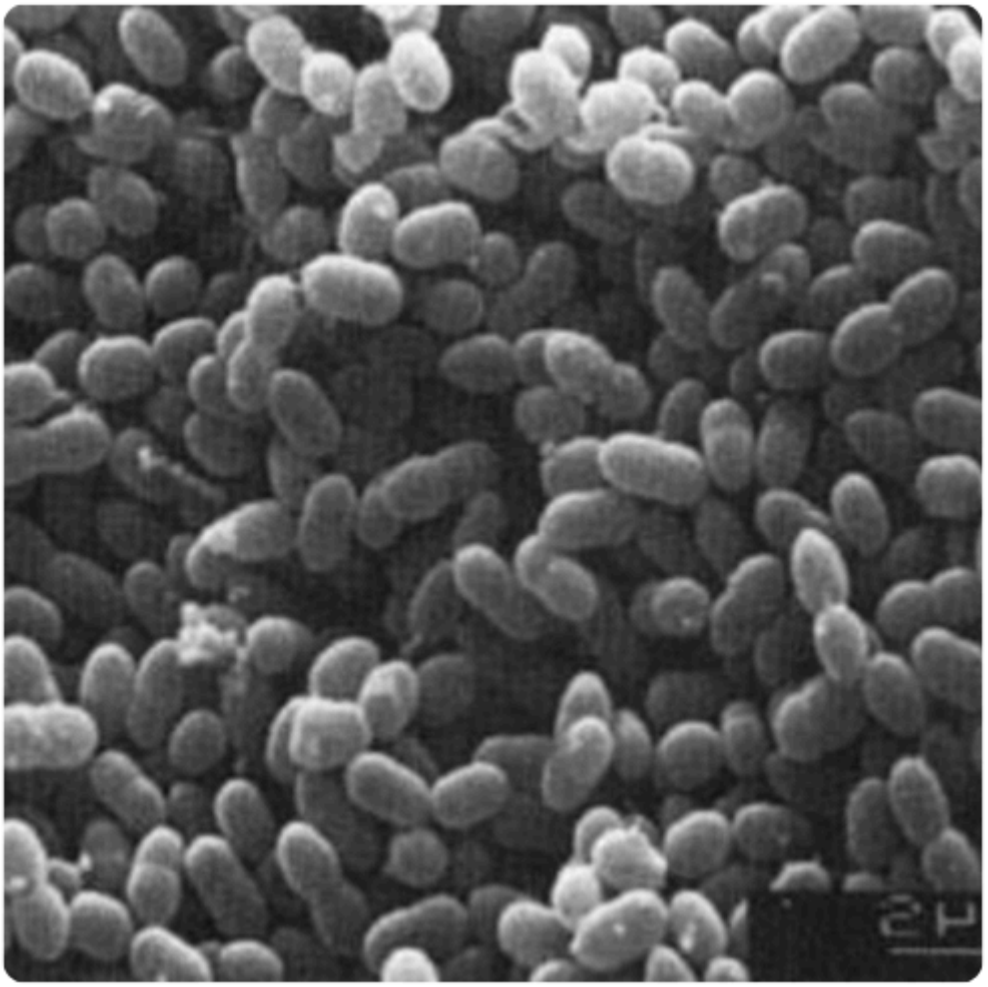
Annotation du génome de *Lactococcus lactis*

Bioinformatique pour la génomique

Rapport de projet
Effectué dans le cadre du parcours
Master 1 Bioinformatique et Biologie des Systèmes
Année universitaire 2023-2024



Morphologie en microscopie électronique de *Lactococcus lactis subsp*. (1)

Université Paul Sabatier - Toulouse III
Projet à rendre le 20/12/2024
Etudiante : RODRIGUES Camille-Astrid
Responsable UE : FICHANT Gwennaelle

RÉSUMÉ

Lactococcus lactis est une bactérie gram-positive jouant un rôle central dans les fermentations laitières et la production de levains, et constitue un modèle clé en recherche scientifique grâce à son génome séquencé. Cette étude se concentre sur la réannotation d'un fragment de son génome, en combinant des outils bioinformatiques avancés tels qu'ORFfinder, GeneMark, GeneMarkHMM, ScanForMatches, BlastP, SignalIP et DeepTMHMM. L'objectif est de caractériser les régions codantes (CDS) en identifiant les cadres de lecture ouverts (ORF), les codons d'initiation et de terminaison, ainsi que les motifs fonctionnels associés, incluant les sites de fixation du ribosome (RBS), les promoteurs sigma, et les terminateurs rho-dépendants.

