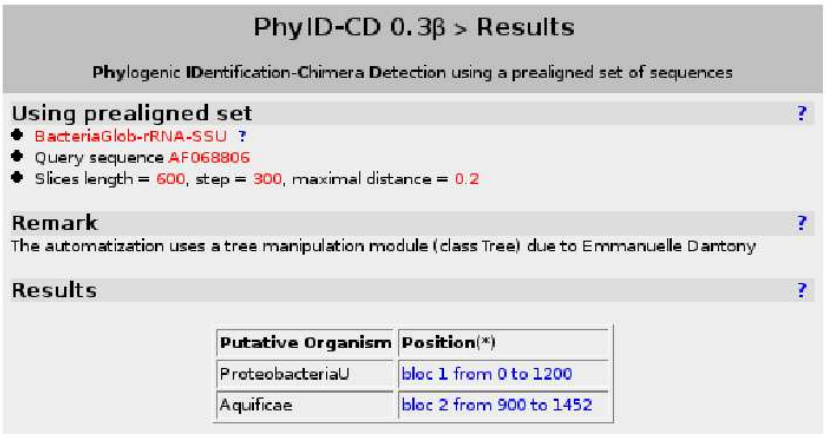
Automatisation de la détection de séquences chimériques basée sur la phylogénie


PhyID-CD 0.3β > Results

Phylogenic **ID**entification-**C**himera **D**etection using a prealigned set of sequences

Using prealigned set ?

- BacteriaGlob-rRNA-SSU ?
- Query sequence AF068806
- Slices length = 600, step = 300, maximal distance = 0.2

Remark ?

The automatization uses a tree manipulation module (class Tree) due to Emmanuelle Dantony

Results ?

Putative Organism	Position(*)
ProteobacteriaU	bloc 1 from 0 to 1200
Aquificae	bloc 2 from 900 to 1452

Anthony Cros - Master 1 aMIV
Mai 2005 - Juillet 2005

Tuteur : Pr. Jean-Pierre Flandrois

Table des matières

Remerciements

Je tiens à remercier le professeur Jean Pierre Flandrois pour m'avoir accueillit dans son laboratoire et guider durant toute la durée de mon stage.

Je remercie également Ghislaine Fardel et Catherine Pichat qui ont pris de leur temps pour me montrer et parfois me faire pratiquer certaines de leurs manipulations, afin de me faire prendre conscience des réalités techniques.

Merci aussi à Emmanuelle Dantony qui bien que stagiaire également m'a aidé à de nombreuses reprises. Merci à Ana Laura Erbino pour son point de vue de biologiste et sa disponibilité.

D'une manière générale, je remercie le laboratoire et son personnel pour un accueil chaleureux et une aide sur laquelle j'ai toujours pu compter.

Résumé

A partir d'un programme préexistant, et à l'aide d'outils créés par différentes personnes de l'UMR CNRS 5558, j'ai conçu une version automatisée du programme PhyID, logiciel permettant d'analyser des séquences nucléiques supposées chimériques. Détecter de telles séquences artefactuelles est une nécessité pour les micro-biologistes qui amplifient ou clonent l'ADN bactérien directement à partir d'écosystèmes complexes. Cette détection est un préalable à toute interprétation des séquences et dépôt dans les banques.
PhyID n'est pas le premier logiciel permettant de détecter ce type de séquence, c'est pourquoi j'ai avant tout étudié les différents programmes existants. Le webiciel final (dans le cadre de ce stage), testé et comparé à la version de départ, a montré des résultats très positifs. L'automatisation implémentée permet un gain de temps considérable par rapport à la version manuelle initiale. Cependant, il est encore possible d'optimiser le programme, différentes améliorations sont étudiées en fin de rapport.

1 Introduction

Une séquence chimérique, aussi appelée chimère, est un mélange de séquences provenant de différents organismes. Les chimères sont une conséquence de la co-amplification par PCR de gênes fortement conservés. Le phénomène se produit lorsqu'un amplicon terminé prématurément se rattache à un brin d'ADN étranger, et est ainsi copié dans les cycles suivant de la PCR. La PCR (Polymerase Chain Reaction) est une technique extrêmement répandue et utilisée tous les jours dans les laboratoires du monde entier. Le séquençage de chimères posent donc un sérieux problème cela entraîne le report dans des bases de données mondiales d'organismes n'existant pas [1][5]. Afin de pallier à ce problème, de nombreuses équipes de recherche se sont penchées sur l'élaboration de techniques permettant de reconnaître ces séquences, et un certain nombre de logiciels ayant cette vocation ont vu le jour depuis ces 10 dernières années. Notre laboratoire, l'UMR CNRS 5558, a lui-même créé un tel logiciel : PhyID.

C'est sur ce logiciel qu'a porté mon travail. Lors de mon arrivée dans le laboratoire, celui-ci ne permettait qu'une analyse "manuelle" d'une séquence douteuse. Après avoir soumis une séquence au webiciel PhyID, l'utilisateur devait activer la construction de nombreux arbres phylogéniques qui nécessitaient chacun une prise de décision de la part d'un biologiste.

Bien que, comme on le verra plus loin, cette option présente l'avantage de donner une grande place au jugement du biologiste quant aux différentes étapes de la détection, il s'est vite avèré qu'il fallait développer une version automatisée du programme. En effet la plupart des biologistes sont à la recherche d'un bon rapport qualité/temps, et le gain de temps que procure une version automatisé est considérable. Outre la recherche et l'étude des logiciels de détection de chimères préexistants, mon travail a principalement consisté à mettre en oeuvre cette automatisation, à l'aide d'un programme écrit en python dans un premier temps, puis de scripts CGI faisait appel à celui-ci dans un deuxième temps. J'ai pu tout au long de mon stage découvrir de nombreux outils, notamment le langage python et LateX grâce auquel j'ai écrit ce rapport.

2 Les logiciels de détection de chimère existants

2.1 Introduction

A l'heure actuelle, il existe 5 logiciels ayant pour objectif de cataloguer des séquences comme potentiellement chimériques ou non. Ces logiciels sont Chimera Check [3], USC Chimera Detection [4], Bellerophon [5], Ccode [6] et Pintail [7]. Tous utilisent des méthodes de calcul basées sur la similitude entre la ou les séquences à analyser et une ou des séquences prises comme référence. La discrimination suggérée repose alors sur un système de score par rapport auquel est déterminé avec plus ou moins de certitude s'il y a présence d'une chimère. Il semble que des 5, le logiciel le plus utilisé soit Chimera Check qui est également le premier à être sorti.

