COMP105 Lecture 10

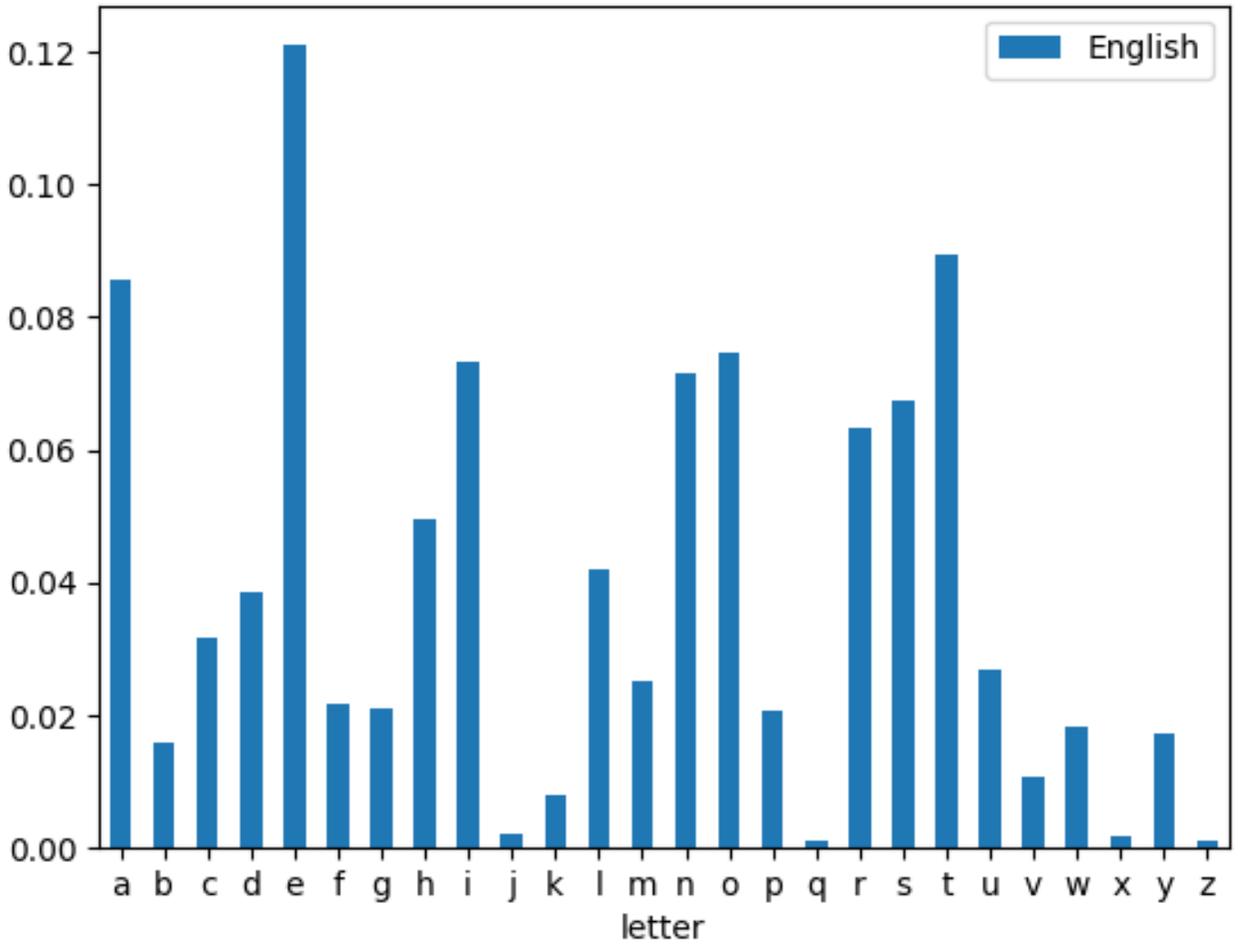
Cracking the Caesar Cipher

Cracking the Caesar cipher

