I have write a program for the homework to help decode subsitution and  Vigen`ere cipher. Which can be found on https://github.com/xuyoji/cryptography-decoder

## Problem 1:

First, by the program, we can find the alphabet 'C' appear much more than other alphabets. It means 'C' stand for 'e', beacause 'e' have similar biggest frequency with 'C'.

Then, we define meaningful n-gram:
Meaningful n-gram is a gram of n alphabet(n >= 2) which appear more than twice in a text, and if it is a sub-string of a k-gram (k > n), its appearence time must be different from the k-gram.(Which means it appears more than the k-gram)
Use the program, and set fixed alphabet = C:e we can find all meaningful n-gram:

```
-----fixed substitution rule-----
C-e

-------all meaningful gram-------
--2gram--:
Ze:7 eG:7 eN:5 Ae:5 GO:5 eK:5 YS:5 Ne:5 FZ:4 GK:4 GY:4 SF:4 ZE:3 eI:3 KG:3 Xe:3 SH:3 eJ:3 XE:3
 KS:3 KU:3 SI:3 Ie:3 eY:3 US:3 GL:3 MG:3 eS:2 Ue:2 EU:2 EO:2 IG:2 IU:2 LK:2 DS:2 KZ:2 UG:2 PK:
2 UM:2 DP:2 NS:2 WY:2

--3gram--:
GOI:3 YSF:3 ZEJ:2 UZe:2 eJU:2 JNe:2 KGO:2 ZeN:2 KSH:2 eKX:2 SAe:2 eKS:2 GAe:2 NeG:2 GOL:2 JeK:
2 eYK:2

--4gram--:
IeGI:2

--5gram--:
FZeeN:3

--6gram--:
eFZeeN:2

--11gram--:
FZeeNDGYYSF:2

-------word possibilty test-------
->test word = 'the'
[('GAe', 0.0009599659765625003), ('SAe', 0.0011744191015625), ('UZe', 0.0014526417578124996),
('FZe', 0.0033228810156249993), ('JNe', 0.004284599765624999)]

--------current text--------
EMGLOSUDeGDNeUSWYSFHNSFeYKDPUMLWGYIeOXYSIPJeKQPKUGKMGOLIeGINeGAeKSNISAeYKZSeKXEeJeKSHYSXeGOIDP
KZeNKSHIeGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZUGFZeeNDGYYSFUSZeNXEOJNeGYEOWEUPXEZGAeGNFGLKNSAeIGOIYe
KXeJUeIUZeFZeeNDGYYSFEUEKUZeSOeFZeeNeIAeZEJNeSHFZEJZEGMXeYHeJUMGKUeY
```

All gram we disscuss below stand for meaningful gram.
Notice the 4-gram 'IeGI'. It has the same begin and end word. And have fix 'e' in the middle of it. In such format, there are only 3 words  to fit it, which is 'dead', 'rear' and 'teat'. We notice in all of these words, G stands for 'a'.

We fix G:a

```
-----fixed substitution rule-----
C-e  G-a

-------all meaningful gram-------
--2gram--:
Ze:7 ea:7 eN:5 Ae:5 aO:5 eK:5 YS:5 Ne:5 FZ:4 aK:4 aY:4 SF:4 ZE:3 eI:3 Ka:3 Xe:3 SH:3 eJ:3 XE:3
 KS:3 KU:3 SI:3 Ie:3 eY:3 US:3 aL:3 Ma:3 eS:2 Ue:2 EU:2 EO:2 Ia:2 IU:2 LK:2 DS:2 KZ:2 Ua:2 PK:
2 UM:2 DP:2 NS:2 WY:2

--3gram--:
aOI:3 YSF:3 ZEJ:2 UZe:2 eJU:2 JNe:2 KaO:2 ZeN:2 KSH:2 eKX:2 SAe:2 eKS:2 aAe:2 Nea:2 aOL:2 JeK:
2 eYK:2

--4gram--:
IeaI:2

--5gram--:
FZeeN:3

--6gram--:
eFZeeN:2

--11gram--:
FZeeNDaYYSF:2

-------word possibilty test-------
->test word = 'the'
[('SAe', 0.0011744191015625), ('UZe', 0.0014526417578124996), ('FZe', 0.0033228810156249993),
('JNe', 0.004284599765624999)]

--------current text--------
EMaLOSUDeaDNeUSWYSFHNSFeYKDPUMLWaYIeOXYSIPJeKQPKUaKMaOLIeaINeaAeKSNISAeYKZSeKXEeJeKSHYSXeaOIDP
KZeNKSHIeaIWYaKKaKaOLDSILKaOIUSIaLEDSPWZUaFZeeNDaYYSFUSZeNXEOJNeaYEOWEUPXEZaAeaNFaLKNSAeIaOIYe
KXeJUeIUZeFZeeNDaYYSFEUEKUZeSOeFZeeNeIAeZEJNeSHFZEJZEaMXeYHeJUMaKUeY
```