2.2 Chimera Check

Chimera Check est uniquement disponible sur le serveur du RDP [3]. Il n'est pas possible de télécharger le logiciel pour l'exécuter sur son propre serveur. Le webiciel est disponible à l'adresse http ://35.8.164.52/cgis/chimera.cgi ?su=SSU et est facile d'utilisation. On y trouve également un lien vers une description bien détaillée du fonctionnement.
Le principe de base du programme est le suivant : une chimère est constitué de 2 fragments ayant chacun des parents plus proches dans les bases de données que la séquence complète. En d'autres termes, si la séquence peut être coupée en 2 de telle manière que l'affiliation phylogénétique des parties ne soit pas consistante avec l'affiliation de la séquence complète, alors on peut suspecter une chimère. C'est le principe de base de la méthode du "nearest-neighbor" [4].
Le programme parcours donc la séquence en plaçant à intervalle régulier un point de séparation potentiel (que l'on appellera "breakpoint" dans le reste de ce rapport), de telle manière qu'à chaque étape on puisse poser la question : est-ce que les scores de similitude obtenus pour chaque partie sont plus élevés que le score de la séquence complète ? Cette question est traduite sous forme d'un rapport entre les deux premiers scores et le troisième. L'emplacement du breakpoint pour lequel ce rapport est le plus élevé est susceptible de correspondre à la frontière séparant 2 organismes. En dessous d'un certain seuil, ce rapport est considéré comme trop faible et ne peut permettre de conclure que l'on est en présence d'une chimère.
Les avantages de ce programme sont qu'il n'utilise aucun autre logiciel et qu'il n'effectue pas d'alignements. Il se base uniquement sur des comparaisons de fragments de séquences (entre 7 et 8 nucléotides de long selon le gêne étudié) [3]. Les scores sont donc obtenus par comparaison de ces oligomères.
Par contre, le principal défaut est qu'il ne considère qu'un seul breakpoint, ce qui fait qu'il ne prend ni en compte le cas d'un organisme inséré au milieu d'un autre (au moins 2 breakpoints), ni le cas d'une chimère constituée de plus de 2 organismes.

2.3 USC Chimera Detection

Ce programme est très proche de Chimera Check, le principe de base est le même. Seul la technique de comparaison des séquences diffère en cela qu'ils ont créé une méthode appelée "chimeric alignment" qui attribue des scores aux comparaisons de séquence par un procédé de "dynamic programming alignment" (se reporter à la référence [4] pour la description de ce procédé). ce programme utilise donc des alignements contrairement à Chimera Check.
Les tests de comparaison par rapport à Chimera Check effectués par les auteurs indiquent une performance très similaire. Ils arrivent en fait à la conclusion que de bon résultats sont obtenus en utilisant les 2 programmes car ils se complémentent bien [4].
On retrouve par contre le même défaut que Chimera Check, un seul breakpoint potentiel est

pris en compte. En outre il semble que le site du webiciel (http ://www-hto.usc.edu/software /mglobalCHI) ait été abandonné depuis un certain temps car la plupart des liens de la page sont morts, y compris celui permettant de soumettre une séquence. Il existe apparemment une version une version téléchargeable qui est peut-être opérationnelle mais n'a pas pu être installée.

2.4 Bellerophon

Le programme est disponible uniquement sous forme de webiciel à l'adresse http ://foo. maths.uq.edu.au/~ huber/bellerophon.pl. Tout comme Chimera Check, il est simple d'utilisation et n'est pas disponible sous forme de programme exportable. On trouve également un tutorial sur le site du webiciel.
Ce programme est le seul à utiliser partiellement la phylogénie. Il diffère des autres sur un point important : il ne permet pas de tester une et une seule séquence considérée comme douteuse, mais de tester un ensemble de séquences, appartenantes à la même librairie d'ADN, dans lequel on soupçonne la présence de chimères.
Le principe est le suivant : des arbres phylogéniques sont déduis de parties de l'alignement multiples des séquences analysées, et la physiologie du branchage est étudiée afin de repérer des contradictions entre les différentes parties, qui indiqueraient la présence d'une chimère. En réalité, aucun arbre n'est construit, les seuls calculs nécessaires sont des calculs de similarité entre des séquences. Une matrice complète de distances pour chaque paire de séquence est calculée pour les fragments de gauche et de droite d'un breakpoint que l'on fait avancer séquentiellement le long des séquences alignées [5]. Encore une fois on retrouve ce défaut majeur qu'est la prise en compte d'un unique breakpoint. Le principe de Bellerophon est néanmoins le plus proche dans l'idée de celui de PhyID, le programme de notre laboratoire (décrit plus tard).

2.5 Ccode

Ccode est disponible sous forme de logiciel téléchargeable à l'adresse http ://www.irnase .csic.es/users/jmgrau/index.html uniquement. Il est écrit en langage C et fonctionne théoriquement sous Linux comme sous Windows. Il n'existe pas de webiciel pour Ccode.
Le principe de base de Ccode est de comparer la séquence à analyser avec des séquences que l'ont sait authentiques. Les programmes précédents sont tous basés sur l'idée qu'une chimère montre différentes similitudes selon la partie de la séquence considérée (début ou fin), alors Ccode observe la variabilité supplémentaire que produit l'insertion d'une séquence chimérique au sein d'un ensemble de référence. En effet si une séquence authentique est introduite, on observera peu variabilité ajoutée alors que dans le cas d'une chimère, celle-ci augmentera de façon remarquable [6].
Le problème est donc qu'il faut impérativement posséder un ensemble de séquences les plus proches possible de la séquence à analyser, ce qui n'est pas toujours évident. Les auteurs de ce programme (voir référence) travaillent actuellement sur un nouveau programme, Blent, une évolution de Ccode permettant d'automatiser le processus de recherche de l'ensemble de référence.

2.6 Pintail

Le plus récent de tous, Pintail est disponible à l'adresse http ://www.cf.ac.uk/biosi/research/biosoft/Pintail/pintail.html et tout comme Ccode, il n'est disponible que sous forme de programme téléchargeable. Il est écrit en Java ce qui présente l'énorme avantage de fonctionner sur n'importe quel système d'exploitation. Un tutorial complet est disponible sur la page de téléchargement.

Le principe de Pintail diffère de celui des logiciels vus précédemment car pour analyser une séquence, une seule autre séquence est prise comme référence. Cela implique qu'il faut être certain de l'authenticité de cette séquence de référence.
Pintail détecte les chimères en comparant les distances en terme d'évolution entre les 2 séquences, progressant séquentiellement le long de la séquence douteuse à l'aide d'un cadre de taille fixe. Lorsque sur une partie de la séquence analysée l'amplitude de déviation évolutive dépasse celle que l'on observe habituellement entre 2 séquences proches, on peut soupçonner la présence d'une chimère.