Les analyses ont permis d'identifier sept protéines hypothétiques, dont deux transmembranaires, et de prédire leurs fonctions par homologie de séquences et leur localisation subcellulaire. Les résultats mettent en évidence la robustesse de l'approche multi-outil pour une caractérisation précise des gènes et de leurs produits, malgré certaines limites méthodologiques, comme l'utilisation d'algorithmes initialement optimisés pour d'autres génomes, notamment celui de *E. coli*.

Des perspectives d'amélioration incluent l'entraînement spécifique des algorithmes aux données génomiques de *L. lactis* et l'intégration de méthodes élargies pour explorer les mécanismes de terminaison alternatifs. Ces travaux apportent une contribution significative à la compréhension moléculaire de cet organisme d'intérêt industriel et scientifique, consolidant son rôle central dans la production alimentaire et la biotechnologie.

Points clés

- Lactococcus lactis est crucial pour les fermentations laitières et constitue un modèle en recherche scientifique.
- Cette étude porte sur l'annotation bioinformatique d'un fragment du génome de L. lactis.
- Les outils bioinformatiques utilisés incluent ORFfinder, GeneMark, GeneMarkHMM, ScanForMatches, BlastP, SignalIP et DeepTMHMM.
- Identification des CDS via l'analyse des ORF, des codons start/stop, et des motifs fonctionnels (RBS, sites sigma, terminateurs rho-dépendants).
- Sept protéines hypothétiques, dont deux transmembranaires, ont été identifiées et caractérisées.
- Des limites méthodologiques sont relevées, notamment l'utilisation d'algorithmes entraînés sur E. coli.
- Des améliorations futures prévoient un entraînement spécifique aux données de L. lactis et une exploration étendue des mécanismes de terminaison.
- Ces travaux enrichissent la compréhension moléculaire et appliquée de L. lactis.

Mots clés

Lactococcus lactis, bactérie gram-positive, fermentations laitières, levains, annotation génomique, bioinformatique, ORFfinder, GeneMark, GeneMarkHMM, ScanForMatches, BlastP, SignalIP, DeepTMHMM, régions codantes (CDS), ORF (Open Reading Frames), protéines hypothétiques, homologie de séquences, localisation subcellulaire, améliorations méthodologiques

INTRODUCTION

Lactococcus lactis est une bactérie gram-positive largement répandue dans les environnements laitiers, notamment dans le lait cru, où elle joue un rôle clé dans la fermentation. Utilisée depuis des siècles dans la fabrication de fromages et autres produits laitiers, cette bactérie se distingue par son métabolisme anaérobie aérotolérant et sa capacité à croître de manière optimale à une température d'environ 30 °C. Deux sous-espèces principales de *L. lactis* sont couramment employées dans la production de levains, ce qui en fait un micro-organisme d'intérêt industriel majeur.

Au-delà de son utilisation industrielle, *L. lactis* constitue également un modèle pour la recherche scientifique. Elle est la première bactérie lactique dont le génome a été entièrement séquencé, ouvrant la voie à une meilleure compréhension des mécanismes moléculaires et des fonctions biologiques associées à ce groupe de micro-organismes.