Then, we look the 3-gram aOI, it is not very make sense if 'I' is 't' or 'r', because 'and' appear more often than them, and 'aOI' is appears 3 times.
Then we can fix O:n, I:d

```
-----fixed substitution rule-----
C-e  G-a  I-d  O-n

-------all meaningful gram-------
--2gram--:
Ze:7 ea:7 eN:5 Ae:5 an:5 eK:5 YS:5 Ne:5 FZ:4 aK:4 aY:4 SF:4 ZE:3 ed:3 Ka:3 Xe:3 SH:3 eJ:3 XE:3
 KS:3 KU:3 Sd:3 de:3 eY:3 US:3 aL:3 Ma:3 eS:2 Ue:2 EU:2 En:2 da:2 dU:2 LK:2 DS:2 KZ:2 Ua:2 PK:
2 UM:2 DP:2 NS:2 WY:2

--3gram--:
and:3 YSF:3 ZEJ:2 UZe:2 eJU:2 JNe:2 Kan:2 ZeN:2 KSH:2 eKX:2 SAe:2 eKS:2 aAe:2 Nea:2 anL:2 JeK:
2 eYK:2

--4gram--:
dead:2

--5gram--:
FZeeN:3

--6gram--:
eFZeeN:2

--11gram--:
FZeeNDaYYSF:2

-------word possibilty test-------
->test word = 'the'
[('SAe', 0.0011744191015625), ('UZe', 0.0014526417578124996), ('FZe', 0.0033228810156249993),
('JNe', 0.004284599765624999)]

--------current text--------
EMaLnSUDeaDNeUSWYSFHNSFeYKDPUMLWaYdenXYSdPJeKQPKUaKManLdeadNeaAeKSNdSAeYKZSeKXEeJeKSHYSXeandDP
KZeNKSHdeadWYaKKaKanLDSdLKandUSdaLEDSPWZUaFZeeNDaYYSFUSZeNXEnJNeaYEnWEUPXEZaAeaNFaLKNSAedandYe
KXeJUedUZeFZeeNDaYYSFEUEKUZeSneFZeeNedAeZEJNeSHFZEJZEaMXeYHeJUMaKUeY
```

Then, I write a test word function in my program, enter the word you want to determine, if can calculate the total variance of this word's each alphabet and the aplabet frequency we have learnt from big data as below.

```
self.outText =
self.alpha_freq_dic = {'e':0.1249, 't':0.0928, 'a':0.0804, 'o':0.0764, 'i':0.0757, 'n':0.0723, 's':0.0651, 'r':0.0628, 'h':0.0505,
'l':0.0407, 'd':0.0382, 'c':0.0334, 'u':0.0273, 'm':0.0251, 'f':0.024, 'p':0.0214, 'g':0.0187, 'w':0.0168,'y':0.0166, 'b':0.0148, 'v':0.0105,
'k':0.0054, 'x':0.0023, 'j':0.0016, 'q':0.0012, 'z':0.001}
```

