Projet d'Analyse des données

Cécile Poulard et Elliot Sisteron

Table des matières

Introduction

Depuis toujours, nous avons rencontré la nécessité de protéger certaines informations de la vue d'autrui. Aujourd'hui où les données s'étalent à n'en plus pouvoir les dénombrer, et où celles-ci représentent parfois des transactions d'une grande importance ; À l'ère où les géants de l'internet récupèrent des renseignements sur chacun d'entre nous, et où nous comptons plus que tout sur la fiabilité de ces services pour conserver notre vie privée, il est primordial de mettre l'accent sur la sécurité de nos informations. C'est pourquoi nous allons nous intéresser dans ce projet aux codes secrets qui ont marqué l'histoire, tant par leur fiabilité apparente au premier abord, que par les conséquences de leur déchiffrement.

Nous nous attarderons dans une première partie sur les chiffres [1] dits de « substitution », qui consistent à remplacer chaque lettre du texte clair par une autre. Parmi ceux-ci, on choisira les codes de César et de Vigenère, mais aussi un chiffre de substitution mono-alphabétique. On se proposera surtout d' « attaquer » ces chiffres sur des cryptogrammes [2], à l'aide des différents outils que le cours de d'analyse des données nous a fourni. Nous présenterons tout d'abord rapidement chacun de ces codes. Puis, nous aurons besoin de faire une analyse statistique sur la langue française et sur les textes à déchiffrer. Nous verrons ici que cette étude statistique fut à la base même de la cryptanalyse [3] pendant plus de 1500 ans. Nous ferons donc des hypothèses grâce aux données précédemment analysées et nous étudierons la véracité de ces suppositions. Si celles-ci s'avèrent cohérentes, nous pourrons enfin les appliquer à la résolution des chiffres et en déduire les textes en clair.

1. Codes secrets
2. Textes chiffrés
3. Étude du déchiffrement

Les premiers codes secrets

1.1 Le chiffre de César

Le chiffre de César doit son nom à l'empereur Jules César qui l'utilisait pour chiffrer ses communications. Celui-ci consistait en un simple « décalage » de l'alphabet :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

Table 1.1 – Un exemple de décalage César

Chaque lettre est substituée à une autre grâce à une clé : un nombre compris entre 1 et 25. Dans l'exemple ci-dessus, la clé est 2 car l'alphabet chiffré est décalé de deux lettres vers la gauche par rapport à l'alphabet clair.

Mathématiquement, si l'on numérote les lettres de l'alphabet de 0 à 25, et que l'on note :

— x la lettre à chiffrer
— y la lettre résultante du chiffrement
— c la clé comprise entre 1 et 25 du chiffre

On a la relation suivante :

$$y \equiv x + c \pmod{26} \Leftrightarrow x \equiv y - c \pmod{26}$$

On constate ici qu'un chiffre comme celui-ci peut facilement être brisé en utilisant une « attaque par force brute » [1] car il n'y a que 25 clés possibles.

1.2 Le chiffre de substitution mono-alphabétique

Ce type de chiffrement fut utilisé pendant longtemps, et le chiffre de César en est un cas particulier. Son utilisation dura plus d'un millénaire, jusqu'à son déchiffrement au IX siècle par le philosophe arabe Al-Kindi.

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**U**	**N**	**E**	**C**	**L**	A	B	D	F	G	H	I	J	K	M	O	P	Q	R	S	T	V	W	X	Y	Z

Table 1.2 – Un exemple de substitution mono-alphabétique

Là aussi, chaque lettre est substituée à une autre grâce à une clé, mais il s'agit maintenant d'un mot. Dans notre exemple, le mot utilisé est « UNECLE », le début de l'alphabet chiffré est définit par ce mot-clé, alors que le reste n'est que la suite ordonnée des lettres non-utilisées dans

1. Tentative de « casser » le chiffre en essayant toutes les clés possibles

la clé. La clé peut aussi être un nouvel alphabet (constitué de lettres connues ou bien de signes, libre à l'imagination de celui qui s'improvise cryptographe). Ce chiffre est bien plus intéressant que le précédent : il y a 26! arrangements de l'alphabet possibles, une « attaque par force brute » a donc très peu de chance d'aboutir. De plus, il est relativement intuitif, pour chiffrer un message on aura tendance à vouloir remplacer une à une chaque lettre du message par une autre. C'est pour ces raisons qu'il fut omniprésent tout au long de l'histoire, même après son déchiffrement ; Ce qui coûta, entre autre, sa vie à Marie Stuart, reine d'Écosse.

1.3 Le chiffre de Vigenère

Ce code secret a vu le jour au XVI siècle, il fut mis en place pour contrer les faiblesses du chiffre mono-alphabétique. Le chiffre de Vigenère a le bon goût d'être un chiffre poly-alphabétique, c'est-à-dire qu'il utilise plusieurs chiffres mono-alphabétique lors du chiffrement. En l'occurence, l'idée principale est de « superposer » plusieurs alphabets de César à l'aide d'un mot-clé, et de les utiliser un à un à chaque lettre.

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet 1	**U**	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
Alphabet 2	**N**	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
Alphabet 3	**E**	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
Alphabet 4	**C**	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
Alphabet 5	**L**	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
Alphabet 6	**E**	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D

TABLE 1.3 – Un exemple de chiffrement Vigenère

La taille du mot-clé définit le nombre d'alphabets que l'on utilisera. Les lettres qui forment le mot-clé définissent quant à elles les alphabets de César utilisés dans le chiffrement. Ici, la clé utilisée est aussi « UNECLE ». Dans cet exemple, pour chiffrer un message, il suffit de chiffrer la première lettre avec l'alphabet 1, la deuxième avec l'alphabet 2, ..., la septième avec l'alphabet 1, etc. Si l'on note k la taille du mot-clé, on a donc 26^k chiffres possibles. Il a été prouvé mathématiquement que ce chiffre est « *incassable* » sous certaines conditions : la clé doit être de la taille du message à chiffrer, unique, et elle se doit aussi d'être générée aléatoirement. Pour des raisons pratiques, on ne l'utilise pas aujourd'hui : la taille de la clé deviendrait trop importante et, surtout, il est impossible de reproduire un phénomène purement aléatoire informatiquement (ce genre de phénomène n'existant que dans la nature).

1.4 Les cryptogrammes à déchiffrer dans ce projet

On essaiera dans ce projet de déchiffrer les cryptogrammes suivants :

— Cryptogramme n°1 :

« FILMK OYFYP CHYHN LYFYM YWLYN MILNF YNUFG OX »

— Cryptogramme n°2 :

« JHCKG TDWNF WAXXV WQKWA GIIKE NQDMT QLRXJ GBGMG IWMEO NDSWL MKWMC SHBWF AHQKW CMVCB AJSLC ESKUS JBMSI CWMFW PMWTD WQOWM GCMVW KIAGJ CWQXA LRTJG FWLWG KMFQH VLFXA VSLOW BLYMW FWFHM MFRNT SATQF DXCLS MZ- WON VECFM FHHCB SGMHC NDSWL XSGHC BSMIA GLMMZ VPWNF WASMK WGMKM FBMML WMKSW QJSJC WZXAZ OLIJR LTWGK MF- QHV LFXAX CKOWB MCFSW MKHBV WSIIJ QXYMS JCSBW WFOEM YCNBV SEIUV HAWEN IFRHV SZXOG IMLWZ TKZCL MTWXV XOBBW ZXJWO NOWGM MHOKN GWLWF BXBJC NDWDT ADWGB WFEWU IMMMF XVXOV MBSWQ JOBAD SFQJC BZIIB DGILI ARXIS JTVUS KIDCK AUSGM KHIIK AHVUO LKGAF MBSWQ KOBAD OICAG JCWAH QSIVW FHKIA FXRSW ICWHC MVWLU WFVQS ZTDAS CMDIB LAGFM JQBRW QAIFH XTSJB MBSWI FGXTS JBMBS GMKIB AIITU GIKML TBVSZ XUWBM YMOGL TSTCU CNXVS ZMFGF MVWLM FHFIA GVWEA XVLTT QKHNX GIKIN CBZUS MBWVN USBBB WXXTW IKZWD HVVGM ZWGLQ EDEME SGBBS EMMFW QKENM USLBU SZWMH WMDOF WMFJC AATXG ILAWO NRGIK LZIBI WBMZW DKMFR KMMBX KGBLB JIVBA CGUWQ TVAEN MEOBA VSFIA BJCAG TQLDX CLSMZ WGXCD SFMFH TUWAX BLFXI MGXZN WVMVS EIUCF UMBTC LSTNS WKMDS WWFZX LGBWM KCB »

— Cryptogramme n°3 :

« OHQSB SAQSA MTOQF BFJAN OQLPB HNQHJ GEPOQ OHBHL FBAQF BWJUF BPLOH TGEOP QBEPL FFHOX KPAGO FOUBA SMTOF OQRBP BRSPA QSAMT OQNTH RIBHS AFFJH BFBSJ APOQO PBKKP JRIOH SNBTS BHSKF TQNOQ RBPBR SPAQS AMTOQ QSBSA QSAMT OQNOF BKJKT FBSAJ HMTOF BSBAF FONOF RIBHS AFFJH BTLGO HSOFB SBAFF ONOFR IBHSA FFJHR JHQAN POPKJ TPBKK PJRIO PFOQR BPBRS PAQSA MTOQN OFBKJ KTFBS AJHHO NKOHN MTOUB AEFOG OHSVJ APOKB QNTSJ TSNOF BSBAF FONOR OFFOR AMTOF OQJHN BLOQJ ASUBA SBTFT XOGEJ TPLJT BTXSB SQTHA QAFQT UUASK JTPJE SOHAP NOQKP RAQAJ HQLBF OQNOK POHNP ONOQR IBHSA FFJHQ NOSBA FFOQL BFOQ »

Calcul des fréquences

Dans cette partie nous étudierons des données sur l'utilisation des lettres de l'alphabet et des bigrammes dans la langue française et dans les textes à déchiffrer. On calculera en particulier les *indices de coïncidence* de chaque cryptogramme, pour savoir par quel cryptosystème [1] est chiffré chaque texte. Nous nous intéresserons aussi aux mots les plus fréquents, car ceux-là auront une forte probabilité d'apparaître dans les cryptogrammes. On se propose de récupérer ces données, pour ensuite essayer de les représenter.

2.1 Génération des données

Nous essayons d'avoir une estimation correcte de la fréquence d'utilisation de chaque lettre de l'alphabet en français. Pour cela, il est nécessaire de choisir un ou plusieurs textes sur lesquels on utilisera un script [2] de calcul des effectifs de chaque lettre. Ici, nous avons choisi les œuvres suivantes :

— *La fille aux yeux d'or*, Honoré de Balzac
— *Les antiquités de Rome*, Joachim du Bellay
— *Madame Bovary*, Gustave Flaubert
— *Notre dame de Paris*, Victor Hugo
— *La science et l'hypothèse*, Henri Poincaré
— *Gargantua*, François Rabelais
— *Le tour du monde en quatre-vingt jours*, Jules Verne
— *Candide*, Voltaire
— *Germinal*, Émile Zola

Étant donné que les caractères spéciaux (é, è, à...) sont remplacés par leurs parents lors du chiffrement, le « é » compte donc pour un « e », le « à » pour un « a », etc.

De plus, nous recherchons aussi la fréquence de chaque bigramme dans la langue française. Pour ce faire, nous appliquons un autre script [3] sur les textes énumérés précédemment.

Enfin, pour pouvoir « casser » des codes César et mono-alphabétique, nous aurons besoin de calculer, de même, la fréquence de chaque lettre, chaque bigramme des textes (en utilisant les scripts précédents). Pour ce qui est d'un chiffre de Vigenère, il faut faire la supposition que le mot-clé ne dépasse pas les k lettres. Autrement dit, il sera nécessaire de trouver le naturel k qui est statistiquement le plus plausible. Il est inutile pour un tel chiffre d'étudier les bigrammes car celui-ci est poly-alphabétique : ces fréquences n'auraient pas de sens.

