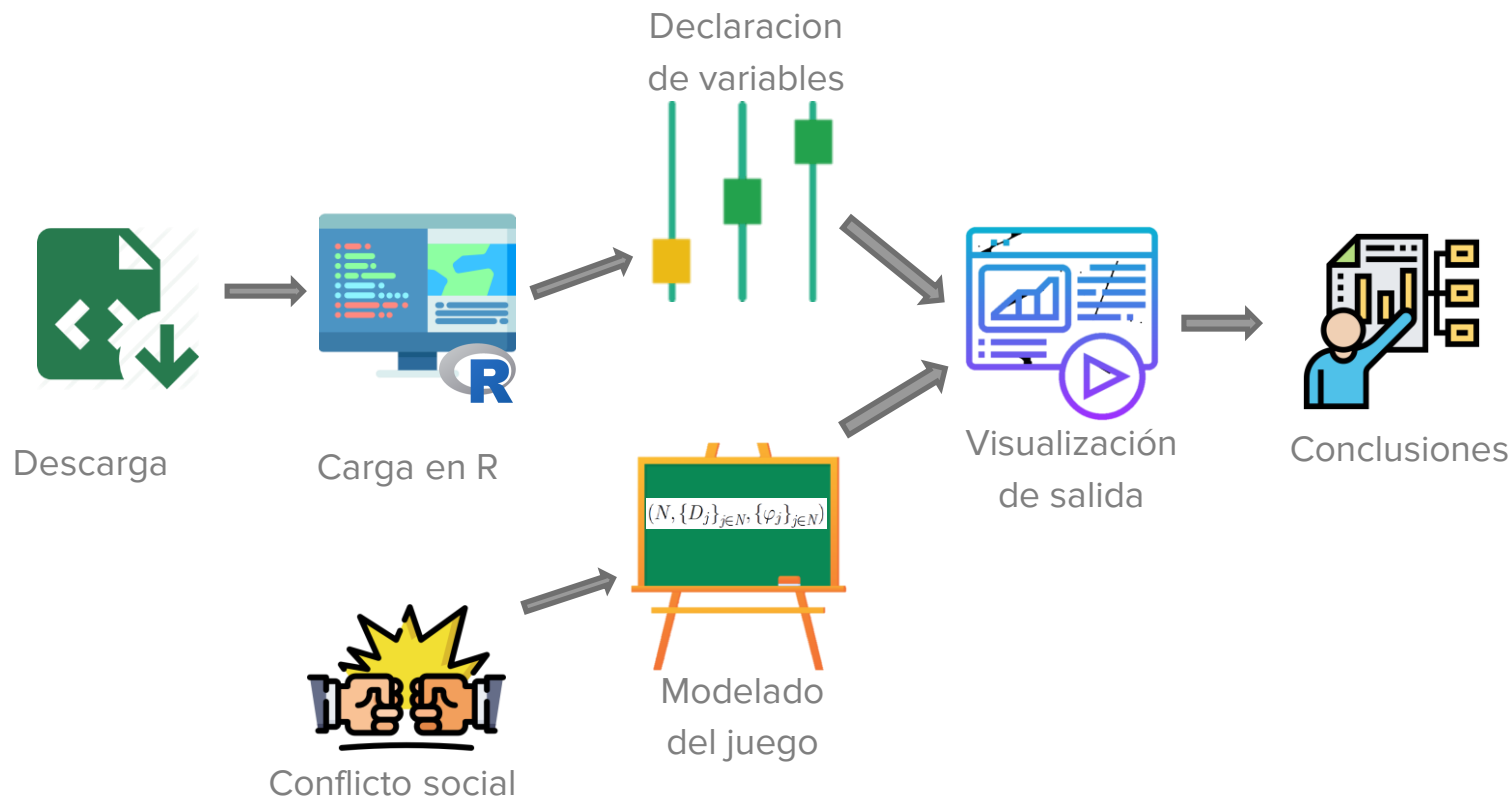


# Uso de los programas

## Modificando el proceso clásico



# Uso de los programas



# Uso de los programas

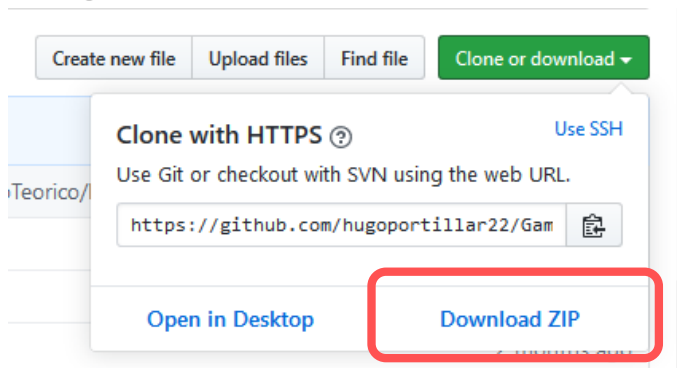
## Descarga de los códigos



@h hugoportillar22 Add files via upload		
1MarcoTeorico		Rename MarcoTeorico/FormaNormal.R
2Dominancias		Add files via upload
3EquilibrioDeNash		Add files via upload
4ECyMA		Add files via upload
5Antagonismos		Add files via upload
6PuntoSilla		Add files via upload
RecopilacionJuegos		Delete dummy.txt
.gitignore		Initial commit
Hugo_Portilla-Teoria_de_Juegos_C...		Add files via upload

Imagen solo con fines visuales

- Descarga directa



- Instrucción descarga Github

Git clone [git://github.com/hugoportillar22/GameTheory](https://github.com/hugoportillar22/GameTheory)

# Uso de los programas

## Carga en R



A.10. CARGA DE ARCHIVOS DE CÓDIGOS A R

115

### A.10. Carga de archivos de códigos a R

Un archivo de código, también conocido como "script", es un archivo de texto que contiene un código escrito como si se fuera a introducir directamente en la línea de comandos (Véase [B10](#)).

Esto permite que se pueda guardar en uno o varios archivos diferentes las declaraciones de funciones u objetos que se usarán para trabajar, sin tener la necesidad de copiar y pegar código, o peor, de reescribirlo en la línea de comandos.

Un archivo de código se puede cargar de diferentes formas dependiendo su origen. Para cargar archivos que se encuentren en el disco de la computadora donde se está usando R se debe conocer su ubicación en directorio, por ejemplo, si el archivo es "Dominancias.R" y se encuentra en la carpeta "Mis Documentos", un ejemplo de la ubicación sería:

```
C:/MisDocumentos/Dominancias.R
```

Y para cargar ese archivo se suele usar, aunque existen otras funciones similares, la función `source()` de la siguiente forma:

```
source("C:/MisDocumentos/Dominancias.R")
```

Con ello, ya podremos ocupar las funciones, variables u objetos que contiene el script "Dominancias.R".

Para saber más del uso y carga de archivos de código en R se deja el siguiente enlace a la [página oficial de R](#) donde se explica a detalle.

[https://cran.r-project.org/doc/contrib/Lemon-kickstart/kr\\_script.html](https://cran.r-project.org/doc/contrib/Lemon-kickstart/kr_script.html)

- Dirección del script

C: /MisDocumentos/j\_reajusteNash.R

- Código para cargarlo

source( "C: /MisDocumentos/j\_reajusteNash.R" )

Imagen solo con fines visuales

# Uso de los programas

## Carga en R



### 3.4.4. Solución del problema con programación

Antes de empezar se debe cargar el script de código que se encuentra en **C.3.4**. Por ejemplo, si el script se encuentra en la siguiente ubicación en la computadora:

```
1 "C:/MisDocumentos/reajusteNash.R"
```

entonces, el script deberá cargarse escribiendo en la línea de comandos de R lo siguiente:

```
1 source("C:/MisDocumentos/reajusteNash.R")
```

Además, como algunas de las funcionalidades de la librería “Tidyverse” respaldan el proceso interno de las funciones que se ocupan mas adelante , se debe cargar esta librería con el siguiente comando:

```
1 library("tidyverse")
```

También es importante mencionar que para este script se ocupan algunas de las funcionalidades de los otros scripts que se ocupan anteriormente en este capítulo. Se deben cargar también.

