JULES GAGNON

Création d'outils pour l'automatisation d'analyses phylogénétiques de génomes d'organites

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise sur mesure en bioinformatique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

FACULTÉ DES ÉTUDES SUPÉRIEURES
UNIVERSITÉ LAVAL

Avril 2004

Résumé

Le traitement des données de séquençage pour les rendre utilisables dans une analyse phylogénétique est long et répétitif. De plus, certaines analyses plus complexes peuvent difficilement être entreprises sans l'automatisation de certaines tâches. La création d'outils bioinformatiques permettrait de diminuer le temps consacré à la préparation des données.

Le but de cette recherche est de développer des outils informatiques permettant d'automatiser le traitement de données provenant du séquençage d'organites. Pour ce faire, il a été nécessaire de créer :

- des bases de données de gènes d'organites ;
- des outils pour l'extraction des séquences génétiques dans différents formats ;
- des outils pour l'identification des gènes d'organismes nouvellement séquencés ;
- des outils de préparation des données pour l'utilisation lors d'analyses phylogénétiques.

Finalement, le bon fonctionnement des outils a été vérifié par l'exécution d'une analyse phylogénétique dont les résultats ont déjà été publiés.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

ADN : Acide DésoxyriboNucléique
ARN : Acide RiboNucléique
ARNt : ARN de transfert
BLAST : "Basic Local Alignment Search Tool"
CPAN : "Comprehensive Perl Archive Network"
EMBOSS : "European Molecular Biology Open Software Suite"
GCG : "Genetics Computer Group"
GTR : "General Time Reversible"
IR : "Inverted Repeat"
MOLPHY : "MOLecular PHYlogenetics"
NCBI : "National Center for Biotechnology Information"
PAUP : "Phylogenetic Analysis Using Parsimony"
PAML : "Phylogenetic Analysis by Maximum Likelihood"
PCR : "Polymerase Chain Reaction"
Perl : "Practical Extraction Report Language"
PHYLIP : "PHYLogeny Inference Package"
RELL : "Resampling Estimated Log Likelihood"

Chapitre 1

Introduction

1.1 Le séquençage d'ADN

Le séquençage d'ADN, c'est-à-dire le processus qui consiste à déterminer la séquence de nucléotides qui composent un fragment d'ADN, a entraîné une révolution de la recherche en biologie moléculaire. Au cours des dernières années, le développement de nouvelles techniques de biologie moléculaire a rendu possible le séquençage à grande échelle. Les améliorations qu'ont subies ces techniques ont beaucoup changé les méthodes de travail dans plusieurs domaines de recherche.

1.1.1 Le clonage moléculaire

Les améliorations des techniques de clonage moléculaire sont en partie responsables de l'augmentation de l'efficacité du processus de séquençage. Ces améliorations se situent prinpalement au niveau des vecteurs de clonage utilisés et des techniques de la réaction de polymérisation en chaîne (PCR).

Les nouveaux vecteurs de clonages permettent de sélectionner de façon plus efficace les clones ayant une insertion. De plus, ils permettent des insertions de plus grande taille et un plus grand nombre de copies du vecteur. Ces nouveaux vecteurs facilitent la manipulation des insertions grâce à des sites d'insertion multiple bordés de promoteurs pour des polymérases.

Les techniques de polymérisation en chaîne ont beaucoup bénéficié du développement de nouvelles variétés d'ADN polymérase plus fiables, avec une meilleure processivité et avec une meilleure résistance aux hautes températures. Ainsi, il est maintenant beaucoup plus facile d'amplifier un fragment de séquence en grande quantité pour en effectuer le séquençage.

1.1.2 Le séquençage automatique

L'apparition de nouvelles techniques de séquençage a permis la conception de séquenceurs automatiques. Ces séquenceurs sont devenus au fil des innovations de plus en plus efficaces.

D'abord, le développement de marqueurs fluorescents a permis la lecture automatisée des gels de séquences. De plus, au fil des ans, ces molécules fluorescentes ont été améliorées pour offrir un signal plus intense, et ainsi plus facilement détectable.

Un autre domaine où d'importantes améliorations ont eu lieu touche les ADN polymérases. Plusieurs variétés d'ADN polymérases ont été développées pour les rendre de plus en plus processives et diminuer leurs taux d'erreur.

Les séquenceurs eux-mêmes ont aussi subi d'importantes modifications. Ils sont maintenant en grande partie automatiques et robotisés. Ils peuvent maintenant effectuer 96 séquences simultanément, et même plus avec les appareils de la toute dernière génération. De plus, avec l'apparition de l'électrophorèse par capillaire, il n'est plus nécessaire de préparer un gel et le temps nécessaire à la préparation d'une réaction de séquence est beaucoup diminué.

Toutes ces améliorations des réactions de séquences ont accéléré grandement le processus. À titre d'exemple, les premiers séquenceurs automatiques permettaient d'effectuer au plus deux séries d'électrophorèse par jour, alors que les derniers modèles peuvent effectuer jusqu'à 24 séries en 24 heures.

1.1.3 Outils informatiques

L'accélération des ordinateurs a permis le développement de nombreux outils pour analyser la grande quantité d'information produite par les séquenceurs automatiques. De plus, le séquençage des génomes de plusieurs organismes modèles, tel que le ver

C. elegans[1], la mouche drosophile[2], la souris[3] et l'être humain[4], a entraîné un raffinement de ces outils et une meilleure standardisation des méthodes utilisées.

Les logiciels d'assemblage de séquence ont beaucoup bénéficié de ces grands projets de séquençage. Ils ont maintenant des algorithmes plus fiables et plus rapides. Ces améliorations permettent d'utiliser la technique de bris aléatoires pour effectuer le séquençage de génomes complets. Cette technique simplifie grandement les manipulations et élimine plusieurs étapes fastidieuses du processus de séquençage. Elle consiste à briser le génome en petits fragments de façon aléatoire et à en effectuer le séquençage. Les logiciels d'assemblage sont ensuite responsables de regrouper les fragments qui se chevauchent et de reconstituer le génome complet.

Les algorithmes de recherche d'homologies ont aussi beaucoup bénéficié des progrès technologiques. Ils permettent maintenant d'identifier facilement et rapidement des séquences ayant un fort degré de similitude. Il est ainsi plus facile de procéder à l'identification et à l'annotation des gènes d'un génome, du moins quand des gènes homologues ont déjà été identifiés chez d'autres espèces.

1.2 Les génomes d'organites

Le laboratoire des Drs Turmel et Lemieux, dans lequel mon projet de maîtrise a été réalisé, se consacre au séquençage de génomes chloroplastiques et mitochondriaux d'algues. Grâce aux améliorations des techniques de séquençage et à l'utilisation de logiciels d'assemblage performants tel que Sequencher[5], il est maintenant possible pour un étudiant gradué de séquencer un génome d'organites en quelques mois.

Actuellement, une quarantaine de génomes chloroplastiques sont disponibles (Tableau 1.1), et presque autant de génomes mitochondriaux d'organismes photosynthétiques (Tableau 1.2). Chaque année, plusieurs nouveaux génomes deviennent disponibles. Tout au long de l'année, plusieurs projets de séquençage sont en cours en simultané. Ainsi, de nouveaux génomes sont régulièrement complétés à l'intérieur du laboratoire. En plus, d'autres groupes rendent disponibles les résultats de leurs recherches, ce qui augmente le nombre de génomes disponibles pour l'étude.

1.2.1 Origine des chloroplastes et des mitochondries

Les chloroplastes et les mitochondries sont des organites qui ont une origine endosymbiotique[6]. Ils sont dérivés de bactéries qui vivaient autrefois de façon indépendante et font maintenant partie intégrante des cellules eucaryotiques. Ces deux organites ont tout de même conservé une indépendance partielle. Ils se multiplient indépendamment de la cellule et ils ont encore un vestige de leur génome originel qui bien que réduit est toujours fonctionnel.

Les chloroplastes sont présents uniquement chez les eucaryotes photosynthétiques, incluant les plantes, les algues et les protozoaires photosynthétiques. Les bactéries qui seraient à leur origine sont les cyanobactéries et le représentant le plus rapproché dont le génome a été séquencé est *Synechosystis* PCC 6803.

Les mitochondries sont responsables de la respiration cellulaire et elles proviendraient d'une alpha-protéobactérie[7] similaire à *Rickettsia*. Elles sont présentes chez pratiquement tous les organismes eucaryotiques.

1.2.2 Caractéristiques des génomes

La majorité des génomes chloroplastiques comptent 100 000 à 200 000 paires de bases (Tableau 1.1), par conséquent, ils sont de 20 à 30 fois plus petits que les génomes de cyanobactéries. Ils possèdent 40 à 250 gènes (Tableau 1.2) et sont très compacts, c'est-à-dire qu'ils comportent très peu de séquences non codantes. La plupart se divisent en quatre parties (Figure 1.1). Deux régions répétées de façon inversée sont séparées par deux régions simple copie. Au cours de l'évolution, ces génomes perdent progressivement des gènes, lesquels sont souvent transférés vers le génome nucléaire.

Les génomes mitochondriaux présentent une très grande variabilité en taille et en structure (Figure 1.2). Leur taille varie de 15 000 à 400 000 paires de base (Tableau 1.3), tandis que leur contenu en gènes (Tableau 1.4) varie de 3 à 67 gènes. Ces génomes ont subi de nombreux réarrangements au fil de l'évolution.

Tableau 1.1 Les génomes chloroplastiques

Liste des organismes dont la séquence du génome chloroplastique fait partie de la base de données construite dans le cadre de ce projet en date du 9 décembre 2003. Le nombre de gènes a été calculé à partir des gènes présents dans la base de données. Le numéro d'accession correspond à celui retrouvé dans la base de données du NCBI. L'abréviation n.d. indique que le génome n'est pas disponible dans la base de données du NCBI.

Organisme	Taille (bp)	Nbre de gènes	Numéro d'accession
Cyanidium caldarium	164 921	224	NC_001840
Cyanidioschyzon merolae	149 987	230	NC_004799
Porphyra purpurea	191 028	242	NC_000925
Vaucheria bursata	123 216	169	n.d.
Odontella sinensis	119 704	155	NC_001713
Guillardia theta	121 524	174	NC_000926
Isochrysis tahiti	105 838	143	n.d.
Pavlova lutheri	95 281	134	n.d.
Cyanophora paradoxa	135 599	174	NC_001675
Euglena gracilis	143 171	87	NC_001603
Nephroselmis olivacea	200 799	129	NC_000927
Pycnococcus prasovali	80 211	98	n.d.
Monomastix species OKE-1	114 528	93	n.d.
Scherffelia dubia	137 161	104	n.d.
Neochloris pseudoalveolaris	145 947	108	n.d.
Chlorella vulgaris	150 613	113	NC_001865
Bryopsis plumosa	123 958	105	n.d.
Derbesia marina	105 614	105	n.d.
Oltmannsiellopsis viridis	151 933	104	n.d.
Pseudendoclonium akinetum	195 867	105	n.d.
Pedinomonas minor	98 340	104	n.d.
Scenedesmus obliquus	161 452	96	n.d.
Chlamydomonas reinhardtii	203 828	94	BK000554
Mesostigma viride	118 360	137	NC_002186
Chlorokybus atmophyticus	149 675	138	n.d.
Zygnema circumcarinatum	165 370	125	n.d.
Staurastrum punctulatum	157 089	121	n.d.
Chaetosphaeridium globosum	131 183	125	NC_004115
Chara vulgaris	184 933	127	n.d.
Anthoceros formosae	161 162	121	NC_004543
Marchantia polymorpha	121 024	120	NC_001319
Physcomitrella patens	122 890	116	NC_005087
Psilotum nudum	138 829	116	NC_003386
Adiantum capillus	150 568	113	NC_004766
Pinus thunbergii	119 707	106	NC_001631
Amborella trichopoda	162 686	111	NC_005086
Calycanthus fertilis	153 337	111	NC_004993
Nicotiana tabacum	155 939	110	NC_001879
Arabidopsis thaliana	154 478	110	NC_000932
Zea mays	140 384	108	NC_001666

Tableau 1.2 Les gènes chloroplastiques

Liste des gènes chloroplastiques présents dans la base de données construite dans le cadre ce projet en date du 9 décembre 2003, groupés selon leurs produits ou leurs fonctions.

Produits des gènes	Gènes
Acetyl-CoA carboxylase	*accA, accB, accD*
Allophycocyanine	*apcA, apcB, apcD, apcE, apcF*
Phycocyanine	*cpcA, cpcB, cpcG*
Phycoerythrine	*cpeA, cpeB*
Protéines de transport	*cysA, cysT, cysW*
Clp protéase	*clpC, clpP*
Chaperones	*groEL, groES*
Acétohydroxyacide synthase	*ilvB, ilvH*
Pyruvate déhydrogenase	*odpA, odpB*
Système de transport du manganèse	*mntA, mntB*
Préprotéine-translocase	*secA, secY*
ARNt synthétase	*syfB, syh*
Photosystème I	*psaA, psaB, psaC, psaD, psaE, psaF, psaI, psaJ, psaK, psaL, psaM*
Photosystème II	*psbA, psbB, psbC, psbD, psbE, psbF, psbH, psbI, psbJ, psbK, psbL, psbM, psbN, psbT, psbV, psbW, psbX, psbY, psbZ*
Cytochrome *b6/f*	*petA, petB, petD, petF, petG, petJ, petL, petM, petN*
ATP synthase	*atpA, atpB, atpD, atpE, atpF, atpG, atpH, atpI*
Biosynthèse de la chlorophylle	*chlB, chlI, chlL, chlN*
Rubisco	*rbcL, rbcR, rbcS*
NADH oxidoréductase	*ndhA, ndhB, ndhC, ndhD, ndhE, ndhF, ndhG, ndhH, ndhI, ndhJ, ndhK*
Protéines ribosomales de la grande sous-unité	*rpl1, rpl2, rpl3, rpl4, rpl5, rpl6, rpl9, rpl11, rpl12, rpl13, rpl14, rpl16, rpl18, rpl19, rpl20, rpl21, rpl22, rpl23, rpl24, rpl27, rpl28, rpl29, rpl31, rpl32, rpl33, rpl34, rpl35, rpl36*
Protéines ribosomales de la petite sous-unité	*rps1, rps2, rps3, rps4, rps5, rps6, rps7, rps8, rps9, rps10, rps11, rps12, rps13, rps14, rps15, rps16, rps17, rps18, rps19, rps20*
ARN polymérase	*rpoA, rpoB, rpoC1, rpoC2*
Facteurs de traduction	*infA, infB, infC, tsf, tufA*
Division du chloroplaste	*ftsH, ftsI, ftsW, minD, minE*

Produits des gènes	Gènes
Protéines de fonction inconnue	*ycf1, ycf3, ycf4, ycf12, ycf16, ycf17, ycf19, ycf20, ycf21, ycf22, ycf23, ycf24, ycf27, ycf29, ycf33, ycf34, ycf35, ycf36, ycf37, ycf38, ycf39, ycf40, ycf41, ycf43, ycf44, ycf45, ycf46, ycf47, ycf48, ycf49, ycf50, ycf51, ycf52, ycf53, ycf54, ycf55, ycf56, ycf57, ycf58, ycf59, ycf60, ycf61, ycf62, ycf63, ycf64, ycf65, ycf66, ycf80, ycf81, ycf82, ycf83, ycf84, ycf85, ycf86, ycf87*
Autres protéines	*acpP, argB, bas1, bioY, carA, cbbX, ccsA, cemA, cobA, crtE, dfr, dnaB, dnaK, dsbD, fabH, fdx, ftrC, glmS, glnB, gltB, hemA, hisH, hlpA, lipB, lpxA, lpxC, menA, menB, menC, menD, menE, menF, moeB, nadA, nblA, ntcA, pbsA, pgmA, preA, rne, thdF, thiG, trpA, trpG, trxA, upp*
ARNs ribosomaux	*rrf, rrl, rrs*
ARNs de transfert	*trnA(ggc), trnA(ugc), trnC(gca), trnD(guc), trnE(uuc), trnF(gaa), trnG(gcc), trnG(ucc), trnH(gug), trnI(cau), trnI(gau), trnK(uuu), trnL(caa), trnL(gag), trnL(uaa), trnL(uag), trnM(cau), trnfM(cau), trnN(guu), trnP(ggg), trnP(ugg), trnQ(uug), trnR(acg), trnR(ccg), trnR(ccu), trnR(ucu), trnS(cga), trnS(gcu), trnS(gga), trnS(uga), trnT(ggu), trnT(ugu), trnV(gac), trnV(uac), trnW(cca), trnY(gua)*
Autres ARNs	*rnpB, ssrA*

Figure 1.1 Exemple de génome chloroplastique

Carte génétique du génome chloroplastique de *Mesostigma viride*[8]. La position des gènes est indiquée par les rectangles pleins. Les deux régions simple copie sont désignées par LSC (Large Single Copy) et SSC (Small Single Copy). Les deux régions répétées inversées sont désignées par IR_A et IR_B (Inverted Repeat).

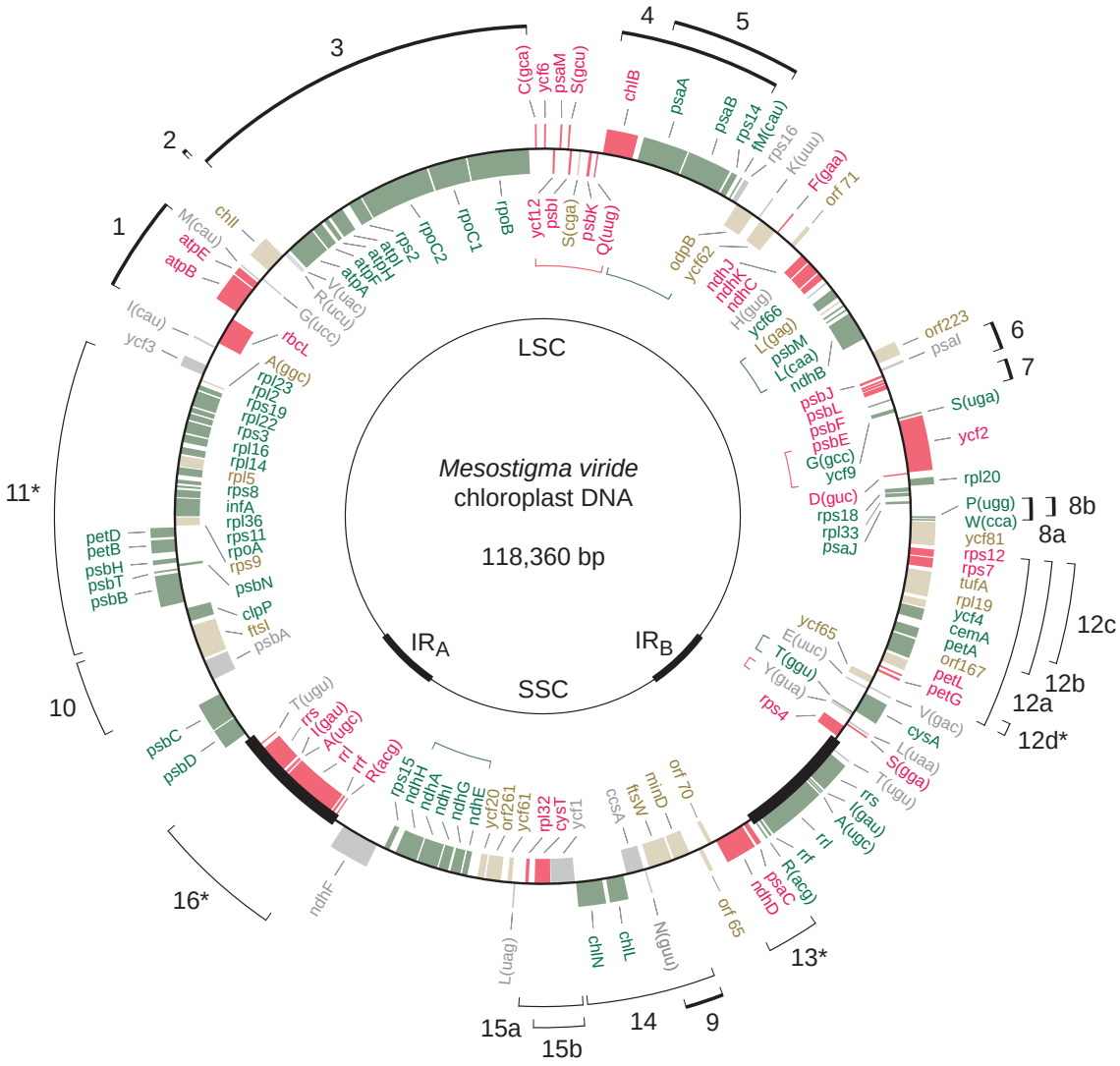

Mesostigma viride
chloroplast DNA
118,360 bp
LSC
IRA
IRB
SSC
1
2
3
4
5
6
7
8a
8b
9
10
11*
12a
12b
12c
12d*
13*
14
15a
15b
16*
C(gca)
ycf6
psaM
S(gcu)
chlB
psaA
psaB
rps14
fM(cau)
rps16
K(uuu)
F(gaa)
orf 71
rpoB
rpoC1
rpoC2
rps2
atpI
atpH
atpF
atpA
V(uac)
R(ucu)
G(ucc)
ycf12
psbI
S(cga)
psbK
Q(uug)
odpB
ycf62
ndhJ
ndhK
ndhC
H(gug)
ycf66
L(gag)
psbM
L(caa)
ndhB
psbJ
psbL
psbF
psbE
G(gcc)
ycf9
D(guc)
rps18
rpl33
psaJ
orf223
psaI
S(uga)
ycf2
rpl20
P(ugg)
W(cca)
ycf81
rps12
rps7
tufA
rpl19
ycf4
cemA
petA
orf167
petL
petG
V(gac)
cysA
L(uaa)
S(gga)
T(ugu)
rrs
I(gau)
A(ugc)
rrl
rrf
R(acg)
psaC
ndhD
orf 65
N(guu)
chlL
chlN
orf 70
minD
ftsW
ccsA
ycf1
rpl32
cysT
orf61
orf20
ndhE
ndhG
ndhI
ndhA
ndhH
rps15
ndhF
L(uag)
T(ugu)
rrs
I(gau)
A(ugc)
rrl
rrf
R(acg)
psbC
psbD
petD
petB
psbH
psbT
psbB
clpP
ftsI
psbA
psbN
rpoA
rps9
rps11
rpl36
infA
rps8
rpl5
rpl14
rpl16
rps3
rpl22
rps19
rpl2
rpl23
A(ggc)
rbcL
I(cau)
ycf3
M(cau)
atpE
atpB
chlI
ycf65
E(uuc)
T(ggu)
Y(gua)
rps4

Figure 1.2 Exemple de génome mitochondrial

Carte génétique du génome mitochondrial de *Mesostigma viride*[9]. La position des gènes est indiquée par les rectangles pleins.

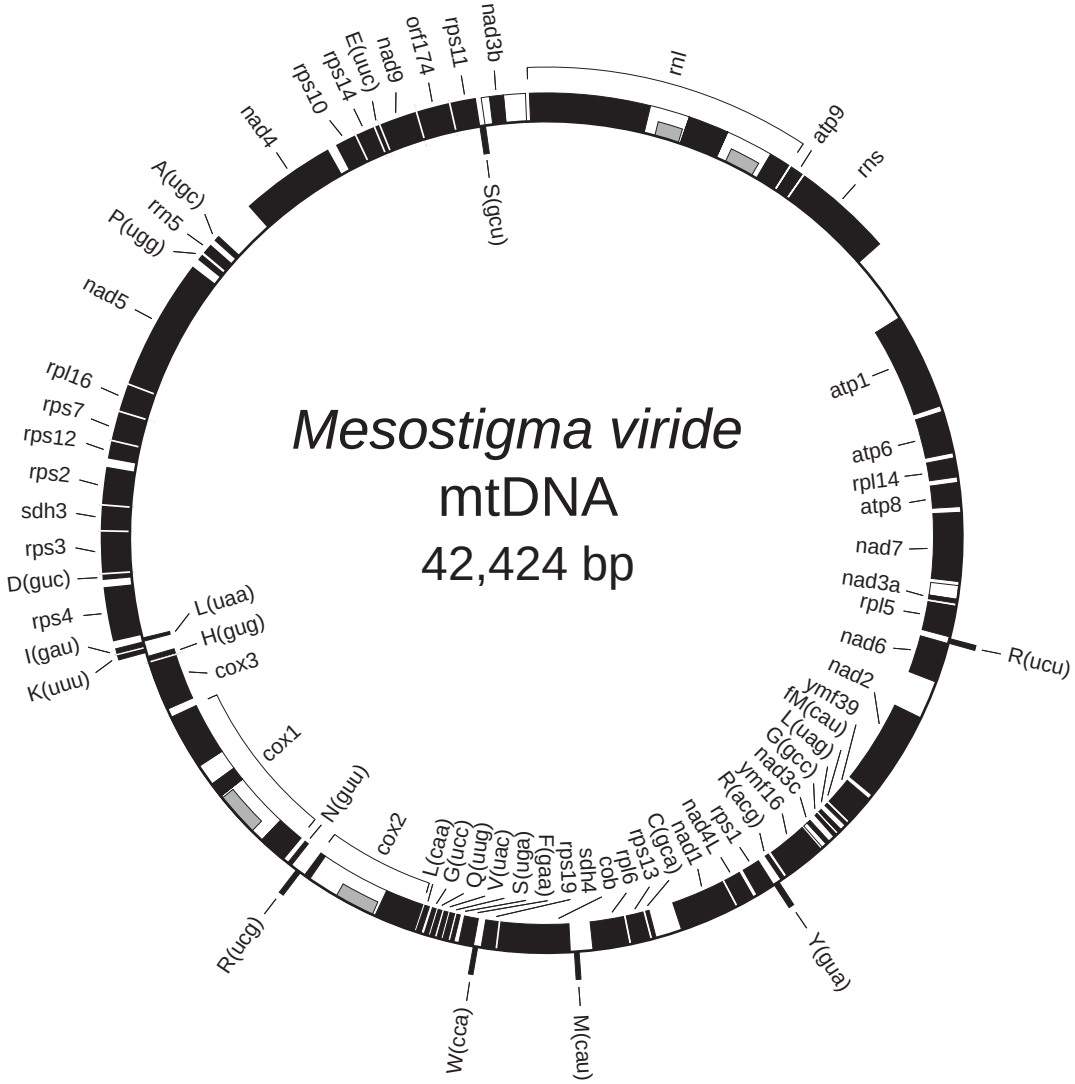

Mesostigma viride
mtDNA
42,424 bp
nad3b
rps11
orf174
nad9
E(uuc)
rps14
rps10
nad4
A(ugc)
rrn5
P(ugg)
nad5
rpl16
rps7
rps12
rps2
sdh3
rps3
D(guc)
rps4
I(gau)
K(uuu)
L(uaa)
H(gug)
cox3
cox1
N(guu)
R(ucg)
cox2
L(caa)
G(ucc)
Q(uug)
V(uac)
S(uga)
F(gaa)
rps19
sdh4
cob
rpl6
rps13
C(gca)
nad1
nad4L
rps1
R(acg)
ymf16
nad3c
G(gcc)
L(uag)
fM(cau)
ymf39
nad2
nad6
rpl5
nad3a
rpl14
atp8
atp6
atp1
Y(gua)
M(cau)
W(cca)
S(gcu)
rnl
atp9
rns
R(ucu)

Tableau 1.3 Les génomes mitochondriaux

Liste des organismes dont la séquence du génome mitochondrial fait partie de la base de données construite dans le cadre de ce projet en date du 9 décembre 2003. Pour les génomes incomplets ou divisés en plusieurs segments, la taille est la somme des tailles de chacun des segments disponibles. Le nombre de gènes a été calculé à partir des gènes présents dans la base de données. Le numéro d'accession correspond à celui retrouvé dans la base de données du NCBI. L'abréviation n.d. indique que le génome n'est pas disponible dans la base de données du NCBI.