2.2 Indice de coïncidence

Étant donné que nous ne savons pas quel code est appliqué sur chaque cryptogramme, il nous est nécessaire d'avoir un outil nous permettant d'avoir une idée du chiffre utilisé. L'indice de coïncidence d'un texte permet de savoir si chaque lettre est uniformément distribuée ou non, ce qui permet d'en déduire la nature mono-alphabétique ou poly-alphabétique du texte étudié. Cet outil a été inventé par William F. Friedman en 1920, et c'est pour cette raison qu'on appelle le

1. Mécanisme de chiffrement et de déchiffrement
2. Ce script est disponible en annexe
3. Disponible en annexe

test appliqué sur les cryptogrammes le test de Friedman. L'indice de coïncidence est défini par la formule suivante :

$$IC = \sum_{l=A}^{Z} \frac{n_l(n_l - 1)}{n(n-1)}$$

avec :

— IC l'indice de coïncidence
— n_l les effectifs de la lettre l dans le texte
— n le nombre total de lettres dans le texte

En fait, si on pose E_l l'événement « La lettre l apparaît deux fois de suite dans le tirage sans remise de deux lettres au hasard dans le texte » on peut dire ici que :

$$IC = \sum_{l=A}^{Z} \mathbb{P}(E_l)$$

Il s'agit donc ici de la probabilité d'obtenir une paire de lettres quelconques en tirant deux lettres dans le texte étudié.

2.3 Description des données

Les scripts précédents nous ont permis d'obtenir les effectifs de chaque lettre/bigramme dans un fichier donné. Toutefois, pour se faire une idée de l'utilisation de chaque élément, il nous est nécessaire de calculer la fréquence d'apparition de ceux-ci. De plus, on aimerait pouvoir se représenter ces données avec des histogrammes. Pour cela, on traite les données des scripts avec MatLab [4].

4. Le code source de ce traitement MatLab est disponible en annexe

2.3.1 Fréquence de chaque lettre en français

	Nb	%
A	279823	8.59
B	33102	1.02
C	100170	3.08
D	120611	3.70
E	557202	17.11
F	35930	1.10
G	34934	1.07
H	32630	1.00
I	239698	7.36
J	13668	0.42
K	508	0.02
L	192759	5.92
M	90806	2.79
N	225182	6.91
O	173499	5.33
P	88040	2.70
Q	38380	1.18
R	213830	6.57
S	257287	7.90
T	238266	7.32
U	207131	6.36
V	51963	1.60
W	348	0.01
X	13849	0.43
Y	11602	0.36
Z	5614	0.17
Total	3256832	100.00

TABLE 2.1 – Résultat des calculs de fréquences

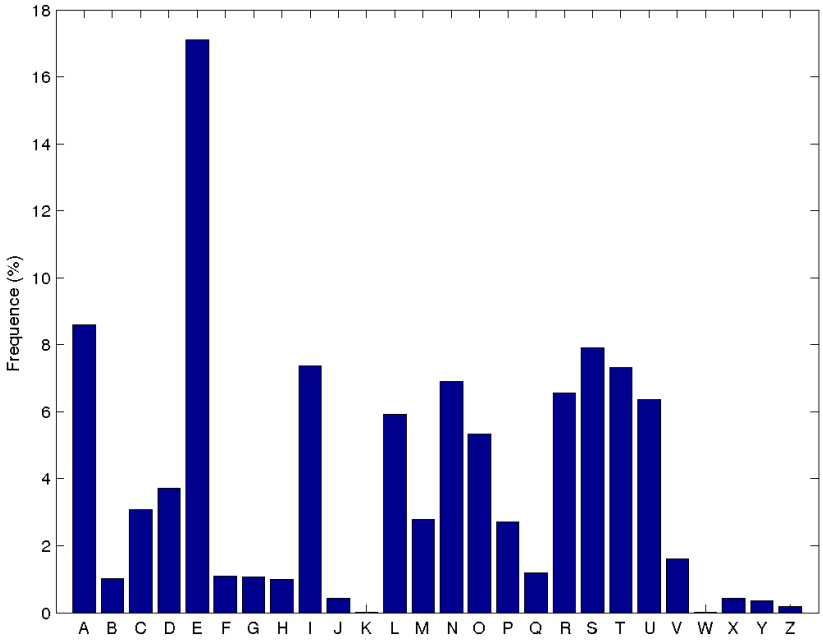

FIGURE 2.1 – Visualisation des fréquences de chaque lettre

	Effectifs	Fréquences
Moyenne	125263	3.85
Variance	16711876115	15.76
Écart-type	129274	3.97
Médiane	89423	2.75

TABLE 2.2 – Description des variables

$$\boxed{IC = 0,0779}$$

TABLE 2.3 – Indice de coïncidence

Cet indice de coïncidence correspond, à 10^{-4} près, à celui que l'on retrouve sur *Wikipédia*, on peut donc en déduire que les données générées sont très représentatives de la langue française. On fera donc l'approximation que les fréquences que l'on a calculées sont les probabilités réelles de chaque lettre.

2.3.2 Fréquence de chaque bigramme en français

Ici, on ne montrera que les 26 bigrammes les plus fréquents en français (les autres ne nous seront pas utiles). De plus, on s'intéresse aussi à la fréquence de chaque paire de lettres.

	Nb	(%)
LE	12124	1.19
EN	11990	1.18
RE	11730	1.15
OU	11326	1.11
ON	11315	1.11
IT	11076	1.09
ES	11034	1.09
NT	10973	1.08
AI	10924	1.07
DE	10907	1.07
QU	10319	1.02
AN	10218	1.01
UR	9756	0.96
ER	9724	0.96
TE	9664	0.95
LA	9473	0.93
IS	9408	0.93
IL	9227	0.91
NE	9195	0.90
SE	9150	0.90
ME	9141	0.90
IE	9100	0.90
EU	9096	0.90
IN	8865	0.87
AR	8734	0.86
PA	8454	0.83

TABLE 2.4 – Fréquences sur les bigrammes

	Nb	(%)
AA	10	0.00
BB	122	0.01
CC	1359	0.13
DD	28	0.00
EE	66	0.01
FF	2588	0.25
GG	631	0.06
HH	6	0.00
II	184	0.02
JJ	0	0.00
KK	0	0.00
LL	7771	0.76
MM	5063	0.50
NN	4413	0.43
OO	103	0.01
PP	2360	0.23
QQ	0	0.00
RR	3954	0.39
SS	6746	0.66
TT	4349	0.43
UU	2	0.00
VV	0	0.00
WW	0	0.00
XX	60	0.01
YY	0	0.00
ZZ	0	0.00

TABLE 2.5 – Fréquences sur les paires

	Effectifs	Fréquences
Moyenne	1503	0.15
Variance	7169716	0.07
Écart-type	2678	0.26
Médiane	46	0.00

TABLE 2.6 – Description des variables

On se restreint à étudier les bigrammes sur un texte plus petit (environ 1 million de bigrammes), car le script bash ne fonctionne pas autrement. Les fréquences sont donc un peu moins précises, mais rien d'affolant. On fera, là aussi, l'approximation que les fréquences calculées correspondent à la probabilité d'apparition de chaque bigramme.

2.3.3 Fréquence de chaque lettre dans les cryptogrammes

Pour le cryptogramme n°1

	Nb	%
A	0	0.00
B	0	0.00
C	1	2.70
D	0	0.00
E	0	0.00
F	5	13.51
G	1	2.70
H	2	5.41
I	2	5.41
J	0	0.00
K	1	2.70
L	4	10.81
M	3	8.11
N	4	10.81
O	2	5.41
P	1	2.70
Q	0	0.00
R	0	0.00
S	0	0.00
T	0	0.00
U	1	2.70
V	0	0.00
W	1	2.70
X	1	2.70
Y	8	21.62
Z	0	0.00
Total	37	100.00

TABLE 2.7 – Résultat des calculs de fréquences

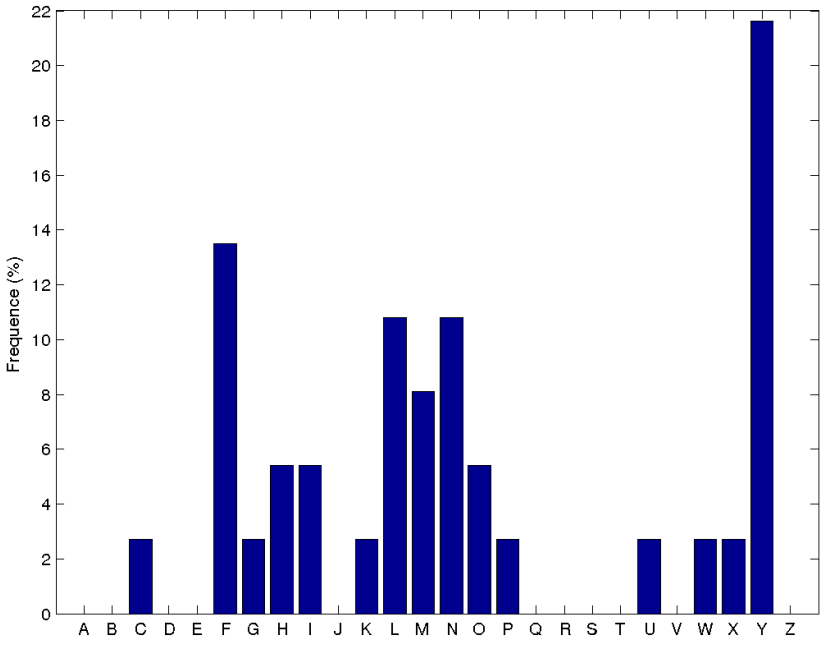

FIGURE 2.2 – Visualisation des fréquences de chaque lettre

	Effectifs	Fréquences
Moyenne	1.42	3.85
Variance	3.85	28.15
Écart-type	1.96	5.31
Médiane	1.00	2.70

TABLE 2.8 – Description des variables

$$\boxed{IC = 0.0841}$$

TABLE 2.9 – Indice de coïncidence

Avec un IC aussi proche du français, on peut faire l'hypothèse que ce cryptogramme résulte d'un chiffrement mono-alphabétique.

Pour le cryptogramme n°2

	Nb	%
A	38	4.53
B	52	6.21
C	38	4.53
D	22	2.63
E	17	2.03
F	44	5.25
G	41	4.89
H	25	2.98
I	44	5.25
J	23	2.74
K	35	4.18
L	36	4.30
M	75	8.95
N	21	2.51
O	21	2.51
P	2	0.24
Q	23	2.74
R	10	1.19
S	52	6.21
T	27	3.22
U	18	2.15
V	33	3.94
W	80	9.55
X	36	4.30
Y	4	0.48
Z	21	2.51
Total	838	100.00

TABLE 2.10 – Résultat des calculs de fréquences

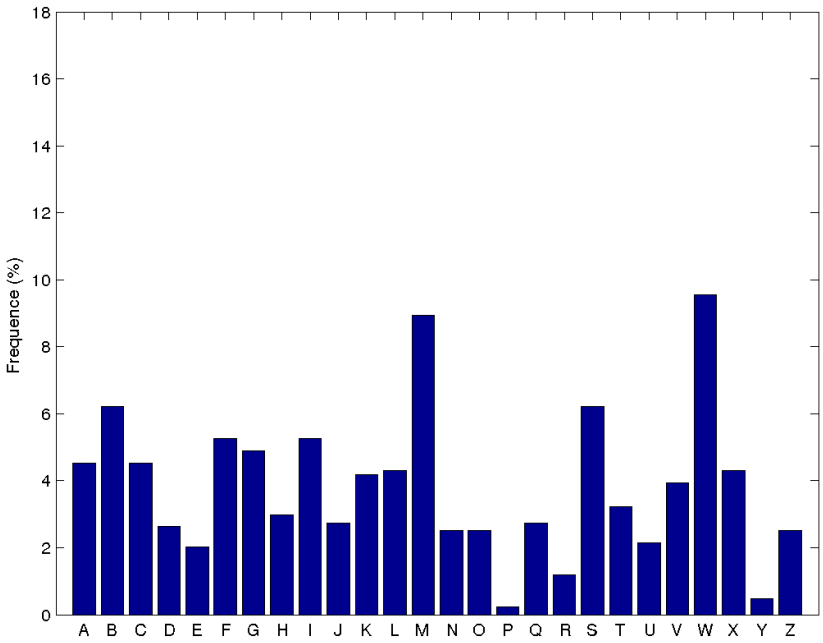

FIGURE 2.3 – Visualisation des fréquences de chaque lettre

	Effectifs	Fréquences
Moyenne	32.23	3.85
Variance	348.90	4.97
Écart-type	18.68	2.23
Médiane	30.00	3.58

TABLE 2.11 – Description des variables

$$\boxed{IC = 0.0497}$$

TABLE 2.12 – Indice de coïncidence

Il est clair que compte tenu de l'histogramme, de la variance et de l'IC aussi différent du français, le chiffre utilisé est poly-alphabétique. Il sera donc inutile d'étudier les bigrammes de ce texte.