Au cours de cette étude, nous allons reproduire l'annotation d'une partie du génome de *L. lactis* (Annexe1) en utilisant des outils bioinformatiques tels qu'ORFfinder (2), GeneMark (GM) (3), GeneMarkHMM (GMH) (4) et ScanForMatches (SFM) (5). Les objectifs de cette étude incluent l'analyse des cadres de lecture ouverts (ORF) à laquelle viennent s'ajouter la recherche de codons d'initiation (codon START) et de signal de terminaison (codon STOP). L'ensemble de ces analyses nous amèneront à considérer des régions codantes (CDS). Nous y adosserons l'identification de la présence de séquences spécifiques telles que les sites de fixation du ribosome (RBS), du facteurs sigma et la présence de terminateurs rho-dépendants. Une fois les CDS identifiés, nous déterminerons la fonction des protéines issues de ces gènes par analogie de séquence (BlastP), ainsi que leur localisation subcellulaire (SignalIP et DeepTMHMM). Afin de permettre une meilleure analyse des résultats obtenus par les différents logiciels, nous développerons un programme permettant la conversion des fichiers de sortie de GM, GMH et SFM au format GFF (general feature format). Il s'agit d'un format de fichier utilisé pour décrire les gènes et d'autres éléments de séquences d'ADN, d'ARN et de protéines.

MATÉRIELS ET MÉTHODES

Quels sont les logiciels et langages utilisés pour cette étude ?

ORFfinder (Open Reading Frame Finder) (2) recherche des ORF dans la séquence d'ADN saisie, renvoyant les coordonnées de chaque ORF ainsi que la traduction de leur séquence en protéines. Ce programme est utilisé pour identifier dans un ADN nouvellement séquencé des segments potentiellement codants pour des protéines.

GeneMark (3) est un algorithme bayésien basé sur les modèles de chaînes de Markov dépendant du cadre (non homogènes) des régions codantes pour les protéines. Il est capable d'identifier le brin d'ADN codant et le cadre de lecture correct, tout en générant un signal distinctif pour localiser un gène. Pour obtenir de bons résultats, la séquence à analyser doit être extraite de la même population statistique que l'ensemble de formation. Ainsi, on ne peut pas s'attendre à ce que l'algorithme entraîné sur le génome d'*E. coli* fonctionne correctement sur les séquences provenant du génome d'autres espèces (telles que *L. lactis*).

GeneMarkHMM (4) est un algorithme utilisé pour améliorer la qualité de la prédiction des gènes en identifiant avec précision les limites des gènes. Il intègre les modèles de GM dans un cadre basé sur les modèles de Markov cachés (HMM), avec les limites des gènes modélisées comme des transitions entre états cachés. Pour affiner les prédictions des codons d'initiation de la traduction, il utilise des motifs spécifiques des sites de liaison au ribosome. GMH est significativement plus précis que GM pour la prédiction exacte des gènes.

ScanForMatches (5) est un outil en C conçu pour rechercher des motifs dans des séquences d'ADN ou de protéines au format FASTA. Il inclut des fonctionnalités pour identifier des motifs complexes, gérer des appariements non standards, et tolérer des variations comme des décalages, des insertions ou des suppressions. Les motifs sont définis par des règles et permettent des recherches basées sur des plages, des appariements inversés ou des boucles (comme les structures en épingle à cheveux). La recherche peut s'effectuer sur les deux brins. Cet algorithme permet également la recherche de répétitions, de motifs spécifiques mais aussi l'utilisation de matrices de poids position.

Python (6) est un langage de programmation créé par Guido van Rossum (1991). Le nom du langage vient de la série humoristique "Monty Python's Flying Circus". Ce langage met l'accent sur la lisibilité du code, en utilisant une syntaxe simple et minimaliste qui permet aux développeurs de se concentrer sur la logique de leur programme. Langage interprété et multiplateforme, Python supporte plusieurs paradigmes de programmation, notamment procédural, orienté objet et fonctionnel, et s'accompagne d'une riche bibliothèque standard pour répondre à divers besoins. Il permet ainsi un large éventail d'applications telles que le développement web, l'analyse des données, l'intelligence artificielle et l'automatisation.