Organisme	Taille (bp)	Nbre de gènes	Numéro d'accession
Cyanidioschyzon merolae	32 211	62	NC_000887
Porphyra purpurea	36 753	49	NC_002007
Chondrus crispus	25 836	50	NC_001677
Rhodomonas salina	48 063	67	NC_002572
Ochromonas danica	41 035	58	NC_002571
Laminaria digitata	38 007	62	NC_004024
Pylaiella littoralis	58 507	61	NC_003055
Nephroselmis olivacea	45 223	65	AF110138
Pycnococcus prasovali	24 321	35	n.d.
Monomastix species OKE-1	60 883	65	n.d.
Scherffelia dubia	78 958	61	n.d.
Prototheca wickerhamii	55 328	61	NC_001613
Oltmannsiellopsis viridis	56 761	54	n.d.
Pseudendoclonium akinetum	95 880	57	n.d.
Pedinomonas minor	25 137	21	NC_000892
Scenedesmus obliquus	42 781	42	NC_002254
Chlamydomonas eugametos	22 897	12	NC_001872
Chlorogonium elongatum	22 704	12	Y13643, Y13644,Y07814
Chlamydomonas reinhardtii	15 758	12	NC_001638
Mesostigma viride	42 424	65	AF353999
Chaetosphaeridium globosum	56 574	67	AF494279
Chara vulgaris	67 737	68	NC_005255
Marchantia polymorpha	186 609	69	NC_001660
Beta vulgaris	368 799	50	NC_002511
Arabidopsis thaliana	366 924	50	NC_001284
Oryza sativa	490 520	54	AB076665, AB076666

Tableau 1.4 Les gènes mitochondriaux

Liste des gènes mitochondriaux présents dans la base de données développée durant ce projet en date du 9 décembre 2003, groupés selon leurs produits ou leurs fonctions.

Produits des gènes	Gènes
Protéines ribosomales de la grande sous-unité	*rpl1, rpl2, rpl5, rpl6, rpl10, rpl11, rpl14, rpl16, rpl18, rpl19, rpl20, rpl27, rpl31, rpl32, rpl34*
Protéines ribosomales de la petite sous-unité	*rps1, rps2, rps3, rps4, rps7, rps8, rps10, rps11, rps12, rps13, rps14, rps19*
NADH déshydrogénase	*nad1, nad2, nad3, nad4, nad4L, nad5, nad6, nad7, nad8, nad9, nad10, nad11*
Succinate :ubiquinone oxidoréductase	*sdh2, sdh3, sdh4*
Ubiquinol :cytochrome c oxidoréductase	*cob*
Cytochrome c oxidase	*cox1, cox2, cox3, cox11*
ATP synthase	*atp1, atp3, atp4, atp6, atp8, atp9*
Biogénèse du cytochrome c	*yejR, yejU, yejV, yejW*
Voie de translocation de protéines sec-indépendente	*mttB*
ARN polymérase	*rpoA, rpoB, rpoC, rpoD*
Autres protéines	*secY, tufA*
ARNs ribosomaux	*rnl, rns, rrn5*
ARNs de transfert	*trnA(ugc), trnC(gca), trnD(guc), trnE(uuc), trnF(gaa), trnG(gcc), trnG(ucc), trnH(gug), trnI(aau), trnI(cau), trnI(gau), trnI(uau), trnK(uuu), trnL(aag), trnL(caa), trnL(cag), trnL(cua), trnL(gag), trnL(uaa), trnL(uag), trnM(cau), trnfM(cau), trnN(guu), trnP(ugg), trnQ(uug), trnR(acg), trnR(ccu), trnR(gcg), trnR(ucg), trnR(ucu), trnS(acu), trnS(gcu), trnS(gga), trnS(uga), trnT(ggu), trnT(ugu), trnV(gac), trnV(uac), trnW(cca), trnW(uca), trnY(gua)*
Autres ARN	*rnpB*

1.3 Informatique

Grâce à l'automatisation et à la puissance grandissante des ordinateurs, il est maintenant possible d'effectuer des analyses qui auraient été difficilement envisageables auparavant. Malgré que le fait que la puissance des ordinateurs double environ tous les 18 mois et que la quantité de données biologiques croît selon un rythme similaire, la disponibilité de chercheurs pouvant analyser ces données ou concevoir des outils pour en accélérer l'analyse reste limitée.

L'étape limitante est maintenant devenue l'analyse des données. Une fois le séquençage d'un génome complété, il reste à identifier les gènes et leurs positions. Dans le laboratoire des Drs Lemieux et Turmel, ce travail, qui était jusqu'alors effectué à l'aide des logiciels de la suite GCG[10], nécessitait d'identifier un par un chaque cadre de lecture ouvert et chaque région codant potentiellement pour un gène en faisant une recherche BLAST[11] sur le serveur du NCBI[12].

1.4 Analyses phylogénétiques

Un des objectifs des recherches effectuées dans mon laboratoire d'accueil est de déterminer les relations phylogénétiques entre les différentes algues étudiées. Principalement, le but est d'identifier les organismes les plus proches des organismes ancestraux étant à l'origine des différentes familles d'organismes photosynthétiques (par exemple, les algues rouges et les plantes terrestres) et de déterminer l'ordre de branchement des différentes familles d'algues.

Les séquences des génomes chloroplastiques et mitochondriaux sont utilisées pour effectuer les analyses phylogénétiques nécessaires à ces recherches. Les génomes d'organites, comparativement aux génomes nucléaires, ont l'avantage d'être plus petits donc séquençable plus rapidement et à moindre coût. De plus, ils évoluent plus lentement et sont plus stables. Cela permet de résoudre des phylogénies faisant intervenir des divergences très anciennes.

Actuellement, les algues vertes sont les principaux sujets d'étude. Elles se divisent en deux grandes classes : les chlorophytes et les streptophytes. Ces derniers incluent, en plus de certaines algues, toutes les plantes terrestres.

Avant d'effectuer une analyse phylogénétique, les gènes doivent avoir été identifiés. Une fois l'identification des gènes effectuée, leurs séquences sont utilisées pour déterminer la position phylogénétique de l'organisme. Cela nécessite plusieurs étapes de traitement qui doivent être effectuées individuellement sur chaque gène. Pour cette raison, il est souhaitable de concevoir des outils bioinformatiques pour automatiser ces tâches. De plus, les analyses phylogénétiques sont très exigeantes en temps de calcul.

1.5 Objectifs du projet

Avec la quantité grandissante d'information provenant du séquençage, il est de plus en plus pénible de faire manuellement les recherches d'homologies et d'identifier les gènes en utilisant seulement les outils traditionnels. En plus, l'extraction des séquences, leur formatage et leur préparation pour effectuer des analyses phylogénétiques comportent de grands risques d'erreurs humaines avec une telle quantité d'information. Il était donc devenu essentiel d'automatiser les tâches de préparation et d'analyse de données pour profiter pleinement des nouveaux génomes disponibles.

Pour ce faire, il sera nécessaire de développer des outils informatiques permettant d'automatiser et d'accélérer le traitement de données provenant du séquençage d'organites. Ces outils effectueront des recherches automatisés d'homologies et l'identification des gènes. Une base de données locale des génomes d'organites sera nécessaire puisqu'un grand nombre de génomes produits par le laboratoire n'ont pas encore été rendu disponibles dans les bases de données publiques et il est souhaitable d'utiliser leurs séquences le plus rapidement possible. En plus, des outils d'extraction et de formatage des séquences seront nécessaires pour maintenir cette base de données des génomes chloroplastiques et mitochondriaux. La structure de cette base de données et les formats employés pour stocker les séquences devront être déterminés.

Finalement, des outils pour extraire les séquences de la base de données et les préparer pour effectuer des analyses phylogénétiques seront aussi nécessaires. Ils devront permettre d'éliminer les risques de manipulation humaine et de rendre facilement reproductibles les résultats en éliminant les choix subjectifs. Ces automatisations rendront possible l'exécution de nouveaux types d'analyse phylogénétique basée sur les séquences d'acides nucléiques plutôt que seulement les séquences protéiques comme cela est actuellement fait. Pour démontrer le bon fonctionnement des outils, une analyse déjà publiée sera répétée et les résultats obtenus seront présentés.

En somme, les objectifs à réaliser dans ce projet peuvent se résumer comme suit :

- concevoir une base de données de gènes d'organites ;
- développer des outils pour maintenir cette base de données à jour ;
- développer des outils pour extraire les séquences de la base de données et les convertir au format désiré ;
- développer des outils pour effectuer automatiquement les recherches d'homologies et identifier les gènes d'un génome nouvellement séquencé ;
- développer des outils pour préparer les données pour effectuer des analyses phylogénétiques ;
- s'assurer du bon fonctionnement des outils en répétant une analyse dont les résultats ont été publiés ;
- documenter les outils de façon à les rendre utilisable par les autres membres du laboratoire.

Chapitre 2

Matériel et méthodes

2.1 Dépendences

Dans le cadre de ce projet, la grande majorité des outils ont été développés sous forme de scripts Perl et certains sous forme de scripts shell. Ils sont appelés scripts, car ils se présentent sous forme de fichiers textes lisibles et modifiables directement par un développeur. Contrairement aux logiciels compilés, ils peuvent être exécutés directement par l'interpréteur sans qu'il soit nécessaire d'effectuer préalablement une compilation. Comme tout programme informatique, les scripts conçus dans le cadre de ce projet ne fonctionnent correctement que si certaines conditions (dépendences) logicielles et matérielles sont remplies.

2.1.1 Système d'exploitation

Tous les scripts ont été testés et sont fonctionnels sur le système Mac OS X.2[13] et Redhat Linux 9[14]. Les machines utilisées sont respectivement un Dual G4 1.25 Ghz et un Pentium III 1.0 Ghz. Les scripts devraient être fonctionnels sans modifications majeures sur la plupart des systèmes de type UNIX sur lesquels perl est disponible. Cependant, il est nécessaire d'indiquer, à l'aide de la variable d'environnement `DBHOME`, l'emplacement des fichiers de données. Si la variable n'est pas définie, l'emplacement utilisé sera `/Users/Shared/db`, c'est-à-dire l'emplacement actuellement utilisé sous Mac OS X. Sous Linux, l'emplacement utilisé est `/home/db`. Il est donc nécessaire de définir `DBHOME` en exécutant `export DBHOME=/home/db` sous shell compatible sh ou `setenv DBHOME /home/db` sous un shell compatible csh. Ces commandes peuvent être

placées dans les fichiers `/etc/bashrc` ou `/etc/cshrc` si l'on veut éviter d'avoir à les exécuter dans chaque nouveau shell. Cependant, plusieurs scripts n'utilisent pas la base de données, donc ils n'ont pas besoin de connaître son emplacement.

Les scripts n'ont pas été adaptés pour fonctionner sous l'environnement Microsoft Windows[15], bien que les modifications à effectuer seraient probablement mineures. La principale source de problème se situe au niveau des noms de fichiers et de répertoires qui ont un format légèrement différent sous l'environnement Windows. Puisque Perl et Bioperl sont disponibles sous Windows, il devrait être relativement facile de rendre les scripts utilisables dans Windows.

2.1.2 Perl

Tous les outils créés dans le cadre de ce projet sont codés en Perl[16], "Practical Extraction and Report Language". Perl est un langage optimisé pour le traitement de fichiers textes, l'extraction d'information depuis ces fichiers et l'impression de résultats à partir de ces informations. C'est aussi un bon langage pour les tâches de gestion de système. Il a comme objectif d'être pratique (facile d'usage, efficace, complet) plutôt que beau (petit, élégant, minimal). De plus, puisque l'interpréteur Perl se charge de la compilation du code au moment de l'exécution, le développement du code est de beaucoup accéléré et la perte d'efficacité est très faible malgré l'utilisation d'un langage de script. De plus, son apprentissage peut être beaucoup plus progressif que plusieurs autres langages qui demandent d'avoir une connaissance beaucoup plus complète du langage avant de pouvoir l'utiliser. Ainsi, il est facilement possible de commencer à l'utiliser en ne connaissant qu'une petite partie de toutes ses possibilités. C'est l'ensemble de ces caractéristiques qui font que c'est un langage très populaire en bioinformatique et qu'il a été choisi pour réaliser ce projet.

Les scripts sont fonctionnels avec les versions perl 5.6.x et 5.8.x et ils devraient être utilisables avec toutes les versions perl 5 plus récentes. Certains modules Perl sont aussi nécessaires. `File::Basename` est utilisé dans certains cas pour faciliter le traitement des noms de fichiers et `Getopt::Long` sert à interpréter les options de la ligne de commande. Tous les autres modules utilisés font partie de la librairie Bioperl. `File::Basename` est inclus dans l'installation régulière de perl, mais `Getopt::Long` doit être installé séparément. Sous environnement Unix, une façon simple de l'installer est d'utiliser le module CPAN en lancant la commande `perl -MCPAN -e 'install Getopt::Long'`.

2.1.3 Bioperl

Bioperl[17] est une collection de plusieurs centaines de modules Perl destinés à la bioinformatique. Les modules de Bioperl sont écrits en style orienté objet. Ainsi, plusieurs modules dépendent les uns des autres pour accomplir la tâche qui leur est demandée. Cependant, il n'est pas nécessaire de très bien connaître la programmation orientée objet pour pouvoir les utiliser.

Les principaux modules qui sont utilisés dans ce projet sont SeqIO, Seq et AlignIO. Bien sûr, ces modules rendent disponibles des méthodes provenant de plusieurs autres modules qui ont aussi été utilisés à l'occasion.

Le module SeqIO est celui qui permet de lire et écrire des fichiers dans différents formats sans avoir à se soucier des spécifications exactes du format. Peu importe le format initial, SeqIO retourne toujours un objet de type Seq et cet objet peut être écrit dans n'importe lequel des formats supportés par SeqIO. De plus, les méthodes pour effectuer des traitements sur ces objets Seq sont les mêmes peu importe la source.

AlignIO est l'équivalent pour le traitement des formats de séquences alignées. En fait, il retourne un objet composé de plusieurs objets Seq. Ainsi, des méthodes spécifiques aux alignements peuvent être utilisées, mais il est aussi possible d'utiliser les méthodes spécifiques aux objets Seq sur chacune des séquences composant l'alignement.

2.1.4 BLAST

BLAST[11], "Basic Local Alignment Search Tool", est un outil de recherche d'homologies entre une séquence et une banque de séquences. La banque et la séquence peuvent être soit une séquence protéique ou nucléotidique. L'algorithme utilisé a été conçu pour être rapide en sacrifiant un minimum de sensibilité.

Pour qu'un utilisateur puisse effectuer une recherche BLAST, il faut avoir créé dans son répertoire usager un fichier `.ncbirc` pour indiquer l'emplacement des fichiers de données nécessaires à BLAST. Ce fichier doit contenir la ligne `[NCBI]` suivi de la ligne `Data=/usr/share/ncbi/data` sous Linux
ou de la ligne `Data=/usr/local/bin/blast/data` sous Mac OS X lorsque BLAST est intallé manuellement. Cette ligne peut varier selon la méthode d'installation utilisée. Elle indique le chemin correct vers les fichiers de données de BLAST, lesquels sont principalement des matrices de substitution.

2.1.5 MSPcrunch

MSPcrunch[18] est un outil de filtration de résultats de recherche BLAST. Il permet entre autre d'éliminer les résultats non significatifs ou provenant d'un biais de composition. De plus, il peut regrouper des résultats qui sont à courte distance les uns des autres de façon à obtenir un meilleur score. Il offre aussi plusieurs formats de sortie qui peuvent faciliter soit l'analyse informatique, soit l'interprétation visuelle.

2.1.6 clustalw et t_coffee

clustalw[19] et t_coffee[20] sont deux outils d'alignement multiple. clustalw utilise un algorithme d'alignement global alors que t_coffee utilise à la fois l'algorithme d'alignement global de clustalw et un algorithme d'alignement local. Il utilise l'information provenant des deux algorithmes pour produire des alignements qui sont souvent de meilleure qualité, mais cela au prix d'une exécution plus lente. Les deux outils procèdent en déterminant d'abord un arbre phylogénétique basé sur l'homologie entre les séquences et ensuite ils alignent successivement les séquences les plus proches en procédant de façon hiérarchique une paire à la fois.

2.1.7 Readseq

Quelques scripts utilisent l'utilitaire readseq[21] pour effectuer des conversions de formats de fichier. Cet utilitaire permet de lire et écrire des fichiers de séquences de différents formats de fichiers de séquences. Il est capable de détecter automatiquement le format du fichier lu et supporte plus de formats de sortie que les modules de Bioperl n'en supportent actuellement.

2.1.8 EMBOSS

EMBOSS[22] est un ensemble de logiciels gratuits pour l'analyse de séquence. L'utilitaire getorf est utilisé par le script findorf.pl pour identifier les cadres de lecture ouverts. Il peut être nécessaire de modifier le code génétique utilisé selon les besoins. findorf.pl utilise par défaut le code génétique 11. Le fichier correspondant est `EGC.11` dans le répertoire `/usr/share/EMBOSS/data/` sous Linux ou `/sw/share/EMBOSS/data/` sous Mac OS X lorsque EMBOSS a été installé en utilisant Fink.

2.1.9 Utilitaires Unix

Quelques utilitaires simples sont nécessaires tels que head, grep et sort. Ils sont habituellement retrouvés dans le paquet coreutils lequel inclus les paquets anciennement appelés fileutils, sh-utils et textutils. L'utilitaire sed doit aussi être installé pour certains scripts. Les scripts shell ont été conçus pour être exécutés sous shell bash. Il est donc nécessaire que bash soit présent sur le système (même si ce n'est pas le shell par défaut) pour pouvoir les utiliser. Tous ces utilitaires sont installés de façon standard avec la plupart des distributions Linux et Mac OS X.

2.2 Les formats de fichiers

Plusieurs formats de fichiers sont employés pour le stockage des données de séquences biologiques. Cela constitue un des grands problèmes de la bioinformatique. Il est donc utile de faire un survol rapide des formats les plus courants pour connaître les particularités de chacun.

2.2.1 Genbank

Dans le laboratoire, le format Genbank (Annexe A.1) est utilisé pour conserver toutes les informations concernant une séquence génomique. C'est un des rares formats qui permet l'annotation de séquences. Ce format ne peut contenir qu'une seule séquence. Cette séquence comporte plusieurs "features" lesquelles sont caractérisées par une position (location) et plusieurs étiquettes (tags) avec une valeur associée. Les "features" les plus courants sont `gene`, `CDS`, `rRNA` et `tRNA`. Toutes les "features" ont une étiquette `gene` qui les identifie. D'autres étiquettes courantes sont `product` et `translation`.

Ce format de fichier a par contre l'inconvénient d'être difficilement modifiable. Il est habituellement produit par Sequin à partir d'un fichier au format ASN.1. Sequin semble être le seul programme capable d'interpréter correctement ce format. Il n'est donc pas envisageable de travailler couramment avec le format ASN.1. Le format Genbank, bien que facilement compréhensible, nécessite quand même l'usage de logiciels pour en extraire l'information. La librairie Bioperl a été choisie pour sa flexibilité et sa simplicité d'utilisation, bien que d'autres auraient pu tout aussi bien convenir.

2.2.2 Fasta

Le format Fasta (Annexe A.2) est le format le plus utilisé dans nos bases de données. Il s'agit d'un format très simple qui contient peu d'informations. Chaque fichier Fasta peut contenir plusieurs séquences. Chaque séquence comporte une ligne d'identification débutant par un symbole > suivi sans espace d'un identifiant qui doit être unique pour le fichier. Tout autre texte présent sur la ligne débutant par > est considéré comme une description. Sur la ligne suivante débute la séquence biologique qui peut s'étendre sur plusieurs lignes jusqu'à la séquence suivante ou la fin du fichier. Ce format peut facilement être modifié à la main au besoin ou par des programmes simples.

Le format Fasta a aussi la particularité de pouvoir être utilisé avec des séquences alignées. Ainsi, Bioperl et Gblocks ont la possibilité de traiter des fichiers Fasta dont les séquences comportent des brèches (gaps) comme des fichiers d'alignement.

2.2.3 Longueur d'exons

Les fichiers de longueurs d'exons (Annexe A.3) ont un format texte simple. Chaque ligne débute par un identifiant d'organisme suivi des longueurs en nucléotides des exons séparées par des espaces. Un organisme dont le gène ne contient pas d'introns n'est tout simplement pas inscrit.

2.2.4 Liste d'organismes

Les listes d'organismes (Annexe A.4) sont une liste d'identifiant d'organismes avec un identifiant par ligne dans l'ordre qui est désiré pour les fichiers de sortie. Chaque identifiant d'organisme est composé des quatre premières lettres du genre en majuscule suivies des cinq premières lettres du nom de l'espèce en minuscule. Les organismes proches au niveau phylogénétique sont habituellement regroupés dans les listes d'organismes et dans les fichiers de séquence. Cela rend plus apparentes les similitudes entre les séquences.

2.2.5 Clustalw

Le format clustalw (Annexe A.5) est un format d'alignement utilisé à la fois par clustalw, d'où sont nom, et par t_coffee, un autre programme d'alignement. La première ligne du fichier identifie le format et le programme ayant produit l'alignement. Après au moins une ligne blanche de séparation, le premier bloc d'alignement débute. Chaque ligne débute par l'identifiant d'organisme. Des espaces sont ajoutés jusqu'à avoir 16 caractères avant le début de la séquence qui comporte au maximum 60 caractères. Une ligne de symboles indiquant le degré d'homologie de la colonne peut être ajoutée après chaque bloc. Au moins, une ligne blanche doit séparer les blocs d'alignement.

2.2.6 Selex

Le format Selex (Annexe A.6) utilisé dans le laboratoire est inspiré du format Selex original, mais il a été adapté à des besoins spécifiques. Il est utilisé pour visualiser les alignements de séquences codantes. Chaque ligne débute par l'identifiant d'organisme suivi par les positions de la séquence qui sont présentes sur cette ligne. Ensuite, débutant à la $23^{ième}$ colonne, vient la séquence avec chaque codon séparé par des espaces. Si des introns sont présents dans cette protéine, leur position est indiquée par le symbole > à la fin de l'exon précédent et < au début de l'exon suivant. Cependant, la séquence de l'intron n'est pas présente dans ces fichiers, seulement la séquence codant pour la protéine est présente. Chaque ligne comporte au maximum 33 codons. Si la séquence n'est pas complète, elle se poursuit dans le bloc suivant. Chaque bloc est séparé du précédent par trois lignes blanches. Les lignes qui indiquent les positions des introns débutent par un dièse (#). Toutes les lignes débutant par un dièse sont ignorées lors de la lecture de la séquence, il est donc possible d'ajouter d'autres informations aux séquences.

Chapitre 3

Résultats

3.1 Les scripts

Dans le cadre de ce projet, de nombreux scripts (Tableau 3.1) ont été développés. Ici sont énumérés leurs fonctionnements et leurs rôles. Des descriptions de leur utilisation typique seront présentées dans les chapitres suivants.

3.1.1 addexons.pl

addexons.pl (Annexe B.1) permet de produire un alignement de séquences codantes au format Selex. Il prend comme premier argument un fichier Fasta contenant les séquences codantes alignées d'une protéine et comme deuxième argument un répertoire contenant un fichier de longueur d'exons portant le même nom. Il produit dans le répertoire courant un fichier contenant les séquences alignées au format Selex.

Il utilise le module AlignIO pour lire l'alignement de séquences codantes en format Fasta. La séquence est d'abord convertie en majuscule pour uniformiser la présentation. Ensuite, un espace est ajouté après chaque trois caractères de séquence valides pour délimiter les codons. La séquence est alors divisée pour être affichée sur plusieurs lignes et les positions de jonction d'exons sont calculées. Si le fichier de longueur d'exons n'existe pas, aucune position de jonction d'exons ne sera ajoutée. Finalement, les entêtes sont ajoutés et l'alignement formaté est écrit dans le fichier Selex portant le même nom que le fichier d'entrée.

Tableau 3.1 Résumé des fonctions des scripts

Résumé des fonctions de chacun des scripts décrits dans ce chapitre.

Script	Fonction
addexons.pl	Crée un alignement Selex à partir de longueurs d'exons et d'un alignement de séquences codantes
align.pl	Aligne plusieurs fichiers en une seule commande avec t_coffee ou clustalw, au choix
choosetpl.pl	Sélectionne les topologies avec une valeur de bootstrap non nulle
codonalign.pl	Aligne des séquences codantes en se basant sur l'alignement des protéines
filterselex.pl	Convertie des fichiers Selex au format Fasta en conservant seulement les séquences demandées
filtertfa.pl	Produit des fichiers Fasta contenant seulement les séquences demandées
findorf.pl	Identifie les cadres de lecture ouverts et les gènes d'un nouveau génome par des recherches BLAST
formatdb.sh	Formate les séquences pour les recherches BLAST
gbkextract.pl	Extrait les informations contenues dans un fichier Genbank
preparednamatrix.pl	Prépare une matrice de séquences codantes pour une analyse phylogénétique
prepareproteinmatrix.pl	Prépare une matrice de protéines pour une analyse phylogénétique
prepsequin.pl	Prépare le fichier pour annoter les protéines dans Sequin
tfa2long.sh	Convertie un identifiant d'organisme en une description longue
updatealign.pl	Met à jour les alignements Selex à partir des alignements de protéines, des séquences codantes et des longueurs d'exons

3.1.2 align.pl

align.pl (Annexe B.2) est un script facilitant l'alignement de plusieurs fichiers de séquences automatiquement. Il aligne la liste de fichiers qui lui est passée en paramètre en faisant appel à t_coffee (par défaut) ou à clustalw (si l'option `-c` est utilisée). Il utilise l'option `-outorder=input` pour s'assurer que l'ordre des séquences dans le fichier d'alignement est le même que dans le fichier d'entrée. Les fichiers d'alignement produits sont en format clustalw puisque c'est le format de sortie par défaut des deux programmes d'alignement. Les autres options sont laissées à leurs valeurs par défaut.