Pour le cryptogramme n°3

	Nb	%
A	37	7.61
B	48	9.88
C	0	0.00
D	0	0.00
E	6	1.23
F	45	9.26
G	6	1.23
H	30	6.17
I	6	1.23
J	25	5.14
K	16	3.29
L	9	1.85
M	9	1.85
N	21	4.32
O	73	15.02
P	27	5.56
Q	37	7.61
R	16	3.29
S	37	7.61
T	27	5.56
U	6	1.23
V	1	0.21
W	1	0.21
X	3	0.62
Y	0	0.00
Z	0	0.00
Total	486	100.00

TABLE 2.13 – Résultat des calculs de fréquences

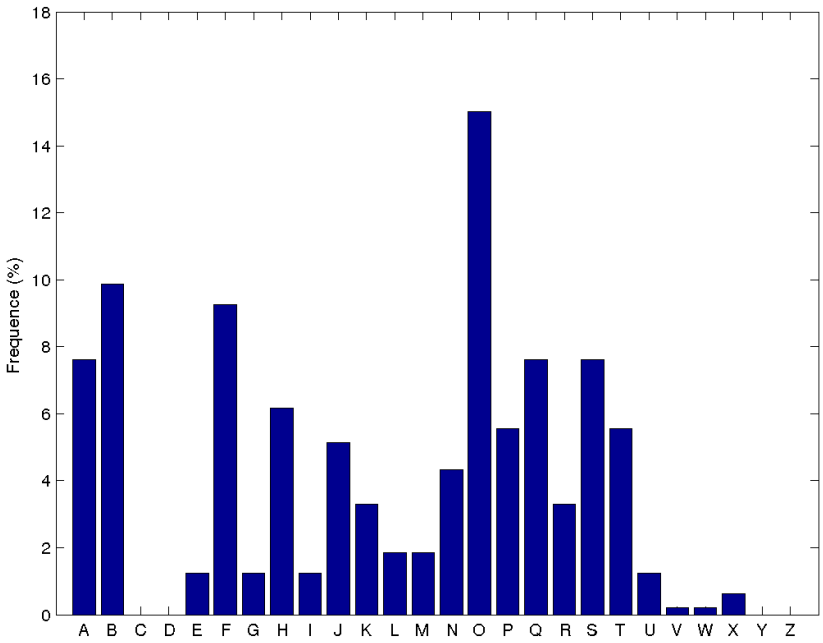

FIGURE 2.4 – Visualisation des fréquences de chaque lettre

	Effectifs	Fréquences
Moyenne	18.69	3.85
Variance	357.34	15.13
Écart-type	18.90	3.89
Médiane	12.50	2.57

TABLE 2.14 – Description des variables

$$IC = 0.0744$$

TABLE 2.15 – Indice de coïncidence

Au vu de ces données, on peut aussi faire l'hypothèse que le cryptosystème utilisé est mono-alphabétique.

2.3.4 Fréquence de chaque bigramme dans les cryptogrammes

Ici encore, on ne s'interessera qu'aux 26 bigrammes les plus fréquents et aux paires de lettres.

Pour le cryptogramme n°1

	Nb	(%)
CH	1	4.17
FG	1	4.17
FI	1	4.17
FY	1	4.17
HN	1	4.17
HY	1	4.17
IL	1	4.17
LM	1	4.17
LN	1	4.17
LY	1	4.17
MI	1	4.17
MK	1	4.17
NF	1	4.17
NU	1	4.17
OX	1	4.17
OY	1	4.17
UF	1	4.17
WL	1	4.17
YF	1	4.17
YH	1	4.17
YM	1	4.17
YN	1	4.17
YP	1	4.17
YW	1	4.17
AA	0	0.00
AB	0	0.00

TABLE 2.16 – Fréquences sur les bigrammes

	Nb	(%)
AA	0	0.00
BB	0	0.00
CC	0	0.00
DD	0	0.00
EE	0	0.00
FF	0	0.00
GG	0	0.00
HH	0	0.00
II	0	0.00
JJ	0	0.00
KK	0	0.00
LL	0	0.00
MM	0	0.00
NN	0	0.00
OO	0	0.00
PP	0	0.00
QQ	0	0.00
RR	0	0.00
SS	0	0.00
TT	0	0.00
UU	0	0.00
VV	0	0.00
WW	0	0.00
XX	0	0.00
YY	0	0.00
ZZ	0	0.00

TABLE 2.17 – Fréquences sur les paires

	Effectifs	Fréquences
Moyenne	0.04	0.15
Variance	0.03	0.60
Écart-type	0.19	0.77
Médiane	0.00	0.00

TABLE 2.18 – Description des variables

Sur un texte aussi petit, on se rend rapidement compte que l'analyse des bigrammes ne sera en rien efficace.

Pour le crytogramme n°3

	Nb	(%)
AE	1	0.74
AF	1	0.74
AG	1	0.74
AJ	1	0.74
AM	1	0.74
AN	1	0.74
AP	1	0.74
AQ	1	0.74
AS	1	0.74
BA	1	0.74
BE	1	0.74
BF	1	0.74
BH	1	0.74
BK	1	0.74
BL	1	0.74
BP	1	0.74
BR	1	0.74
BS	1	0.74
BT	1	0.74
BW	1	0.74
EF	1	0.74
EJ	1	0.74
EO	1	0.74
EP	1	0.74
FB	1	0.74
FF	1	0.74

TABLE 2.19 – Fréquences sur les bigrammes

	Nb	(%)
AA	0	0.00
BB	0	0.00
CC	0	0.00
DD	0	0.00
EE	0	0.00
FF	1	0.74
GG	0	0.00
HH	1	0.74
II	0	0.00
JJ	0	0.00
KK	1	0.74
LL	0	0.00
MM	0	0.00
NN	0	0.00
OO	0	0.00
PP	0	0.00
QQ	0	0.00
RR	0	0.00
SS	0	0.00
TT	0	0.00
UU	1	0.74
VV	0	0.00
WW	0	0.00
XX	0	0.00
YY	0	0.00
ZZ	0	0.00

TABLE 2.20 – Fréquences sur les paires

	Effectifs	Fréquences
Moyenne	0.20	0.15
Variance	0.16	0.09
Écart-type	0.40	0.30
Médiane	0.00	0.00

TABLE 2.21 – Description des variables

Là encore, l'analyse ne nous sera pas très utile (l'effectif des bigrammes les plus fréquents est toujours de 1). Mais, l'étude des paires de lettres pourra nous aider.

2.3.5 Les mots les plus fréquents du français

Après un script de récupération des effectifs de chaque mot et un petit traitement MatLab [5] pour déterminer les 20 mots (de plus de deux lettres) les plus fréquents dans le texte, on a le résultat suivant :

	Nb	**(%)**
LES	12940	1.81
QUE	8184	1.14
CES	7905	1.11
QUI	6679	0.93
UNE	5893	0.82
DANS	5428	0.76
ÉTAIT	4528	0.63
PAS	4495	0.63
ELLE	4202	0.59
SON	3840	0.54
POUR	3720	0.52
SUR	3682	0.51
PLUS	3477	0.49
PAR	3474	0.49
LUI	3156	0.44
MAIS	2839	0.40
COMME	2786	0.39
AVAIT	2712	0.38
TOUT	2608	0.36
VOUS	2592	0.36

TABLE 2.22 – Fréquences sur les mots

5. En annexe

Application aux codes secrets

Dans cette partie, on essaiera de déchiffrer les textes codés. On utilisera principalement des techniques qui s'appliquent à un chiffre de substitution mono-alphabétique, car pour déchiffrer le code de Vigenère on se ramène à ce cas particulier. Pour cela, nous devrons faire des hypothèses sur ces textes grâce aux données précédemment présentées, par exemple :

$$\mathcal{H} : \text{« Dans ce cryptogramme, la lettre } E \text{ représente la lettre } y \text{ ».}$$

Nous aurons donc besoin d'outils qui évaluent la véracité de telles suppositions. Nous pourrons enfin appliquer ces hypothèses aux chiffres.

3.1 Test des hypothèses en cryptanalyse

Dans cette partie, on appelera :
- E un ensemble d'éléments (par exemple les monogrammes, les bigrames, ..., les mots)
- Objet $o \in E$ un élément de E (par exemple, un monogramme, un bigramme, etc.)
- O la variable aléatoire qui représente un objet (par exemple, si on est dans l'ensemble des monogrammes $\mathbb{P}(O = a)$ est la probabilité de tomber sur le monogramme a)

On va donc appliquer des tests qui compareront $\mathbb{P}(O = o)$, que l'on a estimé, et la fréquence d'apparition de chaque objets dans un cryptogramme.,

3.1.1 Distance entre deux éléments

L'hypothèse à vérifier

On cherche à évaluer si un élément (monogramme, bigramme, etc.) du texte chiffré est en relation avec un élément de la langue française. Autrement dit, on s'intéresse aux hypothèses suivantes :
- $\mathcal{H}_0$: « L'objet o de la langue française est chiffré par l'objet o' dans le cryptogramme »
- $\mathcal{H}_1$: « L'objet o de la langue française n'est pas chiffré par l'objet o' dans le cryptogramme »

Définition

L'approche la plus intuitive est d'évaluer l'écart de la fréquence d'apparition de chacun de ces éléments :

$$d(o, o') = |\hat{P}(O = o)_{cryptogramme} - \mathbb{P}(O = o')|$$

Pour avoir un nombre plus significatif, on préférera prendre l'écart entre les effectifs. Dans un texte de n objets, on compte n_o le nombre d'apparitions de l'objet o. On connaît $\mathbb{P}(O = o')$ que l'on a évalué dans la première partie. Et donc,

$$d(o, o') = |n_o - n \times \mathbb{P}(O = o')|$$

Pour trouver l'objet o' qui est le plus probablement représentatif de l'objet o, il suffit de trouver celui qui minimise $d(o, o')$.

3.1.2 Forme d'un texte

L'hypothèse à vérifier

La forme va nous servir à évaluer ce genre d'hypothèses :
— $\mathcal{H}_0$: « Le cryptogramme est déchiffré »
— $\mathcal{H}_1$: « Le cryptogramme n'est pas déchiffré »

Définition

La forme d'un texte évalue concrètement la différence qu'il existe entre ce texte par rapport à un texte source. Typiquement, si l'on déchiffre un texte de plusieurs façons différentes, cet outil nous indiquera quel est le déchiffrement le plus proche de la langue. Dans notre cas, donc, on utilisera cet outil statistique en prenant pour texte source celui étudié précédemment qui dépeint correctement la langue française.

La forme est définie d'après la probabilité d'apparition des n-grammes ou des mots du texte source. En pratique :
— On choisit l'ensemble d'objets E à étudier : monogrammes, bigrammes, trigrammes, ... mots.
— On calcule la fréquence d'apparition de chaque objet O_i dans le texte source, on considère alors que cette fréquence est une très bonne estimation de $\mathbb{P}(O = O_i)$.