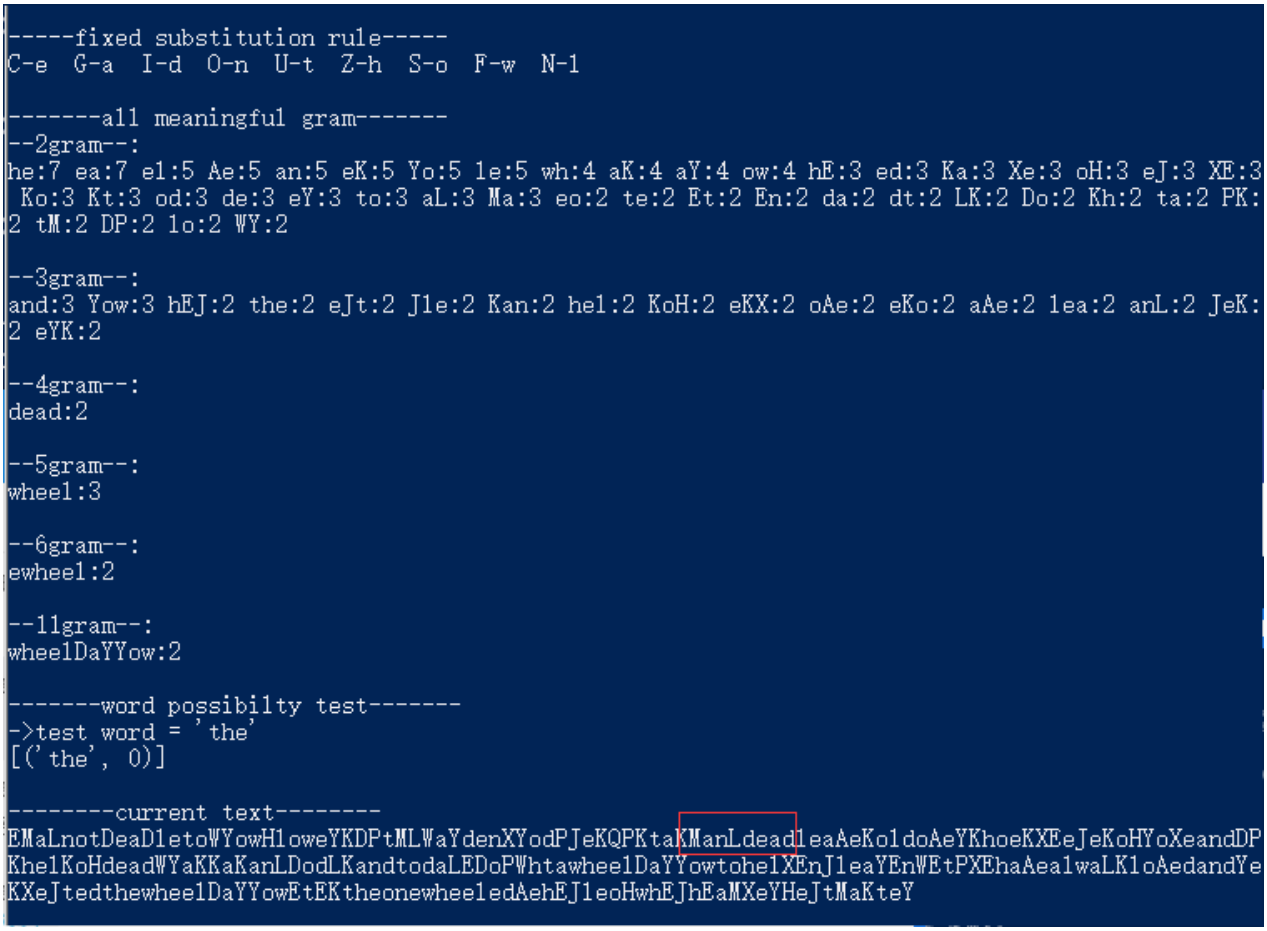
As showed in the blue picture upside, 'SAe' and 'UZe' have similar small variance with 'the'.  And from the first 'UZe' in the cerrent text in the upper image, we konw it follow with 'ed', whcih means 'UZe' must be a word rather than an affix. And we konw easily the 6-gram 'FZeeN' must be a word. Then we know 'UZe' must stand for 'the'. Otherwise, 'Sne' whcih is the word after the second 'UZe',  will be 'tne' which not make sense.
Thus, 'UZe' stand for 'the'. And 'Sne' has no choice, it must be 'one'.