3.1.3 choosetpl.pl

choosetpl.pl (Annexe B.3) sert à filtrer les topologies ayant une valeur de bootstrap non nulle. Il prend comme fichiers d'entrée la liste de topologies produite par la recherche exhaustive de protml et le résultat de l'évaluation de ces topologies qui est produit par protml lorsqu'aucune recherche de topologies n'est demandée. Il vérifie une à une les topologies et s'assure que leur valeur bootstrap est non nulle. Les topologies retenues sont affichées sur la sortie standard.

3.1.4 codonalign.pl

codonalign.pl (Annexe B.4) prend comme paramètres un répertoire contenant les alignements de protéines et une liste de fichiers Fasta contenant les séquences codantes à aligner. Pour chaque fichier de séquence codante, un fichier au format Fasta aligné est produit avec l'extension .dna_aln. Dans cet alignement, chaque acide aminé de l'alignement de protéine est remplacé par les trois nucléotides (le codon) ayant la position correspondante dans le fichier de séquences codantes et chaque symbole de brèche (gap) est remplacé par trois symboles de brèche. Il n'y a pas de vérification qui est faite pour s'assurer que le codon code vraiment pour l'acide aminé qu'il remplace. Cela peut entraîner des erreurs si les fichiers de séquences ne correspondent pas, mais cela évite d'avoir à tenir compte des différents codes génétiques pouvant être utilisés dans un même alignement. Cependant, aucun alignement n'est produit et un message d'erreur est affiché si les longueurs des séquences ne correspondent pas ou si l'ordre des séquences n'est pas la même dans les deux fichiers d'entrée.

3.1.5 filterselex.pl

filterselex.pl (Annexe B.5) prend en entrée une liste d'organismes à conserver et des fichiers Selex à filtrer. Il produit des fichiers au format Fasta dans le répertoire courant qui contiennent seulement la séquence codante des organismes demandés et les séquences sont placées dans le même ordre que les organismes dans la liste. Les seuls caractères qui sont conservés sont 'A', 'C', 'G', 'T' et 'N', de façon à enlever tout l'espacement et les symboles de brèches (gaps). Toutes les lignes débutant par un dièse sont ignorées lors de la lecture.

3.1.6 filtertfa.pl

filtertfa.pl (Annexe B.6) prend en entrée une liste d'organismes à conserver et des fichiers Fasta à filtrer. Il produit des fichiers au format Fasta dans le répertoire `filtered` qui contiennent seulement la séquence des organismes demandés et les séquences sont placées dans le même ordre que les organismes dans la liste. Aucune attention n'est accordée à savoir de quel type de séquence il s'agit ou même si ce sont des séquences valides.

3.1.7 findorf.pl

findorf.pl (Annexe B.7) permet d'identifier les positions des gènes dans un génome nouvellement séquencé. Il procède en deux étapes. D'abord, il utilise le programme getorf de la suite EMBOSS pour trouver les cadres de lecture ouverts. Les séquences de ces cadres de lecture ouverts et leurs traductions sont placés dans deux répertoires et chaque traduction est soumise à une recherche BLAST contre les bases de données locales pour tenter de l'identifier. Si une identification est trouvée, elle est utilisée comme nom de séquence dans le fichier correspondant. Ensuite, le génome est divisé en courtes séquences de 100 bases de longueur et ces séquences sont recherchées dans les bases de données. Dans toutes les recherches BLAST, seuls les résultats provenant du même organite que le génome sont conservés. Les résultats de ces recherches sont présentés dans un fichier tabulé avec pour chaque tranche de cent bases les meilleurs résultats BLAST et les cadres de lecture ouverts présents à cette position.

3.1.8 formatdb.sh

Il s'agit d'un des seuls scripts qui n'est pas écrit en Perl, mais plutôt en langage shell (Annexe B.8). Ce choix a été fait, car il s'agit simplement d'une liste d'instructions qui doivent être exécutées une à la suite de l'autre. Ce script est utilisé pour mettre à jour la base de données utilisée par BLAST pour les recherche d'homologie. Il traite séparamment les séquences protéiques et les séquences d'ARN. Dans les deux cas, il transforme d'abord l'identifiant unique en une description plus détaillée qui inclut le nom du gène, l'organite et le nom complet de l'organisme. Pour ce faire, il utilise tfa2long.sh. Cette opération est effectuée sur tous les fichiers contenus dans `prot_seq` et le résultat est concaténé en un seul fichier nommé `prot_db.tfa`. Ensuite, l'utilitaire BLAST `formatdb` est appelé pour formater la base de données de séquences protéiques utilisée par BLAST. Les mêmes opérations sont ensuite exécutées avec les fichiers contenus dans `RNA_seq` et `formatdb` est appelé avec les options convenables pour une base de données d'acides nucléiques.

3.1.9 gbkextract.pl

gbkextract.pl (Annexe B.9) prend un fichier Genbank comme entrée et il en extrait les "features". Il crée cinq répertoires : `RNA`, `prot`, `cDNA`, `gene`, `exon` et `intron`. `RNA` contient les séquences d'ARN ribosomique et d'ARN de transfert. `prot` contient les séquences protéiques venant de l'étiquette `translation` de chaque "feature" CDS. `cDNA` contient la séquence codante de chaque gène en excluant la séquence des introns. `gene` contient les séquences complètes de chaque gène incluant les introns. `exon` contient les fichiers de longueurs d'exon pour chacun des gènes ayant des introns. `intron` contient la séquence de tous les introns avec un fichier par gène contenant des introns. Un fichier contenant la taille de chaque intron est aussi produit. Il est nécessaire de fournir à gbkextract.pl comme deuxième argument l'identifiant d'organisme pour qu'il puisse être correctement placé dans les différents fichiers.

3.1.10 preparednamatrix.pl

preparednamatrix.pl (Annexe B.10) prend comme entrée des alignements de séquences codantes au format Fasta. Chaque alignement est d'abord filtré par le programme Gblocks qui est utilisé avec l'option `-t=c` pour enlever les parties non conservées des séquences en multiples de trois nucléotides de façon à conserver un nombre

entier de codons. Ensuite, les séquences résultantes sont concaténées, puis un fichier au format nexus contenant la matrice de séquences codantes est produit pour être utilisé avec le logiciel PAUP. Un fichier nommé `Goptns.txt` est aussi produit. Il contient le nombre de gènes et la longueur de chacun d'eux dans la matrice au format nécessaire pour être utilisé par les logiciels de PAML.

3.1.11 prepareproteinmatrix.pl

prepareproteinmatrix.pl (Annexe B.11) prend en entrée des alignements de séquences de protéines au format clustalw. Chaque alignement est d'abord filtré par le programme Gblocks qui est utilisé avec l'option `-t=p` pour enlever les parties non conservées des séquences. Ensuite, les séquences résultantes sont concaténées, puis un fichier au format MOLPHY contenant la matrice de séquences protéiques est produit pour être utilisé avec le logiciel protml. Un fichier nommé `Goptns.txt` est aussi produit. Il contient le nombre de gènes et la longueur de chacun d'eux dans la matrice au format nécessaire pour être utilisé par les logiciels de PAML.

3.1.12 prepsequin.pl

Ce script (Annexe B.12) sert à préparer un fichier contenant toutes les séquences des protéines d'un génome au format demandé par Sequin pour qu'il puisse annoter automatiquement toutes ces protéines. Il demande à l'utilisateur l'identifiant de l'organisme et le nom du fichier à produire. Il utilise le fichier `gene_db.txt` comme source pour obtenir la description du produit des gènes. Lorsque la description d'un gène est introuvable, il laisse la description en blanc et Sequin, lors du chargement du fichier, laisse la possibilité à l'utilisateur d'en entrer une.

3.1.13 tfa2long.sh

Ce script (Annexe B.13) est employé pour transformer la ligne de description des fichiers Fasta de la base de données qui contient seulement l'identifiant d'organisme en une ligne de description plus détaillée nécessaire pour les recherches BLAST. Il prend comme premier paramètre le nom de l'organite d'origine et comme deuxième paramètre, le nom de fichier. Il procède en remplaçant chaque identifiant d'organisme par le nom

de la protéine, le nom de l'organite et le nom complet de l'organisme correspondant à l'identifiant.

3.1.14 updatealign.pl

updatealign.pl (Annexe B.14) est utilisé pour mettre les alignements Selex à jour. Il prend comme entrer la liste complète des organismes et les alignements de protéines à jour. D'abord, il utilise la liste d'organismes pour placer les alignements de protéines et les fichiers de séquences codantes présents dans la base de données dans le même ordre de façon à éviter que codonalign.pl ne produise des messages d'erreurs. Une erreur peut quand même se produire si la séquence d'un organisme est manquante dans l'un des deux fichiers. Ensuite, il appelle codonalign.pl pour aligner les séquences codantes, puis addexons.pl pour produire les fichiers Selex. Il utilise les fichiers contenus dans les répertoires `cDNA_seq` et `exon_db` qui doivent d'abord avoir été mis à jour.

3.2 La base de données

La base de données est actuellement divisée en plusieurs répertoires. Chacun de ces répertoires contient les sous-répertoires `cp` et `mt` pour séparer les informations mitochondriales et chloroplastiques. Ces répertoires sont `cDNA_alignments`, `cDNA_seq`, `complete_genomes`, `exon_db`, `prot_alignments`, `prot_seq` et `RNA_seq` (Figure 3.1). Ils contiennent respectivement les alignements de séquences codantes, les séquences codantes, les séquences de génomes complets, les longueurs d'exons, les alignements de séquences protéiques, les séquences protéiques et les séquences d'ARN. Les informations sont stockées de façon à avoir un fichier par gène, ce fichier contient les informations de tous les organismes possédant ce gène.

3.2.1 La mise à jour

La mise à jour de la base de données se fait en partie de façon manuelle et en partie de façon automatisée pour s'assurer de l'exactitude des données. Puisque la base de données est ensuite utilisée à plusieurs fins à l'intérieur du laboratoire, il est essentiel de s'assurer qu'elle contient un minimum d'erreurs. Plusieurs scripts sont utilisés pour les différentes étapes de la mise à jour (Figure 3.2).

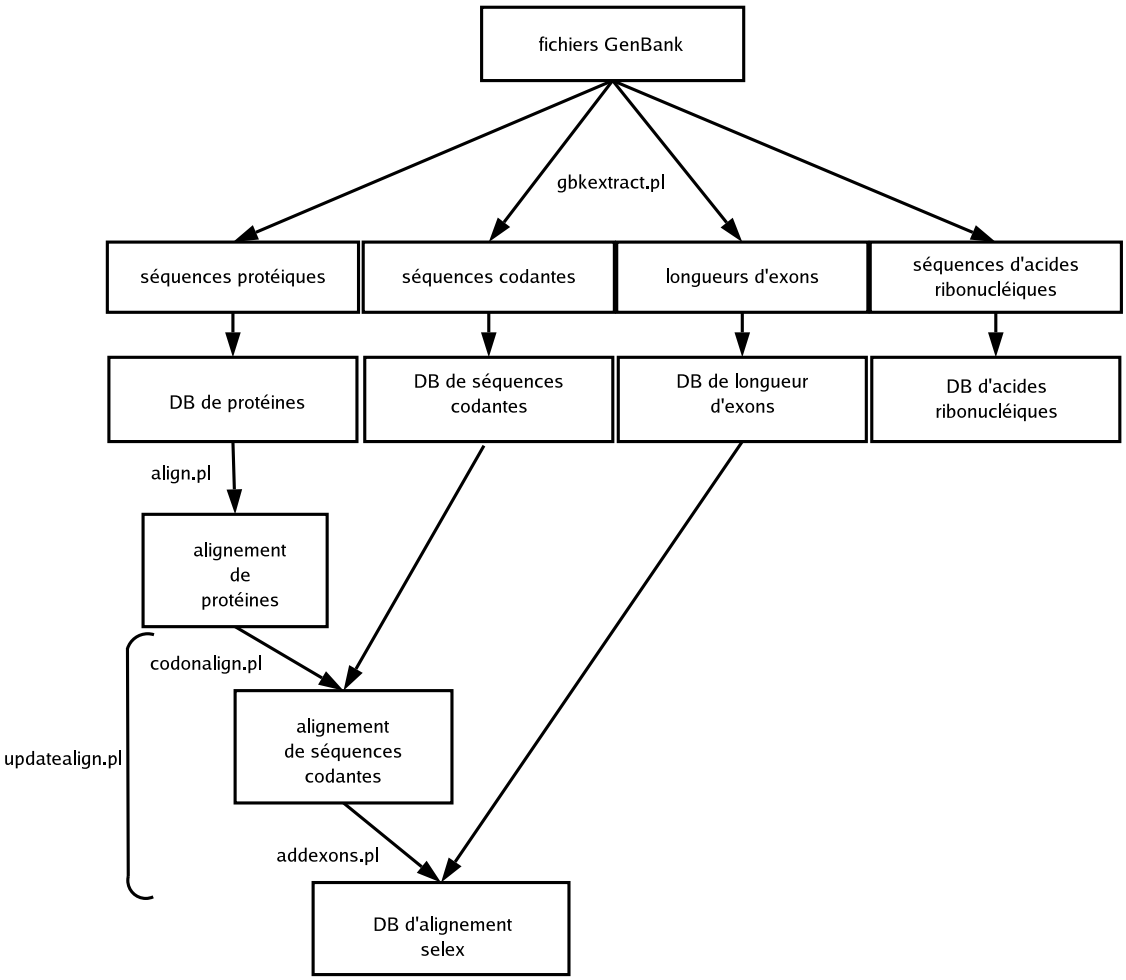
Figure 3.1 Structure de la base de données

Structure des répertoires de la base de données. Les répertoires se terminant par `_old` sont des copies de sauvegarde de la version précédente des répertoires correspondant. Les fichiers à l'intérieur des répertoires ne sont pas montrés.


db
cDNA_alignments
cp
mt
cDNA_seq
cp
cDNA_seq_old
cp
complete_genomes
cp
mt
exon_db
cp
mt
exon_db_old
cp
prot_alignments
cp
prot_seq
cp
mt
prot_seq_old
cp
mt
RNA_seq
cp
mt
RNA_seq_old
cp
mt

Figure 3.2 Maintien de la base de données

Schéma de fonctionnement des scripts servant à la mise à jour de la base de données. L'action d'un script est représentée par une flèche et les résultats sont indiqués dans les boîtes.


fichiers GenBank
gbkextract.pl
séquences protéiques
séquences codantes
longueurs d'exons
séquences d'acides ribonucléiques
DB de protéines
DB de séquences codantes
DB de longueur d'exons
DB d'acides ribonucléiques
align.pl
alignement de protéines
codonalign.pl
alignement de séquences codantes
updatealign.pl
addexons.pl
DB d'alignement selex

À partir d'un fichier Genbank

La première étape consiste à exécuter gbkextract.pl avec comme premier paramètre le fichier Genbank et comme deuxième paramètre l'identifiant que l'on désire donner au nouvel organisme. Il produira alors dans le répertoire courant six répertoires : `RNA`, `prot`, `cDNA`, `gene`, `exon` et `intron`. Les répertoires `gene` et `intron` ne sont pas utilisé actuellement dans la base de données. Les quatre autres correspondent directement à ce qui devrait être ajouté dans les répertoires de la base de données.

Lorsqu'il s'agit d'ajouter un nouveau génome produit à l'intérieur du laboratoire et que quelqu'un a déjà préparé le fichier Genbank correspondant, il est très simple de l'ajouter à la base de données. On suppose alors que le fichier ne comporte aucune erreur. Il suffit, avant d'exécuter gbkextract.pl, de créer les répertoires `RNA`, `prot`, `cDNA` et `exon` et de placer à l'intérieur les fichiers de la base de données. Une fois que gbkextract.pl aura été exécuté, les informations provenant du nouveau génome auront été ajoutées à la fin des fichiers correspondants. Si l'opération s'est bien déroulée, il suffit alors de remplacer les fichiers de la base de données par les fichiers mis à jour.

Cette méthode pourrait être utilisée pour régénérer complètement la base de données si tous les fichiers Genbank étaient disponibles avec une annotation standardisée et sans erreur. Il suffirait alors d'exécuter gbkextract.pl successivement sur chacun des fichiers Genbank pour que toutes les séquences soient extraites dans les mêmes répertoires.

Cependant, les annotations ne suivent pas un standard absolu et peuvent varier beaucoup selon les laboratoires et les versions des logiciels utilisées pour produire le fichier Genbank. De plus, il est fréquent de trouver des erreurs ou des oublis dans des fichiers Genbank provenant des bases de données publiques. Il est donc nécessaire, principalement lorsque les données proviennent de l'extérieur du laboratoire, de vérifier attentivement l'exactitude des annotations avant de faire l'ajout à la base de données.

Les principales étapes de la vérification consistent à aligner les gènes avec les gènes de la base de données pour s'assurer que les codons initiateur et terminateur ont été correctement identifiés, à s'assurer que les jonctions intron-exon ont été correctement déterminés et à vérifier qu'aucun gène n'a été oublié. Ces étapes sont effectuées de façon principalement manuelle avec l'aide de certains scripts décrits plus en détail dans la section suivante. Une fois que ces informations ont été vérifiées, elles peuvent être ajoutées dans les fichiers respectifs de la base de données. L'information, telle que produite par gbkextract.pl, peut être recopiée telle qu'elle à la fin du fichier correspondant dans la base de données.

À partir d'un génome nouvellement séquencé

Lorsqu'un génome vient tout juste d'être complètement séquencé, aucune information précise sur son contenu en gènes n'est connue. La seule chose connue est sa séquence brute. Le script findorf.pl a été développé pour procéder à l'identification des gènes à partir de la séquence brute.

Dans le cas le plus simple, l'utilisateur appelle le script findorf.pl avec comme paramètre un fichier Fasta nommé sous le format `GENRespec_cpDNA.tfa` (`cp` doit être remplacé par `mt` dans le cas d'une séquence mitochondriale et `GENRespec` est l'identifiant d'organisme). Il produit, par la méthode indiquée en 3.1.7, un fichier tabulé et trois répertoires. Il est aussi possible de spécifier comme deuxième paramètre l'organite d'origine du génome plutôt que dans le nom du fichier et un troisième paramètre peut être utilisé pour spécifier la longueur minimale en codons qu'un cadre de lecture ouvert non identifié doit avoir pour qu'il soit conservé dans le fichier de sortie tabulé. Cette valeur est par défaut de 28 codons.

Le fichier tabulé (présenté à l'annexe D.1) contient dans la première colonne la position dans le génome par tranche de 100 nucléotides. Les cinq colonnes suivantes contiennent le ou les résultats de la recherche BLAST contre la base de données préexistante. Les colonnes suivantes indiquent la taille et les positions de début et de fin des cadres de lecture ouverts présents à cette position s'il y en a. De plus, si le cadre de lecture ouvert a été identifié, le nom de la protéine sera aussi indiqué.

Les trois répertoires produits sont nommés d'après l'identifiant d'organisme avec des noms au format `GENRespec_seq`, `GENRespec_trans` et `GENRespec_identified`. Le premier contient la séquence en nucléotides de chacun des cadres de lecture ouverts. Le deuxième contient leurs séquences traduites selon le code génétique standard. Le troisième contient les cadres de lecture ouverts qui ont été identifiés comme correspondant à des protéines présentes dans la base de données.

L'étape suivante consiste à vérifier les cadres de lecture ouverts qui ont été identifiés par findorf.pl par alignement comme indiqué dans la section 3.2.1. Dans le cas des gènes contenant des introns, le fichier tabulé permet de situer approximativement chacun des exons, mais il sera nécessaire de déterminer les jonctions intron-exon de façon précise en déterminant la structure de l'intron. Pour ce qui est des cadres de lecture non identifiés, le fichier tabulé permet d'identifier ceux qui codent potentiellement pour une protéine. Il est ensuite nécessaire de faire une recherche BLAST contre la base de données complète du NCBI pour déterminer s'ils sont homologues à d'autres protéines connues.

Lorsque la séquence des cadres de lecture qui codent pour des protéines est correctement déterminée, il est possible en utilisant prepsequin.pl de préparer un fichier Fasta contenant toutes les séquences et qui peut être utilisé par Sequin pour annoter automatiquement la séquence. Il est cependant nécessaire de réviser tout de même l'annotation pour s'assurer de son exactitude.

Il est aussi possible d'intégrer directement les séquences protéiques vérifiées aux fichiers présents dans le répertoire `prot_seq` de la base de données en les copiant à la fin des fichiers correspondants. Cependant, cela provoque une désynchronisation de la base de données puisque les fichiers des autres répertoires de la base de données n'incluent pas les nouvelles séquences.

findorf.pl permet de trouver la position approximative des ARN de transfert et des ARN ribosomaux, mais pour déterminer leurs positions exactes, il est nécessaire de connaître leurs structures secondaires.

Dans le cas des ARN de transfert, le programme tRNAscan-SE[23] est utilisé. Il prédit la structure secondaire des ARNts et produit un fichier qui contient les positions de début et de fin de chacun des ARN de transfert et il identifie l'anticodon. Cette information peut être entrée dans Sequin manuellement.

La structure secondaire des ARN ribosomaux étant assez bien conservée, il est relativement facile de déterminer leurs structures de façon manuelle. Ensuite, il suffit d'entrer les positions de début et de fin dans Sequin pour que l'annotation soit suffisamment complète pour produire un fichier Genbank utilisable par gbkextract.pl.

Mise à jour des alignements Selex

Les alignements de séquences codantes sont une des rares parties de la base de données qui peut être générée de façon automatisée. Cependant, à la fois les séquences protéiques, les séquences codantes et les longueurs d'exon doivent avoir été mises à jour avant de procéder à cette étape.

D'abord, les séquences protéiques doivent être alignées. Le script align.pl permet d'effectuer automatiquement l'alignement pour plusieurs fichiers en utilisant le programme t_coffee. Ensuite, updatealign.pl utilise les alignements de protéines pour aligner les séquences codantes en utilisant codonalign.pl. Puis, il appelle addexons.pl pour convertir les alignements au format Selex et ajouter les informations concernant les exons.

Il est possible que des messages d'erreur soient produits et cela indique que certaines séquences ne correspondent pas exactement entre les différents répertoires. Il faut alors déterminer la source de l'erreur et la corriger pour obtenir un alignement complet.

Mise à jour des bases de données BLAST

Lorsque les séquences de protéines et d'ARN sont à jour, il est possible de mettre à jour la base de données de recherche BLAST. Cela se fait en exécutant un simple script shell, formatdb.sh, qui exécute les commandes nécessaires pour inclure une ligne de description plus complète de chaque séquence et pour appeler l'outil formatdb de BLAST qui formate les bases de données au format exigé par BLAST. Deux bases de données BLAST sont produites : une contenant les séquences de protéines et une contenant les séquences des ARN.

Ces bases de données seront réutilisées par la suite par findorf.pl lorsque de nouveaux génomes auront été séquencés. Plus la base de données contient d'organismes différents, plus il est facile d'identifier les gènes homologues dans les nouveaux génomes. De plus, le fait d'avoir une base de données locale contenant uniquement des séquences d'organites permet aux recherches BLAST d'être beaucoup plus rapides et de donner des résultats plus spécifiques aux besoins du laboratoire. En plus de findorf.pl, des recherches BLAST sont aussi effectuées durant l'étape du séquençage pour identifier la source et le contenu en gène des séquences individuelles et guider le choix des séquences à faire par la suite.

3.3 Les analyses phylogénétiques

Il est maintenant possible d'obtenir les séquences de génomes d'organites complets dans un délai suffisamment court pour rendre possible leur utilisation à des fins phylogénétiques. Maintenant, la pratique courante dans le laboratoire est d'utiliser toutes les protéines communes entre différents organismes pour déterminer leur relation phylogénétique plutôt que d'utiliser seulement les séquences de quelques protéines ou les séquences des ARN ribosomaux comme c'était le cas au cours des dernières années. Par contre, cette méthode est encore utilisée lorsqu'on désire effectuer une phylogénie comportant de nombreux organismes.

De nombreuses méthodes peuvent être employées pour tenter de déterminer les relations phylogénétiques existant entre différents organismes grâce aux données de sé-

quences d'organites. En autre, plusieurs types d'analyse sont possibles. Certaines analyses utilisent la distance, d'autres utilisent le plus petit nombre de changements (la parcimonie) alors que dans le groupe du Dre Turmel et du Dr Lemieux, les analyses utilisant le "maximum likelihood" (la plus grande probabilité d'être vrai) sont préférées lorsqu'il est possible de les utiliser.

Les analyses de type "maximum likelihood" évaluent un modèle d'évolution d'après les données. Ainsi, plus la valeur de "likelihood" est élevée, plus il est probable que le modèle d'évolution soit vrai et que les séquences utilisées aient vraiment évolué selon ce modèle. Le modèle d'évolution inclut à la fois les probabilités de mutation et l'arbre phylogénétique.

De plus, plusieurs types de données peuvent être utilisés pour effectuer des analyses phylogénétiques : des caractéristiques morphologiques, des séquences de gènes, des séquences de protéines ou des données sur l'ordre de gènes. Jusqu'à maintenant, ce sont surtout les séquences protéiques qui ont été utilisées dans le laboratoire, mais, avec le développement des outils de cette étude, il est maintenant possible d'utiliser les séquences nucléotidiques codant pour les protéines.

Ce chapitre se veut plutôt un guide d'utilisation des outils développés dans le cadre de ce projet qu'une revue des aspects théoriques des analyses. Certains outils sont utilisables avec d'autres types d'analyses utilisant des données structurées de façon similaire, mais seule l'utilisation la plus générique sera présentée ici.