2.7 Récapitulatif

Ce tableau récapitule certaines informations concernant les logiciels étudiés plus haut.

	Chimera Check	**USC Chimera Detection**	**Bellerophon**	**Ccode**	**Pintail**
Origine	U.S.A. (Michigan)	U.S.A. (California)	AUSTRALIE	ESPAGNE	U.K.
Date de sortie	1995	1997	Janvier 2003	Septembre 2004	Mars 2005
Maniabilité du webiciel	bonne	liens morts	bonne	- *	-
Utilise des alignements	non	oui	oui	oui	oui
Utilise la phylogénie	non	non	oui	non	non
Considère plus d'un breakpoint potentiel	non	non	non	oui	oui
Fournit un graphe permettant l'analyse des résultats	oui	oui	non	non	oui
Permet d'analyser plusieurs séquences en même temps	non	non	oui	non	non
Resultats envoyés par e-mail	non	oui	oui	-	-

* : un tiret indique qu'il n'y a pas de webiciel pour le programme en question.

2.8 Les limites

Ces logiciels sont plus ou moins performants, certains ont fait leurs preuves et semblent être des indicateurs relativement fiable, notamment Chimera Check. Néanmoins aucun d'entre eux

n'est basé sur les outils très performants que sont les arbres phylogéniques. C'est à partir de ce constat que Jean Pierre Flandrois a eu l'idée d'adapter l'un de ces programme existant, T4Bi [10] (outil de correction des bases de données bactériennes), et de créer ainsi PhyID le premier logiciel de détection de chimères basé entièrement sur les arbres phylogéniques.

3 PhyID

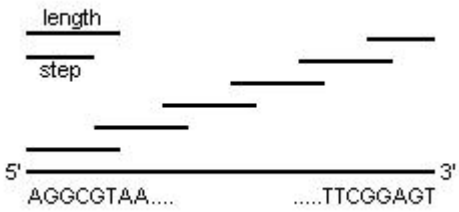
3.1 Présentation

PhyID est un outil de détection de chimères écrit en langage python. L'interface web utilise des scripts CGI faisant appel au programme après avoir récupéré les paramètres voulus par l'utilisateur. Il est disponible sur http ://umr5558-sud-str1.univ-lyon1.fr/phyidcd/phyid01.cgi mais n'est pas encore publié et ne se trouve donc pas sur le site de PBIL[9]. Seuls les utilisateurs munis d'un identifiant avec mot de passe peuvent s'en servir pour le moment.
Cet outil est le résultat du travail de nombreuses personnes : Stéphane Vellay, Emmanuelle Dantony, Ana Laura Erbino, Sophie Mignard, Jean Pierre Flandrois et moi-même.

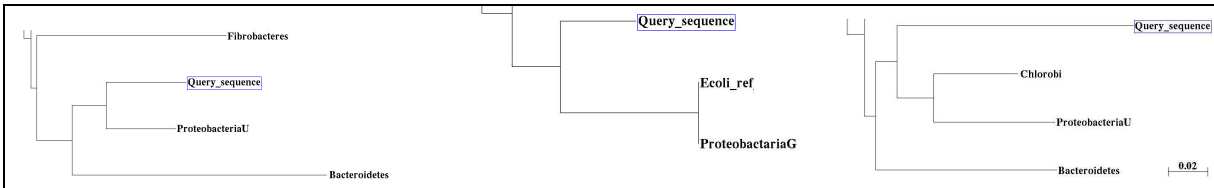
3.2 Le principe de la version de détection automatique

Contrairement aux logiciels cités et décrits dans la 3ème partie, PhyID utilise la phylogénie pour détecter les séquences chimériques.

La séquence à analyser est découpée en fragments se superposant aux extrémités :

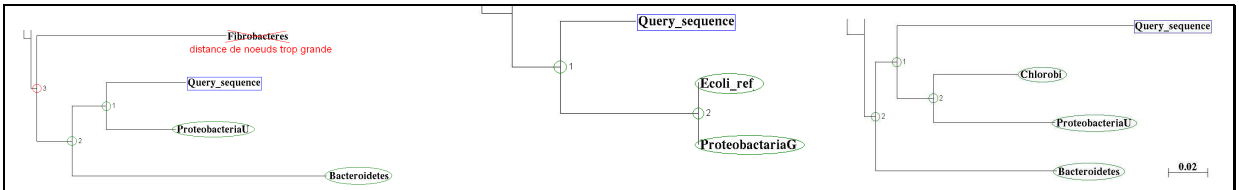


Les fragments sont ensuite stockés dans des fichiers séparés. Chaque fragment est alors aligné par profile avec l'ensemble des séquences de référence (en utilisant MUSCLE[11]) et un arbre phylogénique est construit (en utilisant CLUSTALW[12]). On obtient donc un arbre pour chaque fragment. Exemples d'arbres partiels :



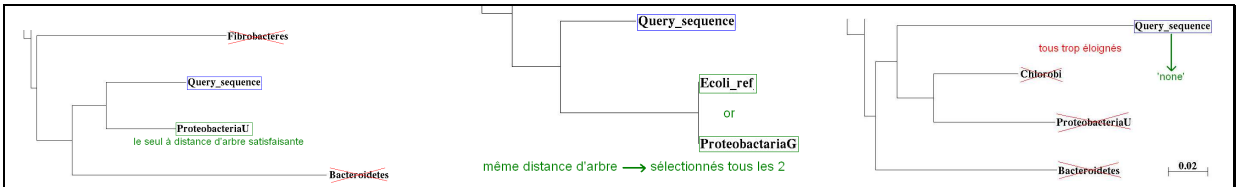
cas 1 : association facile cas 2 : multiples associations cas 3 : association impossible

Pour chaque arbre (associé à un fragment), une liste des voisins (en terme de noeuds) est créée, c'est à dire que les séquences se situant à n noeuds ou moins du fragment sont stockés dans une liste, n étant un paramètre modifiable (fixé à 2 dans tous les tests effectués). Exemples :



cas 1 : association facile cas 2 : multiples associations cas 3 : association impossible

Le candidat le plus proche (cette fois-ci en terme de distance d'arbre) est sélectionné s'il ne se situe pas au delà d'un certain seuil. Plusieurs candidats peuvent être sélectionnés dans le cas improbable mais pas impossible d'équidistances des séquences par rapport au fragment considéré. Exemples :



cas 1 : association facile cas 2 : multiples associations cas 3 : association impossible

Dorénavant lorsque nous parlerons du candidat ou d'organisme associé, il sera implicite qu'il peut occasionnellement en avoir plus d'un.