La forme du texte étudié est ensuite définie par la formule suivante :

$$\gamma = \prod_{\substack{O_i \in texte \\ O_i \neq O_j}} \mathbb{P}(O = O_i)^{n_i}$$

Avec n_i le nombre d'apparition de O_i dans le texte dont on calcule la forme et $\mathbb{P}(O = O_i)$ la probabilité d'apparition de O_i dans le texte source. Autrement dit, c'est le produit des probabilités d'apparition dans le texte source de chaque objet du texte étudié. Pour des raisons pratiques, on préfère prendre le logarithme de la forme :

$$\Gamma = log(\gamma) = log(\prod_{\substack{O_i \in texte \\ O_i \neq O_j}} \mathbb{P}(O = O_i)^{n_i}) = \sum_{\substack{O_i \in texte \\ O_i \neq O_j}} n_i \times log(\mathbb{P}(O = O_i))$$

En effet, le nombre produit étant compris entre 0 et 1, un logarithme nous permet de mieux l'évaluer.

Analyse de la forme

On a un outil, c'est bien. Encore faut-il savoir ce qu'il signifie. On se base par exemple sur la probabilité d'apparition de chaque bigramme en français. Supposons que l'on ait deux textes de même longueur : l'un ne signifie absolument rien (exemple : « ABNNK SETAS CPOAE ANBME EBINS JCLVY X ») et l'autre veut dire quelque chose de terriblement profond (exemple : « Est-ce que votre blanquette est bonne ? »). La forme non-logarithmique γ du premier texte sera très proche de 0 car les bigrammes qui le composent n'apparaissent presque pas en français. Alors que, pour l'autre texte, γ sera moins proche de 0 que le premier.

Lorsque l'on applique le logarithme, donc, la forme du premier texte Γ sera très négative, alors que pour le second texte elle sera plus élevée (mais toujours négative).

Finalement :

Plus Γ est grand, plus le texte est susceptible d'être en français.

Casser le chiffre, c'est donc essayer de maximiser Γ. Une dernière remarque : évidemment, selon les objets que l'on étudie (monogrammes, bigrammes, etc.), la précision de la forme sera plus ou moins intéressante. Avec des quadrigrammes ou des mots, la forme sera très précise, alors qu'avec des monogrammes elle est un peu moins révélatrice. On se contentera des monogrammes puis des bigrammes : c'est déjà assez précis, et on ne peut pas se permettre de travailler avec des mots car le cryptographe ne nous a pas rendu la tâche facile : il a séparé son cryptogramme en bloc de 5 lettres de façon à ce que l'on ne puisse pas discerner les mots.

3.1.3 Ce bon vieux χ^2

L'hypothèse à vérifier

Le test du χ^2 va nous servir aussi à évaluer ces hypothèses :
— $\mathcal{H}_0$: « Le cryptogramme est déchiffré »
— $\mathcal{H}_1$: « Le cryptogramme n'est pas déchiffré »

Le test d'adéquation

En statistiques, le test de χ^2 sert à évaluer s'il y a une relation entre des variables qualitatives. Autrement dit, il sert à tester l'indépendance de ces variables. Dans notre cas, on va l'utiliser pour tester l'adéquation entre l'apparition de certains objets dans un cryptogramme (monogrammes, bigrammes, etc.) avec leur apparition théorique (c'est-à-dire en français).

On notera E l'ensemble des objets étudiés, n_i le nombre d'apparitions de l'objet O_i dans le cryptogramme composé de n objets.

On définit alors la statistique du χ^2 :

$$D(Observations, Theorie) = \sum_{\substack{O_i \in texte \\ O_i \neq O_j}} \frac{(n_i - n \times \mathbb{P}(O = O_i))^2}{n \times \mathbb{P}(O = O_i)} = \sum_{\substack{O_i \in texte \\ O_i \neq O_j}} \frac{d(O_i, O_i)^2}{n \times \mathbb{P}(O = O_i)}$$

Sous l'hypothèse $\mathcal{H}_0$: « Le cryptogramme est déchiffré », c'est-à-dire si toutes les probabilités d'apparition des O_i du cryptogramme sont égales aux probabilités d'apparition des O_i théoriques, cette statistique suit une loi du χ^2 à $card(E) - 1$ degrés de liberté. On pourra donc se ramener aux tables du χ^2 pour récupérer la p-valeur.

On utilisera cette formule ici uniquement pour les monogrammes, ce qui revient à écrire :

$$D(Observations, Theorie) = \sum_{l=A}^{Z} \frac{(n_l - n \times \mathbb{P}(L = l))^2}{n \times \mathbb{P}(L = l)} = \sum_{l=A}^{Z} \frac{d(l, l)^2}{n \times \mathbb{P}(L = l)}$$

Et donc, sous l'hypothèse $\mathcal{H}_0$, la loi du χ^2 suivie est à 25 degrès de liberté.

3.1.4 Analyse de tout cela

Résoudre notre problème, c'est-à-dire casser le chiffre, c'est donc minimiser D (ce qui revient à minimiser chaque écart d) **et** maximiser Γ. Le test de la forme va évaluer si le texte déchiffré est du charabia ou non, alors que le χ^2 nous dira si l'on est en adéquation avec les probabilités théoriques d'apparition.

La solution aura donc une forme élevée (si ce n'est la plus élevée), mais pas forcément le plus petit D possible. En effet, G. Perec a bien écrit un livre sans utiliser une seule fois la lettre e : *La disparition*, qui est pourtant la lettre la plus utilisée en français. La distance du χ^2 nous induirait en erreur dans ce cas.

3.2 Les cryptogrammes mono-alphabétiques

Pour chacun de ces cryptogrammes, nous procéderons de la manière suivante :
— On tentera une attaque par force brute en essayant les 25 clés possibles dans le cas d'un chiffrement César.
— Si cela ne fonctionne pas, on utilisera un algorithme utilisant la forme pour casser le chiffre mono-alphabétique automatiquement.
— Là encore, si le cryptogramme est toujours incompréhensible, on gardera le traitement précédent (qui sera presque correct en théorie), et on se débrouillera à la main en essayant de minimiser la distance du χ^2.

3.2.1 Cryptogramme n°1

Pour rappel, on veut savoir ce qu'il se trame derrière cette expression :

« FILMK OYFYP CHYHN LYFYM YWLYN MILNF YNUFG OX »

Tentative d'attaque par force brute

On applique un programme Pascal [1] au cryptogramme pour en déduire tous les textes « décalés » de César. Après calcul des formes du texte et de la distance du χ^2, on a :

clé	**texte**	**Forme (mono)**	**Forme (big)**	D	**p-valeur (%)**
1	ehkljnxexobg...	-167	-460	3455	$\simeq 0$
2	dgjkimwdwnaf...	-193	-477	17867	$\simeq 0$
			...		
20	lorsquelevi...	-98	-189	13	98
			...		
25	gjmnlpzgzqd...	-161	-440	1109	$\simeq 0$

Table 3.1 – Test de toutes les clés du chiffre de César

Eurêka ! Ça fonctionne : le texte déchiffré est sûrement le 20^{eme}, car la forme est plus élevée mais aussi parce que la distance du χ^2 est faible. En effet on a une p-valeur élevée, ici on a environ 98% de chances d'avoir un texte plus distant du français que celui-là, ce qui est très correct. Finalement, la 20^{eme} clé révèle bien le secret suivant :

« Lorsque le vin entre, le secret sort - Le Talmud »

One down, two to go.

3.2.2 Cryptogramme n°3

On rappelle que l'on cherche à cryptanalyser ce texte :

« OHQSB SAQSA MTOQF BFJAN OQLPB HNQHJ GEPOQ OHBHL FBAQF BWJUF BPLOH TGEOP QBEPL FFHOX KPAGO FOUBA SMTOF OQRBP BRSPA QSAMT OQNTH RIBHS AFFJH BFBSJ APOQO PBKKP JRIOH SNBTS BHSKF TQNOQ RBPBR SPAQS AMTOQ QSBSA QSAMT OQNOF BKJKT FBSAJ HMTOF BSBAF FONOF RIBHS AFFJH BTLGO HSOFB SBAFF ONOFR IBHSA FFJHR JHQAN POPKJ TPBKK PJRIO PFOQR BPBRS PAQSA MTOQN OFBKJ

1. Disponible en annexe

KTFBS AJHHO NKOHN MTOUB AEFOG OHSVJ APOKB QNTSJ TSNOF BSBAF FONOR OFFOR AMTOF OQJHN BLOQJ ASUBA SBTFT XOGEJ TPLJT BTXSB SQTHA QAFQT UUASK JTPJE SOHAP NOQKP RAQAJ HQLBF OQNOK POHNP ONOQR IBHSA FFJHQ NOSBA FFOQL BFOQ »

Tentative d'attaque par force brute

Les résultats de l'attaque sur le cryptogramme sont les suivants :

clé	**texte**	**Forme (mono)**	**Forme (big)**	D	**p-valeur (%)**
1	ngprarzprzls...	-1628	-4396	3321	$\simeq 0$
2	mfoqzqyoqykr...	-1907	-5470	5307	$\simeq 0$
		...			
25	pirtctbrtbn...	-1780	-4855	9546	$\simeq 0$

TABLE 3.2 – Test de toutes les clés du chiffre de César

Sacrebleu ! Cela ne donne rien ! Sortons l'artillerie lourde.

Utilisation d'un algorithme Hill-Climbing

Bon, vu d'ici, on pourrait procéder de 3 manières différentes :
— Essayer de casser le chiffre à la main en utilisant les fréquences précédemment calculées : en « tâtonnant ».
— Tenter une attaque par force brute sur les 26! clés possibles avec un ordinateur.
— Tenter une attaque, via l'outil informatique aussi, mais de façon à minimiser le nombre d'itérations.

Les deux dernières solutions sont intéressantes car elles automatisent le déchiffrement. Toutefois, la deuxième solution est gourmande en ressources : elle demande d'effectuer de l'ordre de 26! itérations. Et, les ressources, en cryptanalyse c'est important : pour casser RSA [2] avec une machine puissante, il faut compter plusieurs années pour une clé de 768 bits (et on utilise aujourd'hui surtout des clés de 2048 bits...). Nous allons donc opter pour la troisième solution.

Nous utiliserons un algorithme de type *Hill-Climbing*[3] en utilisant la forme du texte. Son fonctionnement sera le suivant :

1. Générer une clé aléatoire (qui, ici, est un alphabet) et la tester en calculant la forme du texte ainsi déchiffré.
2. Échanger au hasard deux caractères de la clé, la tester de la même manière.
3. Si la nouvelle forme est plus grande que la précédente, on change la clé par la clé permutée.
4. On retourne à l'étape 2, sauf si l'on a fait 1000 itérations sans changer de clé.

On relance cet algorithme jusqu'à obtenir une forme qui soit sensiblement proche d'un texte français comportant le même nombre de caractères. Autrement dit, nous avons besoin de calculer une forme « correcte » vers laquelle on veut que l'algorithme tende. Calculer cette forme « basique » est relativement simple. Rappelons la formule de la forme :

$$\Gamma = \sum_{\substack{O_i \in texte \\ O_i \neq O_j}} n_i \times log(\mathbb{P}(O = O_i))$$

2. Un cryptosystème récent et très utilisé du fait de sa praticité et de sa sécurité
3. C'est un algorithme qui modifie une solution fausse jusqu'à en trouver une bonne

Avec n_i le nombre d'apparitions de l'objet O_i dans le texte étudié. Pour avoir une forme correcte, il faut que chaque fréquence des objets du texte suive la même loi de probabilité que ces objets en français. Autrement dit, en notant n le nombre d'objets dans le texte, la forme recherchée est la suivante :

$$\Gamma_0 = \sum_{\substack{O_i \in texte \\ O_i \neq O_j}} n \times \mathbb{P}(O = O_i) \times log(\mathbb{P}(O = O_i))$$

La condititon d'arrêt sera donc la suivante : forme $\geq$ forme basique.

