Criptografía Practica 2
Cifrados clásicos

Fresneda Rodríguez, Antonio Manuel
Marín Molina, Ismael

1 Criptografía Practica 2

```
In [1]: import Funciones as fc
        import numpy as np
```

1.1 Texto 1

```
In [2]: texto1 = fc.SustitutionText("IsmaelAntonio1.txt")
        texto1.printStats()

        Stadistics
Longitud: 36669        Media: 1358.111111111111         Indice de coin.: 0.07556177521787245
Maximo: 4991        Minimo: 0        Standar: 1385.57195927205
```

Como vemos, el índice de coincidencia del texto 2 es casi igual que el índice de coincidencia del español. Gracias a esto podemos descartar que el texto se haya cifrado usando un cifrado polialfabético. Luego suponemos que para el cifrado de este texto se ha usado o un cifrado del César o usando un cifrado por sustitución o de trasposición. Vamos a imprimir las frecuencias de aparición de las letras:

```
In [3]: print ("La frecuencia de aparición de las letras es:\n {} \n".format(texto1.frec))
        print ("La función solveCesar devuelve: {}".format(texto1.solveCesar()))

La frecuencia de aparición de las letras es:
 [ 933   37 1440  336 1603    0 1002 2260 2386 1853  603  176 4991 2529
 1638  384 2539  155  315 2568  334 3523  244 4520   61  239    0]

La función solveCesar devuelve: ModeResult(mode=array([0]), count=array([3]))
```

Como vemos, las frecuencias de aparición de las letras no corresponde a la del español (la H por es la letra que aparece más veces), por lo que descartamos que se haya cifrado usando trasposicion.

La función solveCesar calcula el orden de las letras segun su frecuencia de aparicón en el texto, calcula la diferencia con la frecuencia normal del español y saca la moda de esas distancias. Teóricamente esta moda debe de repetirse para la gran mayoría de las letras, ya que la probabilidad de

haber fallado (a la hora de asignar una letra del texto cifrado a la del español) es relativamente baja. Como vemos, el resultado de la función nos dice que la moda es 8 y ha aparecido 4 veces. Esto nos indica que hemos fallado en 23 ocasiones, una probabilidad bastante alta con lo que suponemos que este texto no se ha cifrado con César y por descarte suponemos que se cifró usando un cifrado por sustitución.

Para descifrar este texto, hemos sustituido cada letra del texto por la letra en español siguiendo el orden de aparición, sustituyendo de la siguiente forma:

```
In [4]: texto1.printPerm()
```

```
M->E (0.13610952030325343)
W->A (0.12326488314379994)
U->O (0.09607570427336443)
S->S (0.0700319070604598)
P->R (0.06924104829692657)
N->N (0.06896833837846683)
I->I (0.06506858654449263)
H->D (0.06163244157189997)
J->L (0.050533147890588784)
Ñ->C (0.04466988464370449)
E->T (0.04371539992909542)
C->U (0.03927022825820175)
G->M (0.027325533829665384)
A->P (0.025443835392293217)
K->B (0.016444408083121982)
O->G (0.0104720608688538)
D->Y (0.009163053260247075)
T->V (0.009108511276555127)
R->Q (0.008590362431481632)
V->H (0.006654122010417519)
Y->F (0.006517767051187652)
L->Z (0.004799694564891325)
Q->J (0.0042270037361258825)
X->Ñ (0.0016635305026043796)
B->X (0.0010090266983010172)
F->W (0.0)
Z->K (0.0)
```

Tras estar buscando palabras casi descifradas para ir ajustando el diccionario, el resultado fue: M->A (0.13729778226371717) W->E (0.12395211259707853) U->O (0.08660666722740909) S->N (0.07199932710909245) P->S (0.07118625059578883) N->R (0.07090587938430482) H->L (0.0633638937953851) I->I (0.05969103092494463) J->D (0.05195278548798609) E->C (0.044943505200885976) Ñ->U (0.04432668853562116) C->T (0.040373454453696694) G->P (0.028093195390697285) A->M (0.02615863403145765) K->B (0.01690638405248549) O->V (0.010766254520985785) D->G (0.009420472705862562) T->Q (0.009364398463565762) R->H (0.008831693161746153) V->Y (0.006841057560209717) Y->F (0.0067008719544677155) L->J (0.004934533322118485) Q->Z (0.004345753778002075) B->Ñ (0.0010373734824908178) F->X (0.0) X->W (0.0) Z->K (0.0)

Y el texto descifrado es un fragmento de la obra "Cien años de soledad" escrito por Gabriel García Márquez

1.2 Texto 2

```
In [5]: # Texto 2
        texto2 = fc.SustitutionText("IsmaelAntonio2.txt")
        texto2.printStats()

        Stadistics
Longitud: 18504        Media: 685.3333333333334         Indice de coin.: 0.04037474648891958
Maximo: 1117        Minimo: 317        Standar: 207.32708100612064
```