Pour faciliter la compréhension, une analyse dont les résultats ont déjà été publiés[8] sera utilisée comme exemple. La préparation des données n'a pas à l'origine été effectuée de façon automatique et seules les séquences protéiques avaient été utilisées. Certaines différences minimes sont présentes entre les résultats actuels et ceux publiés en 2000 puisque les logiciels utilisés sont de versions différentes et que certaines méthodes de travail ont été légèrement modifiées.

La préparation des données pour une analyse phylogénétique nécessite plusieurs étapes. Quelques-unes de ces étapes sont communes aux analyses de séquences de protéines (Figure 3.3) et aux analyses de séquences codantes (Figure 3.4) alors que d'autres leur sont particulières.

Figure 3.3 Analyse de séquences protéiques

Schéma de fonctionnement des scripts servant à la préparation des données et au traitement des résultats lors d'une analyse phylogénétique utilisant des séquences protéiques. L'action d'un script est représentée par une flèche et les résultats sont indiqués dans les boîtes.

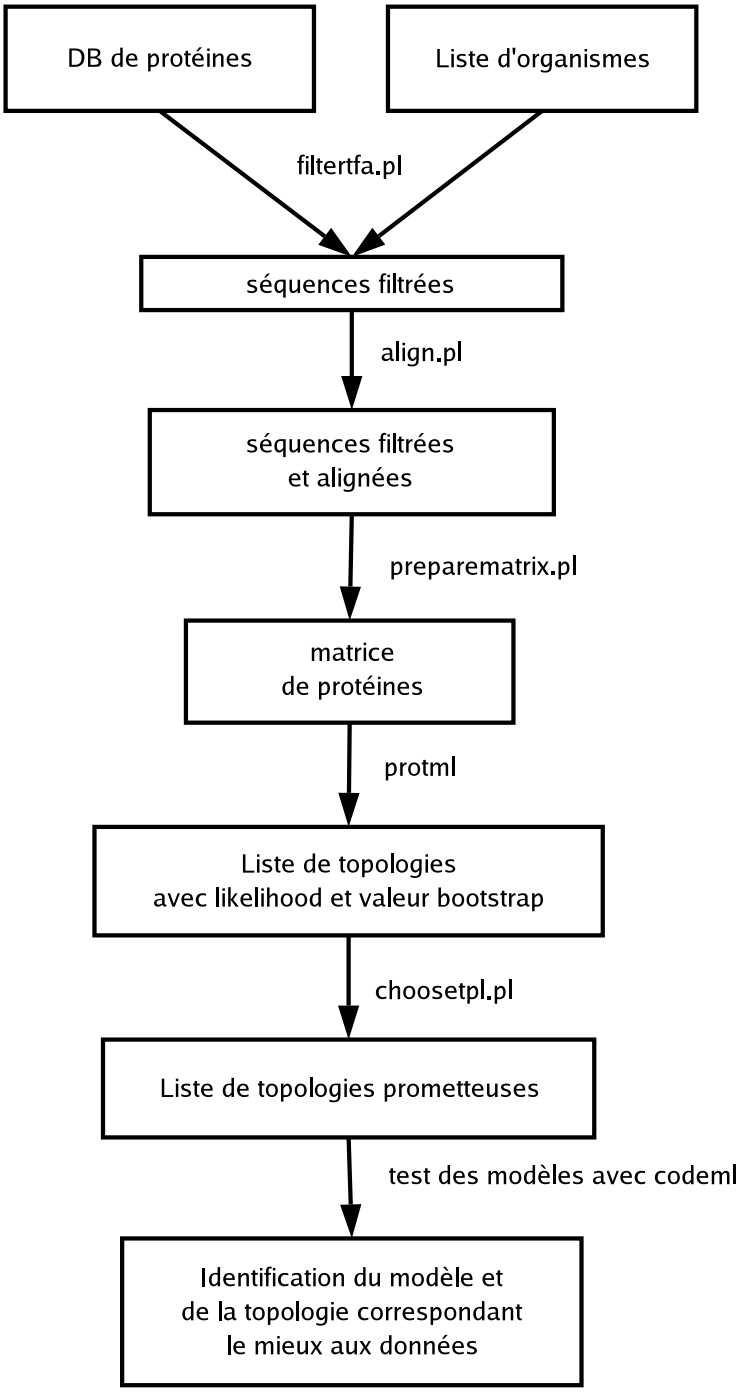
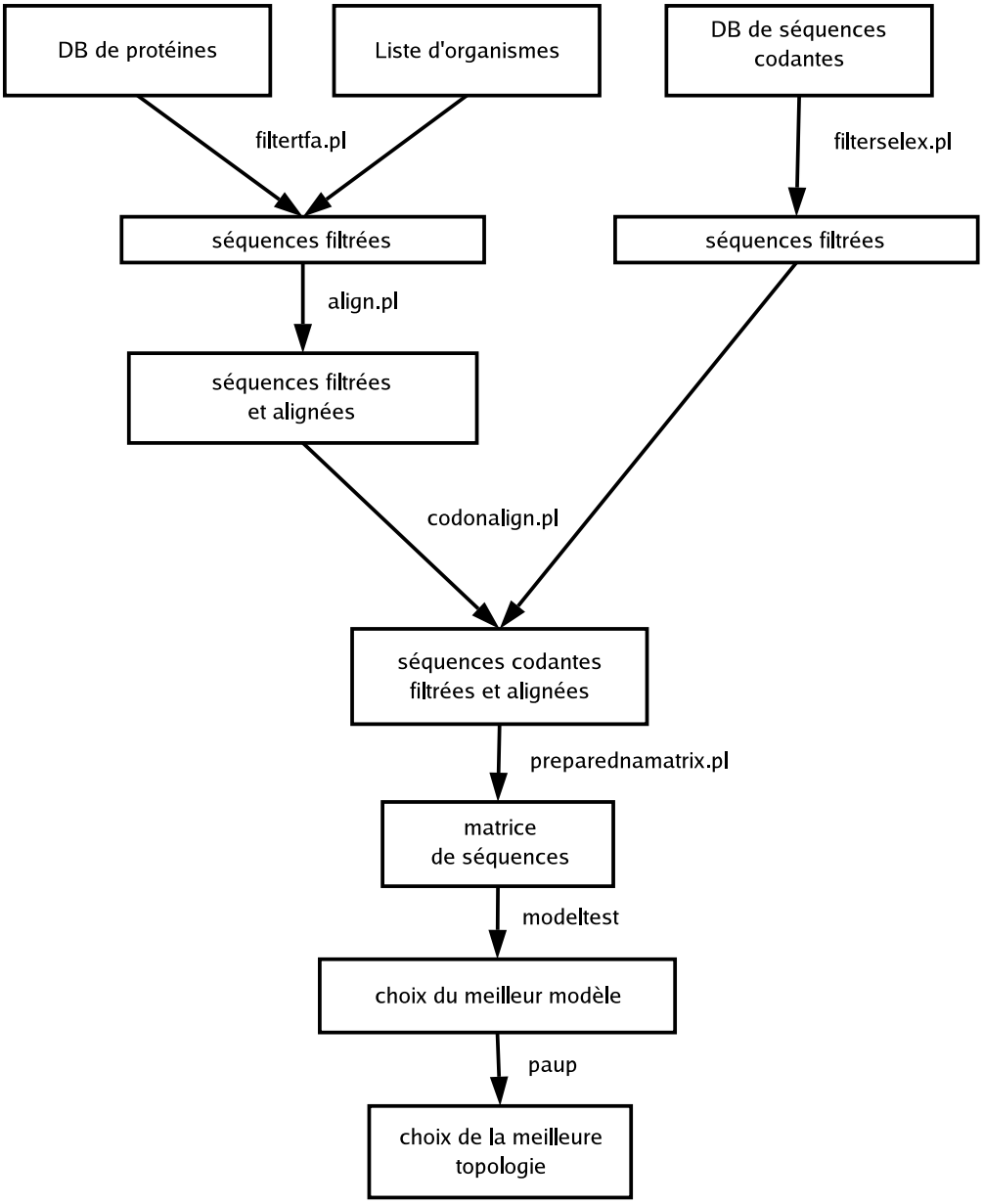

DB de protéines
Liste d'organismes
filtertfa.pl
séquences filtrées
align.pl
séquences filtrées
et alignées
preparematrix.pl
matrice
de protéines
protml
Liste de topologies
avec likelihood et valeur bootstrap
choosetpl.pl
Liste de topologies prometteuses
test des modèles avec codeml
Identification du modèle et
de la topologie correspondant
le mieux aux données

Figure 3.4 Analyse de séquences codantes

Schéma de fonctionnement des scripts servant à la préparation des données lors d'une analyse phylogénétique utilisant des séquences codantes. L'action d'un script est représentée par une flèche et les résultats sont indiqués dans les boîtes.


DB de protéines
Liste d'organismes
DB de séquences codantes
filtertfa.pl
filterselex.pl
séquences filtrées
séquences filtrées
align.pl
séquences filtrées et alignées
codonalign.pl
séquences codantes filtrées et alignées
preparednamatrix.pl
matrice de séquences
modeltest
choix du meilleur modèle
paup
choix de la meilleure topologie

3.3.1 Étapes communes

La première étape consiste à choisir les organismes à étudier. C'est une étape très importante qui est souvent prise à la légère. Il est nécessaire d'avoir des représentants des différents groupes impliqués dans le problème à l'étude. Il faut avoir suffisamment de représentants pour minimiser les longueurs des branches et éviter les effets d'attraction de longues branches, mais chaque organisme supplémentaire fait croître le temps de calcul nécessaire de façon exponentielle. Idéalement, les séquences utilisées devraient avoir une composition qui varie le moins possible entre les organismes. De plus, il faut choisir un groupe de référence (outgroup) qui est clairement extérieur aux autres groupes étudiés, mais sans être trop éloigné et causer de l'attraction.

Pour l'étude en question, le problème consistait à déterminer la position de l'algue *Mesostigma viride* appartenant à la classe des prasinophytes. Sa position fait l'objet d'une controverse puisque tous les autres prasinophytes examinés jusqu'à maintenant se classent à la bases des chlorophytes. Cependant, certaines de ses caractéristiques morphologiques[24] et des analyses phylogénétiques[25][26][27] basées sur un nombre limité de gènes la placent à la base des streptophytes.

Ainsi, trois représentants de chacun de ces deux groupes ont été choisis. Pour les streptophytes : *Marchantia polymorpha*, *Pinus thunbergii* et *Nicotiana tabacum*. Pour les chlorophytes : *Nephroselmis olivacea*, *Chlorella vulgaris* et *Pedinomonas minor*. Le groupe de référence (outgroup) qui a été choisi est *Cyanophora paradoxa*, une algue faisant partie des glaucocystophytes. Les caractéristiques des cyanelles des glaucocystophytes suggèrent qu'elles sont des chloroplastes primitifs ayant une association moins développée avec la cellule hôte[28].

Pour procéder à une analyse, il faut d'abord préparer une liste de leurs identifiants. Dans ce cas-ci, le fichier (couramment nommé Taxa) contiendra :

```
CYANparad
MESOvirid
NEPHoliva
CHLOvulga
PEDIminor
MARCpolym
PINUthunb
NICOtabac
```

Ensuite, il faut utiliser filtertfa.pl avec cette liste d'organismes pour filtrer les données de séquences protéiques. Puisque les chercheurs utilisant ces outils travaillent principalement sous Mac OS X, les exemples de commandes feront référence à l'emplacement de la base de données sous Mac OS X. Donc, filtertfa.pl sera exécuté avec la commande : `filtertfa.pl Taxa /Users/Shared/db/prot_seq/cp/*.tfa`. Cela produira un répertoire nommé `filtered` contenant 54 fichiers (Annexe D.2) contenant uniquement les séquences des protéines communes aux huit organismes contenus dans le fichier Taxa.

Ces séquences doivent ensuite être alignées avec l'aide de align.pl. Il suffit d'exécuter dans le répertoire filtered la commande `align.pl *.tfa` pour lancer l'alignement avec t_coffee. Cette étape peut être assez longue et exigeante en ressources systèmes selon le nombre de protéines et d'organismes. Dans certains cas, il peut être nécessaire d'utiliser l'option -c de align.pl lequel utilisera alors clustalw pour faire l'alignement. Les alignements obtenus avec clustalw sont, dans certains cas, de moins bonne qualité, mais ce programme est moins exigeant en ressources systèmes, surtout au niveau de la mémoire.

Avant d'entreprendre une analyse phylogénétique, il est nécessaire d'enlever les régions non conservées. Pour cela, le programme Gblocks[29] est utilisé. Il procède en déterminant les blocs de l'alignement qui sont conservés chez tous les organismes. Ce critère est essentiel pour effectuer une analyse par "maximum likelihood", car les modèles ne permettent pas de traiter les insertions et les délétions. Les blocs qui sont conservés à la suite d'une filtration par Gblocks ne contiennent aucune brèche et sont encadrés par des résidus conservés. De plus, le nombre de résidus non conservés successifs est limité. Les paramètres de filtration peuvent être ajustés quoique les paramètres par défaut soient habituellement utilisés. Une filtration trop stricte enlève beaucoup d'informations phylogénétiques alors que des paramètres trop souples peuvent produire des résultats divergents et augmenter l'effet des longues branches.

3.3.2 Analyse de séquences protéiques

Dans le cas d'une analyse avec des séquences protéiques, la filtration par Gblocks est effectuée à l'aide de prepareproteinmatrix.pl. Pour effectuer la filtration, il suffit d'exécuter dans le répertoire contenant les alignements `prepareproteinmatrix.pl *.aln`. Le script exécute alors Gblocks sur chacun des fichiers d'alignement et il concatène tous les alignements filtrés pour produire une matrice de séquences utilisable par protml.

Lorsqu'on utilise les 54 fichiers produits précédemment, on obtient une matrice de séquences comportant 10 831 colonnes, donc 10 831 positions conservées. On obtient aussi un fichier nommé Goptns.txt (Annexe D.3) qui contient le nombre de fichiers traités et le nombre de positions conservées par Gblocks pour chacun des alignements.

L'étape suivante est la recherche de topologies. Un des rares programmes qui permet de faire une recherche exhaustive de topologies par "maximum likelihood" avec des séquences protéiques est protml[30]. Cependant, protml ne contient qu'une sélection limitée de matrice de substitution. Ces matrices de substitution sont basées sur des alignements de séquences protéiques conservées et représentent la fréquence de substitution de chaque acide aminé pour un autre. Deux matrices conçues spécifiquement pour les protéines d'organites sont mtREV24[30] et cpREV45[31]. La matrice mtREV24 est incluse dans protml, mais pas cpREV45. Une version modifiée de protml a donc été créée par le remplacement dans le code source de la matrice mtREV24 par la matrice cpREV45. Cette version est nommée protmlcp. Il est nécessaire de comparer les valeurs de "likelihood" obtenus avec différents modèles pour sélectionner celui qui correspond le mieux aux données utilisées.

Pour nos données, il s'agit de cpREV45. On exécutera donc la commande :

```
protmlcp -mfe alignment.ptn > alignment.tpl
```

où alignment.ptn est le fichier produit par prepareproteinmatrix.pl. Une fois l'exécution terminée, alignment.tpl (Annexe D.4) contiendra la liste des 105 meilleures topologies trouvées par protml. Chacune de ces topologies doit ensuite être évaluée de façon exacte en exécutant `protmlcp -mf alignment.ptn alignment.tpl > alignment.ml`. Cela calculera la valeur de "likelihood" et la valeur de "RELL bootstrap" de chaque topologie (Annexe D.5). Cette valeur de "RELL bootstrap"[32] permet d'obtenir une indication de la confiance en une topologie basée sur la constance des données. Plus la valeur de "likelihood" est élevée, meilleure est la correspondance de l'arbre aux données.

Pour les analyses subséquentes, seules les topologies ayant une valeur "RELL bootstrap" non nulle sont conservées. Pour les sélectionner, choosetpl.pl est utilisé. La commande à exécuter est : `choosetpl.pl alignment.tpl alignment.ml > topo.tre`. Le fichier topo.tre (Annexe D.6) contiendra uniquement les topologies ayant une valeur bootstrap non nulle sans que l'ordre par rapport aux fichiers de sortie de protml ne soit changé. Ces topologies peuvent alors être évaluées avec des modèles plus complexes grâce au programme codeml de PAML.

Lorsqu'exécutées avec la matrice de séquences préalablement obtenue, trois topologies sont sélectionnées. Dans les trois topologies, seule la position de *Mesostigma viride* varie. La topologie 1 (Figure 3.5) place *Mesostigma viride* comme provenant d'une lignée basale apparue avant la séparation des chlorophyes et des streptophytes. La topologie 2 (Figure 3.6) place cette algue verte à la base des chlorophytes alors que la topologie 3 (Figure 3.7) la place à la base des streptophytes.

D'après les valeurs de "likelihood", la meilleure des trois topologies serait la topologie 1. Ces résultats sont conformes avec ceux qui avaient été obtenus lors de la publication de l'étude. Cependant, les valeurs de "likelihood" (Annexe D.5) diffèrent légèrement. Cela est principalement dû au gène ycf3 qui avait été oublié lors de la préparation de la matrice de séquences.

3.3.3 Analyse de séquences codantes

La procédure pour l'analyse de séquences codantes est très similaire à celle pour l'analyse de séquences protéiques. Puisque les alignements de séquences codantes sont basés sur ceux de protéines, il faut d'abord avoir aligné les séquences protéiques. Cela implique d'avoir exécuté filtertfa.pl et align.pl comme précédemment. En fait, si les deux types d'analyse sont effectués, les mêmes fichiers peuvent être réutilisés.

Dans le cas d'une analyse avec des séquences codantes, deux étapes supplémentaires sont nécessaires. D'abord, les séquences codantes stockées dans les fichiers Selex doivent être filtrées avec filterselex.pl avec la même liste d'organismes que précédemment. Il faudra donc exécuter la commande :

```
filterselex.pl Taxa /Users/Shared/db/cDNA_alignments/cp/*.selex
```

Les fichiers produits sont au nombre de 54 comme précédemment et ils contiennent les séquences codantes non alignées des protéines. Ces séquences doivent être alignées en se basant sur l'alignement de protéines préalablement effectué. Pour cela, codonalign.pl est appelé par `codonalign.pl prot_alignment *.tfa`. Le premier argument est le répertoire contenant les fichiers de protéines alignées et le deuxième est la liste des fichiers de séquences codantes.

Figure 3.5 La topologie 1

Topologie 1 déterminée par protml. *Mesostigma viride* est positionné comme anscestral à la divergence entre les streptophytes et les chlorophytes.

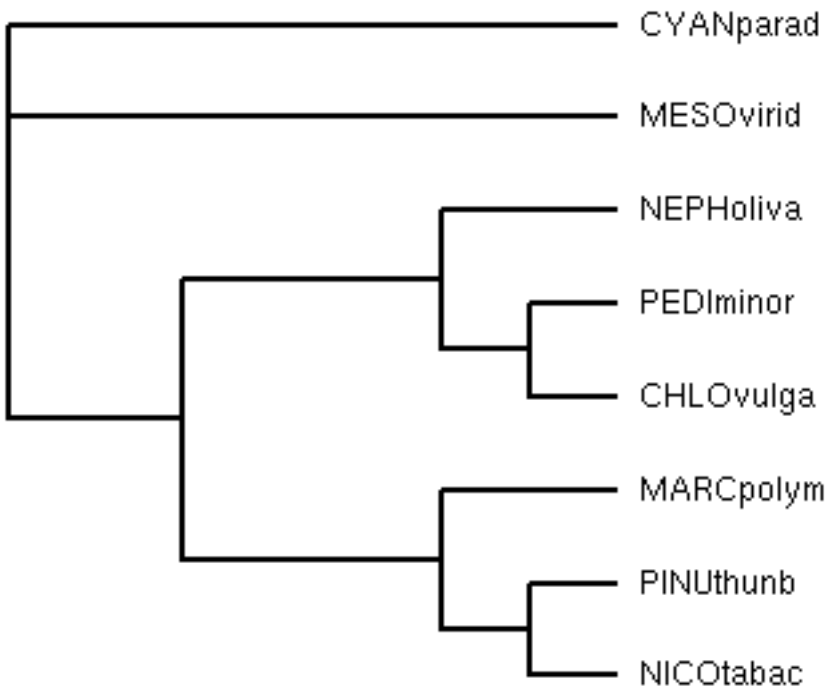

CYANparad
MESOvirid
NEPHoliva
PEDIminor
CHLOvulga
MARCpolym
PINUthunb
NICOtabac

Figure 3.6 La topologie 2

Topologie 2 déterminée par protml. *Mesostigma viride* est positionné à la base des chlorophytes.

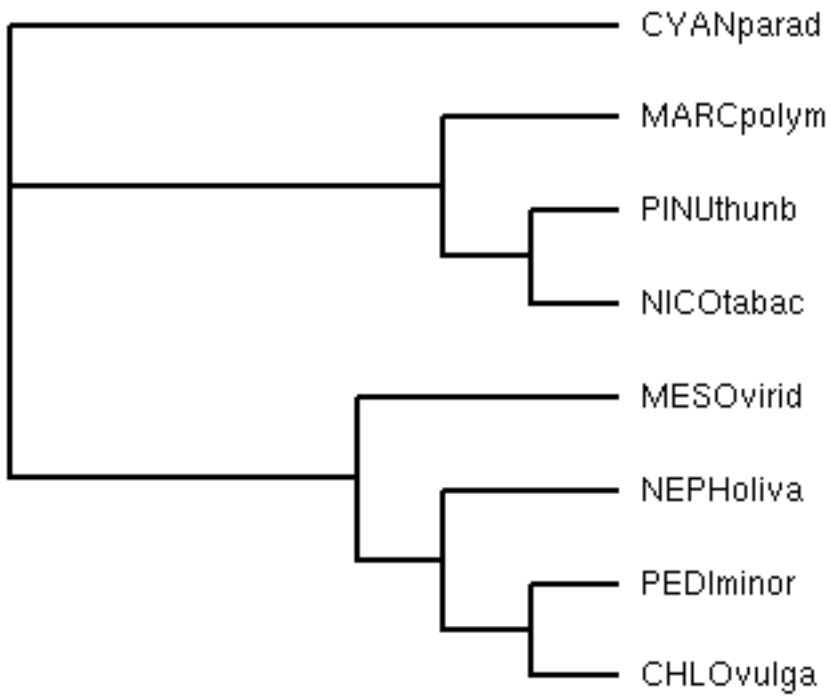
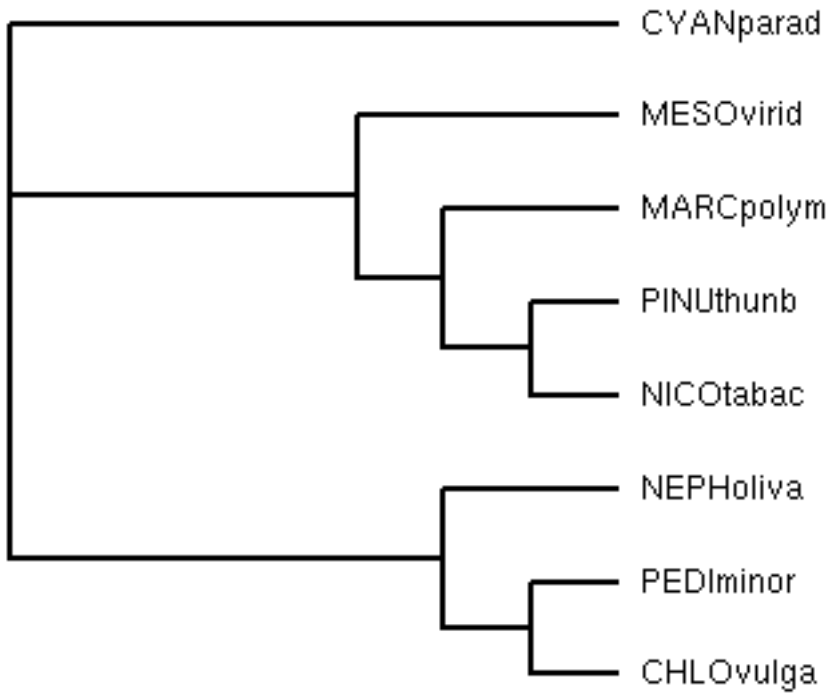

CYANparad
MARCpolym
PINUthunb
NICOtabac
MESOvirid
NEPHoliva
PEDIminor
CHLOvulga

Figure 3.7 La topologie 3

Topologie 3 déterminée par protml. *Mesostigma viride* est positionné à la base des streptophytes.


CYANparad
MESOvirid
MARCpolym
PINUthunb
NICOtabac
NEPHoliva
PEDIminor
CHLOvulga

Comme avec les séquences protéiques, Gblocks doit être utilisé pour filtrer les régions non conservées. La commande `preparednamatrix.pl *.dna_aln` peut être utilisé pour exécuter Gblocks sur tous les alignements et concaténer les séquences conservées en une matrice de séquences codantes au format nexus utilisable par PAUP.

Avec les mêmes organismes que précédemment, la matrice obtenue comporte 34 965 colonnes. Cette matrice ne correspond pas exactement à la matrice protéique. Malgré que Gblocks soit appelé par preparednamatrix.pl avec une option qui délimite les blocs seulement à des jonctions de codons, Gblocks tient uniquement compte de l'identité au niveau des nucléotides. Ainsi, dans certains cas, les acides aminés peuvent être homologues alors que les nucléotides correspondants ne le sont pas. Donc, les limites des blocs peuvent varier.

La matrice peut ensuite être chargée dans PAUP pour effectuer différentes analyses. Pour les analyses utilisant le "maximum likelihood", il existe modeltest qui est un script PAUP testant successivement plusieurs modèles avec différentes complexités. Les résultats de PAUP peuvent ensuite être utilisés pour déterminer le modèle qui convient le mieux aux données. Pour l'ensemble de données discuté précédemment, le modèle le plus complexe est sélectionné, il s'agit du modèle GTR (general time reversible) avec invariants et distribution gamma. Contrairement aux modèles de substitution des acides aminés, les modèles d'évolution de nucléotides sont définis de façon empirique. Le programme PAUP peut faire l'optimisation de tous les paramètres, incluant les probabilités de substitution de chaque nucléotide pour un autre. Ce type d'optimisation est long et serait impensable pour les acides aminés dû au plus grand nombre d'états possibles.

Il est également possible de convertir la matrice de séquence au format PHYLIP à l'aide de readseq pour l'utiliser avec les programmes de PAML si des analyses plus spécifiques sont désirées. Le script preparednamatrix.pl produit lui aussi un fichier contenant les longueurs des séquences conservées par Gblocks qui peuvent être utilisées dans PAML avec certains modèles.