On implémente cet algorithme en Pascal [4], avec la forme des bigrammes (qui est plus précise) et on obtient la clé suivante (au bout d'une petite vingtaine de minute de recherche itérative) :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	B	I	N	G	O	U	E	V	A	W	Z	F	R	H	J	K	M	P	Q	S	T	L	D	X	Y	C

TABLE 3.3 – Clé trouvée par l'algorithme Hill-Climbing

Ce qui donne ce texte déchiffré :

> « ENSTATISTIQUESLALOICESVRANCSNODGRESENANVLAISLAJOFLARVENUDGERS AGREVELLNEXPRIDELEFAITQUELESMARAMTERISTIQUESCUNEMBANTILLONALEATOIRE SERAPPROMBENTCAUTANTPLUSCESMARAMTERISTIQUESSTATISTIQUESCELAPOPULATION QUELATAILLECELEMBANTILLONAUVDENTELATAILLECELEMBANTILLONAMONSICERERPOUR APPROMBERLESMARAMTERISTIQUESCELAPOPULATIONNECEPENCQUEFAIGLEDENTHOIREPAS CUTOUTCELATAILLECEMELLEMIQUELESONCAVESOITFAITAULUXEDGOURVOUAUXETATSUNIS ILSUFFITPOUROGTENIRCESPREMISIONSEVALESCEPRENCRECESEMBANTILLONSCETAILLESEVALES »

Ça marche bien ! On reconnaît du français un peu partout dans le texte. On réécrit ce même texte en mettant en minuscules les caractères que l'on suppose correctement déchiffrés :

> « en statistiques la loi CesVranCsnoDGresenanVlaislaJoflarVenuDGersaGreVellneXpriDe le fait que les MaraMteristiquesCuneMBantillon aleatoire serapproMBentC autant plus CesMaraMteristiques statistiques Cela population que la taille CeleMBantillonauVDente la taille CeleMBantillonaMonsiCerer pour approMBerlesMaraMteristiquesCela population neCepenCquefaiGleDentHoirepasCutoutCe la taille CeMelleMiquelesonCaVe soit fait au luXeDGourVouauX etats unis il suffit pour oGtenirCespreMisionseValesCeprenCreCeseMBantillonsCe tailles eVales »

Ce qui « fixe » dans la clé les caractères suivants :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**B**	I	N	G	**O**	**U**	E	V	**A**	W	Z	**F**	R	**H**	**J**	**K**	**M**	**P**	**Q**	**S**	**T**	L	D	X	Y	C

TABLE 3.4 – Même clé avec les caractères de la clé supposés corrects en gras

4. Disponible en annexe

Déchiffrement du reste du cryptogramme

Nous avons « quasiment » déchiffré le texte en maximisant la forme, nous allons maintenant essayer de minimiser la distance du χ^2. Voici le tableau des fréquences de chaque lettre dans le cryptogramme en partie déchiffré, ainsi que leurs fréquences estimées en français (en pourcents). Les lettres en gras sont supposées correctement déchiffrées (ce sont les lettres au-dessus des caractères considérés corrects de la clé) :

Lettre	Français	Cryptogramme	Distance d
A	8.59	9.87	6.24
B	1.01	1.23	1.06
C	3.07	4.32	6.05
D	3.70	1.23	1.99
E	17.17	15.0	0.14
F	1.10	1.23	0.63
G	1.07	1.23	0.78
H	1.00	0.20	3.86
I	7.35	7.61	1.23
J	0.41	0.20	1.03
K	0.01	0.00	0.07
L	5.91	9.25	6.23
M	2.78	3.29	2.44
N	6.91	6.17	3.60
O	5.32	5.14	0.89
P	2.70	3.29	2.86
Q	1.17	1.85	3.27
R	6.56	5.55	4.90
S	7.89	7.61	1.39
T	7.31	7.61	1.44
U	6.35	5.55	3.90
V	1.59	1.85	1.24
W	0.01	0.00	0.05
X	0.42	0.61	0.93
Y	0.35	0.00	1.73
Z	0.17	0.00	0.83

TABLE 3.5 – Comparaison des fréquences d'apparition de chaque lettre

$$\boxed{D = 33.82}$$

TABLE 3.6 – Distance du χ^2

On a donc une p-valeur de 11%. On voit dans ce tableau que, parmi les lettres considérées comme n'étant pas encore déchiffrées, la fréquence du C n'est pas très proche de celle du français. Revenons sur le texte en partie déchiffré :

> « en statistiques la loi **Ces** VranCsnoDGresenanVlaislaJoflarVenuDGersaGreVellneXpriDe le fait que les MaraMteristiquesCuneMBantillon aleatoire serapproMBent **C autant** plus **Ces** MaraMteristiques statistiques **Ce** la population que la taille CeleMBantillonauVDente la taille CeleMBantillonaMonsiCerer pour approMBerlesMaraMteristiques **Ce** la population neCepenCquefaiGleDentHoirepas **Cu** tout **Ce** la

> taille CeMelleMiquelesonCaVe soit fait au luXeDGourVouauX etats unis il suffit pour oGtenirCespreMisionseValesCeprenCreCeseMBantillonsCe tailles eVales »

On voit ici que le C pourrait très bien représenter la lettre d en français. Calculons la distance entre C et d pour tester cette hypothèse :

$$d(C, d) = |21 - 486 \times 0,0370| = 3,018$$

Ce n'est pas top, mais c'est toujours mieux que la distance $d(C, c)$, on va donc supposer cette hypothèse comme étant juste.

Le cryptogramme devient alors :

> « en statistiques la loi des VrandsnoDGresenanVlaislaJoflarVenuDGersaGreVellneX-priDe le fait que les MaraMteristiques dune MBantillon aleatoire serapproMBent d autant plus des MaraMteristiques statistiques de la population que la taille de leMBantillonauVDente la taille de leMBantillonaMonsiderer pour approMBerlesMaraMteristiques de la population ne depend que faiGleDentHoire pas du tout de la taille de MelleMiquelesondaVe soit fait au luXeDGourVouau etats unis il suffit pour oGtenir des preMisionseVales de prendre des eMBantillons de tailles eVales »

Cela semble correct aussi.

On échange dans la clé le caractère en-dessous du c par celui qui est en-dessous du d. (Ce qui, auparavant, était traduit par un c dans le cryptogramme, est maintenant traduit par un d). La clé devient donc :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**B**	I	G	**N**	**O**	**U**	E	V	**A**	W	Z	**F**	R	**H**	**J**	**K**	**M**	**P**	**Q**	**S**	**T**	L	D	X	Y	C

TABLE 3.7 – Nouvelle clé

$$\boxed{D = 29.28}$$

TABLE 3.8 – Nouvelle distance du χ^2

Cela donne une p-valeur de 25%, ce qui est finalement bien plus correct. Impossible de deviner le mot-clé pour l'instant. On peut même être amené à se demander « Y a-t-il un mot-clé ? Ou est-ce juste un alphabet ? » En fait, il est très probable que le texte soit chiffré avec un mot clé. Si l'on regarde les lettres fixées dans la clé on peut voir une suite de lettre : « **F**R**HJKMPQST** » qui semble presque correspondre à un morceau de l'alphabet dont on a enlevé quelques lettres : comme lors d'un chiffrement par mot-clé ! En suivant cette intuition, on peut en déduire que la lettre à côté du F fixé se doit d'être un G. Cela donne, en échangeant le R à côté du F par le G en-dessous du c :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**B**	I	R	**N**	**O**	**U**	E	V	**A**	W	Z	**F**	**G**	**H**	**J**	**K**	**M**	**P**	**Q**	**S**	**T**	L	D	X	Y	C

TABLE 3.9 – Nouvelle clé

Ce qui donne au niveau du cryptogramme :

> « en statistiques la loi des VrandsnomGresenanVlaislaJoflarVenumGersaGreVelln **eXprime** le fait que les CaraCteristiques dune CBantillon aleatoire serapproCBent d autant plus des *CaraCteristiques* statistiques de la population que la taille de leCBantillonauVmente la taille de leCBantillonaConsiderer pour approCBerlesCaraCteristiques de la population ne depend que faiGlementHoire pas du tout de la taille deCelleCiquelesondaVe soit fait au luXemGourVouau etats unis il suffit pour oGtenir des **preCisions** eVales de prendre des eCBantillons de tailles eVales »

Cela semble correct aussi. On voit apparaître le mot « eXprime », on considère que ce n'est pas un hasard et donc on fixe dans la clé le X. De même, les mots « preCisions » et « CaraCteristiques » apparaissent, on suppose donc que le c est correctement déchiffré, on fixe donc la lettre en-dessous du c dans la clé.

L'alphabet chiffrant devient donc ici :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**B**	I	**R**	**N**	**O**	**U**	E	V	**A**	W	Z	**F**	**G**	**H**	**J**	**K**	**M**	**P**	**Q**	**S**	**T**	L	D	**X**	Y	C

TABLE 3.10 – Nouvelle clé

La distance du χ^2 est alors :

$$\boxed{D = 27.76}$$

TABLE 3.11 – Nouvelle distance du χ^2

La p-valeur passe à 32%, ce qui est mieux.

Difficile de trouver le mot-clé, là encore... Nous allons donc essayer une nouvelle fois de faire des hypothèses sur le cryptogramme :

> « en statistiques la loi des **Vrands** nomGresenanVlaislaJoflarVenumGersaGreVelln exprime le fait que les caracteristiques dune cBantillon aleatoire se **rapprocBent** d autant plus des caracteristiques statistiques de la population que la taille de lecBantillon **auVmente** la taille de lecBantillon a considerer pour **approcBer** les caracteristiques de la population ne depend que faiGlementHoire pas du tout de la taille de celle ci que le **sondaVe** soit fait au **luxemGourV** ou au etats unis il suffit pour **oGtenir** des precisions **eVales** de prendre des ecBantillons de tailles **eVales** »

On suppose ici que :
— le V est en réalité un g (« Vrands » ⇒ « grands », « auVmente » ⇒ « augmente »...)
— le B représente un h (« rapprocBent » ⇒ « rapprochent », « approcBer » ⇒ « approcher »...)
— le G est un b (« oGtenir » ⇒ « obtenir », « luxemGourV » ⇒ « luxembourg »...)

Et donc la clé devient, en échangeant les lettres :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**B**	**E**	**R**	**N**	**O**	**U**	**L**	**I**	**A**	W	Z	**F**	**G**	**H**	**J**	**K**	**M**	**P**	**Q**	**S**	**T**	E	D	**X**	Y	C

La clé nous tombe directement dessus ! Il s'agit du nom du mathématicien Bernoulli ! L'alphabet chiffrant devrait donc être :

Alphabet clair	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet chiffré	**B**	**E**	**R**	**N**	**O**	**U**	**L**	**I**	**A**	**C**	**D**	**F**	**G**	**H**	**J**	**K**	**M**	**P**	**Q**	**S**	**T**	**V**	**W**	**X**	**Y**	**Z**

TABLE 3.12 – Alphabet final

Cela donne au niveau du χ^2 :

$$\boxed{D = 5.34}$$

TABLE 3.13 – Distance finale du χ^2

Cette distance se rapporte à une p-valeur de 99,99%. C'est un très bon signe. Regardons le résultat :

« En statistiques, la loi des grands nombres (en anglais Law of large Numbers, abrégé LLN) exprime le fait que les caractéristiques d'un échantillon aléatoire se rapprochent d'autant plus des caractéristiques statistiques de la population que la taille de l'échantillon augmente. La taille de l'échantillon à considérer pour approcher les caractéristiques de la population ne dépend que faiblement, voire pas du tout, de la taille de celle-ci : que le sondage soit fait au Luxembourg ou aux États-Unis, il suffit, pour obtenir des précisions égales, de prendre des échantillons de tailles égales. »

Une introduction à la loi des grands nombres nous venant tout droit de *Wikipédia*. Et de deux !