On obtient ainsi une liste associant soit un candidat soit 'none' (si aucun ne remplit tous les critères) à chaque fragment de la séquence de départ. Les fragments consécutifs s'étant vus attribuer des organismes identiques sont regroupés en blocs. Un bloc est donc constitué de la concaténation de un ou plusieurs fragments. On peut alors aisément repérer la présence de deux ou plusieurs organismes dans la liste des blocs, et ainsi déduire qu'on est en présence d'une chimère. Exemple trivial de liste résultat :

fragment 1	–>	'none'	–>	'none'	= bloc 1
fragment 2	–>	ProteobacteriaU			
fragment 3	–>	ProteobacteriaU			
fragment 4	–>	ProteobacteriaU	–>	ProteobacteriaU	= bloc 2
fragment 5	–>	ProteobacteriaU			
fragment 6	–>	ProteobacteriaU			
fragment 7	–>	'none'	–>	'none'	= bloc 3
fragment 8	–>	Fibrobacteres			
fragment 9	–>	Fibrobacteres	–>	Fibrobacteres	= bloc 4
fragment 10	–>	Fibrobacteres			
fragment 11	–>	'none'	–>	'none'	= bloc 5

3.3 Répartition des tâches

- Stéphane Vellay : conception de la trame CGI (y compris le style) et du module de découpage de séquence.
- Emmanuelle Dantony : conception d'une classe python de lecture d'arbre, qui permet également de calculer les distances entre 2 séquences d'un arbre (module utilisé initialement par T4Bi[10]).
- Ana Laura Erbino, Sophie Mignard et Jean Pierre Flandrois : élaboration des différentes bases

de données servant d'ensembles de références pour l'analyse.
- Jean Pierre Flandrois : idée originale, création d'un module permettant de traiter des séquences avec MUSCLE [11] dont le nom n'est pas limité en taille, d'une classe python de manipulation de séquences et d'un module d'interaction avec BIBI [8].
- Ana Laura Erbino : nombreux tests avec différents découpages et différents ensembles de référence.
- moi-même : adaptation, utilisation et amélioration des différents modules et scripts CGI cités plus haut, création du module permettant la détection automatique (c.f. Introduction pour plus de détails).

3.4 Le programme en python

Ce programme ne gère que la reconnaissance automatique des fragments. Il est important de noter que la version console du programme est principalement destinée à un usage via des scripts, par conséquent les sorties sur la console et sous forme de fichier servent surtout pour les tests. L'usage pourrait en être simplifié mais il ne s'agit pas ici du but de ce stage.
Le principal avantage de ce programme est qu'il est très facilement exportable. En effet une équipe souhaitant utiliser PhyID sur ses propres machines, et non via notre serveur, peut aisément récupérer le programme. Celui-ci étant intégralement écrit en python peut être exécuté sur toute machine en étant munie.
L'appel depuis une console linux se fait ainsi :

```
python phyid.py reference_sequences_fasta_file query_fasta_file [-outputs]
[-cgi_output cgi_output_path] [-no_slicing] [-l length] [-s step]
[-d maximum_tree_distance] [-list] [-exact] [-complement]
```

Voici le détail des paramètres d'entrée :

paramètres indispensables :
- **reference_sequences_fasta_file** : fichier contenant un ensemble de séquences pré-alignées dites "de référence" au format fasta.
- **query_fasta_file** : fichier contenant la séquence à analyser, en l'occurrence une chimère putative au format fasta également. Ce fichier doit être placé dans le dossier du programme.

paramètres optionnels :
- **[-outputs]** : le programme écrit les résultats dans un fichier en plus de l'affichage à la console (le fichier est créé dans le répertoire du programme).
- **[-cgi_output cgi_output_path]** : le programme écrit les résultats dans un fichier aisément lisible par un script CGI et le place dans le répertoire indiqué par cgi_output_path.
- **[-no_slicing]** : la séquence est analysée en un seul fragment. Dans ce cas les options [-l length] et [-s step] sont ignorées même si spécifiées. Cette option correspond à l'identification automatisée.
- **[-l length]** : indique la taille des fragments (par défaut 180).
- **[-s step]** : indique le pas, soit le décalage entre chaque début de fragment (par défaut 80).
- **[-d maximum_tree_distance]** : indique la distance d'arbre maximale en dessus de laquelle on ne peut plus associer un fragment à un organisme avec confiance (par défaut 0.2).
- **[-list]** : dans le cas où deux séquences d'organismes sont équidistantes du fragment analysé, cette distance étant en dessous du seuil fixé par maximum_tree_distance, -list permet

d'afficher la liste des ex-æquo (par défaut le premier dans l'ordre alphabétique de la liste est affiché).
- **[-exact]** : donne la taille exacte des blocs finaux (on rappelle que les blocs sont des concaténations de un ou plusieurs fragments). Ainsi présentés, les blocs se superposent de length-step aux extrémités (par défaut l'affichage tronque la position de la fin du fragment de manière à former une partition).
- **[-complement]** : complémente la séquence avant de lancer l'analyse.

Voici un exemple d'appel :

```
python phyid.py BacteriaGlob-rRNA-SSU.fst putative_chimera.fst -outputs
-cgi_output /home/user/Desktop -l 250 -s 100 -d 0.15 -list
```