Chapitre 4

Discussion

Ce projet a permis le développement d'une base de données complète des gènes de tous les génomes mitochondriaux et chloroplastiques d'intérêt pour le laboratoire. Cette base de données est facilement consultable et modifiable dans son format actuel. Elle est utilisée de façon courante par les membres du laboratoire. Elle peut être utilisée à la fois pour des recherches d'homologies et pour des analyses phylogénétiques. Sa structure basée sur des répertoires permet d'y ajouter facilement de nouvelles données sans nuire à son fonctionnement.

Des outils pour faciliter la mise à jour de la base de données sont disponibles et fonctionnels. gbkextract.pl facilite l'ajout de nouveaux organismes. updatealign.pl et addexons.pl permettent de formater les alignements de séquences codantes pour leur ajouter les positions des introns. formatdb.sh et tfa2long.sh simplifie le formatage de la base de données pour les recherches d'homologies par BLAST.

filtertfa.pl et filterselex.pl permettent d'extraire les séquences désirés de la base de données. Ils produisent des fichiers au format Fasta. Ce format est un format de base qui est utilisable par la plupart des outils bioinformatiques. Si cela est nécessaire, il peut être converti en d'autres formats grâce à l'utilitaire readseq.

L'identification des gènes dans un génome nouvellement séquencé est maintenant beaucoup facilitée par la recherche automatique des homologies de séquence et l'identification des cadres de lecture ouverts. findorf.pl permet de déterminer les régions d'un génome nouvellement séquencé qui ont des homologies avec les gènes présents dans la base de données. Il permet aussi d'identifier les séquences codant pour des protéines en cherchant les protéines les plus homologues avec chacun des cadres de lecture ouverts

trouvés par l'utilitaire getorf d'EMBOSS. prepsequin.pl permet d'automatiser en partie le processus d'annotation d'un nouveau génome.

Des outils ont été créés qui permettent d'utiliser la base de données pour effectuer la préparation des données en vue d'une analyse phylogénétique. Ces outils s'utilisent en suivant une procédure bien définie qui ne demande pas de décisions subjectives. filtertfa.pl et filterselex.pl servent à extraire les séquences nécessaires, alors que align.pl automatise l'alignement de ces séquences. preparednamatrix.pl et prepareproteinmatrix.pl automatisent l'exécution de Gblocks pour la filtration et se chargent des différentes conversions de format nécessaire à la préparation des données pour effectuer des analyses phylogénétiques. choosetpl.pl permet de sélectionner rapidement les topologies pour des analyses plus poussées.

Les résultats des analyses phylogénétiques utilisant les données préparées par ces outils sont conformes aux résultats obtenus auparavant. Les mêmes topologies sont trouvées avec des valeurs de "likelihood" qui déterminent la même meilleure topologie que les résultats publiés. De plus, il est maintenant possible d'effectuer facilement des analyses basées sur les séquences codantes, alors que cela était difficilement envisageable avant.

Tous les scripts Perl ont été documentés en utilisant le format POD (Plain Old Documentation). Une page de manuel peut facilement être générée en utilisant la commande `perldoc fichier.pl`. Les pages de manuel ainsi générées sont présentées à l'annexe C. De plus, le présent mémoire constitue un guide d'utilisation qui présente les procédures d'utilisation typique de la très grande majorité des scripts produits.

En somme, les objectifs de ce projet de maîtrise ont été réalisés avec succès et l'ensemble des outils bioinformatiques qui ont été conçus permet aux utilisateurs de réduire considérablement le temps nécessaire pour procéder aux analyses de leurs données.

Il est certain que différentes améliorations pourraient être apportées au niveau de la lisibilité et de l'efficacité du code. Ce type de modification est cependant très subjectif et particulier au style du programmeur et au niveau de connaissance du langage. Du point de vue de l'efficacité, la durée d'exécution des scripts pour les données utilisées par le laboratoire est acceptable. Avec l'ajout de beaucoup plus de données, il pourrait être nécessaire de réviser le code pour avoir une efficacité plus optimale. La principale amélioration qui pourrait être ajoutée actuellement est au niveau de la documentation du code, sous forme de commentaires, qui est souvent inconstante entre les scripts et difficile de compréhension pour un nouveau programmeur. Cependant, findorf.pl et gbkextract.pl, les deux scripts les plus complexes, sont documentés en détail.

Un point qui a été un peu négligé est la gestion des erreurs. Dans certains cas, les modules de Bioperl affichent des messages d'erreur. Certains de ces messages sont sans conséquences, mais d'autres sont essentiels pour identifier et résoudre le problème qui cause cette erreur. Dans tous les cas, ces erreurs peuvent porter à confusion et sont difficiles à comprendre. Certains scripts incluent des mesures pour prévenir ces erreurs. Cependant, il serait utile d'ajouter une meilleure gestion des erreurs et d'afficher des messages plus facilement compréhensibles. Cette tâche n'est pas négligeable puisqu'un grand nombre d'erreurs différentes peuvent survenir. Dans le même ordre d'idée, il serait utile d'avoir un outil qui s'assure du synchronisme entre les différents répertoires de la base de données et les fichiers Genbank des génomes, mais cela est complexe dû au manque de standardisation dans les annotations.

Les scripts pourraient être améliorés en les rendant utilisables sur la plateforme Windows et en facilitant leur installation sur les différentes plateformes déjà supportées. Cependant, la direction qui semble la plus avantageuse à prendre est de rendre ces scripts accessibles par l'intermédiaire d'une interface Web. Cela permettrait d'avoir une base de données centralisée sur un seul serveur plutôt qu'une copie de la base de données qui doit être maintenue à jour sur chaque station de travail.

Il serait aussi souhaitable de créer des outils pour charger la base de données qui a été créée dans une base de données relationnelle. Cela permettrait de simplifier le développement de certains outils et rendrait plus facile le développement d'une plateforme Web pour accéder à la base de données sur un serveur central. Cependant, cela rendrait la base de données plus difficilement accessible pour quelqu'un n'ayant que très peu de formation en informatique. Il peut donc être préférable de conserver les deux formats de base de données pour un certain temps. La plupart des scripts produits pourraient facilement être réécrits pour utiliser une base de données relationnelle.

Bibliographie

[1] The Genome Sequencing Center and the Sanger Centre. Genome Sequence of the Nematode *Caenorhabditis elegans*. A Platform for Investigating Biology. *Science*, 282 :2012–2018, 1998.

[2] M.D. Adams, S.E. Celniker, R.A. Holt, C.A. Evans, J.D. Gocayne, P.G. Amanatides, S.E. Scherer, P.W. Li, R.A. Hoskins, R.F. Galle, and coll. The genome sequence of *Drosophila melanogaster*. *Science*, 287 :2185–2195, 2000.

[3] Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420 :520–562, 2002.

[4] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409 :860–921, 2001.

[5] Sequencher. `http://www.genecodes.com/sequencher/index.html`

[6] M.W. Gray. The endosymbiont hypothesis revisited. *Int. Rev. Cytol.*, 141 :233–357, 1992.

[7] S.G. Andersson, A. Zomorodipour, J.O. Andersson, T. Sicheritz-Pontén, U.C. Alsmark, R.M. Podowski, A.K. Näslund, A.S. Eriksson, H.H. Winkler, and C.G. Kurland. The genome sequence of *Rickettsia prowazekii* and the origin of mitochondria. *Nature*, 396 :133–140, 1998.

[8] C. Lemieux, C. Otis, and M. Turmel. Ancestral chloroplast genome in *Mesostigma viride* reveals an early branch of green plant evolution. *Nature*, 403 :649–652, 2000.

[9] M. Turmel, C. Otis, and C. Lemieux. The Complete Mitochondrial DNA Sequence of *Mesostigma viride* Identifies This Green Alga as the Earliest Green Plant Divergence and Predicts a Highly Compact Mitochondrial Genome in the Ancestor of All Green Plants. *Mol. Biol. Evol.*, 19 :24–38, 2002.

[10] GCG Wisconsin Package. `http://www.gcg.com`

[11] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic Local Alignment Search Tool. *J. Mol. Biol.*, 215 :403–410, 1990.

[12] Interface web de BLAST. `http://www.ncbi.nlm.nih.gov/BLAST`

[13] Mac OS X. `http://www.apple.com/macosx/`

[14] Red Hat. `http://www.redhat.com`

[15] Microsoft Windows. `http://www.microsoft.com/windows/default.mspx`

[16] Perl. `http://www.perl.org`

[17] The Bioperl Project. `http://www.bioperl.org`

[18] E.L.L. Sonnhammer and R. Durbin. MSPcrunch : a BLAST enhancement tool for large-scale sequence similarity analysis. 1997.

[19] J. Thompson, D. Higgings, and T. Gibson. ClustalW : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, 22 :4673–4690, 1994.

[20] C. Notredame, D. Higgins, and J. Heringa. T-Coffee : A novel method for multiple sequence alignments. *J. Mol. Biol.*, 302 :205–217, 2000.

[21] Readseq. `http://iubio.bio.indiana.edu/soft/molbio/readseq`

[22] The EMBOSS project. `http://www.emboss.org`

[23] T.M. Lowe and S.R. Eddy. tRNAscan-SE : a program for improved detection of transfer RNA genes in genomic sequence. *Nucl. Acids Res.*, 25 :955–964, 1997.

[24] M. Melkonian. Flagellar apparatus ultrastructure in *Mesostigma viride* (Prasinophycae). *Plant Syst. Evol.*, 164 :93–122, 1989.

[25] M. Melkonian, B. Marin, and B. Surek. *Biodiversity and Evolution.* The National Science Museum Foundation, Tokyo, 1995.

[26] D. Bhattacharya, K. Weber, S. S. An, and W. Berning-Koch. Actin phylogeny identifies *Mesostigma viride* as a flagellate ancestor of the land plants. *J. Mol. Evol.*, 47 :544–550, 1998.

[27] K. G. Karol, R. M. McCourt, M. T. Cimino, and C. F. Delwiche. The closest living relatives of land plants. *Science*, 294 :2351–2353, 2001.

[28] T.A. Helmchen, D. Bhattacharya, and M. Melkonian. Analyses of ribosomal rna sequences from glaucocystophyte cyanelles provide new insights into the evolutionary relationships of plastids. *J. Mol. Evol.*, 41 :203–210, 1995.

[29] J. Castresana. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Mol. Biol. Evol.*, 1 :540–552, 2000.

[30] J. Adachi and M. Hasegawa. MOLPHY version 2.3 : programs for molecular phylogenetics based on maximum likelihood. *Comput. Sci. Monogr.*, 28 :1–150, 1996.

[31] J. Adachi, P.J. Waddell, W. Martin, and M. Hasegawa. Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast dna. *J. Mol. Evol.*, 50 :348–358, 2000.

[32] M. Hasegawa and H. Kishino. Accuracies of the simple methods for estimating the bootstrap probability of maximum likelihood trees. *Mol. Biol. Evol.*, 11 :142–145, 1994.

Annexe A

Exemples de fichiers

A.1 Genbank

```
gene            complement(18877..19155)
                /gene="rpl23"
CDS             complement(18877..19155)
                /gene="rpl23"
                /codon_start=1
                /product="ribosomal protein L23"
                /translation="MIDIVKYPVLTEKATRLLENNQYTFDVDPKANKITIKALIEDFF
                NVKVLSVNTHRPPRKKRRIGRSEGYRPNYKRVIVTLKTGDSIKLLPET"
tRNA            complement(19380..19452)
                /gene="trnA(ggc)"
                /note="codons recognized: GCY"
                /product="tRNA-Ala"
                /anticodon=(pos:19417..19419,aa:Ala)
gene            complement(19380..19452)
                /gene="trnA(ggc)"
gene            19699..20220
                /gene="ycf3"
CDS             19699..20220
                /gene="ycf3"
                /function="putative role in the assembly of photosystem I"
                /codon_start=1
                /product="hypothetical chloroplast RF3"
                /translation="MPRSQKNDNFIDKTFTVVADIILKVLPTTVREKAAFSYYRDGMS
                AQAEGEYAEALQNYYEAMRLEIDPYDRSYILYNIGLIHTSNGEHGKALEYYYQAIERN
                PSLPQALNNIAVIYHYRGEQAIEEGNIATSEILFNQAASYWKQAIRLAPNSYIEAQNW
                LKITGRIEDNINL"
tRNA            21193..21279
```

```
                     /gene="trnI(cau)"
                     /note="codon recognized: AUA; C in the first position of
                     the anticodon assumed to be post-transcriptionally
                     modified to lysidine, which pairs with A rather than G"
                     /product="tRNA-Ile"
                     /anticodon=(pos:21227..21229,aa:Ile)
     gene            21193..21279
                     /gene="trnI(cau)"
```

A.2 Fasta

```
>SYNEs6803
MKGSLYSSKIAEPYAQALIGLAQQQNLTEVFGDNLRSLLTLLQDSPDLSAVLSSPVVKDE
DKKSVLRSVLGDGGNGYLLNFLMLLVDKRRIVFLEAICEQYLALLRQFTNTVLAEVTSAL
KLTDAQKDQVKERVKQLTGAQAVELETKVDGDILGGIVIKVGSQVFDSSLRGQLRRVGLS
LGTAL
>PORPpurpu
MSSNNLVAKIAQPYASALLDLANEKKAIEQISQDMKLIKDILLQSGKLKYFLANPLKTIE
AKKQVIAATFGDQISENTLSFLMVLVDRKRISMLDVIAGKYLELAYAMESLTIANISTSI
ALNSDQENLLIDKIKAMTSAKEVKLVISVDPELIGGFTIQIGSKVIDTSIRGQLKQMASH
LDVAAM
>CYANcalda
MILLLTNSKIIYPYSEALFSIAKDQEKFEVIKNDMELFVTFTKNLNGFKKFLETPLINKN
KKIKVVKDVFSKILNSTTLNFISILINKNRIMFVSNISEKYNQLVLKDKSVKLVKIACAR
QLSEKQAQALSEVLKHKFKCLSVKLIFNIEPELIAGFKIFIESQVIDVSLQGELKEFEWY
LTK
>GUILtheta
MIAMNNKLAQPYAMAFLEFSLDAKQTLDTTIADLTQIKTILHDSVDLSKTLSNPLLSIKA
KKEVIKAIFEPNISKNTLKFLLVLCDRGRSANLSSIIDNTIELAYKKASIEIAYVTTATA
FSSNQQEALVEKLKSMTSTEQIKLNITVDKTLIGGFKVQIGSKVIDTSIQGQLRQLASHL
GSSAI
>ISOCtahit
MLVAKIAVPYAEALLELAKANKSLKETTNDMNIVSQFLANSSDLKKFLGNPLITRDAKKG
VLKDVLGEQIGEKTLTFLMLLVDRNRIAYLDGIAYKFVELSYNEDSIEIAKVTSSVRLSS
EQQKGIAEKLKTITGAKQIKLALKVDPSLIGGFTVEIGSKLIDTSIRGQLKQISSLLGAA
RA
>PAVLluthe
VIDKTVASRVALPYAEALLDFAKKVNGTDESTNDVNVIHQFVSNSKDLERFLANPVITKT
SKKNVLVDLFGDYISVSTIKFLMLLVDKNRIQYLKYVVMAYIALAAKESSSEIAEVTSVT
HLSSDQQEKLKVRLKALRGISNIELVLKVNPRLLGGFVVQIGSSVLDFSLLGALQKMENH
FQSATL
```

A.3 Longueur d'exons

```
ARABthali 145 410
EUGLgraci 5 134 162 251
MARCpolym 145 410
NICOtabac 145 410
PINUthunb 145 410
PSILnudum 145 410
STAUpunct 145 410
ZEAmays 145 407
ANTHformo 145 410
CHARvulga 135 426
ADIAcapil 125 427
BRYOplumo 67 440
CALYferti 145 410
DERBmarin 88 449
AMBOtrich 145 410
PHYSpaten 145 410
```

A.4 Liste d'organismes

```
SYNEs6803
CYANcalda
CYANmerol
PORPpurpu
VAUCbursa
ODONsinen
GUILtheta
ISOCtahit
PAVLluthe
CYANparad
EUGLgraci
NEPHoliva
PYCNpraso
MONOoke-1
SCHEdubia
NEOCpseud
CHLOvulga
BRYOplumo
```

```
DERBmarin
OLTMvirid
PSEUakine
PEDIminor
SCENobliq
CHLAreinh
MESOvirid
CHLOatmop
ZYGNcircu
STAUpunct
CHAEglobo
CHARvulga
ANTHformo
MARCpolym
PHYSpaten
PSILnudum
ADIAcapil
PINUthunb
AMBOtrich
CALYferti
NICOtabac
ARABthali
ZEAmays
```

A.5 Clustalw

```
CLUSTAL W(1.81) multiple sequence alignment


SYNEs6803             MKGSLYSSKIAEPYAQALIGLAQQ-------QNLTEVFGDNLRSLLTLLQDSPDLSAVLS
CYANcalda             MILLLTNSKIIYPYSEALFSIAKD-------QEKFEVIKNDMELFVTFTKNLNGFKKFLE
CYANmerol             -----MKQKIVEPYAQALFRLKDD-------IDL--------TPLWEMARDS-KFMQLLM
PORPpurpu             MSSNNLVAKIAQPYASALLDLANE-------KKAIEQISQDMKLIKDILLQSGKLKYFLA
VAUCbursa             MTNKLSSIKIADPYAEAFFQLGLGLYITNDNPDIFYKLIFDIQDFLELLSETPELNSFLK
ODONsinen             MSINPLASKIAAPYARALFDFSVD-------QNLMHQITADFQNLEVFLNKTPDLTEYLS
GUILtheta             --MIAMNNKLAQPYAMAFLEFSLD------AKQTLDTTIADLTQIKTILHDSVDLSKTLS
ISOCtahit             ----MLVAKIAVPYAEALLELAKA-------NKSLKETTNDMNIVSQFLANSSDLKKFLG
PAVLluthe             VIDKTVASRVALPYAEALLDFAKK-------VNGTDESTNDVNVIHQFVSNSKDLERFLA
CYANparad             MKQSAVVSKITQPYAEALLEMAQK-------YDIVETVNNDITLILNCLQNSTKLQQFLA
                             ::  **: *:: :           .           .     .   :   *
```

```
SYNEs6803           SPVVKDEDKKSVLRSVLGDGGNGYLLNFLMLLVDKRRIVFLEAICEQYLALLRQFTNTVL
CYANcalda           TPLINKNKKIKVVKDVFSKILNSTTLNFISILINKNRIMFVSNISEKYNQLVLKDKSVKL
CYANmerol           NPSIPKEKKWQLFQ-----PFDKLVQSWLEVIWKKKRMNLLAEICASYLELRKKKEGIVT
PORPpurpu           NPLKTIEAKKQVIAATFGDQISENTLSFLMVLVDRKRISMLDVIAGKYLELAYAMESLTI
VAUCbursa           NPLNSEILKKNILNKVLENKVSAHTINFLNLLIDKKRINSIESIGKRFLDKAYEFVCIKF
ODONsinen           NPLISAKSKEEVLNKTLKSQINKETFKFLIVLVNRSRINLLEPIIASYLNLVYNAASVKM
GUILtheta           NPLLSIKAKKEVIKAIFEPNISKNTLKFLLVLCDRGRSANLSSIIDNTIELAYKKASIEI
ISOCtahit           NPLITRDAKKGVLKDVLGEQIGEKTLTFLMLLVDRNRIAYLDGIAYKFVELSYNEDSIEI
PAVLluthe           NPVITKTSKKNVLVDLFGDYISVSTIKFLMLLVDKNRIQYLKYVVMAYIALAAKESSSEI
CYANparad           NPLVKKSSKKNFFEKTLAKEIHPYTFKFLLLVIDRGRISCLEIIAQKYQSLILKLTKTEL
                    .*      *  ..             .:: :: .: *   :  :

SYNEs6803           AEVTSALKLTDAQKDQVKERVKQL---------TGAQAVELETKVDGDILGGIVIKVGSQ
CYANcalda           VKIACARQLSEKQAQALSEVLKHK---------FKCLSVKLIFNIEPELIAGFKIFIESQ
CYANmerol           VFVTSATPLTDTQTQQLEVQLTRM---------CQAKHLQCEYQVDAQLLAGLKIQMNGQ
PORPpurpu           ANISTSIALNSDQENLLIDKIKAM---------TSAKEVKLVISVDPELIGGFTIQIGSK
VAUCbursa           VEVWSTIELTQKQQETIIHKINLILGPVFTEPYVQSSNIQLTLIVDKKILGGLIIKMGSK
ODONsinen           IEVSTAYAFTNLQKNTLIKKLKEL---------TNAREIRLVITVDSSLIGGFLIKTNSK
GUILtheta           AYVTTATAFSSNQQEALVEKLKSM---------TSTEQIKLNITVDKTLIGGFKVQIGSK
ISOCtahit           AKVTSSVRLSSEQQKGIAEKLKTI---------TGAKQIKLALKVDPSLIGGFTVEIGSK
PAVLluthe           AEVTSVTHLSSDQQEKLKVRLKAL---------RGISNIELVLKVNPRLLGGFVVQIGSS
CYANparad           AEVVTAVPLSSEQEAALNNIIKEL---------TNANEVKLVFKIDQNLIGGFIINIGSK
                      :     :.. *   :   :.                :.    ::  ::.*: :   ..

SYNEs6803           VFDSSLRGQLRRVGLSLGTAL--
CYANcalda           VIDVSLQGELKEFEWYLTK----
CYANmerol           LIDTSWQTQLKQLMKSLW-----
PORPpurpu           VIDTSIRGQLKQMASHLDVAAM-
VAUCbursa           VIDLSLRSELQRLGKELDIVL--
ODONsinen           VLDFTIKNQLQKLAKHLDSVLEI
GUILtheta           VIDTSIQGQLRQLASHLGSSAI-
ISOCtahit           LIDTSIRGQLKQISSLLGAARA-
PAVLluthe           VLDFSLLGALQKMENHFQSATL-
CYANparad           VVDASLLGQLLRIGNYLGLETV-
                    :.* :    * ..   :
```

A.6 Selex

```
SYNEs6803/268-292     --- --- --- --- --- GAA ATT AAG GAG --- CAG GAA GTC TTC
CYANcalda/277-301     --- --- --- --- --- GAC GTA AAA GAA --- CAA GAA ATC TTA
CYANmerol/244-268     --- --- --- --- --- GAC GTT AGA CAA --- CAA ATG ATC TAT
PORPpurpu/370-394     --- --- --- --- --- AAG TAT AAA AAA --- AGA TTG GTC TTT
```

```
VAUCbursa/283-307     --- --- --- --- --- ATT TTA AAA AAA --- CAA AAG ATT TTA
ODONsinen/262-289     --- --- --- --- --- AGC GTT CGT TAC TTT GGG CAA TTT ACT
GUILtheta/274-298     --- --- --- --- --- GTA ATA AAG GAA --- CAG GAA GTT TTT
ISOCtahit/310-334     --- --- --- --- --- GAA GCC CAA GAG --- AAA CAA GTT TTT
PAVLluthe/280-304     --- --- --- --- --- AAA GAG TAC TTT --- GAA GAT GTT TTT
CYANparad/274-298     --- --- --- --- --- ATT GTT AAA CAA --- GAT CGT ATT TTC
EUGLgraci/259-283     --- --- --- --- --- CTT GTT AGA AAT --- AAA TAT ATA TTA
#                                             ><
NEPHoliva/292-316     --- --- --- --- --- GAG ACT CAA ATT --- CAA TCG GTA TGG
MONOoke-1/1066-1108   GAA AGA TCA ACT CGT CAA AAA AAA CAA ATT TTT TGG TTT TAT
SCHEdubia/352-388     --- AAA CTT GAA CAA GTT TTT GAA CGA --- AAA TGG ATT TTT
NEOCpseud/262-286     --- --- --- --- --- GAA ATA AAA ATT --- CAA TGG GTT TTA
CHLOvulga/367-391     --- --- --- --- --- ACC TTT CGA ATT --- GAA TGG CTT TTC
BRYOplumo/271-298     --- --- --- --- AAT AAA ATA CGA TTC --- CAA TGG GTT TTT
DERBmarin/277-301     --- --- --- --- --- CAA ATT CAA TTT --- CGA TGG GTT TGT
OLTMvirid/277-301     --- --- --- --- --- ACT TTA AAA TTA --- CAA TGG GTT GTT
PSEUakine/436-460     --- --- --- --- --- GAG GTA CAC CTG --- CAA TGG GTA TTA
PEDIminor/274-298     --- --- --- --- --- GAA ATT AAA TTT --- CAG TGG GTT TTA
SCENobliq/265-289     --- --- --- --- --- CGT ATT TTT TTA --- AAA TGG ATA TTG
CHLAreinh/613-637     --- --- --- --- --- ATA ATC AAA TTA --- AAA TGG GTT TAT
MESOvirid/298-322     --- --- --- --- --- GAA ATT CAA GAA --- CAA GAT GTA TTT
CHLOatmop/292-316     --- --- --- --- --- AAA GTT CAA GAA --- CAA GAT GTT TTT
ZYGNcircu/280-304     --- --- --- --- --- ATC ATT CAA AAA --- CAA ACA GTC TTT
```