- Cargar todos los scripts necesarios:

Juego, perfil algoritmo y soportes

- Cargar librerías

library(“tidyverse”)

Imagen solo con fines visuales

# Uso de los programas

## Ejecución con variables



### Entrada

Recordando que el programa fue diseñado para retornar la lista de estrategias mixtas reajustadas para cada tiempo, especificamos que se deberá iterar un máximo de 20 veces si no se encuentra un punto fijo antes. Se usa la función *reajusteNash*, con el argumento tipo data frame “ej\_reajusteNash” y el argumento de tipo data frame “X\_reajusteNash” para los parámetros “juego” y “perfil de estrategias”, respectivamente. El tercer parámetro es especial, especifica que el algoritmo debe iterar un máximo de 20 veces.

```
1 > reajusteNash(ej_reajusteNash , X_reajusteNash , 20)
```

Imagen solo con fines visuales

- Se llama la función, y entre paréntesis sus argumentos (variables)

*reajusteNash( argumentos )*

- Se explica que variables ocupa cada programa, se especifican

Juego = ej\_reajusteNash ,

Perfil = X\_reajusteNash ,

Iteraciones = 20



# Uso de los programas

## Visualización de salida



### Salida

Se obtiene la tabla visual del juego con Reajuste Nash. Cada columna cuyo nombre empieza con t, representa los pesos de las estrategias en ese tiempo. Como se puede observar, este juego no cae en ningún ciclo, ni en un estado absorbente, pero se observa una convergencia gradual hacia las estrategias “B” y “d”, para los jugadores 1 y 2 respectivamente,

- Se explica como leer la salida.
- La salida siempre expresa la estrategia “elegida” por el algoritmo.

Imagen solo con fines visuales

1	X p	t1	t2	t3	t4
2	A 1	0.3333333	0.2	0.1428571	0.1111111
3	B 0	0.6666667	0.8	0.8571429	0.8888889
4	c 1	1.0000000	1.0	1.0000000	1.0000000
5	d 0	0.0000000	0.0	0.0000000	0.0000000
6	X p	t5	t6	t7	t8
7	A 1	0.0909090	0.0769230	0.0666667	0.0588235
8	B 0	0.9090909	0.9230769	0.9333333	0.9411764
9	c 1	0.9000000	0.7226277	0.5654657	0.4474031
10	d 0	0.1000000	0.2773722	0.4345342	0.5525968
11	X p	t9	t10	t11	t12
12	A 1	0.0526315	0.0476190	0.0434782	0.0400000
13	B 0	0.9473684	0.9523809	0.9565217	0.9600000
14	c 1	0.3617248	0.2990896	0.2523642	0.2166988
15	d 0	0.6382751	0.7009103	0.7476358	0.7833012
16	X p	t13	t14	t15	t16
17	A 1	0.0370370	0.0344827	0.0322580	0.0303030
18	B 0	0.9629629	0.9655172	0.9677419	0.9696969
19	c 1	0.1888680	0.1667109	0.1487532	0.1339678
20	d 0	0.8111319	0.8332890	0.8512467	0.8660321
21	X p	t17	t18	t19	t20
22	A 1	0.0285714	0.0270270	0.0256410	0.0243902
23	B 0	0.9714285	0.9729729	0.9743589	0.9756097
24	c 1	0.1216241	0.1111916	0.1022780	0.0945882
25	d 0	0.8783758	0.8888083	0.8977219	0.9054117
26					

## ¿Qué nos ahorramos ?



### 3.4.3. Solución del problema en forma teórica

Para este problema, por la complejidad que implica, se pr durante 3 iteraciones.