Cet appel lance le programme PhyID qui analyse alors la séquence contenue dans le fichier putative_chimera.fst à l'aide de l'ensemble des séquences de référence contenu dans le fichier BacteriaGlob-rRNA-SSU.fst. Il affiche les résultats sur la console (ceci n'est pas une option), écrit les résultats de l'analyse de manière lisible dans un fichier créé dans le dossier du programme, il écrit ces mêmes résultats dans un fichier facilement récupérable par un script CGI et le place dans /home/user/Desktop. Le découpage se fait par tranches de 250 paires de bases avec un pas de 100. Pour la reconnaissance automatique, 0.15 est le seuil de distance d'arbre entre le fragment de séquence analysée et la séquence d'un organisme au delà duquel aucune association n'est possible. Enfin, l'option -list affiche la liste des candidats ex-æquo lorsqu'il y en a :

```
tony@Tony:~/Desktop/internship/mon_code$ python phyid.py BacteriaGlob-rRNA-SSU.fst putative_chimera.fst -l 250 -s 100 -list

-----------------------------------------------------------------------------------------------------------------
PhyID :
-----------------------------------------------------------------------------------------------------------------

processing...


Choices (regular display) :
        sequence        AB015584_1482_residues~0000~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0100~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0200~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0300~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0400~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0500~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0600~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0700~1482        :       ProteobacteriaA
        sequence        AB015584_1482_residues~0800~1482        :       ProteobacteriaU
        sequence        AB015584_1482_residues~0900~1482        :       Actinobacteria
        sequence        AB015584_1482_residues~1000~1482        :       Actinobacteria
        sequence        AB015584_1482_residues~1100~1482        :       Actinobacteria
        sequence        AB015584_1482_residues~1200~1482        :       Acidobacteria
        sequence        AB015584_1482_residues~1300~1482        :       Actinobacteria

Choices (summary display) :
        from    0       to      850     :       ProteobacteriaU
        from    700     to      950     :       ProteobacteriaA
        from    800     to      1050    :       ProteobacteriaU
        from    900     to      1350    :       Actinobacteria
        from    1200    to      1450    :       Acidobacteria
        from    1300    to      1482    :       Actinobacteria

tony@Tony:~/Desktop/internship/mon_code$
```

FIG. 1 – Exemple de sortie console

3.5 Le Webiciel

Contrairement au programme console, la version webiciel de PhyID possède une interface utilisateur intuitive :

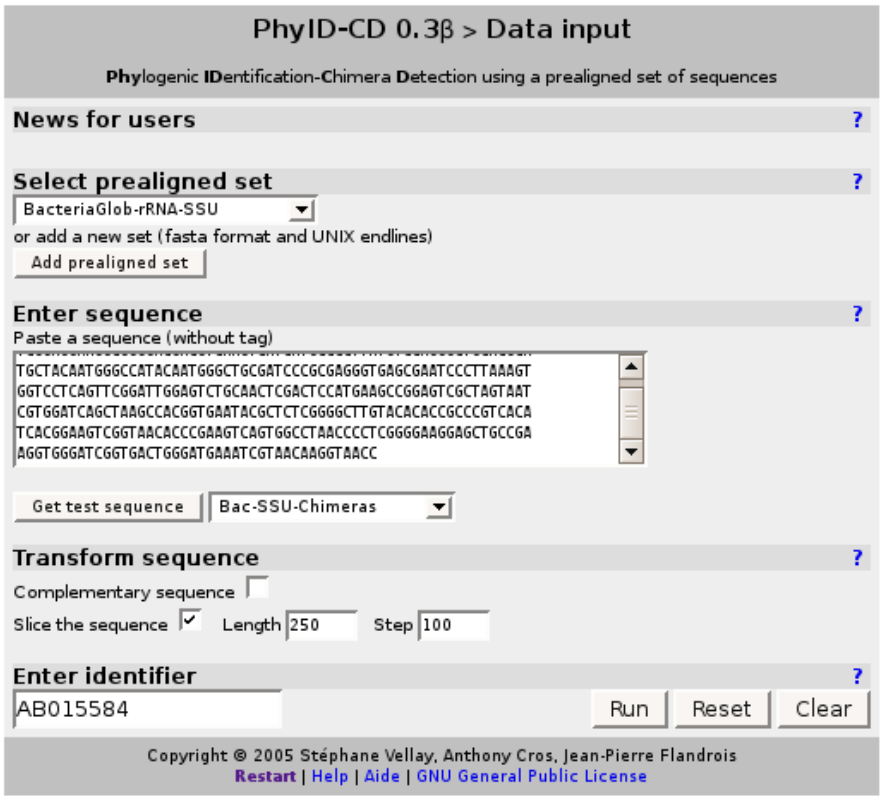


FIG. 2 – Etape 1 - Détection automatique (phyid01.cgi)

Dans un premier temps, on copie une séquence à tester dans le cadre "Enter sequence". A noter que la séquence doit être au format STADEN, c'est à dire qu'elle ne doit pas être précédée de tag. Puis, dans le cadre "Transform sequence" on entre les paramètres de découpage voulus (options [-l length] et [-s step] du programme), ou rien si l'on ne veut pas découper la séquence (option [no_slicing] du programme). Ne pas découper la séquence revient à l'identifier. L'utilisateur peut également demander au programme de complémenter la séquence avant de l'analyser (option [-complement] du programme). Enfin, dans le dernier cadre il est possible, bien que dispensable, de préciser un nom pour la séquence à analyser. Dans ce dernier cadre se situe le bouton "Run" qui permet de passer à l'étape suivante :

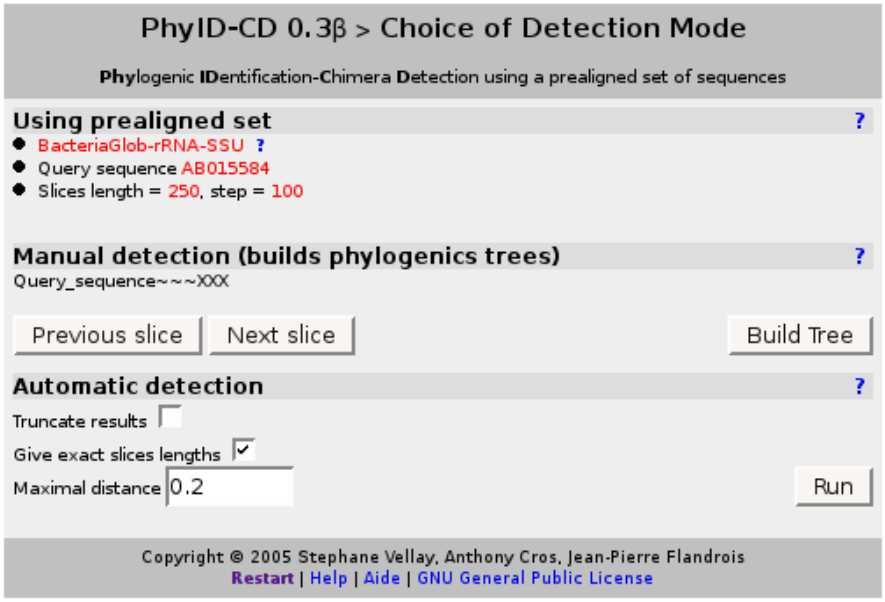


FIG. 3 – Etape 2 - Détection automatique (phyid02AR.cgi)

C'est à cet endroit que l'utilisateur peut choisir d'utiliser la reconnaissance manuelle ou automatique. La deuxième se fait très rapidement (voir section Résultats plus bas) simplement en cliquant sur le bouton "Run" dans le cadre "Automatic detection". La première bien que beaucoup plus longue (proportionnelle au nombre de fragments) permet à un utilisateur expérimenté d'associer ou non un fragment à un organisme selon son propre jugement, à l'aide d'un arbre phylogénique apparaissant à l'écran pour chaque fragment.
Mon travail a porté sur la partie de reconnaissance automatique.