Annexe B

Les scripts

B.1 addexons.pl

```
#!/usr/bin/perl -w

=head1 NAME

addexons.pl - produce Selex files

=head1 SYNOPSIS

addexons.pl file.fasta path_to_loc

=head1 DESCRIPTION

addexons.pl takes as first argument a Fasta file containing the aligned
sequences coding for a given protein and as second argument a directory
containing a file of the same name with the .loc extension
which contains the lengths of the exons for each organism
that features introns in this protein.

It outputs in the current directory a file containing
the aligned sequences in Selex format with the intron positions marked by ><.

=head1 AUTHOR
```

```
Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut
use strict;
use Bio::AlignIO;

my $in = Bio::AlignIO->new( -file => $ARGV[0], -format => "fasta" );
my $out = Bio::AlignIO->newFh( -format => "fasta" );
my $aln = $in->next_aln;
my $maxn       = 20;
my $LINELENGTH = 100;
my @orgtab;
my %lentab;
my @name = split ( /\./, $ARGV[0] );
open( OUTPUT, ">$name[0].selex" );

if ( open( LOC, "$ARGV[1]/$name[0].loc" ) ) {

    while (<LOC>) {
        my @tmp = split (/ /);
        my @new = @tmp[ 1 .. @tmp ];
        $lentab{ $tmp[0] } = \@new;
    }
}

foreach my $seq ( $aln->each_seq ) {
    my ($substring);
    my @outarray;
    my $seqchars = $seq->seq;
    my $count    = 0;
    my $count2   = 1;
    my @tabpos;
    my $seqstr = $seq->seq;
    $seqstr =~ tr/[atcgn]/[ATCGN]/;
    $seqstr =~ s/([ATCGN\.\-]{3})/$1 /g;

    for ( my $n = 0 ; $n < length($seqstr) ; $n++ ) {

        if ( substr( $seqstr, $n, 1 ) =~ /[ATGCN]/ ) {
            $tabpos[$n] = $count2;
            $count2++;
        }
        else {
```

```
            $tabpos[$n] = 0;
        }
    }

    while ( $count < $seq->length ) {
        SWITCH: {
            if ( length($seqchars) >= ( $count + $LINELENGTH * 3 / 4 ) ) {
                $substring = substr( $seqchars, $count, $LINELENGTH * 3 / 4 );
                last SWITCH;
            }
            elsif ( length($seqchars) >= $count ) {
                $substring = substr( $seqchars, $count );
                last SWITCH;
            }
            $substring = "";
        }
        $substring =~ tr/[atcgn]/[ATCGN]/;
        $substring =~ s/([ATCGN\.\-]{3})/$1 /g;

        $count += $LINELENGTH * 3 / 4;

        $aln->set_displayname_flat;
        my $namestr = $aln->displayname( $seq->get_nse );
        $namestr .= '/';
        my $p = 0;
        if ( $substring =~ /[ATCGN]/g ) {
            $p = ( pos($substring) + ( $count * 4 / 3 - $LINELENGTH ) - 1 );

        }
        $namestr .= $tabpos[$p];

        while ( $substring =~ /[ATCGN]/g ) {
            $p = ( pos($substring) + $count * 4 / 3 - $LINELENGTH - 1 );
        }
        $namestr .= "\-" . ( $tabpos[$p] );
        my $add = $maxn - length($namestr) + 2;
        $namestr .= " " x $add;
        push ( @outarray, $namestr . $substring . "\n" );
    }
    $count     = 1;
    $substring = "";
    my @on;
    my @off;
```

```
my $ref = $lentab{ $seq->id };

while ( my $len = shift (@$ref) ) {
    push ( @on, $count );
    $count += $len;
    push ( @off, $count - 1 );

}
my $str = ' ' x ( ( $aln->length / 3 ) * 4 );
my $pos2;
for ( my $t = 0 ; $t < @tabpos ; $t++ ) {
    if ( $tabpos[$t] == 1 ) {
        my $pos = int( $t / 3 ) * 4;
    }
}

foreach my $test (@on) {

    my $pos;
    for ( my $t = 0 ; $t < @tabpos ; $t++ ) {
        if ( $tabpos[$t] == $test ) {
            $pos = $t - 1;
        }
    }
    substr( $str, $pos + 1, 1 ) = '<';
    $test = shift (@off);

    for ( my $t = 0 ; $t < @tabpos ; $t++ ) {
        $pos2 = $t - 1 if ( $tabpos[$t] == $test );
    }
    substr( $str, $pos2 + 1, 1 ) = '>';
}

my @outstr;
$count = 0;
my $namestr = '#';
my $add     = $maxn - length($namestr) + 2;
$namestr .= ' ' x $add;

while ( $count < length($str) ) {
    SWITCH: {
        if ( length($str) >= ( $count + $LINELENGTH ) ) {
            $substring = substr( $str, $count, $LINELENGTH );
```

```
                last SWITCH;
            }
            elsif ( length($str) >= $count ) {
                $substring = substr( $str, $count );
                last SWITCH;
            }
            $substring = "";
        }
        $count += $LINELENGTH;
        push ( @outstr, $substring . "\n" );
    }
    my @output;

    while (@outarray) {
        my $s = shift @outarray;
        if ( $outstr[0] =~ /\<|\>/ ) {
            $s .= $namestr . ( shift @outstr );
        }
        else { shift @outstr; }
        push ( @output, $s );
    }
    push ( @orgtab, \@output );
}

while ( $orgtab[0]->[0] ) {
    foreach my $org (@orgtab) {
        print OUTPUT shift @$org;

    }
    print OUTPUT "\n\n\n";

}
```

B.2 align.pl

```
#!/usr/bin/perl -w

use strict;
```

```
my $aprog = 't_coffee';

if ( $ARGV[0] eq '-c') {
$aprog = 'clustalw';
shift;
}

while (my $f = shift) {
system("$aprog $f -outorder=input");
}
```

B.3 choosetpl.pl

```
#!/usr/bin/perl -w

=head1 NAME

choosetpl.pl - select topologies from protml output

=head1 SYNOPSIS

choosetpl.pl file.tpl file.ml

=head1 DESCRIPTION

choosetpl.pl parses the output of protml and selects the trees that
have a non null bootstrap value.

Those trees are printed to the standard output and the number of trees
selected is printed to STDERR.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut
use strict;

@ARGV > 2 || die "Usage: choosetpl.pl file.tpl file.ml\n";
```

```
open( TPL, shift );
open( ML,  shift );

my $line;
my $i;

while (<ML>) { last if (/-{60}/) }

<TPL>;
while (<TPL>) {
    if ( !/^\#/ ) {
        $line = $_;
        last;
    }
}

while (<ML>) {
    chomp;
    my @ml = split (/ +/);
    if ( $ml[8] > 0 ) {
        $i++;
        my ($tpl) = split ( /;/, $line );
        print "$tpl;\n";
    }
    $line = <TPL>;
}
print STDERR "$i\n";
```

B.4 codonalign.pl

```
#!/usr/bin/perl -w

=head1 NAME

codonalign.pl - align a coding sequence using a protein alignment as reference

=head1 SYNOPSIS

codonalign.pl aln_dir *.tfa
```

```
=head1 DESCRIPTION

codonalign.pl aligns each Fasta file using the corresponding
clustalw .aln file in aln_dir as reference.

In the output alignment in Fasta format, each amino acid is
replaced by the corresponding three nucleotides from the coding sequence
and each gap is replaced by three gaps.

The output files have the same name as the input files but with
the .dna_aln extension. They are put in the current directory.

=head1 CAVEATS

The coding sequence file and alignment file must contain exactly
the same sequences in the same order. There is no verification that
the sequences really correspond.

A file having the same name as an output file will be overwritten.

=head1 AUTHOR

Based on align_on_codons.pl from Bioperl and
modified heavily by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

use strict;
use Bio::SeqIO;
use Bio::AlignIO;
use Bio::LocatableSeq;
use Bio::SimpleAlign;
use File::Basename;

@ARGV >= 2 || die "Usage: codonalign.pl aln_dir *.tfa\n";
my $CODONSIZE = 3;
my $alnpath   = shift;

while ( my $f = shift ) {

    my ( $name, $ext ) = split( /\./, basename($f) );
```

```
my $seqio = new Bio::SeqIO(
    '-format' => 'fasta',
    '-file'   => $f
);
my $alignin = Bio::AlignIO->new(
    '-file'   => "$alnpath/$name.aln",
    '-format' => 'clustalw'
);
my $alignout = new Bio::AlignIO(
    '-format' => 'fasta',
    '-file'   => ">$name.dna_aln"
);

my $aln      = $alignin->next_aln;
my $alnlen   = $aln->length;
my $dnaalign = new Bio::SimpleAlign;
my $seqorder = 0;
my @nucseqs;

while ( my $seq = $seqio->next_seq ) {
    push @nucseqs, $seq;
    $aln->get_seq_by_pos( scalar(@nucseqs) )->id eq $seq->id
      || die "Sequence order mismatch "
      . $aln->get_seq_by_pos( scalar(@nucseqs) )->id . " vs "
      . $seq->id . "\n";
}

foreach my $seq ( $aln->each_seq ) {

    my $newseq;
    my $ngap = 0;
    my $end;
    foreach my $pos ( 1 .. $alnlen ) {
        my $loc = $seq->subseq( $pos, $pos );
        my $dna;
        if ( $loc eq '-' ) { $dna = '---'; $ngap++; }
        else {
            my $start = ( ( $pos - 1 - $ngap ) * $CODONSIZE ) + 1;
            $end = $start + $CODONSIZE - 1;
            die "Sequence length doesn't match for "
              . $seq->id . " in "
              . $name . "\n"
              if ( $start > $nucseqs[$seqorder]->length
```

```
                    || $end > $nucseqs[$seqorder]->length );
                $dna = $nucseqs[$seqorder]->subseq( $start, $end );
            }
            $newseq .= $dna;
        }
        $end == $nucseqs[$seqorder]->length
          || die "Sequence length doesn't match for "
          . $seq->id . " in "
          . $name . "\n";
        my $newdna = new Bio::LocatableSeq(
            -display_id => $seq->id,
            -start      => ( ( $seq->start - 1 ) * $CODONSIZE ) + 1,
            -end        => $seq->end * $CODONSIZE,
            -strand     => $seq->strand,
            -seq        => $newseq
        );

        $dnaalign->add_seq($newdna);
        $seqorder++;
    }
    $dnaalign->set_displayname_flat;
    $alignout->write_aln($dnaalign);
}
```

B.5 filterselex.pl

```
#!/usr/bin/perl -w

=head1 NAME

filterselex.pl - keep only the desired sequence from a Selex file

=head1 SYNOPSYS

filterselex.pl ORG_list /path/to/*.selex

=head1 DESCRIPTION

filterselex.pl takes a file containing the list of the desired organisms
```

```
to be kept and a list of Selex files to be filtered.

It outputs a Fasta file for each Selex file in the current directory and
the Selex files are unmodified. The Fasta files are unaligned.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

use Bio::SeqIO;
use strict;

if ( @ARGV < 2 ) {
    my $USAGE = qq
{
filterselex.pl ORG_list /path/to/*.selex

};
    print $USAGE;
    exit;
}

my $list = shift;

while ( my $f = shift ) {
    open( IN, $f );
    my %name;
    while (<IN>) {
        if ( !/^\#/ ) {
            $name{$1} .= $2
              if (/([\w\-]+)\/\d+\-\d+([ATCGN\- ]+)/);
        }
    }
    my @path = split( /\//, $f );
    my ( $fname, $ext ) = split( /\./, $path[ scalar(@path) - 1 ] );
    my $out = Bio::SeqIO->new( '-file'   => ">$fname.tfa",
                               '-format' => 'Fasta'
                             );

    foreach my $seq ( sort keys %name ) {
        $name{$seq} =~ s/[ \-]//g;
```

```
        $out->write_seq( new Bio::Seq( '-seq' => $name{$seq},
                                       '-id'  => $seq
                                     )
                       );
    }
}

system("filtertfa.pl $list *.tfa");

unlink <*.tfa>;
system("mv filtered/* .");
rmdir "filtered";
```

B.6 filtertfa.pl

```
#!/usr/bin/perl -w

=head1 NAME

filtertfa.pl - keep only the desired organism for a Fasta file

=head1 SYNOPSIS

filertfa.pl ORG_list *.tfa

=head1 DESCRIPTION

filtertfa.pl takes a file containing the list of the desired organisms
to be kept and a list of Fasta files to be filtered.

It creates a directory named filtered which contains all the Fasta files
featuring all and only the desired organisms.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut
use strict;
```

```
use Bio::SeqIO;
use Bio::AlignIO;

if ( @ARGV < 2 ) {
system("perldoc $0");
    exit;
}

mkdir( "filtered", 0755 );
my $numorg;
my %ORGlist;

{
    open( LIST, shift (@ARGV) );

    while ( my $line = <LIST> ) {
        chomp($line);
        if ( ($line) && !( $line =~ /^\#/ ) ) {
            my @tmparray = split ( /\t/, $line );
            if ( @tmparray == 1 ) {
                $numorg++;
                $ORGlist{ $tmparray[0] } = $numorg;

            }
        }
    }
    close(LIST);
}

foreach my $file (@ARGV) {
    my $in = Bio::AlignIO->new( '-file' => $file, '-format' => 'fasta' );
    my $aln = $in->next_aln();
    my @orgtab;
    if ( $aln->no_sequences >= $numorg ) {
        $aln->set_displayname_flat;

        foreach my $seq ( $aln->each_seq() ) {
            if ( $ORGlist{ $seq->id() } ) {
                push ( @orgtab, $seq->id() );
            }
        }
        my @sorted = sort sort_sub(@orgtab);
        my $newaln = Bio::SimpleAlign->new();
```

```
        foreach my $org (@sorted) {
            if ($org) {
                if ( $ORGlist{$org} ) {
                    foreach my $seq2 ( $aln->each_seq() ) {
                        if ( $seq2->id() =~ $org ) {
                            my $newseq = $seq2->seq;
                            $newseq =~ tr/\-//d;
                            my $newseq0 = Bio::LocatableSeq->new(
                                                    -id => $seq2->id,
                                                    -seq => $newseq);
                            $newaln->add_seq($newseq0);
                            last;
                        }
                    }
                }
            }

            if ( $newaln->no_sequences >= $numorg ) {
                my @tmp = split ( /\./, $file );
                my @tmp2 = split ( /\//, $tmp[0] );
                my $fname = pop (@tmp2);
                my $out   = Bio::AlignIO->new(
                  '-file'   => ">filtered/$fname.tfa",
                  '-format' => "fasta"
                );
                $newaln->set_displayname_flat;
                $out->write_aln($newaln);
            }
        }
    }
}

sub sort_sub {
    my $retval;

    $retval = ( $ORGlist{$a} <=> $ORGlist{$b} );

    $retval;
}
```

B.7 findorf.pl

```
#!/usr/bin/perl -w

=head1 NAME

findorf.pl - find ORFs and BLAST the genome

=head1 SYNOPSIS

findorf.pl file.fasta [cp|mt] [min_orf_len]

=head1 DESCRIPTION

findorf.pl uses getorf (from EMBOSS) to identify the open reading frames.
By default, it uses the translation table 11 (which can be modified to
select initiation codons). The default minimum ORF length is 28 codons
but another value can specified as the third command line argument. The
sequences and the products of the found ORFs are put in the _seq and _trans
directory, respectively. The translated sequences are BLASTed, identified
and tagged appropriatly using only the matches from the specified
organite (in the file name or the second argument).

findorf.pl also divides the genome in parts of 100 bp and BLASTs them, keeping
only the results from the specified organite. Then it outputs in tab
delimited format for each position by increment of 100 the result of
the BLAST and the ORF found.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

use strict;
use Bio::SeqIO;
use File::Basename;

#reglage du PATH vers les bases de donnees
my $DBHOME = $ENV{DBHOME} ? $ENV{DBHOME} : '/Users/Shared/db';
```

```
#calcul de la taille minimum des orf
my $min = defined( $ARGV[2] ) ? $ARGV[2] : 28;

#definition des executables
my $blastx    = "blastall -p blastx";
my $blastn    = "blastall -p blastn";
my $blastp    = "blastall -p blastp";
my $mspcrunch = "MSPcrunch -x -";

#parsing du nom de fichier
my $fname = basename( $ARGV[0] );
my ( $name,     $ext )       = split( /\./, $fname );
my ( $organism, $organelle ) = split( /_/,  $name );
$organelle = $ARGV[1] unless $organelle;
$organelle = substr( $organelle, 0, 2 );
( $organelle eq 'cp' || $organelle eq 'mt' )
  || die "Incorrect sequence name!\n";

#creation des repertoires de sortie
mkdir( $organism . '_seq',        0755 );
mkdir( $organism . '_trans',      0755 );
mkdir( $organism . '_identified', 0755 );

#creation du repertoire temporaire pour tout le script
my $tempdir = '/tmp/' . $$;
mkdir( $tempdir, 0755 );

#les variables globales
my @orf_loc;    # tableau de references vers les orf presents sur chaque ligne

#fichier de resultats
open( OUTPUT, ">$organism.xls" );

#produit un fichier Fasta avec tous les orf
system( "getorf -table 11 -minsize 84 -nomethionine -find 3 "
      . "$ARGV[0] $tempdir/$ARGV[0].orf" );
my $in =
  Bio::SeqIO->new( -file => "$tempdir/$ARGV[0].orf", -format => 'Fasta' );

#traite chaque orf
while ( my $seq = $in->next_seq ) {

    #longueur de la proteine
```

```
    my $len = $seq->length / 3;

    #position de debut et fin et orientation
    $seq->desc =~ /\[(\d+) - (\d+)]/;
    my ( $start, $end ) = ( $1, $2 );
    my $compl = ( $start > $end ) ? 1 : 0;

    #elimine les erreurs venant des codes d'ambiguite
    $seq->alphabet('dna');

    #determination de la lettre a ajouter
    #pour eliminer les fichiers de meme nom
    my $letter = "a";
    while ( -e $organism . "_seq/orf$len$letter.tfa" ) { $letter++; }

    #output de la sequence codante
    write_fasta( $organism . "_seq/orf$len$letter.tfa",
        "orf$len$letter", "\($start..$end\)", $seq->seq );

    #output de la traduction pour la recherche
    write_fasta( "$tempdir/orf$len$letter.tfa", $seq->id, $seq->desc,
        $seq->translate->seq );

    #recherche contre la db de proteines
    #le sort et head conserve seulement le meilleur resultat
    my ($firstmsp) = `
$blastp -d $DBHOME/proteines -i $tempdir/orf$len$letter.tfa -F F|$mspcrunch
    `;

    #identification de l'orf et
    #preparation de sa description
    my $orf;
    if ($firstmsp) {
        $firstmsp =~ /^ *(\d+) +\(.+\) +\d+ +\d+ +\d+ +\d+ +(\w+)/;
        my ( $score, $gene ) = ( $1, $2 );
        $orf = $gene . "_orf$len$letter($start..$end)";
        write_fasta( $organism . "_identified/$gene.tfa",
            $organism, '', $seq->translate->seq );
        $seq->display_id($gene);
        $seq->desc("\($start..$end\)");
    }
    else {
        $seq->display_id("orf$len$letter");
```

```
        $seq->desc("\($start..$end\)");
        if ( $len >= $min ) {
            $orf = "orf$len$letter($start..$end)";
        }
    }

    #output de la sequence traduite
    write_fasta( $organism . "_trans/orf$len$letter.tfa",
        $seq->id, $seq->desc, $seq->translate->seq );

    #ajout de l'orf dans le tableau pour lignes
    #ou il est present
    my ( $fl, $ll ) = $compl
      ? ( int( $end / 100 ), int( $start / 100 ) )
      : ( int( $start / 100 ), int( $end / 100 ) );
    if ($orf) {
        foreach my $pos ( $fl .. $ll ) {
            push @{ $orf_loc[$pos] }, $orf;
        }
    }

    #suppression du fichier ayant servi pour la recherche
    unlink "$tempdir/orf$len$letter.tfa";
}

#suppression du fichier produit par getorf
unlink "$tempdir/$ARGV[0].orf";

my $start = 1;    #position dans la sequence

#filtration des caracteres indesirables
open( INPUT, "cat $ARGV[0]|tr -d \":\"|tr -d \'-\'|" );
my $in2         = Bio::SeqIO->new( -fh => *INPUT, -format => 'fasta' );
my $completeseq = $in2->next_seq;

#boucle pour chaque 100 nucleotides
while ( $start < $completeseq->length ) {
    my $end =
      ( $completeseq->length - $start > 100 )
      ? $start + 99
      : $completeseq->length;

    #fichier temporaire pour la recherche
```

```
open( OUT, ">$tempdir/$start" );
print OUT ">$start\n", $completeseq->subseq( $start, $end ), "\n";
close(OUT);

#output a tous les 1000 bps
#if ( substr( $start, -3, 3 ) eq '001' ) { print STDERR $start, ' ' }

#recherche BLAST
my @msp = (
    `$blastx -d $DBHOME/proteines -i $tempdir/$start -F F|$mspcrunch`,
    `$blastn -d $DBHOME/rna       -i $tempdir/$start -F F|$mspcrunch`
);

#suppression du fichier pour la recherche
unlink("$tempdir/$start");

#filtration des resultats
my @grepped;
if ( $organelle eq 'mt' ) { @grepped = grep ( !/chloroplast/,   @msp ) }
if ( $organelle eq 'cp' ) { @grepped = grep ( !/mitochondrion/, @msp ) }
chomp(@grepped);

#choix du meilleur score pour chaque gene
my %score;
foreach my $line (@grepped) {
    $line =~ /^ *(\d+) +\(.+\) +\d+ +\d+ +\d+ +\d+ +(\w+)/;
    $score{$2} = $1 if ( !defined( $score{$2} ) || $score{$2} < $1 );
}

#ligne d'output
my @oline = ( $start, ' ' x 8, '', '', '', '' );
my $col = 1;
foreach my $gene ( sort { $score{$b} <=> $score{$a} } keys %score ) {
    $oline[$col] = "$gene($score{$gene})";
    $col++;
}

#sortie des resultats BLAST
print OUTPUT join( "\t", @oline );

#sortie des orf
print OUTPUT join( "\t", @{ $orf_loc[ ( $start - 1 ) / 100 ] } )
  if defined( $orf_loc[ ( $start - 1 ) / 100 ] );
```

```
    print OUTPUT "\n";

    #passons au bloque de 100 pb suivant
    $start += 100;
}

#suppression du repertoire temporaire
rmdir $tempdir;

#petite procedure pour ecrire un fichier Fasta
sub write_fasta {
    my ( $fname, $id, $desc, $seq ) = @_;
    my $bioseq = Bio::PrimarySeq->new(
        -seq  => $seq,
        -id   => $id,
        -desc => $desc
    );
    my $out = Bio::SeqIO->new(
        -file   => ">>$fname",
        -format => 'Fasta'
    );
    $out->write_seq($bioseq);
}
```

B.8 formatdb.sh

```
#!/bin/bash
DB_PATH=/Users/Shared/db

PAST=‘pwd‘
cd $DB_PATH

mv prot_db.tfa prot_db.bak

ls -1 prot_seq/cp/*.tfa|xargs -n1 tfa2long.sh chloroplast >prot_db.tfa
ls -1 prot_seq/mt/*.tfa|xargs -n1 tfa2long.sh mitochondrion >>prot_db.tfa

formatdb -t proteines -i prot_db.tfa -p T -n proteines -V T
```

```
mv rna_db.tfa rna_db.bak
ls -1 RNA_seq/cp/*.tfa |xargs -n1 tfa2long.sh chloroplast >rna_db.tfa
#Enlever le diese de la ligne suivante lorsque les RNAs mitochondriaux
#seront etiquetes avec des noms courts
ls -1 RNA_seq/mt/*.tfa |xargs -n1 tfa2long.sh mitochondrion >>rna_db.tfa
#Enlever la ligne suivante lorsque les RNAs mitochondriaux seront etiquetes
#avec des noms courts
#cat RNA_seq/mt/*.tfa >> rna_db.tfa
cat Vectors >> rna_db.tfa

formatdb -t rna -i rna_db.tfa -p F -n rna -V T

cd $PAST
```

B.9 gbkextract.pl

```
#!/usr/bin/perl -w

=head1 NAME

gbkextract.pl - extract features from a Genbank file

=head1 SYNOPSIS

gbkextract.pl file.gbk

=head1 DESCRIPTION

gbkextract.pl creates six directories :

=over

=item *
RNA, which contains tRNA and rRNA sequences;

=item *
prot, which contains protein sequence for each CDS;

=item *
```

```
cDNA, which contains gene coding sequences excluding introns;

=item *
gene, which contains gene sequences, including introns;

=item *
exon, which contains lengths of exons usable by addexons.pl;

=item *
intron, which contains intron sequences and lengths.

=back

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

use strict;
use Bio::SeqIO;
use Bio::Location::Split;

#ouverture du fichier d'entree
my $in = Bio::SeqIO->new(
    '-file'   => "$ARGV[0]",
    '-format' => 'Genbank'
);

#lecture de la sequence
my $seq = $in->next_seq;

#extraction de l'organite d'origine
#suppose que c'est le troisieme mot de la description
#ne fonctionne pas toujours
my $desc = $seq->desc;
chop($desc);
my @desc = split( / /, $desc );
chop( $desc[2] );

#normalement $desc[0] est le genre, $desc[1], l'espece et $desc[2], l'organite

#creation des repertoires
```

```
mkdir( "RNA",    0755 );
mkdir( "prot",   0755 );
mkdir( "cDNA",   0755 );
mkdir( "gene",   0755 );
mkdir( "exon",   0755 );
mkdir( "intron", 0755 );

foreach my $feat ( $seq->top_SeqFeatures ) {

    #extraction de tous les genes, incluant tRNA, rRNA ou sequence codante
    if ( $feat->primary_tag eq "gene" ) {
        if ( $feat->has_tag('gene') ) {
            foreach my $gene ( $feat->each_tag_value('gene') ) {
                my $partseq = $feat->seq;
                $partseq->id($gene);
                my @name = split( /\//, $gene );
                $partseq->desc("$desc[2] [$desc[0] $desc[1]]");
                my $out = Bio::SeqIO->new(
                    '-file'   => ">>gene/$name[0].tfa",
                    '-format' => 'Fasta'
                );
                $out->write_seq($partseq);
            }
        }
    }

    #extraction des cDNA, sequence de proteines et RNA
    if (   $feat->primary_tag eq 'tRNA'
        || $feat->primary_tag eq 'rRNA'
        || $feat->primary_tag eq 'CDS' )
    {
        if ( $feat->has_tag('gene') ) {
            foreach my $gene ( $feat->each_tag_value('gene') ) {
                my $partseq = $feat->seq;

                #on ajuste la ligne de description
                $partseq->id( $ARGV[1] );
                $partseq->desc('');
                my @name =
                  split( /\//, $gene ); #pas de slash dans un nom de fichier
                my @tab_exon_len;
                my $loc = $feat->location;
```

```
if ( $loc->to_FTstring =~ /join/ ) {    #il y a des introns
    my $realseq;
    my @introns;
    my $prev_end;
    foreach my $location ( $loc->sub_Location ) {

        #sequence codante de chaque exon
        $realseq .=
          $feat->entire_seq->subseq( $location->start,
            $location->end );

        #longueur de chaque exon
        push( @tab_exon_len, $location->length );

        #l'intron
        if ( defined($prev_end)
            && ( $prev_end + 1 < $location->start - 1 ) )
        {
            my $newintron =
              $feat->entire_seq->trunc( $prev_end + 1,
                $location->start - 1 );
            $newintron->id( $ARGV[1] );
            $newintron->desc( ( $prev_end + 1 ) . '..'
                  . ( $location->start - 1 ) );
            push @introns, $newintron;

        }
        $prev_end = $location->end;
    }

    #assignation de la sequence codante seulement
    $partseq->seq($realseq);

    #inverse si on est sur le brin complementaire
    if ( $loc->strand == -1 ) {
        @tab_exon_len = reverse @tab_exon_len;
        $partseq      = $partseq->revcom;
        @introns      = reverse @introns;
        foreach (@introns) { $_ = $_->revcom; }
    }
    open( INTRON, ">>intron/$ARGV[1]_size.xls" );
    print INTRON "$name[0]\t";
    foreach (@introns) {
```

```
            my $outintron = Bio::SeqIO->new(
                '-file'   => ">>intron/$name[0].tfa",
                '-format' => 'Fasta'
            );
            $outintron->write_seq($_);
            print INTRON $_->length, "\t";
        }
        print INTRON "\n";
        close(INTRON);

    }
    if (   $feat->primary_tag eq 'tRNA'
        || $feat->primary_tag eq 'rRNA' )
    {
        my $out = Bio::SeqIO->new(
            '-file'   => ">>RNA/$name[0].tfa",
            '-format' => 'Fasta'
        );
        $out->write_seq($partseq);
    }

    if ( $feat->primary_tag eq 'CDS' ) {

        #ecriture du fichier de cDNA
        my $out = Bio::SeqIO->new(
            '-file'   => ">>cDNA/$name[0].tfa",
            '-format' => 'Fasta'
        );
        $out->write_seq(
            $partseq->trunc( 1, $partseq->length - 3 ) );

        if ( scalar(@tab_exon_len) > 1 ) {

            #ecriture du fichier de longueur d'exon
            open( OUTPUT, ">>exon/$name[0].loc" );
            print OUTPUT join ' ', $ARGV[1], @tab_exon_len;
            print OUTPUT "\n";
            close(OUTPUT);
        }
        if ( $feat->has_tag('translation') ) {
            foreach
              my $prot ( $feat->each_tag_value('translation') )
            {
```

```
                            #ecriture du fichier de sequence proteique
                            my $protseq = Bio::PrimarySeq->new(
                                -seq      => $prot,
                                -id       => $ARGV[1],
                                -alphabet => 'protein'
                            );
                            my $outprot = Bio::SeqIO->new(
                                '-file'   => ">>prot/$name[0].tfa",
                                '-format' => 'Fasta'
                            );
                            $outprot->write_seq($protseq);
                        }
                    }
                }
            }
        }
        else { print $feat->primary_tag, " has no tag gene\n"; }
    }
}
```