Perl (Practical Extraction and Report Language) (7) est un langage de programmation créé par Larry Wall (1987). Ses principales caractéristiques sont sa flexibilité, sa capacité à gérer des expressions régulières puissantes, sa syntaxe concise et son interprétation dynamique. Perl est principalement conçu pour le traitement de texte et l'extraction de données. Il offre également une portabilité multiplateforme et bénéficie d'une grande quantité de modules. Il est donc largement utilisé dans l'administration système, la bioinformatique, le développement web et l'automatisation de tâches. Cependant, Perl peut être difficile à lire en raison de sa syntaxe flexible, et bien qu'il soit rapide pour certaines tâches, il peut être moins performant que des langages compilés. Bien qu'il ait perdu de sa popularité au profit de langages comme Python, il reste pertinent pour des tâches spécifiques dans des environnements Unix/Linux.

Quelles sont les étapes mises en place pour cette étude ?

Identifier les régions codantes

Nous avons d'abord identifié les **ORF** éventuels avec ORFfinder. Nous avons sélectionné le code génétique Standard et restreint l'analyse à des tailles d'ORF de 300 nucléotides. " 'ATG' et codon d'initiation alternatif" a été sélectionné. En effet, chez les procaryotes notamment, il y a différents codons start possibles.

Pour mieux définir les **CDS** présents sur notre fragment génomique d'intérêt, nous avons utilisé deux algorithmes : GM et GMH. GM nous permet de localiser les **codons start et stop** éventuels ainsi que les **régions** probablement **codantes**. GMH permet de vérifier la présence de **gènes typiques ou atypiques** (transfert horizontal) en plus des informations fournies par GM.

Les paramètres utilisés pour GM sont respectivement pour l'espèce, la taille de la fenêtre, le pas d'étude et le seuil : lactococcus lactis II1403 ; 120 ; 12 et 0,5. Une seconde étude avec GM a été menée en modifiant le seuil à 0,4. Concernant, GMH, nous avons sélectionné la même espèce que GM afin d'avoir des résultats comparables.

L'utilisation des deux algorithmes vient conforter la présence de CDS éventuels. Cependant, ces deux algorithmes ont été entraînés sur le génome de *E. coli*, il nous faut donc de nouvelles approches pour venir appuyer la présence d'un gène codant pour une protéine.

Pour cela, nous avons choisi de rechercher la présence de **RBS**, de **site de fixation du facteur sigma** et des **sites de terminaison rho-dépendant**. Ces trois analyses ont été effectuées en ligne de commande avec l'outil SFM (Annexe2) et les matrices et pattern correspondants (Annexe3).

Définir la fonction et la localisation des protéines issues des régions codantes identifiées

Une fois les CDS identifiés, nous avons récupéré les séquences protéiques à l'aide de la suite EMBOSS. Nous avons **découpé la séquence** pour en extraire les CDS avec Extractseq et **traduit les séquences** génomiques en séquences protéiques avec Transeq. Les séquences protéiques hypothétiques obtenues ont ensuite été utilisées comme query d'un BlastP. Cette étape permet d'identifier les **fonctions des protéines** par analogie de séquences.

La localisation des protéines a été étudiée par la recherche de **peptide signal** sur la protéine qui indiquerait un adressage à la membrane (protéine exportée ou sécrétée) avec l'outil SignalIP. Ensuite, une analyse menée avec DeepTMHMM a prédit la **position subcellulaire** de la protéine (intracellulaire, membranaire ou extracellulaire) ainsi que sa **structure tertiaire** venant appuyer les informations obtenues sur sa localisation subcellulaire.

RÉSULTATS

Les résultats de ORFfinder (Annexe4), GM (Annexe5), GMH (Annexe6), SFM (Annexe7), SignalIP (Annexe8) et DeepTMHMM (Annexe9) ont été retranscrits dans les tableaux ci-dessous.

