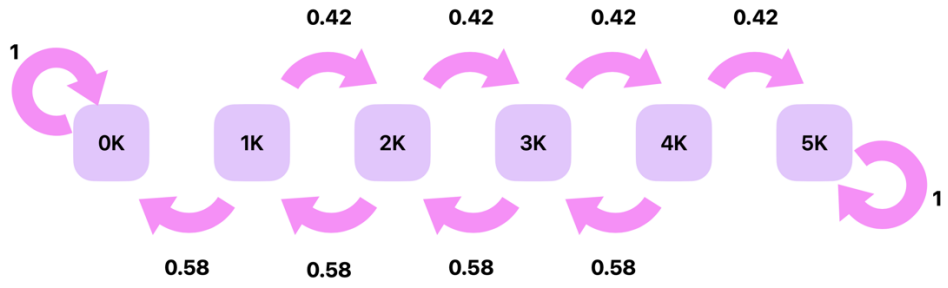
Práctica 2, segundo intento.

Reto 5: ¿Cómo modelar sistemas dinámicos con cadenas de Markov al estilo del algoritmo PageRank de Google?

Paula Corbatón Álvarez

Ejercicio 1:

Dinero antes de jugar	Dinero después de jugar					
	0k	1k	2k	3k	4k	5k
0k	1	0	0	0	0	0
1k	0.58	0	0.42	0	0	0
2k	0	0.58	0	0.42	0	0
3k	0	0	0.58	0	0.42	0
4k	0	0	0	0.58	0	0.42
5k	0	0	0	0	0	1



Ejercicio 2:

Generamos la matriz de transición P:

```
labels<-c("0K","1K","2K","3K","4K","5K")
byRow <- TRUE
x <- c(1, 0, 0, 0, 0, 0, 0.58, 0, 0.42, 0, 0, 0, 0, 0.58, 0, 0.42, 0, 0, 0, 0, 0.58, 0, 0.42, 0, 0, 0, 0, 0.58,
0, 0.42, 0, 0, 0, 0, 0, 1)
P<-matrix(data=x,byrow=byRow,nrow=6,dimnames=list(labels,labels))
P
```

```
##      0K   1K   2K   3K   4K   5K
## 0K 1.00 0.00 0.00 0.00 0.00 0.00
## 1K 0.58 0.00 0.42 0.00 0.00 0.00
## 2K 0.00 0.58 0.00 0.42 0.00 0.00
## 3K 0.00 0.00 0.58 0.00 0.42 0.00
## 4K 0.00 0.00 0.00 0.58 0.00 0.42
## 5K 0.00 0.00 0.00 0.00 0.00 1.00
```

Podemos comprobar que la suma de las probabilidades de cada una de las filas es 1 con el siguiente código:

```
#apply(P, 1, sum)
rowSums(P)
```

```
## 0K 1K 2K 3K 4K 5K
##  1  1  1  1  1  1
```

Podemos calcular la suma de las probabilidades de cada una de las columnas con el siguiente código:

```
colSums(P)
```

```
##   0K   1K   2K   3K   4K   5K
## 1.58 0.58 1.00 1.00 0.42 1.42
```

Ejercicio 3:

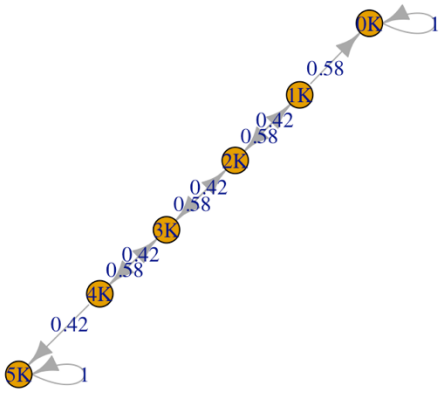
Tal y como indica el ejercicio, creamos la cadena de Markov discreta (mcP) definida por la matriz de transiciones P:

```
library("markovchain")
mcP<-new("markovchain",states=labels,byrow=byRow,transitionMatrix=P,name=" Caesars Palace")
mcP
```

```
##  Caesars Palace
##  A  6 - dimensional discrete Markov Chain defined by the following states:
##  0K, 1K, 2K, 3K, 4K, 5K
##  The transition matrix  (by rows)  is defined as follows:
##      0K   1K   2K   3K   4K   5K
## 0K 1.00 0.00 0.00 0.00 0.00 0.00
## 1K 0.58 0.00 0.42 0.00 0.00 0.00
## 2K 0.00 0.58 0.00 0.42 0.00 0.00
## 3K 0.00 0.00 0.58 0.00 0.42 0.00
## 4K 0.00 0.00 0.00 0.58 0.00 0.42
## 5K 0.00 0.00 0.00 0.00 0.00 1.00
```

Hacemos el diagrama:

```
plot(mcP,package="diagram",cex=.6)
```



Un estado absorvente es aquel que no se puede abandonar (desde ese estado no se puede transitar a ningún otro). En este caso 0K y 5K son estados absorventes porque como el enunciado indica:

> "Esta persona apostará cada vez 1,000 dólares y parará, o bien cuando se quede en bancarrota (0 dólares), o bien cuando consiga 5,000 dólares"

Esto nos indica que si la persona llega a 0 dólares y entra en bancarrota abandonará el juego (estamos seguros al 100% de que no va a ganar más dinero ni a endeudarse puesto que no jugará). Lo mismo sucede si la persona consigue 5K, del mismo modo abandonará la mesa por lo que no tendrá la posibilidad de ganar ni de perder dinero.

La matriz de transición muestra que existe un 100% de probabilidad de permanecer en estos estados una vez que se alcanzan.