**Iteración 1** Empezamos con  $X_0 = [(1,0)(1,0)]$

- Se obtiene  $E_j(X_0)$  para el jugador 1.

$$E_1[(1,0)(1,0)] = (1)(1)(2) + (1)(0)(3) + (0)(1)(4) + (0)(0)(5) = 2 + 0 + 0 + 0 = 2$$

- Se obtiene  $c_j^1(X_0)$  para la estrategia A del jugador 1.

$$c_{1-A}^1(X_0) = \max\{E_1[(1,0)(1,0)] - E_1(X_0), 0\} = c_{1-A}^1(X_0) = \max\{E_1[(1,0)(1,0)] - E_1(X_0), 0\} = c_{1-A}^1(X_0) = \max\{0, 0\} = 0$$

- Se obtiene  $c_j^2(X_0)$  para la estrategia B del jugador 1.

$$c_{1-B}^2(X_0) = \max\{E_1[(1,0)(1,0)] - E_1(X_0), 0\}$$

56

CAPÍTULO 3. EQUILIBRIO DE NASH

Donde

$$E_1[(0,1)(1,0)] = (0)(1)(2) + (0)(0)(3) + (1)(1)(4) + (1)(0)(5) = 4$$

Por lo que

$$c_{1-B}^2(X_0) = \max\{4 - 2, 0\} = 2$$

- Se obtiene  $E_j(X_0)$  para el jugador 2.

$$E_2[(1,0)(1,0)] = (1)(1)(4) + (1)(0)(-3) + (0)(1)(1) + (0)(0)(4 + 0 + 0 + 0 = 2$$

- Se obtiene  $c_j^1(X_0)$  para la estrategia c del jugador 2.

$$c_{2-c}^1(X_0) = \max\{E_2[(1,0)(1,0)] - E_2(X_0), 0\}$$

$$c_{2-c}^1(X_0) = \max\{E_2(X_0) - E_2(X_0), 0\}$$

- Se obtiene  $c_j^2(X_0)$  para la estrategia d del jugador 2

$$c_{2-d}^2(X_0) = \max\{E_2[(1,0)(1,0)] - E_2(X_0), 0\}$$

Donde

$$E_2[(1,0)(0,1)] = (1)(0)(4) + (1)(1)(-3) + (0)(0)(1) +$$

Por lo que

$$c_{2-d}^2(X_0) = \max\{-3 - 2, 0\} = 0$$

- Se obtiene  $\sum_{\sigma \in D_1} c_j^1(X_0)$  para el jugador 1.

$$\sum_{\sigma \in D_1} c_j^1(X_0) = c_{1-A}^1(X_0) + c_{1-B}^1(X_0) = 0 + 0 = 0$$

- Se obtiene  $\sum_{\sigma \in D_1} c_j^2(X_0)$  para el jugador 2.

$$\sum_{\sigma \in D_1} c_j^2(X_0) = c_{2-c}^2(X_0) + c_{2-d}^2(X_0) = 0 + 0 = 0$$

57

CAPÍTULO 3. EQUILIBRIO DE NASH

Donde

$$E_2[(1/3, 2/3)(1,0)] = (1/3)(1)(2) + (1/3)(0)(3) + (2/3)(1)(4) + (2/3)(0)(5) = 2/3 + 0 + 8/3 + 0 = 10/3$$

- Se obtiene  $c_j^1(X_1)$  para la estrategia A del jugador 1.

$$c_{1-A}^1(X_1) = \max\{E_1[(1,0)(1,0)] - E_1(X_1), 0\}$$

$$c_{1-A}^1(X_1) = \max\{2 - 10/3, 0\} = 0$$

58

CAPÍTULO 3. EQUILIBRIO DE NASH

- Se obtiene  $c_j^2(X_1)$  para la estrategia B del jugador 1.