ORF	***ORFfinder***					
	Label	*Strand*	*Frame*	*Start*	*Stop*	*Length (nt | aa)*
0	*/*	*/*	*/*	*/*	*/*	*/*
3	*ORF3*	*+*	*2*	*455*	*3436*	*2982 | 993*
4	*ORF4*	*+*	*3*	*3447*	*5249*	*1803 | 600*
1	*ORF1*	*+*	*1*	*5230*	*6486*	*1257 | 418*
5	*ORF5*	*+*	*3*	*6483*	*7220*	*738 | 245*
2	*ORF2*	*+*	*1*	*7261*	*9270*	*2010 | 669*
6	*/*	*/*	*/*	*/*	*/*	*/*

Tableau A : Résultats obtenus avec ORFfinder

Figure 1 Analyse des régions codantes et des protéines issues.
Identification des régions codantes par la présence de codon initiateur de la transcription, codon stop, RBS, site de fixation du facteur sigma et site de terminaison rho-dépendant. La probabilité qu'une région soit codante ajoutée aux données précédentes permet la délimitation de gènes. L'étude est ensuite poursuivie par l'analyse fonctionnelle des protéines hypothétiques générées par homologie de séquences.

ORF	***GeneMark (seuil 0.5)***				***GeneMark (seuil 0.4)***				***GeneMarkHMM***				
	LEnd	*REnd*	*Strand*	*Frame*	*LEnd*	*REnd*	*Strand*	*Frame*	*Strand*	*LeftEnd*	*RightEnd*	*GeneLength*	*Class*
0	*/*	*/*	*/*	*/*	*/*	*/*	*/*	*/*	*+*	*174*	*347*	*174*	*2*
3	*425*	*3436*	*direct*	*fr 2*	*425*	*3436*	*direct*	*fr 2*	*+*	*455*	*3436*	*2982*	*2*
4	*3444*	*5249*	*direct*	*fr 3*	*3444*	*5249*	*direct*	*fr 3*	*+*	*3447*	*5249*	*1803*	*2*
1	*5185*	*6486*	*direct*	*fr 1*	*5185*	*6486*	*direct*	*fr 1*	*+*	*5242*	*6486*	*1245*	*2*
5	*6453*	*7220*	*direct*	*fr 3*	*6453*	*7220*	*direct*	*fr 3*	*+*	*6486*	*7220*	*738*	*2*
2	*7210*	*9270*	*direct*	*fr 1*	*7210*	*9270*	*direct*	*fr 1*	*+*	*7222*	*9270*	*2049*	*1*
6	*9388*	*9558*	*direct*	*fr 1*	*/*	*/*	*/*	*/*	*+*	*9475*	*>9555*	*81*	*1*

Tableau B : Résultats obtenus avec GeneMark et GeneMarkHMM

ORF	RBS	sigma	rho	Blastp
0	161-176	111-151	557-582	gallidermin/nisin_family_lantibiotic
3	282-298	399-435	3986-4009	ABC_transporter_ATP-binding_protein
4	2531-2549	3294-3333	6683-6717	lantibiotic_dehydratase
1	4581-4599	4317-4348	6683-6717	Nisl/Spal_family_lantibiotic_immunity_lipoprotein
5	6469-6485	6344-6389	7224-7248	lanthionine_synthetase_C_family_protein
2	7205-7224	6873-6916	9277-9310	Nisl/Spal_family_lantibiotic_immunity_lipoprotein
6	9397-9411	9307-9353	/	response_regulator_transcription_factor

Tableau C : Résultats obtenus avec ScanForMatches et BlastP (NCBI)

ORF	SignalIP					DeepTMHMM	
	Prediction	SP(Sec/SPI)	TAT(Tat/SPI)	LIPO(Sec/SPII)	OTHER	Type	Localisation
0	OTHER	0.217604	0.004405	0.009520	0.768471	Globulaire	Inside
3	OTHER	0.125556	0.002389	0.016899	0.855156	alpha-TM	Transmembranaire
4	OTHER	0.003432	0.000448	0.001506	0.994614	Globulaire	Inside
1	OTHER	0.019379	0.002161	0.003979	0.974481	alpha-TM	Transmembranaire
5	OTHER	0.017380	0.000367	0.021509	0.960745	Globulaire	Inside
2	OTHER	0.004834	0.000481	0.000758	0.993927	Globulaire	Inside
6	OTHER	0.001969	0.000097	0.000325	0.997610	Globulaire	Inside