3.3 Cryptogramme de Vigenère

On cherche à cryptanalyser ce texte :

« JJHCKG TDWNF WAXXV WQKWA GIIKE NQDMT QLRXJ GBGMG IWMEO NDSWL MKWMC SHBWF AHQKW CMVCB AJSLC ESKUS JBMSI CWMFW PMWTD WQOWM GCMVW KIAGJ CWQXA LRTJG FWLWG KMFQH VLFXA VSLOW BLYMW FWFHM MFRNT SATQF DXCLS MZ- WON VECFM FHHCB SGMHC NDSWL XSGHC BSMIA GLMMZ VPWNF WASMK WGMKM FBMML WMKSW QJSJC WZXAZ OLIJR LTWGK MF- QHV LFXAX CKOWB MCFSW MKHBV WSIIJ QXYMS JCSBW WFOEM YCNBV SEIUV HAWEN IFRHV SZXOG IMLWZ TKZCL MTWXV XOBBW ZXJWO NOWGM MHOKN GWLWF BXBJC NDWDT ADWGB WFEWU IMMMF XVXOV MBSWQ JOBAD SFQJC BZIIB DGILI ARXIS JTVUS KIDCK AUSGM KHIIK AHVUO LKGAF MBSWQ KOBAD OICAG JCWAH QSIVW FHKIA FXRSW ICWHC MVWLU WFVQS ZTDAS CMDIB LAGFM JQBRW QAIFH XTSJB MBSWI FGXTS JBMBS GMKIB AIITU GIKML TBVSZ XUWBM YMOGL TSTCU CNXVS ZMFGF MVWLM FHFIA GVWEA XVLTT QKHNX GIKIN CBZUS MBWVN USBBB WXXTW IKZWD HVVGM ZWGLQ EDEME SGBBS EMMFW QKENM USLBU SZWMH WMDOF WMFJC AATXG ILAWO NRGIK LZIBI WBMZW DKMFR KMMBX KGBLB

JIVBA CGUWQ TVAEN MEOBA VSFIA BJCAG TQLDX CLSMZ WGXCD SFMFH TUWAX BLFXI MGXZN WVMVS EIUCF UMBTC LSTNS WKMDS WWFZX LGBWM KCB »

3.3.1 Approximation de la longueur du mot-clé

Test de Kasiski (version moderne)

Cette approximation s'appelle le test de Kasiski, et elle part de l'idée que lorsqu'une séquence de lettres apparaît plusieurs fois dans le cryptogramme, c'est que ces séquences ont été chiffrées avec la même clé. Nous allons utiliser l'indice de coïncidence pour trouver la longueur de cette clé. Notons k la taille du mot-clé utilisé. On a alors les 1^{ere}, $(k+1)^{eme}$, $(2k+1)^{eme}\dots$ lettres qui sont codées par le même chiffre de César. On divise ainsi le texte en k parties, de telle sorte que chaque partie résulte du même chiffre de substitution mono-alphabétique. On a alors environ $\frac{n}{k}$ lettres dans chacune de ces parties.

On sait que :

$$IC = \sum_{l=A}^{Z} \mathbb{P}(E_l)$$

Avec $\mathbb{P}(E_l)$ la probabilité de tomber sur deux lettres identiques en piochant deux lettres au hasard dans le texte.

Si l'on note I_l l'IC de la langue francaise et I_u l'IC « uniforme », c'est-à-dire l'IC d'un texte où chaque lettre est parfaitement uniformément répartie, on a :

— Pour deux lettres chiffrées de la même manière, la probabilité que ce soit les mêmes est d'environ I_l, car il s'agit d'un chiffre mono-alphabétique

— Pour deux lettres chiffrées différemment, la probabilité que ce soit les mêmes est d'environ I_u car le chiffre est poly-alphabétique et donc la répartition de chaque lettre est presque uniforme

Et donc, en notant p le nombre de paires de lettres chiffrées de la même manière et q le nombre de paires de lettres chiffrées différemment :

$$IC = \frac{p \times I_l + q \times I_u}{\frac{n(n-1)}{2}}$$

En effet :

— $p \times I_l$: parmi toutes les paires de lettres possibles dans l'ensemble des lettres chiffrées de la même manière, on compte uniquement celles où les lettres sont identiques

— $q \times I_u$: même chose, mais dans l'ensemble des lettres chiffrées de manière différente

— $p \times I_l + q \times I_u$: c'est donc une approximation du nombre total de paires de lettres identiques dans le cryptogramme de Vigenère

— $\frac{n(n-1)}{2}$: on divise par le nombre de lettres au premier tirage (n) et au second tirage ($n-1$), et on ne tient pas compte de l'ordre (on divise par 2 pour enlever les permutations)

Rappelons que l'on cherche une approximation de k. Il ne nous reste donc plus qu'à calculer I_u, puis p et q en fonction de k, pour enfin isoler k.

Calculons I_u : pour une répartition uniforme, la probabilité de tomber sur la $l - ieme$ lettre est toujours de $\frac{1}{26}$, en considérant que l'alphabet est de 26 lettres.

$$I_u = \sum_{l=A}^{Z} \mathbb{P}(E_l) = \sum_{l=A}^{Z} (\frac{1}{26})^2 = \frac{1}{26}$$

Trouvons p, le nombre de paires de lettres chiffrées avec le même décalage César. On tire la première lettre dans le cryptogramme au hasard (n lettres). La seconde est tirée dans les $\frac{n}{k} - 1$ lettres restantes chiffrées de la même façon. Et, comme on ne tient pas compte de l'ordre, on divise le tout par 2, le nombre de permutations possibles.

Finalement :

$$p = \frac{1}{2} \times n \times (\frac{n}{k} - 1)$$

Enfin, calculons q, le nombre de paires de lettres chiffrées différemment. On tire la première lettre parmi n, et la seconde est tirée dans les $\frac{n}{k} \times (k-1)$ restantes. Là encore, on divise le résultat par 2.

On a donc :

$$q = \frac{1}{2} \times n \times (\frac{n(k-1)}{k})$$

Enfin,

$$IC = \frac{n \times (\frac{n}{k} - 1) \times I_l + n \times (\frac{n(k-1)}{k}) \times I_u}{\frac{2n(n-1)}{2}} = \frac{(\frac{n}{k} - 1) \times I_l + (\frac{n(k-1)}{k}) \times I_u}{(n-1)} = \frac{(n-k) \times I_l + n(k-1) \times I_u}{k(n-1)}$$

Et donc

$$k = \frac{(n-k) \times I_l + n(k-1) \times I_u}{IC(n-1)}$$

$$\Leftrightarrow k(1 + \frac{I_l - n \times I_u}{IC(n-1)}) = \frac{n \times I_l - n \times I_u}{IC(n-1)}$$

$$\Leftrightarrow k = \frac{n \times I_l - n \times I_u}{IC(n-1) + I_l - n \times I_u}$$

On en déduit enfin :

$$\boxed{k \simeq \frac{n \times (I_l - I_u)}{(I_l - IC) + n(IC - I_u)}}$$

Application au cryptogramme

En utilisant la formule ci-dessus [5] sur nos données, on trouve :

$$\boxed{k \simeq 3.4837}$$

Diantre ! Ce n'est pas très précis. On en déduit que la longueur du mot-clé est très probablement de 3 ou 4 caractères, mais on préférerait connaître la taille exacte du mot-clé...

Test de Kasiski (version old-school)

On part ici de la même constatation que plus haut, celle de Kasiski (ou plutôt celle de Charles Babbage quelques années avant lui) : On peut décomposer le cryptogramme en plusieurs morceaux, tous codés avec un chiffre de César. Admettons que l'on retrouve dans le cryptogramme plusieurs fois une même suite de lettres (par exemple, dans notre cryptogramme, « WQK » apparaît deux fois), alors cela signifie que ces suites représentent très probablement la même

5. Avec un traitement MatLab disponible en annexe

chose en français et qu'elles ont des chances d'avoir été codées par la même « partie » de la clé. Or, si elles sont chiffrées par la même « partie » de clé, alors le nombre de lettres qui sépare ces suites est nécessairement un diviseur de la taille de la clé.

Donc, en résumé, si l'on note d la distance (le nombre de caractères) qui sépare deux mêmes suites et k la taille de la clé :

$$\forall d, d \equiv 0 \mod k$$

Il suffit de trouver tous les d possibles et d'essayer d'en déduire k. Dans notre cas, on ne prendra que quelques d, car l'on sait que k vaut 3 ou 4.

On trouve ces relations :

Suite	d	k possibles
WNF	200	1, 2, 4, ..., 100, 200
VWL	100	1, 2, 4, ..., 50, 100
ENM	76	1, 2, 4, ..., 38, 76
...		

TABLE 3.14 – Quelques diviseurs probables d de k

Ici, on en deduit clairement que k vaut 4, car aucune des distances entre les suites étudiées n'est divisible par 3.

3.3.2 Attaque par mot probable

Il s'agit de l'attaque « classique » lancée contre le chiffre de Vigenère. Elle peut s'avérer très puissante si l'on connaît la longueur du mot-clé. Cela tombe bien, c'est notre cas. Pour commencer, on va sélectionner une quinzaine de mots probables en français, de la taille du mot-clé. Nous allons utiliser un algorithme dont le fonctionnement est le suivant pour chaque mot probable :

1. Pour toutes les positions i possibles, supposer que le mot probable se trouve à la i-ème position dans le texte
2. Calculer alors le mot-clé qui découle de cette supposition
3. Déchiffrer le texte avec ce mot-clé et trouver la forme de ce déchiffrement
4. Retourner le déchiffrement dont la forme est maximale

On implémente cet algorithme, là encore, en Pascal[6]. Le résultat est immédiat (enfin, le programme tourne exactement 2.0 secondes), le mot probable « VOUS » fait tomber le mot-clé qui est « OTIS » et le résultat est le suivant :

« VOUSSAVEZMOIJENECROISPASQUILYAITDEBONNEOUDEMAUVAISESITUATION MOISIJEDOISRESUMERMAVIEAUJOURDHUIAVECVOUSJEDIRAISQUECESTDABORD DESRENCONTRESDESGENSQUIMONTTENDULAMAINPEUTETREAUNMOMENTOUJENEPOUVAIS PASOUJETAISSEULCHEZMOIETCESTCURIEUXDESEDIREQUELESHASARDSLESRENCONTRES FORGENTUNEDESTINEEPARCEQUEQUANDONALEGOUTDELACHOSEQUANDONALEGOUTDELACHOSEBIEN FAITELEBEAUGESTEPARFOISONNETROUVEPASLINTERLOCUTEURENFACEJEDIRAISLEMIROIRQUIVOUS AIDEAAVANCERALORSCENESTPASMONCASCOMMEJEDISAISLAPUISQUEMOIAUCONTRAIREJAIPUETJEDISMERCI

6. En annexe

ALAVIEJELUIDISMERCIJECHANTELAVIEJEDANSELAVIEJENESUISQUAMOU-
RETFINALEMENTQUANDBEAUCOUP DEGENSMEDISENTMAISCOMMENT-
FAISTUPOURAVOIRCETTEHUMANITEJELEURREPONDSTRESSIMPLEMENTJELEUR
DISQUECESTCEGOUTDELAMOURQUIMAPOUSSEAUJOURDHUIAENTREPREN-
DREUNECONSTRUCTIONMECANIQUEMAIS DEMAINQUISAITPEUTETRESEU-
LEMENTAMEMETTREAUSERVICEDELACOMMUNAUTEAFAIRELEDONLEDONDESOI »

Ce qui donne, avec la ponctuation :

« Vous savez, moi je ne crois pas qu'il y ait de bonne ou de mauvaise situation. Moi, si je dois résumer ma vie aujourd'hui avec vous, je dirais que c'est d'abord des rencontres. Des gens qui m'ont tendu la main, peut-être à un moment où je ne pouvais pas, où j'étais seul chez moi. Et c'est curieux de se dire que les hasards, les rencontres forgent une destinée... Parce que quand on a le goût de la chose, quand on a le goût de la chose bien faite, le beau geste, parfois on ne trouve pas l'interlocuteur en face je dirais, le miroir qui vous aide à avancer. Alors ce n'est pas mon cas, comme je disais là, puisque moi au contraire, j'ai pu : et je dis merci à la vie, je lui dis merci, je chante la vie, je danse la vie... Je ne suis qu'amour ! Et finalement, quand beaucoup de gens me disent "Mais comment fais-tu pour avoir cette humanité ?", je leur réponds très simplement, je leur dis que c'est ce goût de l'amour qui m'a poussé aujourd'hui à entreprendre une construction mécanique, mais demain qui sait ? Peut-être seulement à me mettre au service de la communauté, à faire le don, le don de soi... »

Et de trois ! Il s'agit du monologue du personnage Otis, joué par Edouard Baer, dans le film *Astérix et Obélix : Mission Cléopâtre*. Comme quoi le chiffre de Vigenère, même s'il est poly-alphabétique, n'est pas des plus sûr...

