Frekvensanalyse af cæsar kryptering

import math

Et ordinært frekvenanalyse for alle de engelske ord ser således ud

distribution = {

"a" : 8.1

"b" : 1.8

"c" : 2.3

"d" : 4.1

"e" : 12.8

"f" : 2.3

"g" : 2

"h" : 6.1

"i" : 7

"j" : 0.2

"k" : 0.9

"l" : 4

"m" : 2.1

"n" : 6.9

"o" : 7.5

"p" : 2

"q" : 0.18

"r" : 6

"s" : 6.3

"t" : 9

"u" : 2.8

"v" : 1

"w" : 2.1

"x" : 0.2

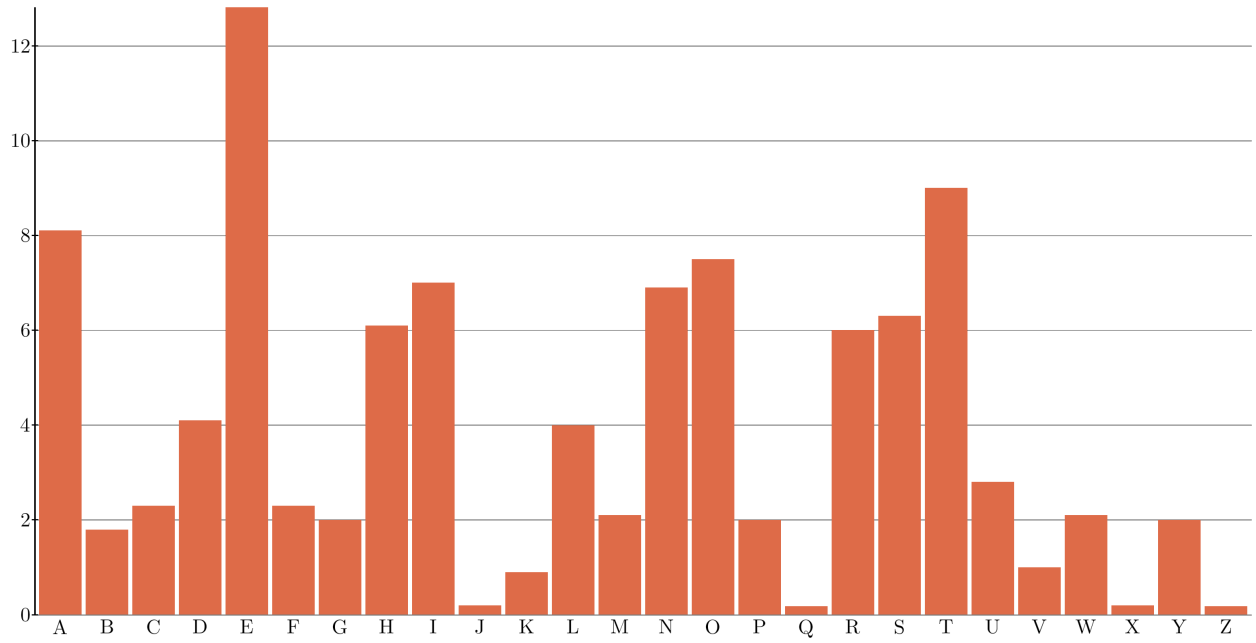
"y" : 2

"z" : 0.18

}

s = sum($seq(distribution[i], i=distribution)$)