B.10 preparednamatrix.pl

```
#!/usr/bin/perl -w

use strict;
use Getopt::Long;
use Bio::AlignIO;

=head1 NAME

preparednamatrix.pl - create a protein matrix from aligned protein files

=head1 SYNOPSIS

preparednamatrix.pl *.aln

=head1 DESCRIPTION
```

```
preparednamatrix.pl takes as input alignments in Fasta format and
filters them with Gblocks. The filtered alignments are concatenated
to alignment.nex in nexus format.

It also produces a file called Goptns.txt which contains the length of
each filtered alignment for use in PAML.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

my $USAGE = qq {

};
open( GOPT, ">Goptns.txt" );
print GOPT "G " . scalar(@ARGV);
while ( my $f = shift(@ARGV) ) {
    my @split;
    @split = split( /\./, $f );
    `Gblocks $f -t=c` =~ /^Gblocks.+:\s+(\d+) positions/m;
    print GOPT " $1";
    my $in = Bio::AlignIO->new(
        '-file'   => "$f-gb",
        '-format' => 'fasta'
    );
    my $out = Bio::AlignIO->new(
        '-file'   => ">$split[0].aln-gb",
        '-format' => 'clustalw'
    );

    while ( my $aln = $in->next_aln() ) {
        $aln->set_displayname_flat();
        $out->write_aln($aln);
    }
    unlink("$split[0].tfa");
    unlink("$split[0].tfa-gb");

    #    unlink("$f-gb.htm");
    unlink("$split[0].dnd");
}
print GOPT "\n";
```

```
system( "cat *.aln-gb |grep -v \"CLUSTAL W\""
      . "|clus2mol.pl|mol2phy.pl|readseq -a -p -f17 >alignment.nex" );
```

B.11 prepareproteinmatrix.pl

```
#!/usr/bin/perl -w

use strict;
use File::Basename;
use Bio::AlignIO;

=head1 NAME

prepareproteinmatrix.pl - create a protein matrix from aligned protein files

=head1 SYNOPSIS

prepareproteinmatrix.pl [--nogblocks] *.aln

=head1 DESCRIPTION

prepareproteinmatrix.pl takes as input alignments in clustal format and
filters them with Gblocks. The filtered alignments are concatenated to
alignment.ptn in MOLPHY format.

It also produces a file called Goptns.txt which contains the length of
each filtered alignment for use in codeml.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

if ( @ARGV == 0 ) {
system("perldoc $0");
    exit;
}
my $gblocks = 1;
```

```
if ($ARGV[0] eq '--nogblocks') { $gblocks = 0; shift;}
open( GOPT, ">Goptns.txt" );
print GOPT "G " . scalar(@ARGV);

while ( my $f = shift ) { &filter($f,$gblocks) }

system("cat *.aln-gb |grep -v \"CLUSTAL \"|clus2mol.pl>alignment.ptn");

print GOPT "\n";

sub filter {
    my ($f,$gblocks)= @_;
    my ($name,$ext) = split(/\./, basename($f));

    if ($gblocks) {
    {
        my $in1 = Bio::AlignIO->new(
          '-file'   => $f,
          '-format' => 'clustalw'
        );

        my $out1 = Bio::AlignIO->new(
          '-file'   => ">$name.tfa",
          '-format' => 'fasta'
        );

        while ( my $aln = $in1->next_aln ) {
            $aln->set_displayname_flat;
            $out1->write_aln($aln);
        }
    }
    `Gblocks $name.tfa -t=p`;

    my $in = Bio::AlignIO->new(
      '-file'   => "$name.tfa-gb",
      '-format' => 'fasta'
    );
    my $out = Bio::AlignIO->new(
      '-file'   => ">$name.aln-gb",
      '-format' => 'clustalw'
    );

    while ( my $aln = $in->next_aln ) {
```

```
        $aln->set_displayname_flat;
        $out->write_aln($aln);
    }

    unlink("$name.dnd");
    unlink("$name.tfa");
    unlink("$name.tfa-gb");
}
my $in = Bio::AlignIO->new(
      '-file'   => "$name.aln-gb",
      '-format' => 'clustalw'
    );
    my $aln = $in->next_aln;
    print GOPT ' ',$aln->length;
}
```

B.12 prepsequin.pl

```
#!/usr/bin/perl -w

=head1 NAME

prepsequin.pl - Prepare protein sequences for input by Sequin

=head1 SYNOPSIS

prepsequin.pl *.tfa

=head1 DESCRIPTION

prepsequin.pl requests the organism name and the output file,
then selects all protein sequences from the organism in
the input files specified on the command line, and reformats them
in a format suitable for input by Sequin.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>
```

```
=cut

my $DBHOME = $ENV{DBHOME} ? $ENV{DBHOME} : '/Users/Shared/db';

print("Please enter the organism name: ");
my $organism = <STDIN>;
chomp($organism);

print("Please enter the filename you want to print the results to: ");
my $file = <STDIN>;
chomp($file);

open( OUTPUT, ">$file" );

open( DB, "$DBHOME/gene_db.txt" );
my %gene_db;

my $line;
my @tmp;

while ( $line = <DB> ) {
    chomp($line);
    @tmp = split ( /\t/, $line );
    $gene_db{ $tmp[0] } = $tmp[1];
}
close(DB);
$line = "";
my $counter = 0;

while ( my $f = shift ) {
    open( INPUT, $f );
    while ( $line = <INPUT> ) {

        while ( $line =~ /^>/ ) {

            if ( $line =~ /$organism/ ) {

                my @tmp3;
                @tmp3 = split ( /\./, $f );
                if ( !( $gene_db{ $tmp3[0] } ) ) {
                     $gene_db{ $tmp3[0] } = "";
                }
                $counter++;
```

```
                print OUTPUT (
">prot_$counter [gene=$tmp3[0]] [prot=$gene_db{$tmp3[0]}] $tmp3[0]\n"
                );
                my $seq;

                while ( ( $seq = <INPUT> ) && !( $seq =~ />/ ) ) {
                    print OUTPUT ("$seq");
                }
                $line = <INPUT>;

            }
            else { $line = <INPUT>; }
        }
    }
}
```

B.13 tfa2long.sh

```
#!/bin/bash
FILE=‘echo $2|cut -d’/’ -f3‘
PROT=‘echo $FILE|cut -d. -f1‘
O=$1
cat $2 |
sed -e s/">CHAEglobo"/">$PROT $O [Chaetosphaeridium globosum]"/ |
sed -e s/">PAVLluthe"/">$PROT $O [Pavlova lutheri]"/ |
sed -e s/">SYNEs6803"/">$PROT $O [Synechocystis PCC6803]"/ |
sed -e s/">PORPpurpu"/">$PROT $O [Porphyra purpurea]"/ |
sed -e s/">CYANcalda"/">$PROT $O [Cyanidium caldarium]"/ |
sed -e s/">GUILtheta"/">$PROT $O [Guillardia theta]"/ |
sed -e s/">ISOCtahit"/">$PROT $O [Isochrysis tahiti]"/ |
sed -e s/">ODONsinen"/">$PROT $O [Odontella sinensis]"/ |
sed -e s/">CYANparad"/">$PROT $O [Cyanophora paradoxa]"/ |
sed -e s/">MESOvirid"/">$PROT $O [Mesostigma viride]"/ |
sed -e s/">EUGLgraci"/">$PROT $O [Euglena gracilis]"/ |
sed -e s/">NEPHoliva"/">$PROT $O [Nephroselmis olivacea]"/ |
sed -e s/">PYCNpraso"/">$PROT $O [Pycnococcus prasovali]"/ |
sed -e s/">MONOoke-1"/">$PROT $O [Monomastix species OKE-1]"/ |
sed -e s/">SCHEdubia"/">$PROT $O [Scherffelia dubia]"/ |
```

```
sed -e s/">PSEUakine"/">$PROT $O [Pseudendoclonium akinetum]"/ |
sed -e s/">NEOCpseud"/">$PROT $O [Neochloris pseudoalveolaris]"/ |
sed -e s/">PEDIminor"/">$PROT $O [Pedinomonas minor]"/ |
sed -e s/">CHLOvulga"/">$PROT $O [Chlorella vulgaris]"/ |
sed -e s/">SCENobliq"/">$PROT $O [Scenedesmus obliquus]"/ |
sed -e s/">CHLAreinh"/">$PROT $O [Chlamydomonas reinhardtii]"/ |
sed -e s/">MARCpolym"/">$PROT $O [Marchantia polymorpha]"/ |
sed -e s/">PINUthunb"/">$PROT $O [Pinus thunbergii]"/ |
sed -e s/">NICOtabac"/">$PROT $O [Nicotiana tabacum]"/ |
sed -e s/">RECLameri"/">$PROT $O [Reclinomonas americana]"/ |
sed -e s/">CYANmerol"/">$PROT $O [Cyanidioschyzon merolae]"/ |
sed -e s/">CHONcrisp"/">$PROT $O [Chondrus crispus]"/ |
sed -e s/">RHODsalin"/">$PROT $O [Rhodomonas salina]"/ |
sed -e s/">OCHRdanic"/">$PROT $O [Ochromonas danica]"/ |
sed -e s/">PROTwicke"/">$PROT $O [Prototheca wickerhamii]"/ |
sed -e s/">CHLAeugam"/">$PROT $O [Chlamydomonas eugametos]"/ |
sed -e s/">CHLOelong"/">$PROT $O [Chlorogonium elongatum]"/ |
sed -e s/">OLTMvirid"/">$PROT $O [Oltmannsiellopsis viridis]"/ |
sed -e s/">ARABthali"/">$PROT $O [Arabidopsis thaliana]"/ |
sed -e s/">STAUpunct"/">$PROT $O [Staurastrum punctulatum]"/ |
sed -e s/">BETAvulga"/">$PROT $O [Beta vulgaris]"/ |
sed -e s/">PSILnudum"/">$PROT $O [Psilotum nudum]"/ |
sed -e s/">CHLOatmop"/">$PROT $O [Chlorokybus atmophyticus]"/ |
sed -e s/">ANTHformo"/">$PROT $O [Anthoceros formosae]"/ |
sed -e s/">CHARvulga"/">$PROT $O [Chara vulgaris]"/ |
sed -e s/">ZYGNcircu"/">$PROT $O [Zygnema circumcarinatum]"/ |
sed -e s/">BRYOplumo"/">$PROT $O [Bryopsis plumosa]"/ |
sed -e s/">DERBmarin"/">$PROT $O [Derbesia marina]"/ |
sed -e s/">ADIAcapil"/">$PROT $O [Adiantum capillus]"/ |
sed -e s/">CALYferti"/">$PROT $O [Calycanthus fertilis]"/ |
sed -e s/">PHYSpaten"/">$PROT $O [Physcomitrella patens]"/ |
sed -e s/">AMBOtrich"/">$PROT $O [Amborella trichopoda]"/ |
sed -e s/">VAUCbursa"/">$PROT $O [Vaucheria bursata]"/ |
sed -e s/">LAMIdigit"/">$PROT $O [Laminaria digitata]"/ |
sed -e s/">PYLAlitto"/">$PROT $O [Pylaiella littoralis]"/ |
sed -e s/">ORYZsativ"/">$PROT $O [Oryza sativa]"/ |
sed -e s/">ZEAmays"/">$PROT $O [Zea mays]"/
```

B.14 updatealign.pl

```
#!/usr/bin/perl -w

use strict;
use Bio::AlignIO;

=head1 NAME

updatealign.pl - update the Selex alignments

=head1 SYNOPSIS

updatealign.pl ORG_list *.aln

=head1 DESCRIPTION

updatealign.pl takes as first argument the list of all organisms
in the order they are to appear in the alignments and as second
argument the alignment files of the proteins.

First, it reorders the sequences in the protein alignments and in the coding
sequence files from the database. Then it calls codonalign.pl to create the
alignment of coding sequences. Finally, it calls addexons.pl to reformat
the alignment and to add the exon positions.

=head1 AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

open( LIST, shift );

my $dbhome = $ENV{DBHOME} ? $ENV{DBHOME} : '/Users/Shared/db';
my $numorg;
my %ORGlist;

while ( my $line = <LIST> ) {
    chomp($line);
    if ( ($line) && !( $line =~ /^\#/ ) ) {
```

```
        my @tmparray = split( /\t/, $line );
        if ( @tmparray == 1 ) {
            $numorg++;
            $ORGlist{ $tmparray[0] } = $numorg;

        }
    }
}
close(LIST);

while ( my $f = shift ) {
    my @path = split( /\//, $f );
    my ( $fname, $ext ) = split( /\./, $path[ scalar(@path) - 1 ] );
    {
        my $in = Bio::AlignIO->new( '-file' => $f, '-format' => 'clustalw' );
        my $aln = $in->next_aln;
        $aln->set_displayname_flat;
        my @orgtab;

        foreach my $seq ( $aln->each_seq ) {
            if ( $seq->id ) {
                push( @orgtab, $seq->id );
            }
        }
        my @sorted = sort sort_sub(@orgtab);
        my $newaln = Bio::SimpleAlign->new;

        foreach my $org (@sorted) {
            if ($org) {
                if ( $ORGlist{$org} ) {
                    foreach my $seq2 ( $aln->each_seq ) {
                        if ( $seq2->id =~ $org ) {
                            $newaln->add_seq($seq2);
                            last;
                        }
                    }
                }
            }
        }
        system("mv $f $fname.old");
        my $out = Bio::AlignIO->new(
            '-file'   => ">$fname.aln",
            '-format' => "clustalw"
```

```
    );
    $newaln->set_displayname_flat;
    $out->write_aln($newaln);
}
{
    my $in = Bio::AlignIO->new(
        '-file'   => "$dbhome/cDNA_seq/cp/$fname.tfa",
        '-format' => 'fasta'
    );
    my $aln = $in->next_aln;
    $aln->set_displayname_flat;
    my @orgtab;

    foreach my $seq ( $aln->each_seq ) {
        if ( $seq->id ) {
            push( @orgtab, $seq->id );
        }
    }
    my @sorted = sort sort_sub(@orgtab);
    my $newaln = Bio::SimpleAlign->new();

    foreach my $org (@sorted) {
        if ($org) {
            if ( $ORGlist{$org} ) {
                foreach my $seq2 ( $aln->each_seq ) {
                    if ( $seq2->id =~ $org ) {
                        $newaln->add_seq($seq2);
                        last;
                    }
                }
            }
        }
    }
    my $out = Bio::AlignIO->new(
        '-file'   => ">$fname.fasta",
        '-format' => "fasta"
    );
    $newaln->set_displayname_flat;
    $out->write_aln($newaln);
}
system("codonalign.pl . $fname.fasta");
system("addexons.pl $fname.dna_aln $dbhome/exon_db/cp");
```

```
}

sub sort_sub {
    my $retval;

    $retval = ( $ORGlist{$a} <=> $ORGlist{$b} );

    $retval;
}
```

Annexe C

Manuel d'utilisation

C.1 addexons.pl

Produce Selex files

SYNOPSIS

addexons.pl file.fasta path_to_loc

DESCRIPTION

addexons.pl takes as first argument a Fasta file containing the aligned sequences coding for a given protein and as second argument a directory containing a file of the same name with the .loc extension which contains the lengths of the exons for each organism that features introns in this protein.

It outputs in the current directory a file containing the aligned sequences in Selex format with the intron positions marked by ><.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.2 choosetpl.pl

Select topologies from protml output

SYNOPSIS

choosetpl.pl file.tpl file.ml

DESCRIPTION

choosetpl.pl parses the output of protml and selects the trees that have a non null bootstrap value.

Those trees are printed to the standard output and the number of trees selected is printed to STDERR.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

=cut

use strict ;

@ARGV > 2 || die "Usage : choosetpl.pl file.tpl file.ml\n" ;

open(TPL, shift) ; open(ML, shift) ;

```
my $line; my $i;

while (<ML>) { last if (/-{60}/) }

<TPL>;

while (<TPL>) { if (!/^\#/ ) { $line = $_ ; last; } }

while (<ML>) { chomp; my @ml = split (/ +/); if ( $ml[8] > 0 ) { $i++; my
($tpl) = split ( /;/, $line ); print "$tpl;\n"; } $line = <TPL>; }

print STDERR "$i\n";
```

C.3 codonalign.pl

Align a coding sequence using a protein alignment as reference

SYNOPSIS

codonalign.pl aln_dir *.tfa

DESCRIPTION

codonalign.pl aligns each Fasta file using the corresponding clustalw .aln file in aln_dir as reference.

In the output alignment in Fasta format, each amino acid is replaced by the corresponding three nucleotides from the coding sequence and each gap is replaced by three gaps.

The output files have the same name as the input files but with the .dna_aln extension. They are put in the current directory.

CAVEATS

The coding sequence file and alignment file must contain exactly the same sequences in the same order. There is no verification that the sequences really correspond.

A file having the same name as an output file will be overwritten.

AUTHOR

Based on align_on_codons.pl from Bioperl and modified heavily by Jules Gagnon <eonwe@users.sourceforge.net>

C.4 filterselex.pl

Keep only the desired sequence from a Selex file

SYNOPSYS

filterselex.pl ORG_list /path/to/*.selex

DESCRIPTION

filterselex.pl takes a file containing the list of the desired organisms to be kept and a list of Selex files to be filtered.

It outputs a Fasta file for each Selex file in the current directory and the Selex files are unmodified. The Fasta files are unaligned.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.5 filtertfa.pl

Keep only the desired organism for a Fasta file

SYNOPSIS

filertfa.pl ORG_list *.tfa

DESCRIPTION

filtertfa.pl takes a file containing the list of the desired organisms to be kept and a list of Fasta files to be filtered.

It creates a directory named filtered which contains all the Fasta files featuring all and only the desired organisms.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.6 findorf.pl

Find ORFs and BLAST the genome

SYNOPSIS

findorf.pl file.fasta [cp|mt] [min_orf_len]

DESCRIPTION

findorf.pl uses getorf (from EMBOSS) to identify the open reading frames. By default, it uses the translation table 11 (which can be modified to select initiation codons). The default minimum ORF length is 28 codons but another value can specified as the third command line argument. The sequences and the products of the found ORFs are put in the _seq and _trans directory, respectively. The translated sequences are BLASTed, identified and tagged appropriatly using only the matches from the specified organite (in the file name or the second argument).

findorf.pl also divides the genome in parts of 100 bp and BLASTs them, keeping only the results from the specified organite. Then it outputs in tab delimited format for each position by increment of 100 the result of the BLAST and the ORF found.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.7 gbkextract.pl

Extract features from a Genbank file

SYNOPSIS

gbkextract.pl file.gbk

DESCRIPTION

gbkextract.pl creates six directories :

- RNA, which contains tRNA and rRNA sequences ;
- prot, which contains protein sequence for each CDS ;

– cDNA, which contains gene coding sequences excluding introns ;
– gene, which contains gene sequences, including introns ;
– exon, which contains lengths of exons usable by addexons.pl ;
– intron, which contains intron sequences and lengths.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.8 preparednamatrix.pl

Create a protein matrix from aligned protein files

SYNOPSIS

preparednamatrix.pl *.aln

DESCRIPTION

preparednamatrix.pl takes as input alignments in Fasta format and filters them with Gblocks. The filtered alignments are concatenated to alignment.nex in nexus format.

It also produces a file called Goptns.txt which contains the length of each filtered alignment for use in PAML.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.9 prepareproteinmatrix.pl

Create a protein matrix from aligned protein files

SYNOPSIS

prepareproteinmatrix.pl [–nogblocks] *.aln

DESCRIPTION

prepareproteinmatrix.pl takes as input alignments in clustal format and filters them with Gblocks. The filtered alignments are concatenated to alignment.ptn in MOLPHY format.

It also produces a file called Goptns.txt which contains the length of each filtered alignment for use in codeml.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.10 prepsequin.pl

Prepare protein sequences for input by Sequin

SYNOPSIS

prepsequin.pl *.tfa

DESCRIPTION

prepsequin.pl requests the organism name and the output file, then selects all protein sequences from the organism in the input files specified on the command line, and reformats them in a format suitable for input by Sequin.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

C.11 updatealign.pl

Update the Selex alignments

SYNOPSIS

updatealign.pl ORG_list *.aln

DESCRIPTION

updatealign.pl takes as first argument the list of all organisms in the order they are to appear in the alignments and as second argument the alignment files of the proteins.

First, it reorders the sequences in the protein alignments and in the coding sequence files from the database. Then it calls codonalign.pl to create the alignment of coding sequences. Finally, it calls addexons.pl to reformat the alignment and to add the exon positions.