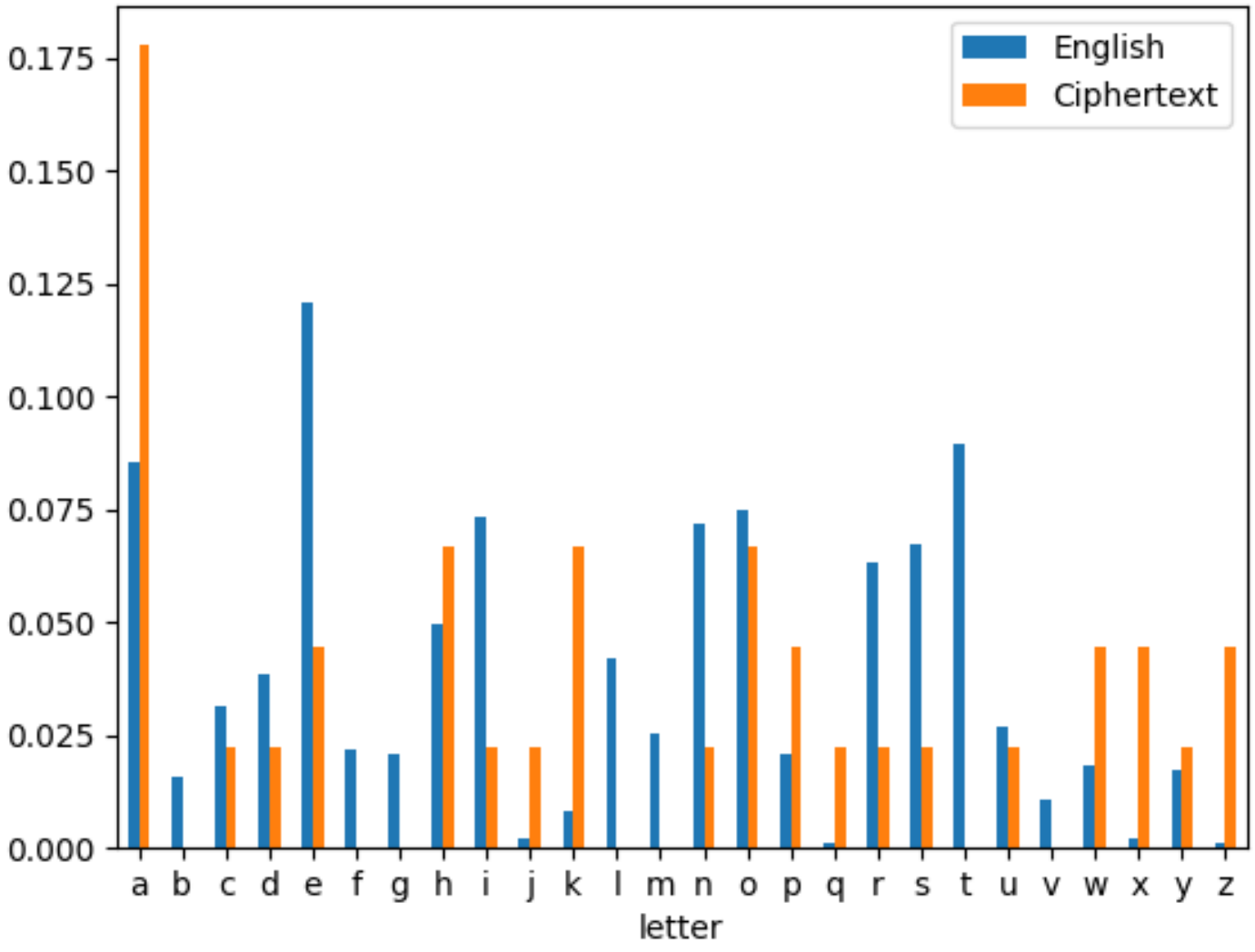
How would we **crack** this text?

```
"ukq sehh jaran xa wxha pk zaykza pdeo iaoowca"
```