$$c_{1-B}^2(X_1) = \max\{E_1[(0,1)(1,0)] - E_1(X_1), 0\}$$

Donde

$$E_1[(0,1)(1,0)] = (0)(1)(2) + (0)(0)(3) + (1)(1)(4) + (1)(0)(5) = 4$$

Por lo que

$$c_{1-B}^2(X_1) = \max\{4 - (10/3), 0\} = 2/3$$

- Se obtiene  $E_j(X_1)$  para el jugador 2.

$$E_2[(1/3, 2/3)(1,0)] = (1/3)(1)(4) + (1/3)(0)(-3) + (2/3)(1)(1) + (2/3)(0)(2) = 4/3 + 0 + 2/3 + 0 = 2$$

- Se obtiene  $c_j^1(X_1)$  para la estrategia c del jugador 2.

$$c_{2-c}^1(X_1) = \max\{E_2[(1/3, 2/3)(1,0)] - E_2(X_1), 0\}$$

$$c_{2-c}^1(X_1) = \max\{E_2(X_1) - E_2(X_1), 0\}$$

- Se obtiene  $c_j^2(X_1)$  para la estrategia d del jugador 2.

$$c_{2-d}^2(X_1) = \max\{E_2[(1/3, 2/3)(1,0)] - E_2(X_1), 0\}$$

Donde

$$E_2[(1/3, 2/3)(0,1)] = (1/3)(0)(4) + (1/3)(1)(-3) + (2/3)(0)(1) + (2/3)(1)(2) = -3/3 + 4/3 = 1/3$$

Por lo que

$$c_{2-d}^2(X_1) = \max\{1/3 - 2, 0\} = 0$$

- Se obtiene  $\sum_{\sigma \in D_1} c_j^1(X_1)$  para el jugador 1.

$$\sum_{\sigma \in D_1} c_j^1(X_1) = c_{1-A}^1(X_1) + c_{1-B}^1(X_1) = 0 + 2/3 = 2/3$$

### 3.4. FUNCIÓN DE REAJUSTE DE NASH

61

- Se obtiene  $\sum_{\sigma \in D_1} c_j^1(X_2)$  para el jugador 1.

$$\sum_{\sigma \in D_1} c_j^1(X_2) = c_{1-A}^1(X_2) + c_{1-B}^1(X_2) = 0 + 2/5 = 2/5$$

- Se obtiene  $\sum_{\sigma \in D_1} c_j^2(X_2)$  para el jugador 2.

$$\sum_{\sigma \in D_2} c_j^2(X_2) = c_{2-c}^2(X_2) + c_{2-d}^2(X_2) = 0 + 0 = 0$$

- Se obtiene  $\hat{x}_{\sigma_1}^j$  para la estrategia A del jugador 1.

$$\hat{x}_{\sigma_1-A}^1 = \frac{x_{\sigma_1-A}^1 + c_{\sigma_1-A}^1(X_2)}{1 + \sum_{\sigma \in D_1} c_{\sigma_1}^1(X_2)} = \frac{1/5 + 0}{1 + 2/5} = 1/5 \approx 0.1429$$

- Se obtiene  $\hat{x}_{\sigma_1}^j$  para la estrategia B del jugador 1.

$$\hat{x}_{\sigma_1-B}^1 = \frac{x_{\sigma_1-B}^1 + c_{\sigma_1-B}^1(X_2)}{1 + \sum_{\sigma \in D_1} c_{\sigma_1}^1(X_2)} = \frac{4/5 + 2/5}{1 + 2/5} = 6/7 \approx 0.8571$$

- Se obtiene  $\hat{x}_{\sigma_2}^j$  para la estrategia c del jugador 2.

$$\hat{x}_{\sigma_2-c}^2 = \frac{x_{\sigma_2-c}^2 + c_{\sigma_2-c}^2(X_2)}{1 + \sum_{\sigma \in D_2} c_{\sigma_2}^2(X_2)} = \frac{1 + 0}{1 + 0} = 1$$

- Se obtiene  $\hat{x}_{\sigma_2}^j$  para la estrategia d del jugador 2.

$$\hat{x}_{\sigma_2-d}^2 = \frac{x_{\sigma_2-d}^2 + c_{\sigma_2-d}^2(X_2)}{1 + \sum_{\sigma \in D_2} c_{\sigma_2}^2(X_2)} = \frac{0 + 0}{1 + 0} = 0$$

$$\therefore X_3 = [(1/7, 6/7)(1,0)]$$

Y como  $X_2 \neq X_3$  entonces no se llegó a un punto fijo, por lo que se debe continuar con las siguientes iteraciones necesarias ...



