Paquete Numbers

Grupo 13: Daniel Ruskov, Ander Alonso, Adrian San Segundo

13 de noviembre 2017

```
> library(numbers)
```

1. Introducción

Para trabajar en R a veces es necesario descargar e instalar paquetes que contienen funciones que no están en el paquete base de R.Cuando se necesita un paquete nuevo hay que instalarlo con el comando install.packages("nombre") para instalarlo y library(nombre) para cargar las funciones que contiene.

El primer objetivo de este laboratorio es aprender a utilizar un paquete específico que contiene funciones en teoría de números, el paquete numbers. El segundo es resolver los ejercicios del final de este documento utilizando lasa funciones del paquete.

Al final del laboratorio hay que entregar un documento que contenga las repuestas a los ejercicios propuestos.

2. El paquete numbers

Para utilizar un paquete nuevo hay que instalarlo y cargar las funciones que contiene usando los comandos: install.packages("numbers") y library(numbers). Una vez que está instalado, cuando se cierra la sesión y se abre una nueva, no es necesario instalarlo de nuevo, sólo es necesario cargar de nuevo las finciones del paquete utilizando library(numbers).

En el índice del paquete aparecen las funciones que contiene. Algunas de esas funciones, que necesitamos para el algoritmo RSA, son las siguientes:

1. div(n,m)
2. mod(n. m)
3. modinv(m,n)
4. primeFactors
5. Primes(n)
6. Primes(n,m)

7. nextPrime(n)
8. previousPrime(n)
9. coprime(n,m)
10. modpower(n, k m)
11. GCD(n, m)
12. LCM(n, m)
13. extGCD(a, b),
14. eulersPhi

Ejercicios

1. Usa la función help() para aprender a utilizar las funciones del paquete numbers recogidas en el apartado 2.
2. Utiliza dichas funciones para responder a las siguientes preguntas.

   2.1 **Congruencia mod n**. Decir si son verdaderas o falsas las siguientes afirmaciones:
   $2 \equiv 4 \pmod 2$; $13 \equiv -2 \pmod 5$; $15 \equiv 3 \pmod 3$; $20 \equiv 4 \pmod 7$.

   ```
   > mod(2-4,2)
   [1] 0
   > mod (13-(-2), 5)
   [1] 0
   > mod (15-3, 3)
   [1] 0
   > mod (20-4, 7)
   [1] 2
   ```

   Conclusión: Las verdaderas son la 1, 2 y 3. La falsa es la ultima.

   2.2 **Suma y producto**.

   1) Calcular: $15 + 5 \pmod{35}$, $28 + 11 \pmod{13}$, $112 + 4 \pmod{16}$.

   ```
   > mod (15+5, 35)
   [1] 20
   > mod (28+11, 13)
   [1] 0
   ```

```
> mod (112+4, 16)
[1] 4
```

2) Calcular: $3 \cdot 5 \pmod{11}$, $2 \cdot 10 \pmod{11}$, $4 \cdot 7 \pmod{11}$.

```
> mod (3*5, 11)
[1] 4
> mod (2*10, 11)
[1] 9
> mod (4*7, 11)
[1] 6
```

2.3 **Inversos modulares.** Calcular, si existen, los siguiente inversos modulares:

- $3^{-1} \pmod{10}$.

```
> modinv(3, 10)
[1] 7
```

- $5^{-1} \pmod{12}$.

```
> modinv(5, 12)
[1] 5
```

- $7^{-1} \pmod{16}$.

```
> modinv(7, 16)
[1] 7
```

- $6^{-1} \pmod{17}$.

```
> modinv(6, 17)
[1] 3
```

2.4 **Funciones de euler.** Calcular: $\phi(5)$. $\phi(67)$, $\phi(15)$.

```
> eulersPhi(5)
[1] 4
> eulersPhi(67)
[1] 66
> eulersPhi(15)
[1] 8
```

2.5 **Potencias modulares.** Calcular: $2^3 \pmod{7}$, $3^3 \pmod{4}$, $4^2 \pmod{11}$.

```
> modpower(2, 3, 7)
[1] 1
> modpower(3, 3, 4)
[1] 3
> modpower(4, 2, 11)
[1] 5
```

3. Crea una función que dado como argumento un número n calcule aleatoriamente un número e relativamente primo con n. Para ello: dado un número n, elegir aleatoriamente un número e menor o igual que n. Si e es par, hacer e=e+1.Mientras $mcd(e, n) \neq 1$, hacer $e < -e + 2$. El resultado e

```
> busqueda_coprimo <- function(n){
+
+   repeat{
+
+     e<-sample(1:n,1,replace = F)
+     coprimo<-coprime(n,e)
+     if (!coprimo){
+       if(mod(e,2)==0){
+         e<-e+1
+           return(e)
+       }else{
+         e<-e+2
+         return(e)
+       }
+     }
+     else{
+       break
+       }
+   }
+    return(e)
+ }
> coprimo<-busqueda_coprimo(76896743)
> print(coprimo)

[1] 19093007
```

4. Dado un vector **v**, la siguiente función admite como parámetro un vector **v** y da como resultado un vector en el que a cada componente de **v** le suma 10.

```
> Vectores<-function(vector){
+ l<-length(vector)
+ resultado<-rep(0,l)
+ for (i in 1:l) {resultado[i]<-vector[i]+10}
+ print("Resultado")
+ print(resultado)
+ }
> v<-c(2,1,6)
> Vectores(v)

[1] "Resultado"
[1] 12 11 16
```

Ejercicio. Codificar un mensaje. El resultado es un vector numérico **v**. Crear una función que admita como parámetro un vector y devuelva un vector que tiene cada componente de **v** elevada al cuadrado.

```
> Vectores<-function(vector){
+ l<-length(vector)
+ resultado<-rep(0,l)
+ for (i in 1:l) {resultado[i]<-vector[i]*vector[i]}
+ v<-strtoi (charToRaw("El resultado es"), 16L)
+ print(v)
+ print(resultado)
+ }
> v<-c(2,1,6)
> Vectores(v)

 [1]  69 108  32 114 101 115 117 108 116  97 100 111  32 101 115
[1]  4  1 36
```