AUTHOR

Written by Jules Gagnon <eonwe@users.sourceforge.net>

Annexe D

Résultats d'analyses

D.1 Fichier produit par findorf.pl

```
14101                                   orf45a(14117..14251)
14201                                   orf45a(14117..14251)
14301
14401                                   rpl20_orf117a(14478..14828)
14501   rpl20(170)                      rpl20_orf117a(14478..14828)
14601   rpl20(167)                      rpl20_orf117a(14478..14828)
14701   rpl20(160)                      rpl20_orf117a(14478..14828)
14801                                   rpl20_orf117a(14478..14828)
14901   rps18(83)                       rps18_orf82c(15194..14949)
15001   rps18(152)                      rps18_orf82c(15194..14949)
15101   rps18(139)                      rpl33_orf73g(15416..15198)      rps18_orf82c(15194..14949)
15201   rpl33(184)                      rpl33_orf73g(15416..15198)
15301   rpl33(161)                      rpl33_orf73g(15416..15198)
15401                                   rpl33_orf73g(15416..15198)
15501                                   psaJ_orf47ab(15729..15589)
15601   psaJ(170)                       psaJ_orf47ab(15729..15589)
15701                                   psaJ_orf47ab(15729..15589)
15801
15901                                   orf64d(16165..15974)
16001                                   orf64d(16165..15974)
16101   trnP_ugg(260)                   orf64d(16165..15974)
16201
16301   trnW_cca(250)                   orf62b(16375..16560)
16401   petG(149)                       orf62b(16375..16560)    petG_orf37bc(16519..16409)
16501                                   orf62b(16375..16560)    petG_orf37bc(16519..16409)
16601   petL(87)                        petL_orf38ak(16757..16644)
16701                                   petL_orf38ak(16757..16644)
16801
16901
```

D.2 Protéines communes

```
atpA.tfa  petB.tfa  psaJ.tfa  psbI.tfa  rbcL.tfa   rpoC1.tfa  rps3.tfa
atpB.tfa  petD.tfa  psbA.tfa  psbJ.tfa  rpl14.tfa  rpoC2.tfa  rps4.tfa
atpE.tfa  petG.tfa  psbB.tfa  psbK.tfa  rpl16.tfa  rps11.tfa  rps7.tfa
atpF.tfa  petL.tfa  psbC.tfa  psbL.tfa  rpl20.tfa  rps12.tfa  rps8.tfa
atpH.tfa  psaA.tfa  psbD.tfa  psbM.tfa  rpl2.tfa   rps14.tfa  ycf3.tfa
ccsA.tfa  psaB.tfa  psbE.tfa  psbN.tfa  rpl36.tfa  rps18.tfa  ycf4.tfa
clpP.tfa  psaC.tfa  psbF.tfa  psbT.tfa  rpoA.tfa   rps19.tfa
petA.tfa  psaI.tfa  psbH.tfa  psbZ.tfa  rpoB.tfa   rps2.tfa
```

D.3 Fichier Goptns.txt

G 54 491 470 119 157 81 190 178 266 215 160 35 26 738 733 79 30 39 349 506 460 339 73 34 57 35 35 38 38 31 40 29 61 475 115 131 266 107 28 154 825 452 545 119 120 93 52 92 217 177 137 155 104 166 169

D.4 Topologies sélectionnées par la recherche exhaustive de protml

```
105 / 10395 protmlcp 2.3b3 "mtREV24-F" 8 OTUs 10831 sites.
# <= 105 trees (top ranking for approx. ln L) in the top 30.0% range of TBL
# range    TBL      trees
#   <    110.06         0
#   5%   111.25         6
#  10%   112.44        24
#  15%   113.62        40 *
#  20%   114.81        54 **
#  25%   115.99       125 ****
#  30%   117.18       126 ****
#  35%   118.37        97 ***
#  40%   119.55       189 *******
#  45%   120.74       334 ************
#  50%   121.92       455 *****************
#  55%   123.11       628 ************************
#  60%   124.29       882 *********************************
#  65%   125.48       748 ****************************
#  70%   126.67       693 *************************
#  75%   127.85      1082 ****************************************
#  80%   129.04      1286 ************************************************
#  85%   130.22      1276 ************************************************
```

```
#  90%   131.41      1229 **********************************************
#  95%   132.60       900 *********************************
# 100%   133.78       218 ********
#          over         3
# approx. ln L -95085.9 ... -96324.5 diff 1238.7, TBL 110.1 ... 114.6 diff 4.5
((CYANparad,MESOvirid),(NEPHoliva,(PEDIminor,CHLOvulga)),((NICOtabac,PINUthunb),MARCpolym)); 0.0
(CYANparad,((NEPHoliva,(PEDIminor,CHLOvulga)),MESOvirid),((NICOtabac,PINUthunb),MARCpolym)); 95.2
(CYANparad,(NEPHoliva,(PEDIminor,CHLOvulga)),(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 108.0
(((CYANparad,MESOvirid),(PEDIminor,CHLOvulga)),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 173.9
((CYANparad,MESOvirid),NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),(PEDIminor,CHLOvulga))); 204.1
(CYANparad,(NEPHoliva,(MESOvirid,(PEDIminor,CHLOvulga))),((NICOtabac,PINUthunb),MARCpolym)); 225.7
((CYANparad,(PEDIminor,CHLOvulga)),NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 231.0
(CYANparad,NEPHoliva,((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),(PEDIminor,CHLOvulga))); 252.5
(CYANparad,NEPHoliva,((((NICOtabac,PINUthunb),MARCpolym),(PEDIminor,CHLOvulga)),MESOvirid)); 267.9
(CYANparad,NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),(MESOvirid,(PEDIminor,CHLOvulga)))); 334.4
((CYANparad,(MESOvirid,(PEDIminor,CHLOvulga))),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 341.1
(CYANparad,((NEPHoliva,MESOvirid),(PEDIminor,CHLOvulga)),((NICOtabac,PINUthunb),MARCpolym)); 349.7
(CYANparad,(NEPHoliva,MESOvirid),(((NICOtabac,PINUthunb),MARCpolym),(PEDIminor,CHLOvulga))); 358.6
(((CYANparad,(PEDIminor,CHLOvulga)),MESOvirid),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 372.2
((CYANparad,(PEDIminor,CHLOvulga)),(NEPHoliva,MESOvirid),((NICOtabac,PINUthunb),MARCpolym)); 387.1
((CYANparad,MESOvirid),(NEPHoliva,(PEDIminor,CHLOvulga)),((NICOtabac,MARCpolym),PINUthunb)); 425.8
((CYANparad,MESOvirid),(NEPHoliva,(PEDIminor,CHLOvulga)),(NICOtabac,(MARCpolym,PINUthunb))); 434.0
(CYANparad,((NEPHoliva,(PEDIminor,CHLOvulga)),MESOvirid),((NICOtabac,MARCpolym),PINUthunb)); 520.7
(CYANparad,((NEPHoliva,(PEDIminor,CHLOvulga)),MESOvirid),(NICOtabac,(MARCpolym,PINUthunb))); 527.8
(CYANparad,(NEPHoliva,(PEDIminor,CHLOvulga)),(((NICOtabac,MARCpolym),PINUthunb),MESOvirid)); 531.3
(CYANparad,(NEPHoliva,(PEDIminor,CHLOvulga)),((NICOtabac,(MARCpolym,PINUthunb)),MESOvirid)); 540.0
((CYANparad,MESOvirid),((NEPHoliva,PEDIminor),CHLOvulga),((NICOtabac,PINUthunb),MARCpolym)); 591.6
((CYANparad,MESOvirid),((NEPHoliva,CHLOvulga),PEDIminor),((NICOtabac,PINUthunb),MARCpolym)); 596.3
(((CYANparad,MESOvirid),(PEDIminor,CHLOvulga)),NEPHoliva,((NICOtabac,MARCpolym),PINUthunb)); 602.1
(((CYANparad,MESOvirid),(PEDIminor,CHLOvulga)),NEPHoliva,(NICOtabac,(MARCpolym,PINUthunb))); 609.7
((CYANparad,MESOvirid),NEPHoliva,(((NICOtabac,MARCpolym),PINUthunb),(PEDIminor,CHLOvulga))); 629.2
((CYANparad,MESOvirid),NEPHoliva,((NICOtabac,(MARCpolym,PINUthunb)),(PEDIminor,CHLOvulga))); 638.2
(CYANparad,(NEPHoliva,(MESOvirid,(PEDIminor,CHLOvulga))),((NICOtabac,MARCpolym),PINUthunb)); 653.3
((CYANparad,(PEDIminor,CHLOvulga)),NEPHoliva,(((NICOtabac,MARCpolym),PINUthunb),MESOvirid)); 657.1
(CYANparad,(NEPHoliva,(MESOvirid,(PEDIminor,CHLOvulga))),(NICOtabac,(MARCpolym,PINUthunb))); 660.3
((CYANparad,(PEDIminor,CHLOvulga)),NEPHoliva,((NICOtabac,(MARCpolym,PINUthunb)),MESOvirid)); 665.8
(CYANparad,NEPHoliva,((((NICOtabac,MARCpolym),PINUthunb),MESOvirid),(PEDIminor,CHLOvulga))); 679.0
(CYANparad,NEPHoliva,(((NICOtabac,(MARCpolym,PINUthunb)),MESOvirid),(PEDIminor,CHLOvulga))); 687.9
(CYANparad,(((NEPHoliva,PEDIminor),CHLOvulga),MESOvirid),((NICOtabac,PINUthunb),MARCpolym)); 698.3
(CYANparad,NEPHoliva,((((NICOtabac,MARCpolym),PINUthunb),(PEDIminor,CHLOvulga)),MESOvirid)); 698.5
(CYANparad,(((NEPHoliva,CHLOvulga),PEDIminor),MESOvirid),((NICOtabac,PINUthunb),MARCpolym)); 703.5
(CYANparad,NEPHoliva,(((NICOtabac,(MARCpolym,PINUthunb)),(PEDIminor,CHLOvulga)),MESOvirid)); 707.0
(CYANparad,((NEPHoliva,PEDIminor),CHLOvulga),(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 717.5
(CYANparad,((NEPHoliva,CHLOvulga),PEDIminor),(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 724.9
(CYANparad,NEPHoliva,(((NICOtabac,MARCpolym),PINUthunb),(MESOvirid,(PEDIminor,CHLOvulga)))); 762.2
(CYANparad,NEPHoliva,((NICOtabac,(MARCpolym,PINUthunb)),(MESOvirid,(PEDIminor,CHLOvulga)))); 770.2
((CYANparad,(MESOvirid,(PEDIminor,CHLOvulga))),NEPHoliva,((NICOtabac,MARCpolym),PINUthunb)); 770.5
(CYANparad,((NEPHoliva,MESOvirid),(PEDIminor,CHLOvulga)),((NICOtabac,MARCpolym),PINUthunb)); 777.9
((CYANparad,(MESOvirid,(PEDIminor,CHLOvulga))),NEPHoliva,(NICOtabac,(MARCpolym,PINUthunb))); 778.0
(CYANparad,((NEPHoliva,MESOvirid),(PEDIminor,CHLOvulga)),(NICOtabac,(MARCpolym,PINUthunb))); 785.3
(CYANparad,(NEPHoliva,MESOvirid),(((NICOtabac,MARCpolym),PINUthunb),(PEDIminor,CHLOvulga))); 786.7
(CYANparad,(NEPHoliva,MESOvirid),((NICOtabac,(MARCpolym,PINUthunb)),(PEDIminor,CHLOvulga))); 795.6
(((CYANparad,(PEDIminor,CHLOvulga)),MESOvirid),NEPHoliva,((NICOtabac,MARCpolym),PINUthunb)); 801.7
(((CYANparad,(PEDIminor,CHLOvulga)),MESOvirid),NEPHoliva,(NICOtabac,(MARCpolym,PINUthunb))); 809.3
((CYANparad,(PEDIminor,CHLOvulga)),(NEPHoliva,MESOvirid),((NICOtabac,MARCpolym),PINUthunb)); 813.7
((CYANparad,(PEDIminor,CHLOvulga)),(NEPHoliva,MESOvirid),(NICOtabac,(MARCpolym,PINUthunb))); 822.3
(((CYANparad,CHLOvulga),PEDIminor),NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 856.4
(((CYANparad,PEDIminor),CHLOvulga),NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 878.1
((CYANparad,MESOvirid),NEPHoliva,((((NICOtabac,PINUthunb),MARCpolym),PEDIminor),CHLOvulga)); 935.3
((((CYANparad,CHLOvulga),PEDIminor),MESOvirid),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 956.3
(CYANparad,(NEPHoliva,CHLOvulga),((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),PEDIminor)); 957.7
```

```
((((CYANparad,MESOvirid),CHLOvulga),PEDIminor),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 959.7
(CYANparad,(NEPHoliva,PEDIminor),((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),CHLOvulga)); 963.4
((CYANparad,MESOvirid),NEPHoliva,((((NICOtabac,PINUthunb),MARCpolym),CHLOvulga),PEDIminor)); 968.6
((((CYANparad,PEDIminor),CHLOvulga),MESOvirid),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 970.5
(((CYANparad,CHLOvulga),PEDIminor),(NEPHoliva,MESOvirid),((NICOtabac,PINUthunb),MARCpolym)); 995.4
(CYANparad,NEPHoliva,(((((NICOtabac,PINUthunb),MARCpolym),PEDIminor),CHLOvulga),MESOvirid)); 1014.2
(((CYANparad,PEDIminor),CHLOvulga),(NEPHoliva,MESOvirid),((NICOtabac,PINUthunb),MARCpolym)); 1014.6
((((CYANparad,MESOvirid),PEDIminor),CHLOvulga),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 1016.5
(((CYANparad,MESOvirid),CHLOvulga),(NEPHoliva,PEDIminor),((NICOtabac,PINUthunb),MARCpolym)); 1018.6
((CYANparad,MESOvirid),((NEPHoliva,PEDIminor),CHLOvulga),((NICOtabac,MARCpolym),PINUthunb)); 1022.8
((CYANparad,MESOvirid),(NEPHoliva,CHLOvulga),(((NICOtabac,PINUthunb),MARCpolym),PEDIminor)); 1024.9
((CYANparad,MESOvirid),((NEPHoliva,CHLOvulga),PEDIminor),((NICOtabac,MARCpolym),PINUthunb)); 1027.6
((CYANparad,MESOvirid),((NEPHoliva,PEDIminor),CHLOvulga),(NICOtabac,(MARCpolym,PINUthunb))); 1030.9
((CYANparad,MESOvirid),((NEPHoliva,CHLOvulga),PEDIminor),(NICOtabac,(MARCpolym,PINUthunb))); 1035.7
(CYANparad,NEPHoliva,(((((NICOtabac,PINUthunb),MARCpolym),CHLOvulga),PEDIminor),MESOvirid)); 1038.3
(((CYANparad,MESOvirid),PEDIminor),(NEPHoliva,CHLOvulga),((NICOtabac,PINUthunb),MARCpolym)); 1042.0
((CYANparad,MESOvirid),(NEPHoliva,PEDIminor),(((NICOtabac,PINUthunb),MARCpolym),CHLOvulga)); 1054.3
(CYANparad,(NEPHoliva,MESOvirid),((((NICOtabac,PINUthunb),MARCpolym),PEDIminor),CHLOvulga)); 1057.1
((CYANparad,CHLOvulga),NEPHoliva,((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),PEDIminor)); 1060.1
(((CYANparad,MESOvirid),CHLOvulga),NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),PEDIminor)); 1068.1
(CYANparad,(((NEPHoliva,MESOvirid),CHLOvulga),PEDIminor),((NICOtabac,PINUthunb),MARCpolym)); 1068.7
(CYANparad,NEPHoliva,(((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),PEDIminor),CHLOvulga)); 1069.6
((CYANparad,PEDIminor),NEPHoliva,((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),CHLOvulga)); 1070.4
(CYANparad,(((NEPHoliva,MESOvirid),PEDIminor),CHLOvulga),((NICOtabac,PINUthunb),MARCpolym)); 1072.3
(CYANparad,NEPHoliva,(((((NICOtabac,PINUthunb),MARCpolym),MESOvirid),CHLOvulga),PEDIminor)); 1074.1
(CYANparad,(NEPHoliva,MESOvirid),((((NICOtabac,PINUthunb),MARCpolym),CHLOvulga),PEDIminor)); 1090.1
(CYANparad,(((NEPHoliva,CHLOvulga),MESOvirid),PEDIminor),((NICOtabac,PINUthunb),MARCpolym)); 1092.7
(CYANparad,(((NEPHoliva,PEDIminor),MESOvirid),CHLOvulga),((NICOtabac,PINUthunb),MARCpolym)); 1096.4
(((CYANparad,MESOvirid),PEDIminor),NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),CHLOvulga)); 1097.6
(CYANparad,(NEPHoliva,((MESOvirid,CHLOvulga),PEDIminor)),((NICOtabac,PINUthunb),MARCpolym)); 1098.6
(CYANparad,(NEPHoliva,((MESOvirid,PEDIminor),CHLOvulga)),((NICOtabac,PINUthunb),MARCpolym)); 1109.7
((CYANparad,CHLOvulga),(NEPHoliva,PEDIminor),(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 1120.8
(CYANparad,(((NEPHoliva,PEDIminor),CHLOvulga),MESOvirid),((NICOtabac,MARCpolym),PINUthunb)); 1128.5
(CYANparad,(((NEPHoliva,CHLOvulga),PEDIminor),MESOvirid),((NICOtabac,MARCpolym),PINUthunb)); 1133.9
(CYANparad,(((NEPHoliva,PEDIminor),CHLOvulga),MESOvirid),(NICOtabac,(MARCpolym,PINUthunb))); 1135.7
(CYANparad,(((NEPHoliva,CHLOvulga),PEDIminor),MESOvirid),(NICOtabac,(MARCpolym,PINUthunb))); 1141.0
((CYANparad,PEDIminor),(NEPHoliva,CHLOvulga),(((NICOtabac,PINUthunb),MARCpolym),MESOvirid)); 1141.9
(CYANparad,((NEPHoliva,PEDIminor),CHLOvulga),(((NICOtabac,MARCpolym),PINUthunb),MESOvirid)); 1145.6
(CYANparad,((NEPHoliva,CHLOvulga),PEDIminor),(((NICOtabac,MARCpolym),PINUthunb),MESOvirid)); 1153.2
(CYANparad,((NEPHoliva,PEDIminor),CHLOvulga),((NICOtabac,(MARCpolym,PINUthunb)),MESOvirid)); 1154.3
(CYANparad,((NEPHoliva,CHLOvulga),PEDIminor),((NICOtabac,(MARCpolym,PINUthunb)),MESOvirid)); 1161.9
(CYANparad,((NEPHoliva,(MESOvirid,CHLOvulga)),PEDIminor),((NICOtabac,PINUthunb),MARCpolym)); 1203.8
(CYANparad,((NEPHoliva,(MESOvirid,PEDIminor)),CHLOvulga),((NICOtabac,PINUthunb),MARCpolym)); 1204.9
(CYANparad,NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),((MESOvirid,CHLOvulga),PEDIminor))); 1209.0
((CYANparad,((MESOvirid,CHLOvulga),PEDIminor)),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 1209.9
((CYANparad,((MESOvirid,PEDIminor),CHLOvulga)),NEPHoliva,((NICOtabac,PINUthunb),MARCpolym)); 1214.8
(CYANparad,NEPHoliva,(((NICOtabac,PINUthunb),MARCpolym),((MESOvirid,PEDIminor),CHLOvulga))); 1216.5
(CYANparad,((NEPHoliva,CHLOvulga),(MESOvirid,PEDIminor)),((NICOtabac,PINUthunb),MARCpolym)); 1238.6
(CYANparad,((NEPHoliva,PEDIminor),(MESOvirid,CHLOvulga)),((NICOtabac,PINUthunb),MARCpolym)); 1238.7
```

D.5 Évaluation du likelihood et des valeurs de RELL bootstrap

protmlcp 2.3b3 mtREV24-F 105 trees 8 OTUs 10831 sites.

Tree	ln L	Diff ln L	S.E.	#Para	AIC	Diff AIC	TBL	RELL-BP
1	-94733.6	0.0	<-best	32	189531.2	0.0	ME	0.9775
2	-94796.2	-62.6	25.7	32	189656.4	125.2	0.3	0.0064
3	-94789.0	-55.4	26.6	32	189641.9	110.7	0.3	0.0161
4	-94932.7	-199.0	32.5	32	189929.3	398.1	1.1	0.0000
5	-94877.4	-143.8	36.7	32	189818.8	287.6	0.6	0.0000
6	-95026.4	-292.8	45.0	32	190116.9	585.7	1.6	0.0000
7	-94932.6	-199.0	47.0	32	189929.2	398.0	0.9	0.0000
8	-94988.8	-255.2	43.2	32	190041.5	510.3	1.3	0.0000
9	-94994.8	-261.2	47.2	32	190053.7	522.4	1.3	0.0000
10	-95078.9	-345.3	46.0	32	190221.8	690.5	2.0	0.0000
11	-95058.3	-324.7	45.0	32	190180.6	649.4	2.0	0.0000
12	-94997.0	-263.4	46.5	32	190058.0	526.8	1.2	0.0000
13	-94969.9	-236.3	48.8	32	190003.8	472.6	1.2	0.0000
14	-94991.6	-258.0	48.3	32	190047.1	515.9	1.4	0.0000
15	-94985.7	-252.1	50.7	32	190035.4	504.2	1.3	0.0000
16	-95168.0	-434.4	43.2	32	190400.1	868.9	1.8	0.0000
17	-95167.8	-434.2	43.2	32	190399.6	868.4	1.8	0.0000
18	-95234.4	-500.8	49.9	32	190532.9	1001.7	2.2	0.0000
19	-95231.0	-497.4	50.1	32	190525.9	994.7	2.1	0.0000
20	-95225.0	-491.4	50.4	32	190514.1	982.9	2.2	0.0000
21	-95222.4	-488.8	50.4	32	190508.8	977.6	2.1	0.0000
22	-95361.6	-628.0	52.5	32	190787.3	1256.1	2.6	0.0000
23	-95356.7	-623.0	52.7	32	190777.3	1246.1	2.5	0.0000
24	-95346.1	-612.5	53.6	32	190756.2	1225.0	2.8	0.0000
25	-95352.5	-618.9	53.4	32	190769.0	1237.7	2.8	0.0000
26	-95312.0	-578.3	55.9	32	190687.9	1156.7	2.5	0.0000
27	-95313.8	-580.2	56.0	32	190691.6	1160.4	2.5	0.0000
28	-95460.9	-727.3	62.4	32	190985.8	1454.6	3.4	0.0000
29	-95363.3	-629.7	63.8	32	190790.7	1259.5	2.8	0.0000
30	-95460.2	-726.5	62.4	32	190984.3	1453.1	3.3	0.0000
31	-95362.9	-629.3	63.7	32	190789.9	1258.7	2.7	0.0000

32	-95427.0	-693.4	61.0	32	190918.0	1386.8	3.3	0.0000
33	-95426.5	-692.9	60.9	32	190917.0	1385.7	3.2	0.0000
34	-95436.7	-703.1	58.4	32	190937.4	1406.2	3.0	0.0000
35	-95439.3	-705.7	63.7	32	190942.6	1411.3	3.3	0.0000
36	-95429.1	-695.5	58.8	32	190922.3	1391.1	2.9	0.0000
37	-95437.0	-703.4	63.8	32	190938.0	1406.8	3.2	0.0000
38	-95413.2	-679.6	58.9	32	190890.3	1359.1	2.9	0.0000
39	-95408.7	-675.1	59.3	32	190881.3	1350.1	2.8	0.0000
40	-95514.2	-780.6	63.2	32	191092.5	1561.2	3.9	0.0000
41	-95514.4	-780.8	63.3	32	191092.8	1561.6	3.8	0.0000
42	-95473.0	-739.4	62.4	32	191010.1	1478.8	3.7	0.0000
43	-95440.2	-706.6	63.0	32	190944.3	1413.1	3.1	0.0000
44	-95478.4	-744.8	62.2	32	191020.7	1489.5	3.7	0.0000
45	-95439.0	-705.4	63.0	32	190942.1	1410.8	3.0	0.0000
46	-95418.8	-685.2	64.6	32	190901.6	1370.4	3.2	0.0000
47	-95416.8	-683.2	64.7	32	190897.5	1366.3	3.1	0.0000
48	-95405.2	-671.6	64.7	32	190874.4	1343.2	3.0	0.0000
49	-95410.8	-677.2	64.5	32	190885.6	1354.4	3.1	0.0000
50	-95426.2	-692.6	66.4	32	190916.4	1385.1	3.2	0.0000
51	-95426.3	-692.7	66.3	32	190916.6	1385.3	3.1	0.0000
52	-95616.2	-882.6	76.1	32	191296.4	1765.2	3.8	0.0000
53	-95623.1	-889.5	76.0	32	191310.2	1778.9	3.9	0.0000
54	-95642.3	-908.7	70.2	32	191348.6	1817.4	3.8	0.0000
55	-95743.4	-1009.7	82.0	32	191550.7	2019.5	4.6	0.0000
56	-95820.1	-1086.5	76.6	32	191704.1	2172.9	4.8	0.0000
57	-95751.9	-1018.3	67.8	32	191567.8	2036.5	4.3	0.0000
58	-95809.1	-1075.5	75.6	32	191682.2	2151.0	4.9	0.0000
59	-95614.9	-881.3	71.6	32	191293.8	1762.6	3.4	0.0000
60	-95752.3	-1018.7	81.7	32	191568.7	2037.5	4.7	0.0000
61	-95722.8	-989.2	83.3	32	191509.5	1978.3	4.4	0.0000
62	-95816.6	-1083.0	80.6	32	191697.2	2166.0	4.8	0.0000
63	-95728.7	-995.1	83.2	32	191521.4	1990.2	4.5	0.0000
64	-95744.3	-1010.7	68.4	32	191552.6	2021.4	4.1	0.0000
65	-95792.6	-1059.0	70.0	32	191649.2	2118.0	4.7	0.0000
66	-95806.3	-1072.7	67.9	32	191676.7	2145.5	4.6	0.0000
67	-95774.4	-1040.8	71.2	32	191612.8	2081.6	4.2	0.0000
68	-95802.3	-1068.7	68.0	32	191668.5	2137.3	4.5	0.0000
69	-95804.1	-1070.5	68.1	32	191672.3	2141.0	4.5	0.0000
70	-95800.9	-1067.3	68.1	32	191665.8	2134.6	4.4	0.0000
71	-95788.7	-1055.1	81.9	32	191641.4	2110.2	4.4	0.0000

72	-95790.9	-1057.3	70.7	32	191645.8	2114.6	4.4	0.0000
73	-95745.3	-1011.7	72.1	32	191554.6	2023.4	4.1	0.0000
74	-95776.1	-1042.5	81.9	32	191616.3	2085.0	4.5	0.0000
75	-95868.7	-1135.1	77.0	32	191801.4	2270.2	5.1	0.0000
76	-95863.6	-1130.0	70.9	32	191791.2	2260.0	5.1	0.0000
77	-95851.9	-1118.3	78.6	32	191767.8	2236.6	4.6	0.0000
78	-95802.4	-1068.8	74.8	32	191668.7	2137.5	4.5	0.0000
79	-95865.1	-1131.5	77.3	32	191794.2	2263.0	5.2	0.0000
80	-95846.6	-1112.9	79.1	32	191757.1	2225.9	4.5	0.0000
81	-95804.8	-1071.2	74.5	32	191673.6	2142.4	4.6	0.0000
82	-95751.0	-1017.4	83.1	32	191566.0	2034.8	4.2	0.0000
83	-95929.7	-1196.1	78.6	32	191923.5	2392.3	5.0	0.0000
84	-95924.4	-1190.8	78.1	32	191912.7	2381.5	5.3	0.0000
85	-95830.1	-1096.5	72.4	32	191724.2	2193.0	4.7	0.0000
86	-95936.1	-1202.5	78.4	32	191936.1	2404.9	5.4	0.0000
87	-95937.8	-1204.2	78.4	32	191939.5	2408.3	5.3	0.0000
88	-95777.9	-1044.3	76.5	32	191619.9	2088.7	4.3	0.0000
89	-95883.0	-1149.3	72.5	32	191829.9	2298.7	4.9	0.0000
90	-95875.9	-1142.3	72.9	32	191815.8	2284.6	4.8	0.0000
91	-95878.4	-1144.8	72.7	32	191820.8	2289.6	4.8	0.0000
92	-95872.2	-1138.6	73.0	32	191808.4	2277.2	4.7	0.0000
93	-95786.6	-1053.0	77.6	32	191637.1	2105.9	4.3	0.0000
94	-95857.9	-1124.3	72.8	32	191779.9	2248.7	4.8	0.0000
95	-95853.7	-1120.1	73.2	32	191771.4	2240.2	4.7	0.0000
96	-95854.3	-1120.7	73.0	32	191772.6	2241.4	4.7	0.0000
97	-95850.7	-1117.1	73.3	32	191765.4	2234.1	4.6	0.0000
98	-96033.5	-1299.8	79.2	32	192130.9	2599.7	5.9	0.0000
99	-96030.9	-1297.3	79.5	32	192125.7	2594.5	5.9	0.0000
100	-96032.1	-1298.5	82.3	32	192128.2	2597.0	6.1	0.0000
101	-96005.4	-1271.8	82.0	32	192074.8	2543.6	6.0	0.0000
102	-96005.5	-1271.9	82.2	32	192075.0	2543.8	6.0	0.0000
103	-96031.9	-1298.3	82.5	32	192127.8	2596.6	6.0	0.0000
104	-95927.9	-1194.3	78.5	32	191919.9	2388.6	5.0	0.0000
105	-95919.6	-1186.0	77.9	32	191903.2	2371.9	5.2	0.0000

D.6 Topologies avec des valeurs de RELL bootstrap non nulles (T1, T2 et T3)

```
((CYANparad,MESOvirid),(NEPHoliva,(PEDIminor,CHLOvulga)),((NICOtabac,PINUthunb),MARCpolym));
(CYANparad,((NEPHoliva,(PEDIminor,CHLOvulga)),MESOvirid),((NICOtabac,PINUthunb),MARCpolym));
(CYANparad,(NEPHoliva,(PEDIminor,CHLOvulga)),(((NICOtabac,PINUthunb),MARCpolym),MESOvirid));
```