Dans le cadre "Automatic recognition" il est possible de d'activer/désactiver les options [-exact], [-list] et [-d maximum_tree_distance] du programme décrites plus haut.
Cocher la case "Truncate results" désactive l'option [-list] (par défaut la case est décochée).
Cocher la case "Give exact slices lengths" active l'option [-exact] (par défaut la case est cochée).
Il est possible de préciser la valeur de maximum_tree_distance (par défaut celle-ci vaut 0.2)
Cliquer sur le bouton "Run" de ce cadre lance l'analyse automatique.

Le temps d'attente est d'autant plus long que :
- la taille de la séquence à analyser est grande
- le nombre de séquences dans l'ensemble de référence est grand
- la séquence analysée diffère des séquences de l'ensemble de référence

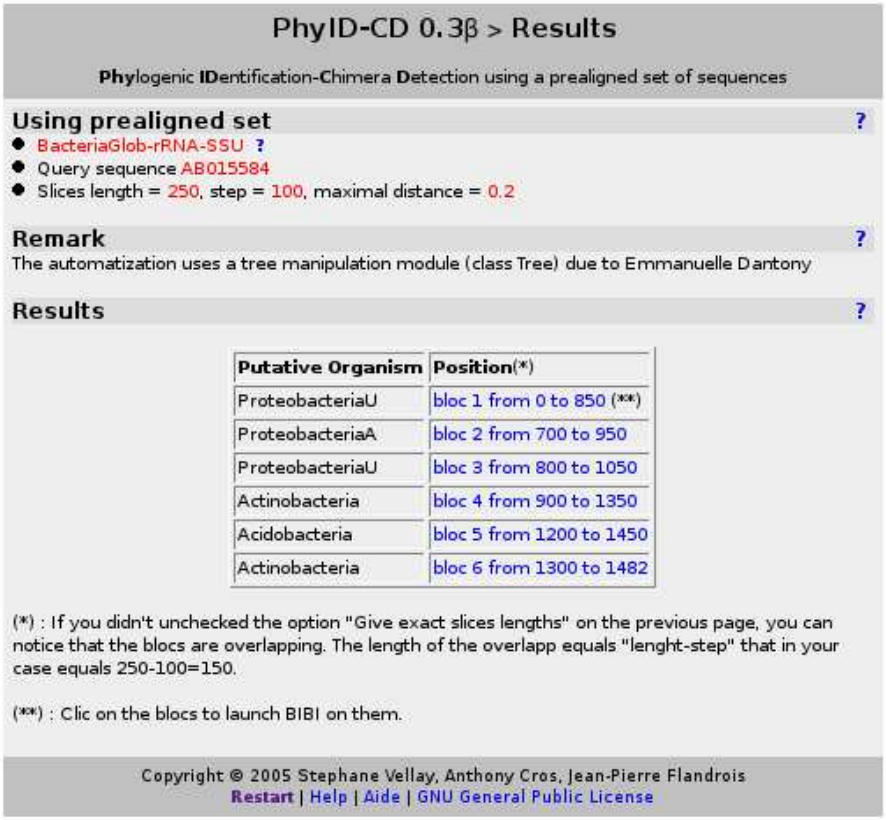


FIG. 4 – Etape 3 - Détection automatique (phyid03AR.cgi)

La page de résultats comporte les informations sur l'analyse, à savoir le nom de l'ensemble de référence utilisé, le nom de la séquence analysée si précisé dans la première page ainsi que les paramètres de découpage et la valeur de la limite. Plus bas est affiché un tableau présentant d'un côté le nom d'un organisme ou 'none', et de l'autre côté les positions de début et de fin des blocs qui y sont associés.
Il est possible de lancer BIBI[8] sur chaque bloc en cliquant sur son nom. La page suivante permet de vérifier que les données à envoyer à BIBI[8] sont correctes et cliquer sur le bouton "Run BiBi on PBIL now" envoie les données (c.f. la documentation de BIBI[8] pour en connaître le fonctionnement). Cette vérification permet d'évaluer la qualité de l'association d'un bloc à un organisme :

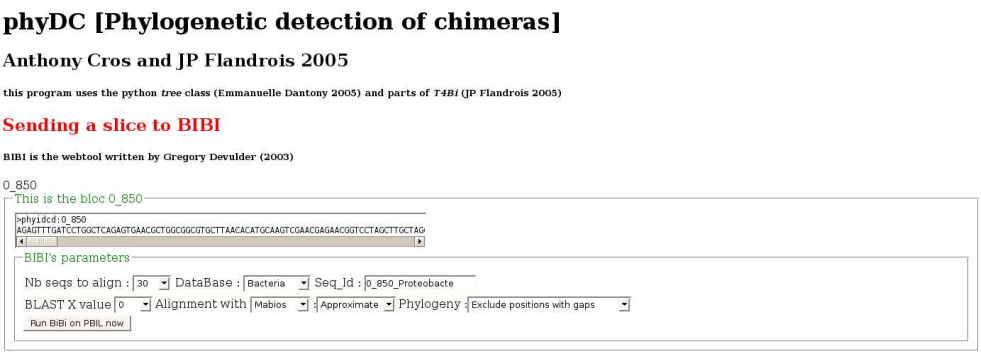


FIG. 5 – Verification optionelle (bibi_call.cgi)

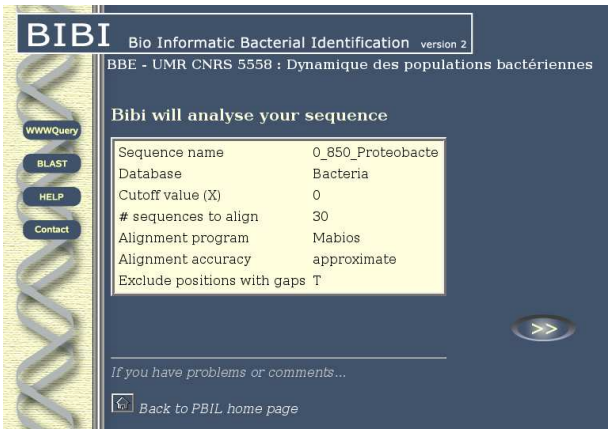


FIG. 6 – Verification optionelle (script propre à BIBI)