Ejercicio 4:

Una distribución estacionaria es una distribución de probabilidad que se mantiene constante a lo largo del tiempo, es decir, una distribución de probabilidades que no cambia después de cada transición en la cadena. En el caso de nuestra cadena, puesto que tiene como estados absorventes 0K y 5K es lógico deducir que estos a su vez estarán asociados a una distribución estacionaria. Por lo tanto habrá dos distribuciones estacionarias, una en 5K (0, 0, 0, 0, 0, 1) y otra en 0K (1, 0, 0, 0, 0, 0).

Podemos calcularlo con R de la siguiente forma:

```
DistEst = steadyStates(mcP)
DistEst
```

```
##      0K 1K 2K 3K 4K 5K
## [1,]  0  0  0  0  0  1
## [2,]  1  0  0  0  0  0
```

Como he mencionado anteriormente y podemos comprobar con el código de R, hay **2 distribuciones estacionarias**.

Calculo de los vectores propios de la matriz P:

```
lambda <- eigen(P)
lambda$values
```

```
## [1]  1.0000000  1.0000000  0.7985944 -0.7985944 -0.3050359  0.3050359
```

Como podemos observar la matriz P tiene **2 vectores propios con valor propio 1**.

Ejercicio 5:

Para calcular el mínimo número natural (al que elevar la matriz P) que satisface que el valor de cada uno de los valores de las columnas asociadas a los estados no absorventes es menor que 10^{-4} podemos utilizar el siguiente programa:

```
n <- 1 #n es el número al que está elevado nuestra matriz
Pn <- P # Pn es donde guardaremos la matriz elevada a la n potencia
while (TRUE) { #hasta que le indiquemos que pare, nuestro "programa" repetirá el siguiente código:
  Pn <- P %*% Pn #multimplica la matriz elevada a la n potencia por la matriz P (esto lo eleva a la n potencia)
  n <- n + 1 #(sumamos 1 a n para seguir la pista de a qué potencia hemos elevado P)
  non_absorb_cols <- which(colSums(P%*%P) < 1) # seleccionamos las columnas asociadas a los estados no absorbente
s (aquellas cuya suma es menor que 1)
  if (all(Pn[,non_absorb_cols] < 0.0001)) { #si todas las columnas asociadas a los estados no absorbentes tienen
un valor menor a 0.0001 (10^-4) paramos
    break
  }
}
n #imprimimos n
```

```
## [1] 40
```

Ejercicio 6:

Reordeno P siguiendo las intrucciones del enunciado y creo una nueva matriz (P_canonica) con el resultado:

```
labels<-c("0K","5K","1K","2K","3K","4K")
byRow <- TRUE
x <- c(1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0.58, 0, 0, 0.42, 0, 0, 0, 0, 0.58, 0, 0.42, 0, 0, 0, 0, 0.58, 0, 0.4
2, 0, 0.42, 0, 0, 0.58, 0)
P_canonica<-matrix(data=x,byrow=byRow,nrow=6,dimnames=list(labels,labels))
P_canonica
```

```
##      0K   5K   1K   2K   3K   4K
## 0K 1.00 0.00 0.00 0.00 0.00 0.00
## 5K 0.00 1.00 0.00 0.00 0.00 0.00
## 1K 0.58 0.00 0.00 0.42 0.00 0.00
## 2K 0.00 0.00 0.58 0.00 0.42 0.00
## 3K 0.00 0.00 0.00 0.58 0.00 0.42
## 4K 0.00 0.42 0.00 0.00 0.58 0.00
```

Como podemos observer P_canonica sigue la estructura descrita por el enunciado:

	0K	5K	1K	2K	3K	4K
0K	1.00	0.00	0.00	0.00	0.00	0.00
5K	0.00	1.00	0.00	0.00	0.00	0.00
1K	0.58	0.00	0.00	0.42	0.00	0.00
2K	0.00	0.00	0.58	0.00	0.42	0.00
3K	0.00	0.00	0.00	0.58	0.00	0.42
4K	0.00	0.42	0.00	0.00	0.58	0.00

- P_canonica es una matriz cuadrada de 6 filas y 6 columnas.
- Ik (en amarillo) es la matriz identidad de 2 filas y 2 columnas.
- 0 (en naranja) es la matriz nula de 2 filas y 4 columnas.
- A (en verde) es una matriz de 4 filas y 2 columnas.
- B (en rosa) es una matriz de 4 filas y 4 columnas.

Ejercicio 7:

Creamos:

- I: Matriz identidad de 4 filas y 4 columnas.
- B: Matriz de 4 filas y 4 columnas tal y como indica el enunciado.
- A: Matriz de 2 filas y 4 columnas tal y como indica el enunciado.

```
I <- diag(rep(1, 4))

b <- c(0, 0.42, 0, 0, 0.58, 0, 0.42, 0, 0, 0.58, 0, 0.42, 0, 0, 0.58, 0)
B <-matrix(data=b,byrow=TRUE,nrow=4)

A <- matrix(c(0.58, 0, 0, 0, 0, 0, 0, 0.42), nrow = 4, ncol = 2)
```

Resolvemos la ecuación tal y como nos indica el resultado:

$$S = (I_{6-k} - B)^{-1} A$$

```
S <- solve(I - B) %*% A #La función solve calculará la inversa de I-B y a este resultado se le multiplicará A
colnames(S) <- c("0K", "5K")
rownames(S) <- c("1K", "2K", "3K", "4K")
S
```

```
##           0K         5K
## 1K 0.9052874 0.09471257
## 2K 0.7744939 0.22550613
## 3K 0.5938742 0.40612580
## 4K 0.3444470 0.65555296
```

La probabilidad de empezar con 3K y terminar en 0K es 59.38742%

La probabilidad de empezar con 1K y terminar en 5K es 9.471257%