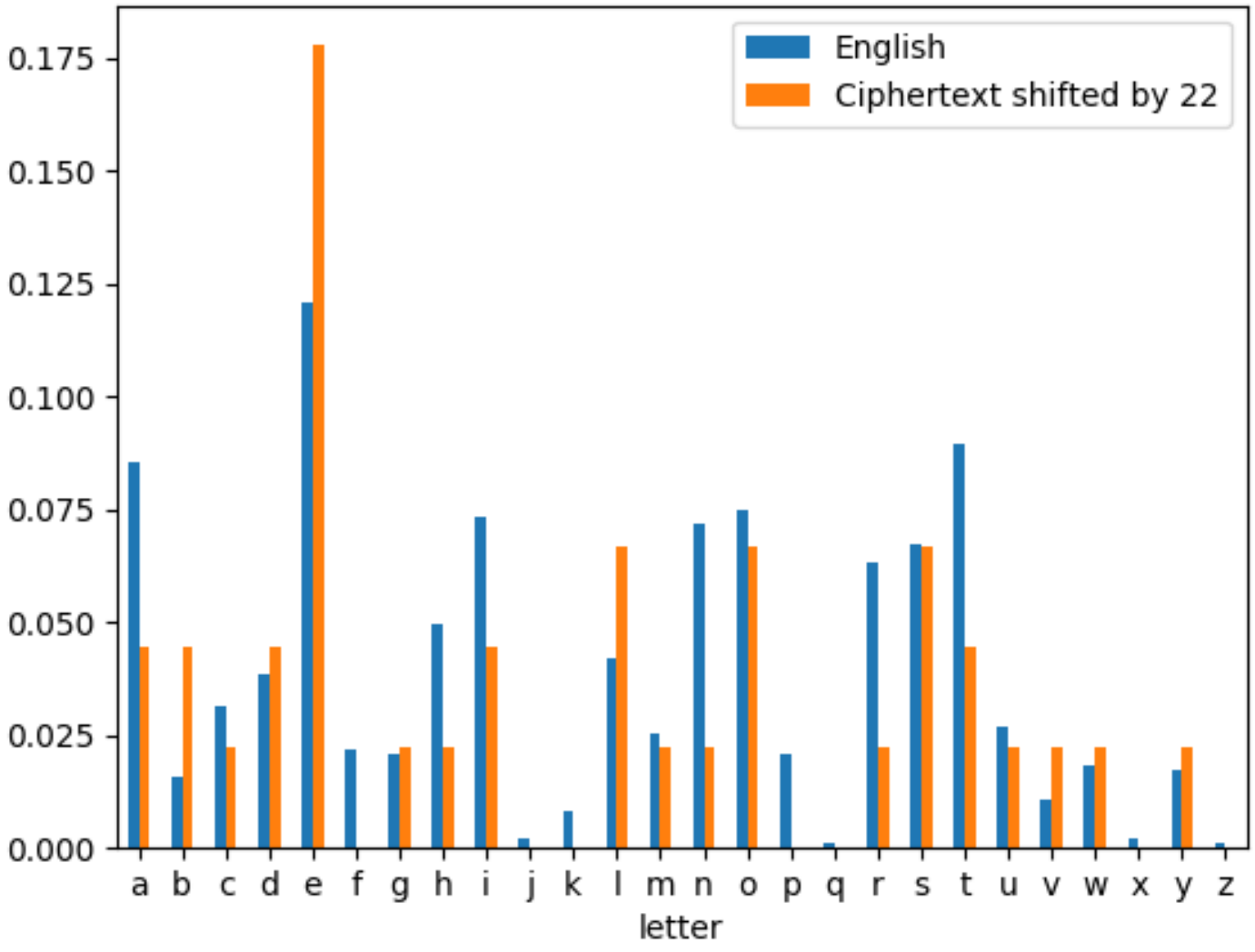
The letter frequencies for the English language



The letter frequencies for the cipertext



The letter frequencies for the cipertext shifted by 22



Cracking the Caesar cipher

Let's guess that the message was **shifted by 22**

```
ghci> let msg = "ukq sehh jaran xa wxha pk zaykza
                                              pdeo iaoowca"
ghci> caesar_dec msg 22
"you will never be able to decode this message"
```

Cracking the Caesar cipher

Idea:

- Try decoding the text with a particular offset
- Compute the letter frequencies for the decoded text
- Check if the frequencies are close to the English frequencies

How to tell if two frequency lists are close?