# Uso de los programas

## Replanteamiento



- Función

reajusteNash( *argumentos* )

- Argumentos:

Juego = ej\_reajusteNash ,

Perfil = X\_reajusteNash ,

Iteraciones = 20



- Misma función

reajusteNash( *argumentos* )

- Nuevos argumentos:

Juego = ej\_reajusteNash ,

Perfil = X\_reajusteNash\_2 ,

Iteraciones = 30



Nuevo  
Resultado

## Replanteamiento



- ¿ Nuevos argumentos ?

Juego = ej\_reajusteNash ,

Perfil = X\_reajusteNash\_2 ,

Iteraciones = 30

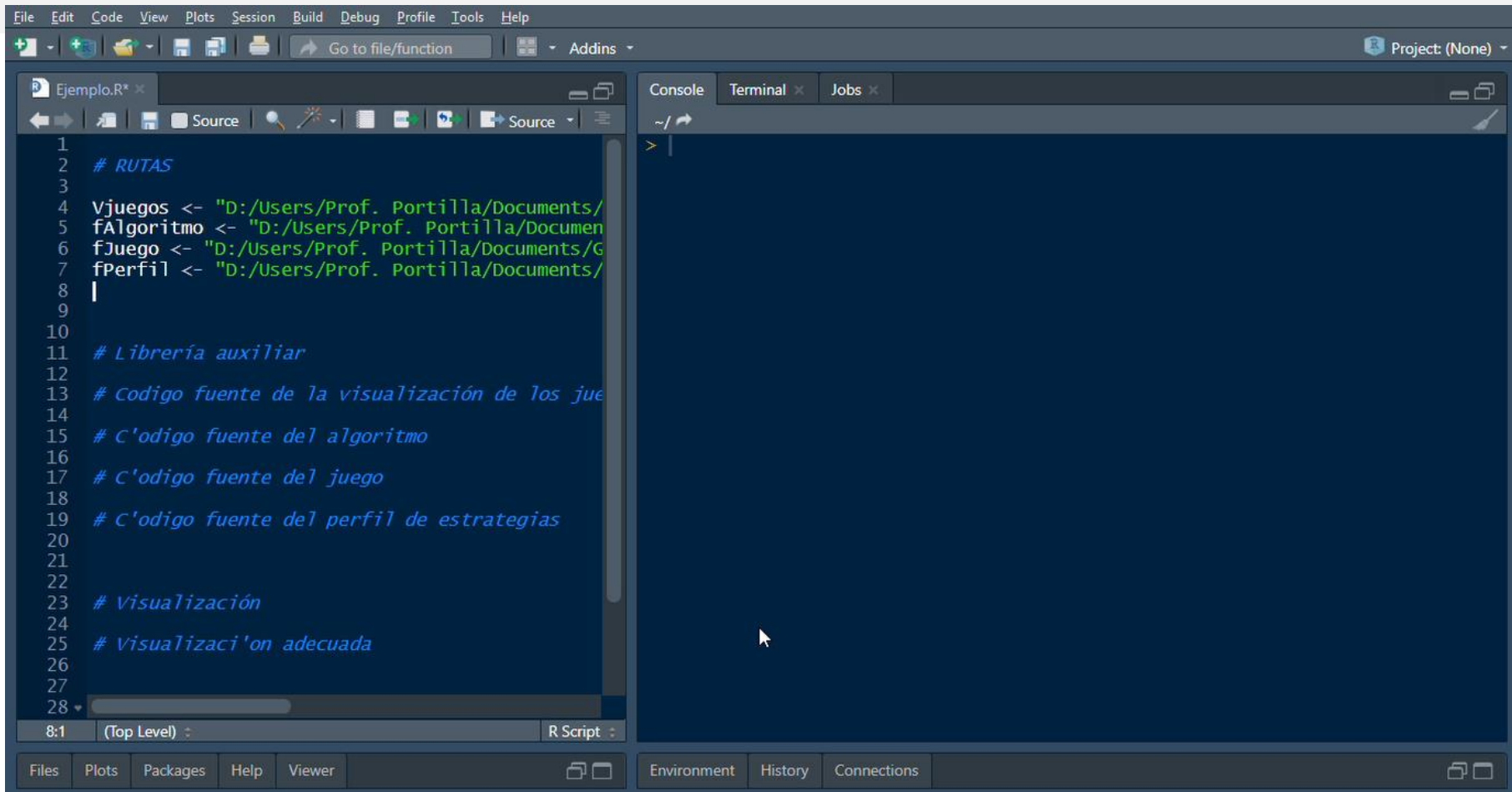
### ➡ 1.2.2. Declaración de juegos en el entorno de R