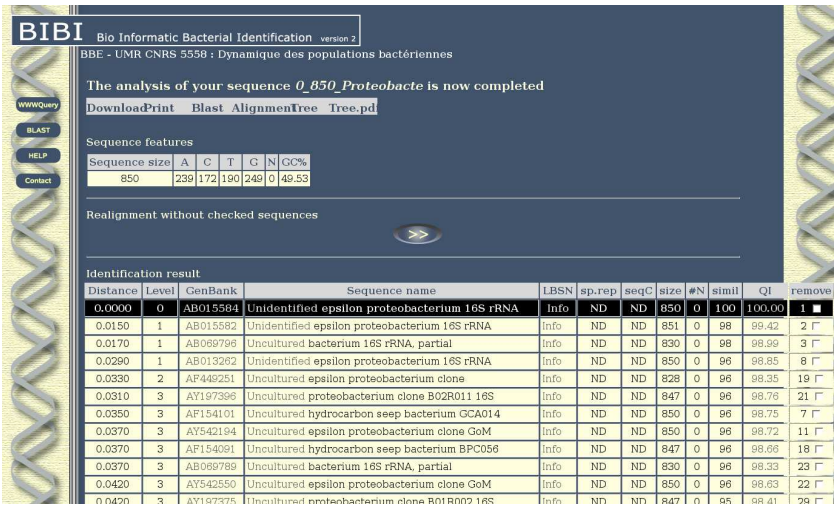


FIG. 7 – Verification optionelle (script propre à BIBI)

4 Résultats

Les résultat obtenus sont très satisfaisants. En effet tous les tests effectués parallèlement en automatique et en manuel se sont avérés positifs. Si l'on a trouvé parfois quelques différences entre les 2 types de résultats, ceux-ci étaient minimes et dû à des nuances dans l'appréciation du biologiste par rapport au programme.
Voici quelques exemples de tests effectués par Ana Laura Erbino et moi-même :

Test 1 :
séquence AB015584 présumée être une chimère de Proteobacteria Epsilon (désigné par ProteobacteriaU ici) et d'**Actinobacteria** [2]
Résultats de PhyID avec length = 250 et step = 100 :

	Détection manuelle	**Détection automatique**
Temps nécessaire	1 min 50 secs	19 secs
Résultats	Query = ProteobacteriaU Slice 1 = ProteobacteriaU Slice 2 = ProteobacteriaU Slice 3 = ProteobacteriaU Slice 4 = ProteobacteriaU Slice 5 = ProteobacteriaU Slice 6 = ProteobacteriaU Slice 7 = ProteobacteriaU Slice 8 = ProteobacteriaA Slice 9 = ProteobacteriaU Slice 10 = Actinobacteria Slice 11 = **Acidobacteria** Slice 12 = Actinobacteria Slice 13 = Acidobacteria Slice 14 = Actinobacteria	bloc 1 from 0 to 850 = ProteobacteriaU bloc 2 from 700 to 950 = ProteobacteriaA bloc 3 from 800 to 1050 = ProteobacteriaU bloc 4 from 900 to 1350 = Actinobacteria bloc 5 from 1200 to 1450 = Acidobacteria bloc 2 from 1300 to 1482 = Actinobacteria

Test 2 :
séquence AB068806 présumée être une chimère de Proteobacteria Epsilon (désigné par ProteobacteriaU ici) et d'**Aquificae** [2]
Résultats de PhyID avec length = 250 et step = 100 :

	Détection manuelle	**Détection automatique**
Temps nécessaire	1 min 44 secs	21 secs
Résultats	Query = ProteobacteriaU Slice 1 = **ProteobacteriaU** Slice 2 = none Slice 3 = Chlorobi Slice 4 = ProteobacteriaU Slice 5 = ProteobacteriaU Slice 6 = ProteobacteriaU Slice 7 = ProteobacteriaU Slice 8 = ProteobacteriaU Slice 9 = ProteobacteriaU Slice 10 = Gemmatimonadetes Slice 11 = Aquificae Slice 12 = Aquificae Slice 13 = Aquificae Slice 14 = Aquificae	bloc 1 from 0 to 350 = none bloc 2 from 200 to 450 = Chlorobi bloc 3 from 300 to 1050 = ProteobacteriaU bloc 4 from 900 to 1150 = Gemmatimonadetes bloc 5 from 1000 to 1452 = Aquificae

Test 3 : séquence D83371 authentique Staphylococcus saprophyticus (Firmicutes)
Résultats de PhyID avec length = 250 et step = 100 :

	Détection manuelle	**Détection automatique**
Temps nécessaire	1 min 43 secs	25 secs
Résultats	Query = FirmicutesA Slice 1 = FirmicutesA Slice 2 = FirmicutesA Slice 3 = FirmicutesA Slice 4 = FirmicutesA Slice 5 = FirmicutesA Slice 6 = FirmicutesA Slice 7 = FirmicutesA Slice 8 = FirmicutesA Slice 9 = **none** Slice 10 = FirmicutesA Slice 11 = FirmicutesA Slice 12 = FirmicutesA Slice 13 = FirmicutesA Slice 14 = FirmicutesA	bloc 1 from 0 to 1478 = FirmicutesA

Étant donné la similitude entre les résultats fournis par les 2 méthodes et le temps que prennent chacune d'elles pour chaque test, l'efficacité de l'automatisation est satisfaisante. Par ailleurs, s'il s'avère que le biologiste a un doute quant à un résultat fournit automatiquement, il lui est tout à fait possible d'utiliser le webiciel en manuel sur la même séquence, avec les mêmes paramètres, lui permettant ainsi d'estimer la qualité de la reconnaissance.
Bien sûr, de nombreuses séquences présentes dans les grandes bases de données telles GenBank[13] restent difficile à cataloguer comme chimériques ou authentique. Diverses raisons peuvent être à l'origine de cette difficulté accrue d'authentifier les séquences :
- séquence trop vague (trop de 'N')
- chimère d'organismes très proches
- chimères constituée de nombreux petits fragments
- chimère pour laquelle un des points de séparations se trouve très proche d'une extrémité
- ...
En dehors de ces cas particuliers, la version automatique du webiciel PhyID permet une analyse rapide et pertinente des séquences douteuses. Il serait intéressant de comparer les résultats donnés par PhyID et ceux donnés par les logiciels décrits dans la partie 2 afin d'en confronter l'efficacité. Par manque de temps je n'ai malheureusement pas pu effectuer ces tests. Il reste à noter que des améliorations sont envisageable et donc que PhyID possède encore un potentiel de progression.