- Use the **chi-squared** score:

$$\sum_{i=\mathtt{a}}^{\mathtt{z}} \frac{(\mathtt{freq}_i - \mathtt{english}_i)^2}{\mathtt{english}_i}$$

- The chi-squared score will be lower when `freq` is close to `english`

Cracking the Caesar cipher

Algorithm:

- Try all 26 possible shifts
- For each one compute the letter frequency distribution of the decoded text, and the chi-squared score
- Use the shift with the lowest chi-squared score to decode the string

Exercise

- Think about how you would code this in Haskell

Counting frequencies in strings

```
count _ [] = 0
count c (x:xs)
    | c == x    = 1 + rest
    | otherwise = rest
    where rest = count c xs


freq c list = fromIntegral (count c list) /
              fromIntegral (length list)
```

```
ghci> count 'a' "aabaa"
4
ghci> freq 'a' "aabaa"
0.8
```

Getting the table of frequencies for a string

```
get_freqs _      26 = []
get_freqs string c  = freq (int2char c) string
                      : get_freqs string (c + 1)
```

```
ghci> get_freqs "abc" 0
[0.33333334,0.33333334,0.33333334,0.0,0.0,...
```

get_freqs returns a table with exactly 26 elements

Implementing the chi-squared score

$$\sum_{i=\mathtt{a}}^{\mathtt{z}} \frac{(\mathtt{freq}_i - \mathtt{english}_i)^2}{\mathtt{english}_i}$$

```
chi_squared []     []     = 0
chi_squared (x:xs) (y:ys) =
    (x - y)**2/y + chi_squared xs ys
```

```
ghci> chi_squared [0.1, 0.9] [0.8, 0.2]
3.0624998
```

The table of English frequencies

```
eng_freqs = [0.0855, 0.0160, 0.0316, 0.0387, 0.1210,
             0.0218, 0.0209, 0.0496, 0.0733, 0.0022,
             0.0081, 0.0421, 0.0253, 0.0717, 0.0747,
             0.0207, 0.0010, 0.0633, 0.0673, 0.0894,
             0.0268, 0.0106, 0.0183, 0.0019, 0.0172,
             0.0011]
```

Getting the chi-squared score for a string

```
chi_squared_string string =
    let
        string_freqs = get_freqs string 0
    in
        chi_squared string_freqs eng_freqs
```

```
ghci> chi_squared_string "hello there"
1.5819808
```

Getting the list of chi-squared scores for a string

```
chi_squared_list _ 26 = []
chi_squared_list string i =
    let
        decoded = caesar_dec string i
        score = chi_squared_string decoded
     in
        (score, decoded) : chi_squared_list string (i+1)
```

```
ghci> chi_squared_list "ifmmp" 0
[(9.637143,"ifmmp"),(4.4730797,"hello"),
 (22.258533,"gdkkn"),(76.40909,"fcjjm"),...
```

Finding the offset with the lowest score

```
get_best [(score, string)] = (score, string)
get_best ((score, string):xs) =
    let
        (tail_score, tail_string) = get_best xs
    in
        if score < tail_score
        then (score, string)
        else (tail_score, tail_string)
```

```
ghci> get_best (chi_squared_list "ifmmp" 0)
(4.4730797,"hello")
```

Tieing it all together

```
caesar_crack string =
    let
        scores = chi_squared_list string 0
        (score, best) = get_best scores
    in
        best
```

```
ghci> caesar_crack "lbh jvyy arire qrpbqr guvf"
"you will never decode this"
```