Para explicar como se declara un juego en R ejemplificaremos con el siguiente conflicto de dos jugadores cuya argumentación y declaración del juego pueden encontrarse en [3.1.2](#) y [D.2](#) respectivamente.

Códigos proveídos y fáciles de editar

```
5 j_reajusteN <- data.frame(matrix(NA, nrow = filas , ncol = j*2) ,  
  stringsAsFactors=FALSE)  
6 names(j_reajusteN) <- c('S1' , 'S2' , 'PHI1' , 'PHI2')  
7 j_reajusteN[1,] <- c('A' , 'c' , 2, 4)  
8 j_reajusteN[2,] <- c('A' , 'd' , 3, -3)  
9 j_reajusteN[3,] <- c('B' , 'c' , 4, 1)  
10 j_reajusteN[4,] <- c('B' , 'd' , 5, 2)  
  
1 filas <- length(pe_j1)+length(pe_j2)  
2 X_reajusteN <- data.frame(matrix(NA, nrow = filas , ncol = 2) ,  
  stringsAsFactors=FALSE)  
3 names(X_reajusteN) <- c('X' , 'p')  
4 X_reajusteN[1,] <- c('A' , 1)  
5 X_reajusteN[2,] <- c('B' , 0)  
6 X_reajusteN[3,] <- c('c' , 1)  
7 X_reajusteN[4,] <- c('d' , 0)
```

# Muestra de uso



The screenshot displays the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a 'Go to file/function' search bar. The main editor window shows a script named 'Ejemplo.R\*' with the following R code:

```
1 # RUTAS
2
3
4 Vjuegos <- "D:/Users/Prof. Portilla/Documents/
5 fAlgoritmo <- "D:/Users/Prof. Portilla/Documen
6 fJuego <- "D:/Users/Prof. Portilla/Documents/G
7 fPerfil <- "D:/Users/Prof. Portilla/Documents/
8 |
9
10
11 # Librería auxiliar
12
13 # Codigo fuente de la visualización de los jue
14
15 # C'odigo fuente del algoritmo
16
17 # C'odigo fuente del juego
18
19 # C'odigo fuente del perfil de estrategias
20
21
22
23 # Visualización
24
25 # Visualizaci'on adecuada
26
27
28
```

The right-hand pane contains the Console, Terminal, and Jobs tabs. The Console tab is active, showing a prompt '> |'. The bottom status bar indicates the current position is '8:1 (Top Level) : R Script :'. The bottom-most bar shows tabs for Files, Plots, Packages, Help, Viewer, Environment, History, and Connections.