Therefore, we get U:t,  Z:h, S:o

```
-----fixed substitution rule-----
C-e  G-a  I-d  O-n  U-t  Z-h  S-o

-------all meaningful gram-------
--2gram--:
he:7 ea:7 eN:5 Ae:5 an:5 eK:5 Yo:5 Ne:5 Fh:4 aK:4 aY:4 oF:4 hE:3 ed:3 Ka:3 Xe:3 oH:3 eJ:3 XE:3
 Ko:3 Kt:3 od:3 de:3 eY:3 to:3 aL:3 Ma:3 eo:2 te:2 Et:2 En:2 da:2 dt:2 LK:2 Do:2 Kh:2 ta:2 PK:
2 tM:2 DP:2 No:2 WY:2

--3gram--:
and:3 YoF:3 hEJ:2 the:2 eJt:2 JNe:2 Kan:2 heN:2 KoH:2 eKX:2 oAe:2 eKo:2 aAe:2 Nea:2 anL:2 JeK:
2 eYK:2

--4gram--:
dead:2

--5gram--:
FheeN:3

--6gram--:
eFheeN:2

--11gram--:
FheeNDaYYoF:2

-------word possibilty test-------
->test word = 'the'
[('the', 0)]

--------current text--------
EMaLnotDeaDNetoWYoFHNoFeYKDPtMLWaYdenXYodPJeKQPKtaKManLdeadNeaAeKoNdoAeYKhoeKXEeJeKoHYoXeandDP
KheNKoHdeadWYaKKaKanLDodLKandtodaLEDoPWhtaFheeNDaYYoFtoheNXEnJNeaYEnWEtPXEhaAeaNFaLKNoAedandYe
KXeJtedtheFheeNDaYYoFEtEKtheoneFheeNedAehEJNeoHFhEJhEaMXeYHeJtMaKteY
```

Now, the 5-gram 'GheeN' must be 'sheep' or 'wheel'. we notice there is a word 'GheeNed'. Apparently, 'sheeped' not make sense.

So. We fix G:s, N:l




Now it's much easy to guess the key, from the begining of the text, it easy to guess 'I MaL not be able to' . And 'MaL' is 'can' or 'may', but we have fix 'O-n'. So, it is 'I may not be able to'

Then 'D:b', 'L:y', 'M:m', 'E:i'

```
-----fixed substitution rule-----
C-e  G-a  I-d  O-n  U-t  Z-h  S-o  F-w  N-1  D-b  L-y  M-m  E-i

-------all meaningful gram-------
--2gram--:
he:7 ea:7 e1:5 Ae:5 an:5 eK:5 Yo:5 1e:5 wh:4 aK:4 aY:4 ow:4 hi:3 ed:3 Ka:3 Xe:3 oH:3 eJ:3 Xi:3
 Ko:3 Kt:3 od:3 de:3 eY:3 to:3 ay:3 ma:3 eo:2 te:2 it:2 in:2 da:2 dt:2 yK:2 bo:2 Kh:2 ta:2 PK:
2 tm:2 bP:2 1o:2 WY:2

--3gram--:
and:3 Yow:3 hiJ:2 the:2 eJt:2 J1e:2 Kan:2 he1:2 KoH:2 eKX:2 oAe:2 eKo:2 aAe:2 1ea:2 any:2 JeK:
2 eYK:2

--4gram--:
dead:2

--5gram--:
wheel:3

--6gram--:
ewheel:2

--11gram--:
wheelbaYYow:2

-------word possibilty test-------
->test word = 'the'
[('the', 0)]

--------current text--------
imaynotbeabletoWYowH1oweYKbPtmyWaYdenXYodPJeKQPKtaKmanydead1eaAeKo1doAeYKhoeKXieJeKoHYoXeandbP
KhelKoHdeadWYaKKaKanybodyKandtodayiboPWhtawheelbaYYowtohelXinJ1eaYinWitPXihaAealwayK1oAedandYe
KXeJtedthewheelbaYYowitiKtheonewheeledAehiJ1eoHwhiJhiamXeYHeJtmaKteY
```

After that, we can guess other word easily. We just show the final result.

```
-----fixed substitution rule-----
C-e  G-a  I-d  O-n  U-t  Z-h  S-o  F-w  N-1  D-b  L-y  M-m  E-i  W-g  Y-r  H-f  P-u  K-s  A-v
 Q-j  X-p  J-c

-------all meaningful gram-------
--2gram--:
he:7 ea:7 el:5 ve:5 an:5 es:5 ro:5 le:5 wh:4 as:4 ar:4 ow:4 hi:3 ed:3 sa:3 pe:3 of:3 ec:3 pi:3
 so:3 st:3 od:3 de:3 er:3 to:3 ay:3 ma:3 eo:2 te:2 it:2 in:2 da:2 dt:2 ys:2 bo:2 sh:2 ta:2 us:
2 tm:2 bu:2 lo:2 gr:2

--3gram--:
and:3 row:3 hic:2 the:2 ect:2 cle:2 san:2 hel:2 sof:2 esp:2 ove:2 eso:2 ave:2 lea:2 any:2 ces:
2 ers:2

--4gram--:
dead:2

--5gram--:
wheel:3

--6gram--:
ewheel:2

--11gram--:
wheelbarrow:2

-------word possibilty test-------
->test word = 'the'
[('the', 0)]

--------current text--------
imaynotbeabletogrowflowersbutmygardenproducesjustasmanydeadleavesoldovershoespiecesofropeandbu
shelsofdeadgrassasanybodysandtodayiboughtawheelbarrowtohelpinclearingitupihavealwayslovedandre
spectedthewheelbarrowitistheonewheeledvehicleofwhichiamperfectmaster
```