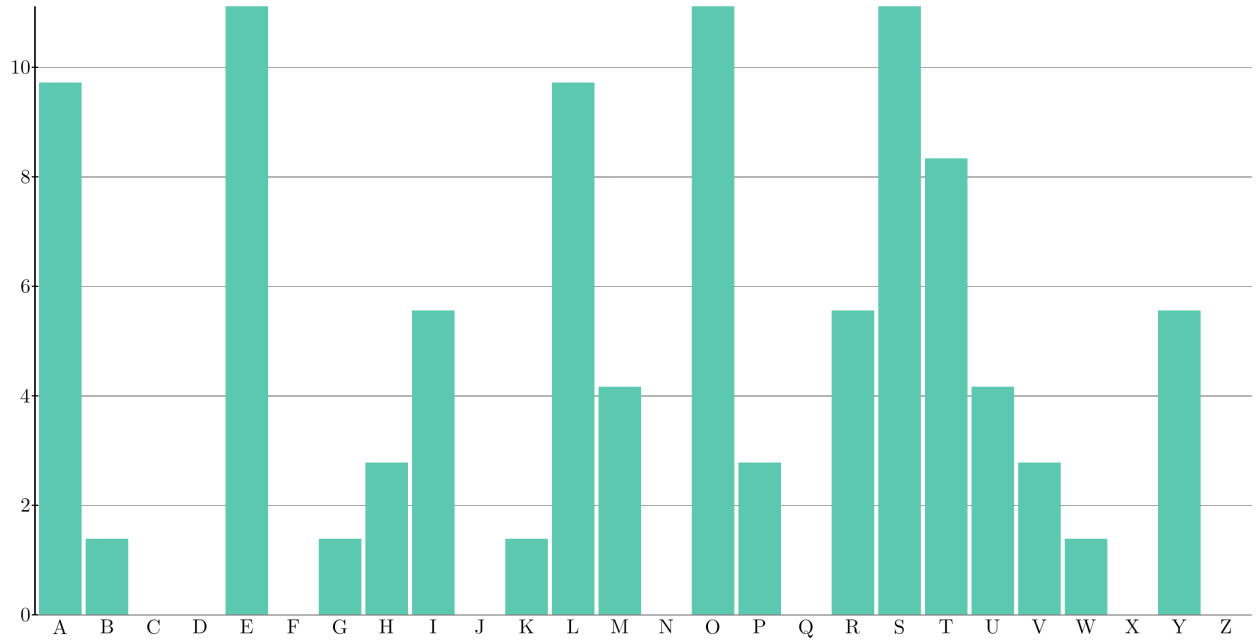
> For alle *key* i distribution
>
> distribution[key] = $\frac{distribution[key]}{s}$



Givet en linje
linje = "*hello this is a message to Ceasar, someome will propbaly try to kill you. Stay save. Love Brutus*"
Frekvensen af denne er kan findes ved nedenstående funktion
alfabet = "*abcdefghijklmnopqrstuvwxyz*"

```
Funktion frekvensanalyse()
    res = {
    }

    For alle bogstav i alfabet
        res[bogstav] = count(bogstav)

    s = sum(seq(res[i], i=res))

    For alle key i res
        res[key] = res[key] / s

    return res
```

frekvenser = frekvensanalyse(*linje*)
Dertil kan nedenstående frekvensgraf laves



Dette minder om den første frekvenanalyse.
En cæsar kryptering skubber eller "rotere" alle bogstaver en vist antal
Funktionen for cæsar kryptering er herunder

```
Funktion krypter()
    s = ""
```

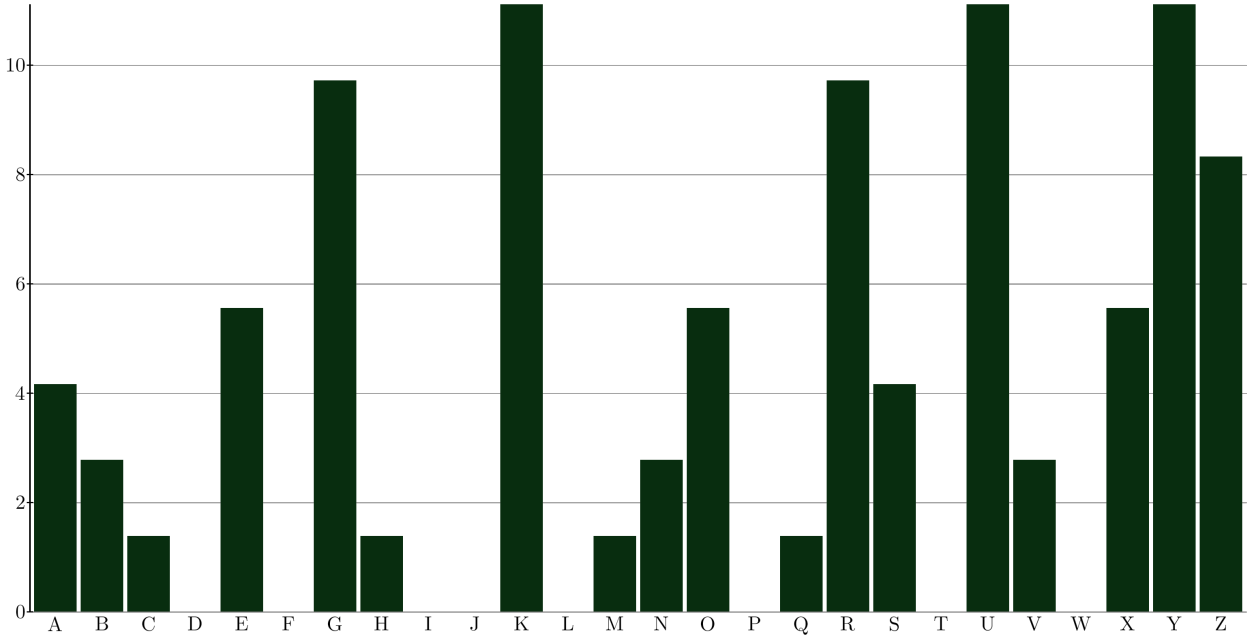
```
    For alle bogstav i linje
        Hvis bogstav not in alfabet:
            Spring over linje
        s = alfabet[(alfabet.index(bogstav.lower()) + n) % 26]

    return s
```