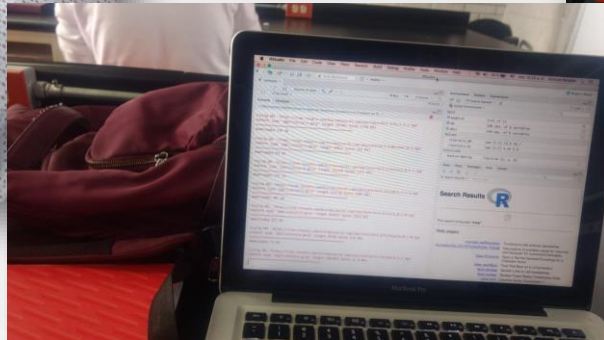
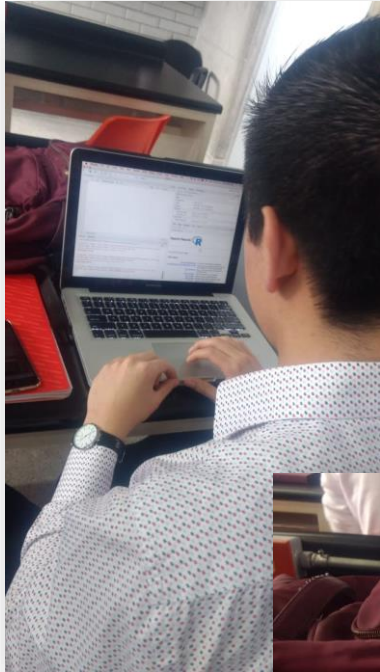
# Conclusiones

- Se presentaron programas para la mayoría de técnicas
- Se creo un texto que explica su uso
- Tanto los programas como el texto están en un github
- Al cierre de cada tema se orienta para continuar experimentando

El máximo fin de esta tesis es aportar a la docencia didáctica de la materia, lo cual se ha hecho con la presentación del material, pero su capacidad aún se está probando y como cualquier proceso de enseñanza, está siendo evaluado y retroalimentado por los estudiantes.



# Conclusiones





# Trabajos futuros

- **Probarse en el salón de clases**
- Ampliar capacidades y temas
- Paquetería e integración
- Ser apoyo para otros trabajos

*Gracias*

---

## Matemáticas

- \* R. Gibbons. *Un primer curso de teoría de juegos*. Antoni Bosch, 1992. España.
- \* D. Morton. *Game Theory: A Nontechnical Introduction*. Dover Publications, 1983. Estados Unidos.
- \* P. Zapata Lillo. *Economía, política y otros juegos: una introducción a los juegos no cooperativos*. Las prensas de Ciencias. UNAM., 1997. México.
- \* Osborne M. y Rubinstein A. *A Course in Game Theory*. The MIT Press, 2011. Estados Unidos.
- \* E. Ventsel. *Elementos de la teoría de juegos*. MIR, 1977. Rusia.
- \* Martínez Hernández A. y Sánchez Mendiola M. *Formación docente en la UNAM: Antecedentes y la voz de su profesorado*. CDIC. México. Páginas 415 - 430.

## Computacionales

- \* V. Knight. *Axelrod python library documentation*. <https://axelrod.readthedocs.io/en/stable/>
- \* M. Canty. *Resolving Conflicts with Mathematica*. Elsevier Academic Press, 2000. Alemania.
- \* J. García Cano, E. y Solano Gálvez. *Guía práctica de estudio: Pseudocódigo*. [http://odin.fi-b.unam.mx/salac/practicasFP/fp\\_p4.pdf](http://odin.fi-b.unam.mx/salac/practicasFP/fp_p4.pdf)
- \* The Nashpy project developers. *A two players equilibria python library*. <https://nashpy.readthedocs.io/en/stable/>
- \* M. Crawley. *The R Book*. Wiley, 2007. Inglaterra.
- \* J. Vilella C. Cano-Berlanga, S. *The R package gametheory*. <https://cran.r-project.org/web/packages/GameTheory/index.html>
- \* A. Carillo Ledesma e I. González Rosas. *Introducción a la programación*. <http://132.248.182.159/Herramientas/Lenguajes/IntroduccionALaProgramacion.pdf>