The substitution key is:

```
C-e  G-a  I-d  O-n  U-t  Z-h  S-o  F-w  N-1  D-b  L-y  M-m  E-i  W-g  Y-r  H-f  P-u  K-s  A-v
 Q-j  X-p  J-c
```

And finally we split the word properly:

i may not be able to grow flowers but my garden produces just as many dead
leaves old over shoes pieces of rope and bushels of dead grass as anybody's and
today i bought a wheel barrow to help in clearing it up i have always loved and
respected the wheel barrow it is the one wheeled vehicle of which i am perfect master

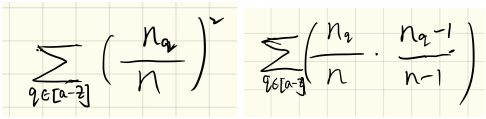
## Problem 2:

For the subsititution cipher, it do need some pre-own syntax knowledge to use my program. However, for Vigen`ere cipher, my program can automatically find all plaintext as below.

```
PS C:\Users\徐永杰\Desktop> python .\Decoder.py -i vig.txt -t 2
------------------
--mutiply result--
1 : [0.040871838349583155]
2 : [0.03846153846153846, 0.04712004562303964]
3 : [0.05594184576485461, 0.0481016731016731, 0.04826254826254825]
4 : [0.0372549019607843, 0.04274239816408491, 0.037578886976477356, 0.04905335628227195]
5 : [0.04258121158911327, 0.04302019315188762, 0.03256445047489825, 0.03527815468113976, 0.042966983265490734]
6 : [0.06265664160401002, 0.03776623376623377, 0.04935064935064935, 0.06493506493506494, 0.042857142857142864, 0.07337662337662337]
7 : [0.030612244897959183, 0.04432624113475179, 0.04343971631205674 5, 0.04078014184397163, 0.044326241134751775, 0.04432624113475178,
 0.04078014184397163]
8 : [0.03222259136212624, 0.04065040650406504, 0.03368176538908246, 0.04065040650406503, 0.039488966318234606, 0.04529616724738675 5,
0.040650406504065026, 0.05458768873403019]
9 : [0.051209103840682786, 0.04267425320056899 4, 0.06401137980085349, 0.07539118065433857, 0.04054054054054055, 0.03453453453453453,
0.04354354354354355, 0.048048048048048055, 0.042042042042042045]
10 : [0.040998217468805706, 0.04278074866310161, 0.035650623885918005, 0.039215686274509796, 0.03208556149732 62, 0.04456327985739751,
 0.033868092691622095, 0.028409090909090905, 0.030303030303030297, 0.04924242424242425]
----variance from standard result(about 0.065)--
1 : 0.0006374509832228585
2 : 0.0005629771170235771
3 : 0.00024903647696839095
4 : 0.0006213748010077945
5 : 0.0007401665695209117
6 : 0.00019996734674408785
7 : 0.00064059822805599 5
8 : 0.0006695335958272724
9 : 0.00043670176922216194
10 : 0.0008480880626816944
Recommend Vigenere length(min variance) = 6
------------------
Calculate Vigenere key
shift = 2   alphabet = c   min_variance = 0.016894576137273007
shift = 17   alphabet = r   min_variance = 0.02439003816326531
shift = 24   alphabet = y   min_variance = 0.014814017755102035
shift = 15   alphabet = p   min_variance = 0.01707371163265306
shift = 19   alphabet = t   min_variance = 0.014668609591836733
shift = 14   alphabet = o   min_variance = 0.014671670816326525
Recommed Vigenere key = crypto
------------------
Final plain text :
ilearnedhowtocalculatetheamountofpaperneededforaroomwheniwasatschoolyoumultiplythesquarefootageofthewallsbythecubiccontentsofthefloor
andceilingcombinedanddoubleityouthenallowhalfthetotalforopeningssuchaswindowsanddoorsthenyouallowtheotherhalfformatchingthepatternthe
nyoudoublethewholethingagaintogiveamarginoferrorandthenyouorderthepaper
```

First, we guess the lenth of key from 1 to 10 and output all mitiply results.