Der anvendes nu n=6
Dermed bliver voers linje fra før nu
cipher = krypter(*linje*, 6)

cipher

nkrruznoyoygskyygmkzukgygxyuskuskcorrvxuvhgrezxezuqorreuazgeygbkubkxazay

Frekvensanalysen af den krypteret sætning er således

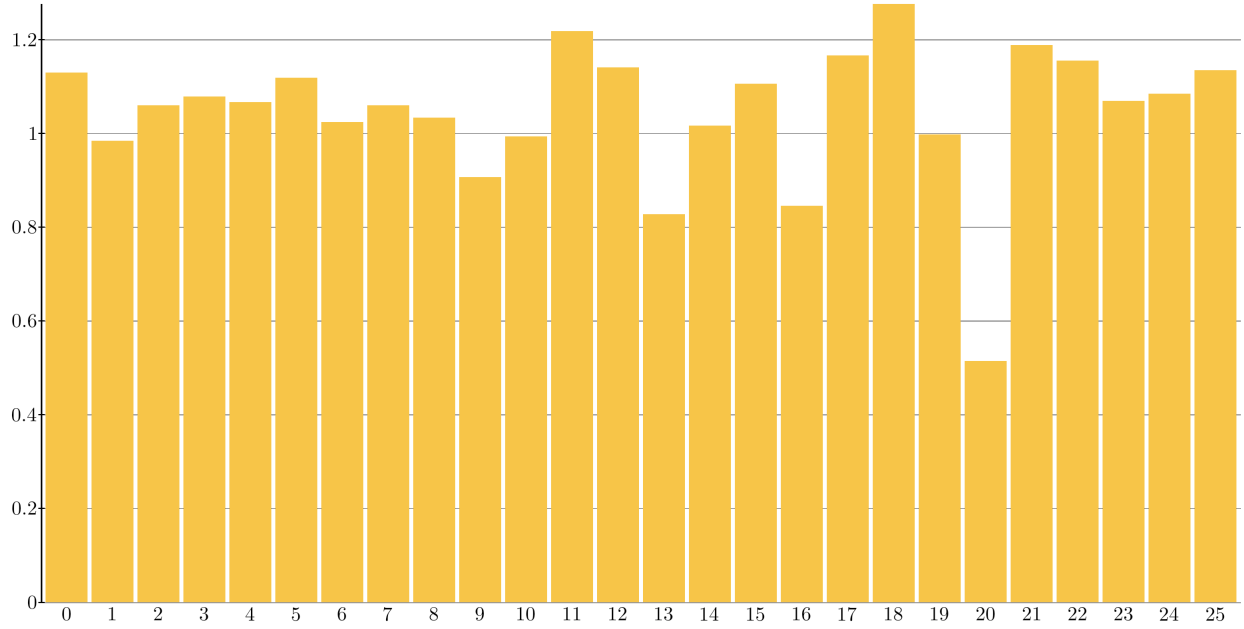


Vi vil nu prøve at finde n ved at gætte og sammenligne med den "rigtige" distribution af bogstaver
Dette gøres ned nedenstående funktion

```
Funktion kneakKoden()
    scores = []

    For alle i i range(26)
        guess = krypter(cipher, i)
        guessfreq = frekvensanalyse(guess)
        score = 0

        For alle bogstav i frekvenser
            score = abs(distribution[bogstav]-guessfreq[bogstav])

        append(score)

    return scores
```

```
resultat = kneakKoden(cipher)
chart = Bar( ,)
style(10)
```

```
For alle i, res i enumerate(resultat)
    add(i, res)
```



Dette indikere at for at rotere cipheren tilbage skal der bruges $n=20$

26-20

6

Dermed er den orginale er $n=6$
Og dekrpyteringen kan fortages med

krypter(*cipher*, 20)

hellothisisamessagetoeasarsomeomewillpropbalytrytokillyoutaysaveoverutus

NB:
I praksis lavede man ikke alle analyse, man gættede bare på forflyningen og så mønstre