Conclusion

Résumons-nous. Nous avons vu dans ce projet que la cryptologie [7] est une science à part entière, qui s'est révélée au cours de l'histoire de manière parfois très intéressante. On s'est approprié le déchiffrement de certains codes secrets, prouvant ainsi que la sécurité de certains chiffres n'est pas infaillible.

Nous nous sommes grandement aidé de certains outils de description : l'analyse des fréquences d'utilisation de certains éléments de la langue, l'indice de coïncidence... Grâce à eux, nous avons alors appliqué différents tests : le test de Friedman, le test de la forme, le test d'adéquation du χ^2, les deux tests de Kasiski. Nous nous sommes aussi appuyé sur l'outil informatique pour attaquer de manière automatique et rapide les différents chiffres. Ainsi, nous avons démontré que les statistiques et l'informatique occupent souvent une place primordiale en cryptologie. Ce projet s'inclue dans l'enseignement GM.

Nous terminerons ici ce projet par un petit cryptogramme, libre à vous d'essayer d'en percer le secret...

21, 1, 21, 77, 1, 37, 9, 77, 37, 2, 1, 21, 26, 1, 37

Indice : *La clé tend vers sa moyenne, le chiffre garde un trésor secret depuis près de deux siècles.*

7. Terme englobant la cryptographie (ou l'art de chiffrer) et la cryptanalyse (ou l'art de déchiffrer)

Scripts de récupération des données

A.1 Calcul des effectifs de chaque lettre

```
for i in ‘grep . alphabet.txt‘;
do echo "$i ‘grep . texte.txt | sed -e ’s/./&\n/g’ | grep $i -ic‘";
done > resultat.txt
```

Le fichier *alphabet.txt* contient 37 caractères (les 26 caractères de l'alphabet et les 11 caractères spéciaux supplémentaires) en colonne. *texte.txt* contient le texte source. Enfin, on envoie les effectifs de chaque lettre dans le fichier *resultat.txt*.

A.2 Calcul des effectifs de chaque bigramme

```
for i in ‘echo {a..z}{a..z} | sed -e ’s/\ /\n/g’‘;
do echo "$i ‘grep $i -ic texte.txt‘" | awk -F" " ’{print $1,$2}’;
done > resultat.txt
```

On parcoure les $26 * 26 = 676$ bigrammes possibles. De même, le fichier *texte.txt* contient le texte source et les effectifs sont envoyés dans *resultat.txt*.

A.3 Calcul des effectifs de chaque mot

```
cat texte.txt  |strings  | tr "[" " "|tr "]" " "| tr -d "^$" |
tr -d ’0-9’ | tr "A-Z" "a-z" | tr ’\011’ ’ ’| tr -d ’\042’ |
sed -e ’s/[.$=‘*_~(/\);:?,! ]/\n/g’ | sed -e ’s/-$//g’ -e ’s/^-//g’|
sort | grep -v ’^$’ |uniq -c | awk -F" " ’{print $2,$1}’ > resultat.txt
```

On parcoure tous les mots du fichier *texte.txt*, en prenant soin de supprimer les caractères spéciaux. Les effectifs sont, là encore, envoyés dans *resultat.txt*.

Traitements MatLab

B.1 Description des données

B.1.1 Traitement pour la description des lettres

```
% Script de traitement des donnees : Effectif des lettres - TraitementLettres.m

% Importation des donnees
fichier = input('Rentrez le chemin fichier a etudier : ', 's');
tmp = importdata(fichier);
dt = tmp.data;

% Calcul des effectifs reels
effectifs = [dt(1)+dt(2)+dt(3); dt(4); dt(5); dt(6); ...
    dt(7)+dt(8)+dt(9)+dt(10)+dt(11);
    dt(12); dt(13); dt(14); dt(15)+dt(16)+dt(17); dt(18); dt(19); dt(20);
    dt(21); dt(22); dt(23)+dt(24); dt(25); dt(26); dt(27); dt(28); dt(29); ...
        dt(30)+dt(31)+dt(32);
    dt(33); dt(34); dt(35); dt(36); dt(37)];

% Calcul des frequences et des totaux
freq = 100*effectifs./sum(effectifs);
total = [sum(effectifs) sum(freq)];

% Exportation dans un Cell Array
numbers = [effectifs freq; total];
name = {'A'; 'B'; 'C'; 'D'; 'E'; 'F'; 'G'; 'H'; 'I'; 'J'; 'K'; 'L'; 'M'; 'N'; ...
    'O'; 'P'; 'Q';
     'R'; 'S'; 'T'; 'U'; 'V'; 'W'; 'X'; 'Y'; 'Z'; 'Total'};
res = {name numbers};

% Exportation de l'histogramme dans le fichier res.png
h = figure;
bar(freq);
set(gca,'XTick', 1:26 ,'XTickLabel',['A':'Z']');
ylim([0 18]);
xlim([0 27]);
ylabel('Frequence (%)');
print(h, '-dpng', 'res.png');
close(h);

% Exportation des donnees au format tableau LaTeX dans le fichier ...
    resultatFrequences.txt
filename = 'resultatFrequences.txt';
fid = fopen(filename, 'w');

for row=1:26
    fprintf(fid, '\t\\textbf{%s} & % 7.0f & % 02.2f\\\\\n\t\\hline\n',
                res{1}{row}, res{2}(row,1), res{2}(row,2));
end

fclose(fid);

% Calcul des moyennes, medianes, variances et ecart-types
```

```
46 moyenne = [mean(effectifs) mean(freq)];
47 variance = [var(effectifs) var(freq)];
48 ecartType = [std(effectifs) std(freq)];
49 mediane = [median(effectifs) median(freq)];
50 name = {'Moyenne'; 'Variance'; 'Ecart-type'; 'Mediane'};
51 numbers = [moyenne; variance; ecartType; mediane];
52 res = {name numbers};
53
54 % Exportation des donnees au format tableau LaTeX dans le fichier ...
       resultatDescription.txt
55 filename = 'resultatDescription.txt';
56 fid = fopen(filename, 'w');
57
58 for row=1:4
59     fprintf(fid, '\t\\textbf{%s} & % 7.2f & % 02.2f\\\\\n\t\\hline\n', ...
           res{1}{row}, res{2}(row,1), res{2}(row,2));
60 end
61
62 fclose(fid);
63
64 % Calcul et affichage de l'IC
65 IC = sum((effectifs.*(effectifs - 1))./(sum(effectifs)*(sum(effectifs) - 1)))
```

B.1.2 Traitement pour la description des bigrammes

```
1  % Script de traitement des donnees : Effectif des bigrammes - TraitementBigrammes.m
2
3  % Importation des donnees
4  fichier = input('Rentrez le chemin fichier a etudier : ', 's');
5  tmp = importdata(fichier);
6  effectifs = tmp.data;
7  name = tmp.textdata;
8
9  % Calcul des frequences
10 freq = 100*effectifs./sum(sum(effectifs));
11
12 % Tri des donnees dans l'ordre decroissant
13 tmp = {upper(name) effectifs freq};
14 [unused order] = sort(effectifs, 'descend');
15 res = cellfun(@(x)x(order),tmp,'un',0);
16
17 % Exportation des bigrammes au format tableau latex dans le fichier ...
       'resultatBig.txt'
18 filename = 'resultatBig.txt';
19 fid = fopen(filename, 'w');
20
21 for row=1:26
22     fprintf(fid, '\t\\textbf{%s} & % 7.0f & % 02.2f\\\\\n\t\\hline\n', ...
           res{1}{row}, res{2}(row), res{3}(row));
23 end
24
25 fclose(fid);
26
27 % Recherche des paires de lettres
28 for i = 1:26
29     namePaires{i} = name{26*(i-1) + i};
30     effectifsPaires(i) = effectifs(26*(i-1) + i);
31     freqPaires(i) = freq(26*(i-1) + i);
```

```
32 end
33 res = {upper(namePaires) effectifsPaires freqPaires};
34
35 % Exportation des paires de lettres au format tableau latex dans le fichier ...
       'resultatPaires.txt'
36 filename = 'resultatPaires.txt';
37 fid = fopen(filename, 'w');
38
39 for row=1:26
40     fprintf(fid, '\t\\textbf{%s} & % 7.0f & % 02.2f\\\\\n\t\\hline\n', ...
           res{1}{row}, res{2}(row), res{3}(row));
41 end
42
43 fclose(fid);
44
45 % Calcul de moyenne & cie
46 moyenne = [mean(effectifs) mean(freq)];
47 variance = [var(effectifs) var(freq)];
48 ecartType = [std(effectifs) std(freq)];
49 mediane = [median(effectifs) median(freq)];
50 name = {'Moyenne'; 'Variance'; 'Ecart-type'; 'Mediane'};
51 numbers = [moyenne; variance; ecartType; mediane];
52 res = {name numbers};
53
54 % Exportation des resultats de la description au format tableau latex dans le ...
       fichier 'resultatDescriptionBig.txt'
55 filename = 'resultatDescriptionBig.txt';
56 fid = fopen(filename, 'w');
57
58 for row=1:4
59     fprintf(fid, '\t\\textbf{%s} & % 7.2f & % 02.2f\\\\\n\t\\hline\n', ...
           res{1}{row}, res{2}(row,1), res{2}(row,2));
60 end
61
62 fclose(fid);
```

B.1.3 Traitement pour la description des mots

```
1  % Script de traitement des donnees : Effectif des mots - TraitementMots.m
2
3  % Importation des donnees
4  fichier = input('Rentrez le chemin fichier a etudier : ', 's');
5  tmp = importdata(fichier);
6  effectifs = tmp.data;
7  name = tmp.textdata;
8
9  % Calcul des frequences
10 freq = 100*effectifs./sum(effectifs);
11
12 % Tri des donnees dans l'ordre decroissant
13 tmp = {upper(name) effectifs freq};
14 [unused order] = sort(effectifs, 'descend');
15 res = cellfun(@(x)x(order),tmp,'un',0);
16
17 % Suppression des mots de moins de trois lettres
18 i = 1;
19 m = size(res{1});
20 n = m(1);
```

```
while i ≤ n
    str = res{1}{i};
    if length(str) < 3
        res{1}(i,:) = [];
        res{2}(i,:) = [];
        res{3}(i,:) = [];
    else
        i = i + 1;
    end
    m = size(res{1});
    n = m(1);
end

% Exportation des mots au format tableau latex dans le fichier 'resultatMots.txt'
filename = 'resultatMots.txt';
fid = fopen(filename, 'w');

for row=1:20
    fprintf(fid, '\t\\textbf{%s} & % 7.0f & % 02.2f\\\\\n\t\\hline\n', ...
        res{1}{row}, res{2}(row), res{3}(row));
end

fclose(fid);
```

B.2 Utilisation des données

B.2.1 Calcul de la forme d'un texte (monogrammes)

```
% Script de calcul de la forme d'un texte avec les monogrammes — ...
    CalculDeFormeMono.m

% Chargement des frequences du fichier source
TraitementLettres;

% Sauvegarde des frequences reelles dans une variable
frqReelle = freq/100;

% Chargement des effectifs du fichier etudie
TraitementLettres;

% Calcul de la forme
forme = 0;
for i = 1:26
forme = forme + effectifs(i)*log(frqReelle(i));
end;

% Affichage de la forme
forme
```

B.2.2 Calcul de la forme d'un texte (bigrammes)

```
% Script de calcul de la forme d'un texte avec les bigrammes — CalculDeFormeBig.m

% Chargement des frequences du fichier source
TraitementBigrammes;

% Sauvegarde des frequences reelles dans une variable
frqReelle = freq/100;
for i = 1:676
    if frqReelle(i) == 0
        frqReelle(i) = 0.0000001;
    end;
end;

% Chargement des effectifs du fichier etudie
TraitementBigrammes;

% Calcul de la forme
forme = 0;
forme = 0;
for i = 1:676
    if frqReelle(i) ≠ 0
        forme = forme + effectifs(i)*log(frqReelle(i));
    end;
end;

% Affichage de la forme
forme
```