Como vemos, el indice de coincidencia de este texto es más bajo que el IC del español. Esto indica que se han homogeneizado las frecuencias de aparición de las letras. El único algoritmo de cifrado que hemos visto que provoque esto es el cifrado de Vignere. Comenzamos calculando la longitud de la clave con la función implementada:

```
In [6]: print (fc.vignere_key_length(str(texto2)))

14
```

Vemos que la longitud de la clave que se ha usado para cifrar tiene una longitud de 14 caracteres. Sabiendo esto, lo unico que tenemos que hacer es prácticamente resolver un César, sabiendo que para cifrar se ha ido cogiendo las letras de 14 en 14 y cifrandose con las letras de la clave. Asi que para resolverlo:

```
In [7]: for i in texto2.frecuenciasVigenere(14):
            print (texto2.solveCesar(np.argsort(i)[::-1]))

ModeResult(mode=array([7]), count=array([8]))
ModeResult(mode=array([4]), count=array([10]))
ModeResult(mode=array([11]), count=array([8]))
ModeResult(mode=array([8]), count=array([12]))
ModeResult(mode=array([15]), count=array([12]))
ModeResult(mode=array([2]), count=array([5]))
ModeResult(mode=array([4]), count=array([12]))
ModeResult(mode=array([13]), count=array([11]))
ModeResult(mode=array([20]), count=array([11]))
ModeResult(mode=array([18]), count=array([9]))
ModeResult(mode=array([8]), count=array([12]))
ModeResult(mode=array([19]), count=array([12]))
ModeResult(mode=array([12]), count=array([11]))
ModeResult(mode=array([15]), count=array([7]))
```

Y con lo anterior tenemos que la clave es Heliocentrismo

1.3 Texto 3

```
In [8]: texto3 = fc.SustitutionText("IsmaelAntonio3.txt")
        texto3.printStats()

        Stadistics
Longitud: 6731        Media: 249.2962962962963        Indice de coin.: 0.07537222710207567
Maximo: 927        Minimo: 0        Standar: 254.0816163441589
```

Seguimos el procedimiento del texto 1:

```
In [9]: print ("La frecuencia de aparición de las letras es:\n {} \n".format(texto3.frec))
        print ("La función solveCesar devuelve: {}".format(texto3.solveCesar()))

La frecuencia de aparición de las letras es:
 [792  79 262 324 927  41  64  61 395  22   0 342 211 477  10 616 157  85
 450 617 318 303  71   0   6  81  20]

La función solveCesar devuelve: ModeResult(mode=array([0]), count=array([12]))
```

Esto nos dice que no se ha cambiado el valor de las letras, si no la posición de las mismas. El único cifrado que provoca esto es el de trasposición.

```
In [10]: max_apariciones = 0
         iteracion_max = 0
         for i in range (1,100):
             occ_que = fc.ocurrencias('QUE',str(texto3), i)
             if occ_que > max_apariciones:
                 max_apariciones = occ_que
                 iteracion_max = i
         print ('El valor de los saltos es de {}, la cadena "que" aparece {} veces'
                .format(iteracion_max,max_apariciones))

El valor de los saltos es de 27, la cadena "que" aparece 71 veces
```

Con esto suponemos que el cifrado da saltos de 27 en 27, asi que desciframos el texto usando este valor:

```
In [11]: text3_des = fc.descifra_transposicion(str(texto3), iteracion_max)
         print (text3_des[:50])

DASLASCOSASSERCRIADASAMANERADECONTIENDAOBATALLADIC
```

Con esto vemos que el texto corresponde a un fragmento de la Celestina de Fernando de Rojas.

1.4 Texto 4

```
In [12]: texto4 = fc.SustitutionText("IsmaelAntonio4.txt")
         texto4.printStats()

        Stadistics
Longitud: 3218        Media: 119.18518518518519         Indice de coin.: 0.07697801823091396
Maximo: 413        Minimo: 0        Standar: 124.21308961769331
```

```
In [13]: print ("La frecuencia de aparición de las letras es:\n {} \n".format(texto4.frec))
         print ("La función solveCesar devuelve: {}".format(texto4.solveCesar()))

La frecuencia de aparición de las letras es:
 [ 17 279  58  24 213 313 159 107  19   0  10  32   8 394  46 112 145 413
  21  35  23 228  18   0 134 109 301]

La función solveCesar devuelve: ModeResult(mode=array([13]), count=array([8]))
```

Como vemos, en este caso se repite más veces la moda que en el texto 1. Probamos a resolver el texto con un César de 13:

```
In [14]: print (texto4.decodeCesar(13)[:50])

SEESTIMAQUEUNMILLONDEMENORESTRABAJANACTUALMENTEENM
```

Vemos como el texto se ha descifrado correctamente.