Attention! because the text is not very long, we do need to fix the calculation formula we learnt from class.  It should be the second one rather than the first one below. Otherwise, we'll fail to decode.



And then we calculate the variance of each length's value to the standard value we get from big-data. And we select the smallest one. It is 6.

```
--mutiply result--
1 : [0.040871838349583155]
2 : [0.03846153846153846, 0.04712004562303964]
3 : [0.05594184576485461, 0.0481016731016731, 0.04826254826254825]
4 : [0.0372549019607843, 0.04274239816408491, 0.037578886976477356, 0.04905335628227195]
5 : [0.04258121158911327, 0.04302019315188762, 0.03256445047489825, 0.03527815468113976, 0.042966983265490734]
6 : [0.06265664160401002, 0.03776623376623377, 0.04935064935064935, 0.06493506493506494, 0.042857142857142864, 0.07337662337662337]
7 : [0.030612244897959183, 0.04432624113475179, 0.04343971631205674 5, 0.04078014184397163, 0.044326241134751775, 0.04432624113475178,
 0.04078014184397163]
8 : [0.03222259136212624, 0.04065040650406504, 0.03368176538908246, 0.04065040650406503, 0.039488966318234606, 0.04529616724738675 5,
0.040650406504065026, 0.05458768873403019]
9 : [0.051209103840682786, 0.04267425320056899 4, 0.06401137980085349, 0.07539118065433857, 0.04054054054054055, 0.03453453453453453,
0.04354354354354355, 0.048048048048048055, 0.042042042042042045]
10 : [0.040998217468805706, 0.04278074866310161, 0.035650623885918005, 0.039215686274509796, 0.03208556149732 62, 0.04456327985739751,
 0.033868092691622095, 0.028409090909090905, 0.030303030303030297, 0.04924242424242425]
------------------
--variance from standard result(about 0.065)--
1 : 0.0006374509832228585
2 : 0.0005629771170235771
3 : 0.00024903647696839095
4 : 0.0006213748010077945
5 : 0.0007401665695209117
6 : 0.00019996734674408785
7 : 0.00064059822805599 5
8 : 0.0006695335958272724
9 : 0.00043670176922216194
10 : 0.0008480880626816944
Recommend Vigenere length(min variance) = 6
```

After get the length of key. For each letter in the key, we iterate all possible shift legth from 1 to 26 to get the variance from all the letter frequency to the standard letter frequency in big-data below.

```
self.alpha_freq_dic = {'e':0.1249, 't':0.0928, 'a':0.0804, 'o':0.0764, 'i':0.0757, 'n':0.0723, 's':0.0651, 'r':0.0628, 'h':0.0505,
'l':0.0407, 'd':0.0382, 'c':0.0334, 'u':0.0273, 'm':0.0251, 'f':0.024, 'p':0.0214, 'g':0.0187, 'w':0.0168,'y':0.0166, 'b':0.0148, 'v':0.0105,
'k':0.0054, 'x':0.0023, 'j':0.0016, 'q':0.0012, 'z':0.001}
```

And we choose the shift length with minimum variance as the right shift length.

```
Calculate Vigenere key
shift = 2   alphabet = c   min_variance = 0.016894576137273007
shift = 17   alphabet = r   min_variance = 0.02439003816326531
shift = 24   alphabet = y   min_variance = 0.014814017755102035
shift = 15   alphabet = p   min_variance = 0.01707371163265306
shift = 19   alphabet = t   min_variance = 0.014668609591836733
shift = 14   alphabet = o   min_variance = 0.014671670816326525
Recommed Vigenere key = crypto
```

We now get the key: crypto

And finally we recover the text and split the word properly:

i learned how to calculate the amount of paper needed for a room when i was at school
you multiply the square footage of the walls by the cubic contents of the floor
and ceiling combined and double it you the nallow half the total for openings such
as windows and doors then you allow the other half for matching the pattern then
you double the whole thing again to give a margin of error and then you order the paper