B.2.3 Calcul de D, la distance du χ^2

```
% Script de calcul de la distance du chi2 et de la p—valeur avec les ...
    monogrammes — CalculDistanceKhi2.m

% Chargement des frequences du fichier source
TraitementLettres;

% Sauvegarde des frequences reelles dans une variable
frqReelle = freq/100;

% Chargement des effectifs du fichier etudie
TraitementLettres;

% Calcul de n
n = sum(effectifs);

% Calcul de la distance
khi2 = 0;
for i = 1:26
    khi2 = khi2 + ((effectifs(i) — n*frqReelle(i))^2)/(n*frqReelle(i));
end;

% Affichage de la distance et de la p—valeur (en pourcents)
khi2
pval = (1 — cdf('chi2', khi2, 25))*100
```

B.2.4 Calcul de k

```
% Script de calcul de k, le nombre de lettres du mot-cle - CalculDeK.m

% Chargement des effectifs, des frequences et de l'IC du fichier
TraitementLettres;

% Affectation des constantes
Il = 0.0779;
Iu = 1/26;
n = sum(effectifs);

% Calcul et affichage de k
k = (n*(Il - Iu))/((Il - IC)+n*(IC - Iu))
```

Programmes Pascal

C.1 Attaque par force brute du chiffre de César

```
1  program cesar;
2
3  uses
4      sysutils, importerExporter;
5      (*Unit creee pour importer et exporter des fichiers*)
6
7  function dechiffrerLettre(l : Char; cle : Integer) : Char;
8      var res : Char;
9          i : Integer;
10 begin
11     res := l;
12     for i := 1 to cle do
13     begin
14         case res of
15             'A' : res := 'Z';
16             else
17                 res := pred(res);
18         end;
19     end;
20     dechiffrerCaractere := res
21 end;
22
23 function dechiffrerTexte(texte : Ansistring; cle : Integer) : Ansistring;
24     var res : Ansistring;
25         i : Integer;
26 begin
27     res := '';
28     for i := 1 to Length(texte) do
29         res := res + dechiffrerLettre(texte[i], cle);
30     dechiffrerTexte := Lowercase(res)
31 end;
32
33     var texte : Ansistring;
34         i : Integer;
35 begin
36     texte := importer('cryptogramme.txt');
37     for i := 1 to 25 do
38         exporter('res' + IntToStr(i) + '.txt', dechiffrerTexte(texte, i));
39 end.
```

C.2 Attaque d'un chiffre mono-alphabétique par un algorithme Hill-Climbing

```
1  program hillClimbing;
2
3  uses
4      sysutils, importerExporter;
5
6  {$J-}
7  const
8    PROBAREELLESBIG: array['A'..'Z','A'..'Z'] of Real = (* Il s'agit du tableau ...
         de 26*26 frequences (en pourcents), je ne l'ecris pas ici car cela prend ...
         de la place pour rien *)
9    ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
10 {$J+}
11
12 type
13     Bigrammes = array['A'..'Z','A'..'Z'] of LongWord;
14
15 function compterBigrammes(texte : Ansistring) : Bigrammes;
16     var l, m : Char;
17         i : LongWord;
18         res : Bigrammes;
19 begin
20     for l := 'A' to 'Z' do
21     begin
22         for m := 'A' to 'Z' do
23             res[l,m] := 0;
24     end;
25     for i := 2 to length(texte) do
26         res[texte[i-1],texte[i]] := res[texte[i-1],texte[i]] + 1;
27     compterBigrammes := res;
28 end;
29
30 function calculerFormeBigrammes(texte : Ansistring): Real;
31     var l, m : Char;
32         forme : Real;
33         effectifs : Bigrammes;
34 begin
35     forme := 0;
36     effectifs := compterBigrammes(texte);
37     for l := 'A' to 'Z' do
38     begin
39         for m := 'A' to 'Z' do
40             forme := forme + effectifs[l,m]*ln(PROBAREELLESBIG[l,m]/100);
41     end;
42     calculerFormeBigrammes:= forme
43 end;
44
45 function formeBigrammesBasique(n : LongWord): Real;
46     var l, m : Char;
47         forme : Real;
48 begin
49     forme := 0;
50     for l := 'A' to 'Z' do
51     begin
52         for m := 'A' to 'Z' do
53         begin
```

```
54              forme := forme + ...
                    n*(PROBAREELLESBIG[l,m]/100)*ln(PROBAREELLESBIG[l,m]/100);
55          end;
56      end;
57      formeBigrammesBasique:= forme
58  end;
59
60  procedure echanger(var a : Char; var b : Char);
61      var tmp : Char;
62  begin
63      tmp := a;
64      a := b;
65      b := tmp
66  end;
67
68  procedure permuterDeuxLettres(var str : String);
69      var i, j : LongWord;
70  begin
71      repeat
72          i := random(length(str) + 1);
73          j := random(length(str) + 1);
74      until ((i <> j) and (i <> 0) and (j <> 0));
75      echanger(str[i], str[j])
76  end;
77
78  function cleAleatoire() : String;
79      var i, j, n : Integer;
80          c : Char;
81          dejaLa : Boolean;
82          res : String;
83  begin
84      res := ALPHABET;
85      for i := 1 to 26 do
86      begin
87          repeat
88              dejaLa := false;
89              n := random(26);
90              c := Chr(ord('A') + n);
91              for j := 1 to i - 1 do
92              begin
93                  if c = res[j] then
94                      dejaLa := true;
95              end;
96          until not(dejaLa);
97          res[i] := c;
98      end;
99      cleAleatoire:= res;
100 end;
101
102 function dechiffrerLettre(lettre : Char; cle : String) : Char;
103     var pos, i : LongWord;
104 begin
105     for i := 1 to 26 do
106     begin
107         if cle[i] = lettre then
108             pos := i;
109     end;
110     dechiffrerLettre:= ALPHABET[pos]
111 end;
112
113 function dechiffrerTexte(texte : Ansistring; cle : String) : Ansistring;
114     var i : LongWord;
```

```
115         res : Ansistring;
116 begin
117     res := '';
118     for i := 1 to length(texte) do
119     begin
120         res := res + dechiffrerLettre(texte[i], cle);
121     end;
122     dechiffrerTexte:= res;
123 end;
124
125 procedure hillClimbing(texte : Ansistring; var cle : String; var formeMax : Real);
126     var
127         cletmp : String;
128         forme : Real;
129         cpt : LongWord;
130 begin
131     cletmp := cle;
132     formeMax := calculerFormeBigrammes(dechiffrerTexte(texte, cletmp));
133     cpt := 0;
134     while (cpt < 1000) do
135     begin
136         permuterDeuxLettres(cletmp);
137         forme := calculerFormeBigrammes(dechiffrerTexte(texte, cletmp));
138         if (forme > formeMax) then
139         begin
140             cpt := 0;
141             formeMax := forme;
142             cle := cletmp;
143         end;
144         cpt := cpt + 1;
145     end;
146 end;
147
148     var res, texte : Ansistring;
149         cle : String;
150         forme, formeMax : Real;
151         i : Integer;
152 begin
153     randomize;
154     texte := importer('cryptogramme.txt');
155     cle := cleAleatoire();
156     formeMax := calculerFormeBigrammes(dechiffrerTexte(texte, cle));
157     i := 0;
158     repeat
159         i := i + 1;
160         hillClimbing(texte,cle,forme);
161         if (forme > formeMax) then
162         begin
163             formeMax := forme;
164             res := dechiffrerTexte(texte, cle);
165             writeln('Iteration : ', i);
166             writeln('Cle : ', cle);
167             writeln('Cryptogramme dechiffre : ', res);
168             writeln('Forme : ', forme, #13#10);
169         end;
170         if (i mod 2000 = 0) then
171             cle := cleAleatoire();
172     until (formeMax ≥ formeBigrammesBasique(length(texte)));
173     exporter('resultat.txt', 'Cle : ' + cle + #13#10 + 'Cryptogramme dechiffre ...
        : ' + res)
174 end.
```

C.3 Attaque par mot probable du chiffre de Vigenère

```
1  program hillClimbing;
2
3  uses
4      sysutils, formeBigrammes, importerExporter, cesar; (* Utilisation des ...
           algorithmes precedents *)
5
6  {$J-}
7  const
8    NBMOTSPROBABLES = 15;
9    MOTSPROBABLES: array[1..NBMOTSPROBABLES] of String =
10   ('DANS','ELLE','POUR','PLUS','MAIS','TOUT','VOUS',
11     'AVEC','NOUS','QUIL','DEUX','BIEN','CEST','SANS','DUNE');
12   TAILLEMOTCLE = 4;
13 {$J+}
14
15 function dechiffrerTexte(texte : Ansistring; motCle : String) : Ansistring;
16     var res : Ansistring;
17         i, j : Integer;
18 begin
19     res := texte;
20     for i := 1 to length(motCle) do
21     begin
22         for j := 0 to (length(texte) div TAILLEMOTCLE) do
23         begin
24             res[i + TAILLEMOTCLE*j] := cesar.dechiffrerCaractere(texte[i + ...
                   TAILLEMOTCLE*j], -ord('A') + ord(motCle[i]));
25         end;
26         dechiffrerTexte := res
27     end;
28 end;
29
30 function motCle(texte : Ansistring; motProbable : String; pos : Integer) : String;
31     var res : String;
32         i, difference : Integer;
33 begin
34     res := motProbable;
35     difference := 0;
36     for i := 1 to length(motProbable) do
37     begin
38         if ord(motProbable[i]) - ord(texte[pos+i-1]) ≥ 0 then
39             difference := ord(motProbable[i]) - ord(texte[pos+i-1])
40         else
41             difference := -ord(motProbable[i]) + ord(texte[pos+i-1]);
42         res[i] := cesar.dechiffrerCaractere('A', difference);
43     end;
44     motCle := res
45 end;
46
47 function attaqueMotPrb(texte : Ansistring; tailleMotCle : Integer) : Ansistring;
48     var
49         forme, formeMax : Real;
50         res : Ansistring;
51         cle : String;
52         i, j : Integer;
53 begin
54     res := texte;
55     formeMax := calculerFormeBigrammes(texte);
56     for i := 1 to NBMOTSPROBABLES do
```

```
57      begin
58          for j := 1 to length(texte) - tailleMotCle do
59          begin
60              cle := motCle(texte, MOTSPROBABLES[i], j);
61              forme := calculerFormeBigrammes(dechiffrerTexte(texte, cle));
62              if (forme > formeMax) then begin
63                  formeMax := forme;
64                  res := dechiffrerTexte(texte, cle);
65              end;
66          end;
67      end;
68      attaqueMotPrb:= res
69  end;
70
71      var texte : Ansistring;
72  begin
73      texte := importer('cryptogramme.txt');
74      exporter('res.txt', attaqueMotPrb(texte, TAILLEMOTCLE))
75  end.
```

Bibliographie

[1] *Singh*, Simon *Histoire des codes secrets*, Lgf, septembre 1999.

[2] *Marc Béveraggi*, *Cours d'Analyse des données*, INSA Rouen.

[3] *Stéphane Canu*, *Cours de statistiques et traitement de données (M8)*, INSA Rouen.

[4] *Julien Barnier*, "*Tout ce que vous n'avez jamais voulu savoir sur le χ^2 sans jamais avoir eu envie de le demander*" ENS-Lyon, 38 pages, 2013.

[5] *Christine Badoc*, "*Codes et cryptologie*" Université Bordeaux 1, 59 pages, 2010.

[6] *arrobe - Société de services en ingénierie informatique*, "*Codes et cryptologie*" Arrobe, 50 pages, 2009.

[7] `http://www.practicalcryptography.com/` (Valide à la date du 03/06/2013) Très bon site web pour cryptanalyser avec les statistiques.

[8] `http://fr.wikipedia.org/wiki/Test_du_%CF%87%C2%B2#Test_du_.CF.87.C2.B2_d.27ad.C3.A9quation` (Valide à la date du 03/06/2013)

[9] `http://www.apprendre-en-ligne.net/crypto/vigenere/motprobvig.html` (Valide à la date du 03/06/2013)

[10] `http://www.dcode.fr/tous-les-outils` (Valide à la date du 03/06/2013) De très bon outils, vaut le coup d'oeil aussi.