5 Conclusion

En conclusion, l'automatisation du programme est convaincante, elle permet à l'utilisateur un gain de temps très intéressant sans perte de qualité notable. De plus, la version initiale manuelle est toujours à disposition de l'utilisateur, lui permettant de vérifier par lui-même les résultats trouvés, ou même de mieux comprendre le fonctionnement du programme. La mise à jour créée est donc un module ajouté à la version originale et n'en diminue aucunement la capacité initiale de détection. Autrement dit, il s'agit d'un raccourci pour les utilisateurs désireux d'obtenir des résultats rapidement.
Néanmoins, si l'automatisation mise en oeuvre est satisfaisante, il n'en demeure pas moins que le principe même du programme peut et devra être amélioré. Il reste deux problèmes majeurs.
Le premier est celui de la stratégie optimale de découpage. En effet si la taille des fragments est trop petite, l'affiliation phylogénique de petits bouts étant difficile, celle-ci devient imprécise même au sein d'une séquence authentique. A l'inverse si la taille est trop grande les affiliations obtenues pour les fragments s'étendant de part et d'autre des "break points" sont aberrantes, rendant plus difficile la discrimination entre séquence chimérique et authentique.
Une idée serait d'effectuer plusieurs types de découpages en partant de fragments de grande taille et en diminuant séquentiellement la taille jusqu'à l'obtention de résultats pertinents. On pourrait également se baser sur un nombre fixe de découpages, 4 par exemples, et afficher les 4 résultats obtenus laissant ainsi l'utilisateur en tirer ses propres conclusions. On pourrait également implémenter une fonction capable de sélectionner le résultat le plus intéressant des 4 et l'afficher.
Ana Laura Erbino dans le cadre de son travail a effectué de nombreux tests afin de trouver des stratégies optimales que ce soit au niveau du découpage ou bien du choix de l'ensemble de référence. A ce jour, il semblerait que pour détecter des chimères grossières, un découpage par fragment de 600 bps de longueur et un pas de 300 serait adéquate pour les séquences de 1500 bps du 16SrRNA.
L'autre problème est que selon le type de chimère analysé, l'ensemble des séquences de référence ne peut être toujours le même. En effet le programme actuel est efficace pour la détection

de chimères grossières, c'est à dire constituées d'organismes relativement distants. Dans le cas d'une chimère d'organismes très proches, il est nécessaire de se baser sur un ensemble de référence beaucoup plus précis. Pour résoudre ce problème on pourrait effectuer plusieurs tests en commençant par l'ensemble de référence le plus général, puis en utilisant des ensembles de plus en plus précis jusqu'à l'obtention éventuelle d'un résultat satisfaisant. Les suggestions proposées pour le problème de la taille des fragments sont également applicables à ce problème, comme la création d'une fonction reconnaissant le meilleur résultat.
Il est également envisageable de cumuler les 2 idées, c'est à dire tester plusieurs ensembles de références et pour chacun, tester diffèrents découpages. Ceci ne serait bien sûr possible que via l'option de détection automatique, seule capable de produire un tel résultat en un temps raisonnable.

Références

[1] Grace C.-Y WANG and Yue WANG
Frequency of Formation of Chimeric Molecules as a Consequence of PCR Coamplification of 16S rRNA Genes from Mixed Bacterial Genomes.
(Applied and Environmental Microbiology, Dec. 1997, p. 4645-4650)

[2] Philip HUGENHOLTZ and Thomas HUBER (2003) *Chimeric 16S rDNA sequences of diverse origin are accumulating in the public databases*
(Int J Syst Evol Microbiol 53 (2003), 289-293 ; DOI 10.1099/ijs.0.02441-0)

[3] J.R. COLE, B. CHAI, T.L. MARSH, R.J. FARRIS, Q. WANG, S.A. KULAM, S. CHANDRA, D.M. McGARRELL, T.M. SCHMIDT, G.M. GARRITY, J.M. TIEDJE
The Ribosomal Database Project (RDP-II) : previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy.
(Nucleic Acids Res 2003 Jan 1 ;31(1) :442-3)

[4] George A. KOMATSOULIS and Michael S. WATERMAN.
A New Computattional Method for Detection of Chimeric 16S rRNA Artifacts Generated by PCR Amplification from Mixed Bacterial Populations.
(Applied and Environmental Microbiology, June 1997, p. 2338-2346)

[5] Thomas HUBER, Geoffrey FAULKNER and Philip HUGENHOLTZ
Bellerophon : A Program to Detect Chimeric Sequences in Multiple Sequence Alignments.
(Bioinformatics Vol. 20 no. 14 2004, pages 2317-2319)

[6] Juan M. GONZALEZ, Johannes ZIMMERMANN and Cesareo SAIZ-JIMENEZ
Evaluating putative chimeric sequences from PCR-amplified products.
(Bioinformatics Vol. 21 no. 3 2005, pages 333-337)

[7] Kevin ASHELFORD
Pintail - a program for detecting and analysing 16S rRNA chimeras.
http ://www.cf.ac.uk/biosi/research/biosoft/Pintail/pintail.html

[8] G. DEVULDER, G. PERRIERE, F. BATY and J.P. FLANDROIS (2003)
BIBI, a Bioinformatics Bacterial Identification Tool.
(J. Clin. Microbiol. 41 :(4)1785-1787.)

[9] PBIL - Pôle Bio-Informatique Lyonnais - http ://pbil.univ-lyon1.fr/ - January 1998

[10] T4Bi
J.P. FLANDROIS, S. MIGNARD, E. DANTONY, M. GOUY and G. DEVULDER *Génération et visualisation de la phylogénie des Bacteria pour l'étude des incohérences taxinomie-phylogénie*
(JOBIM 2005 LYON - July 6, 7 and 8)

[11] EDGAR, C. ROBERT (2004)
MUSCLE : multiple sequence alignment with high accuracy and high throughput.
(Nucleic Acids Research 32(5), 1792-97.)

[12] J.D. THOMPSON, D.G. HIGGINS and T.J. GIBSON
CLUSTAL W : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.
(Nucleic Acids Research, Vol 22, Issue 22 4673-4680)

[13] Dennis A. BENSON, Ilene KARSCH-MIZRACHI, David J. LIPMAN, James OSTELL and David L. WHEELER
GenBank : Update.
(Nucleic Acids Research, 2004, Vol. 32, Database issue D23-D26)