Tableau D : Résultats obtenus avec SignalIP et DeepTMHMM

Afin de mieux comparer les résultats, un programme pour convertir les fichiers a été créé et utilisé (documents complémentaires 4 et 5). Les analyses montrent la présence supposée de sept protéines dont deux transmembranaires. Les protéines 0 et 6 ne sont pas identifiées avec ORFfinder puisque leur taille est inférieure à celle de la recherche. En effet, elles seraient tronquées et donc les analyses sur leur promoteur et terminateur seraient approximatives. Tous les gènes sont dans le même sens (direct). La protéine 3 est une ABC transporter et est bien transmembranaire à trois domaines transmembranaires et une partie extracellulaire. La protéine 1 est une lipoprotéine responsable d'une réponse immunitaire et est membranaire. Cependant, ces résultats ne se retrouvent pas dans l'analyse faite par SignalIP.

CONCLUSION & DISCUSSION

L'étude menée sur le génome de Lactococcus lactis a permis d'explorer diverses méthodes de prédiction des régions codantes et d'analyse des fonctions protéiques. L'utilisation combinée de plusieurs outils bioinformatiques, tels qu'ORFfinder, GeneMark, GeneMarkHMM, ScanForMatches, SignalIP et DeepTMHMM, a fourni une vue d'ensemble des gènes potentiels et de leurs caractéristiques. Ces approches ont permis d'identifier sept protéines hypothétiques, dont deux transmembranaires.

Cependant, des limites ont été constatées, notamment l'utilisation de logiciels initialement développés pour d'autres organismes, comme E. coli, ce qui peut altérer la précision des prédictions pour L. lactis. De même, la recherche s'est concentrée uniquement sur les terminateurs rho-dépendants, négligeant d'autres

types de terminaison, et certains promoteurs sigma n'ont pu être analysés en raison de problèmes techniques.

Pour les études futures, l'amélioration de la pertinence des outils bioinformatiques via un entraînement spécifique au génome de L. lactis, ainsi que l'intégration d'analyses complémentaires pour détecter d'autres types de terminateurs, seront nécessaires. Ces ajustements permettraient de renforcer la fiabilité et la robustesse des résultats, contribuant ainsi à une compréhension plus approfondie de cet organisme d'intérêt industriel et scientifique.

RÉFÉRENCES

[1] Mémoire de Magistère : Isolement et sélection des souches de bactéries lactiques productrices des métabolites antibactériennes, par BELARBI Fatima (2010-2011) doi:10.13140/RG.2.2.13373.82405
[2] Wheeler, D L et al. "Database resources of the National Center for Biotechnology Information." Nucleic acids research vol. 28,1 (2000): 10-4. doi:10.1093/nar/28.1.10
[3] Borodovsky M. and McIninch J. "GeneMark: parallel gene recognition for both DNA strands." Computers & Chemistry, 1993, Vol. 17, No. 19, pp. 123-133
[4] Lukashin, A V, and M Borodovsky. "GeneMark.hmm: new solutions for gene finding." Nucleic acids research vol. 26,4 (1998): 1107-15. doi:10.1093/nar/26.4.1107
[5] https://blog.theseed.org/servers/2010/07/scan-for-matches.html By The SEED Team on July 16, 2010. The utility was written by Ross Overbeek; David Joerg and Morgan Price wrote sections of an earlier version. It is worth noting that it was strongly influenced by the elegant tools developed and distributed by David Searls.
[6] Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.
[7] Wall, L., & others. (1994). The Perl programming language. Prentice Hall Software Series.

ANNEXES

>seq_L_lactis_lactis

GAAACATTAACAAATCTAAAACAGTCTTAACCCCGGATGAGAATTCCGGGGTTTTTTTCTATTGGTAATATAACGCGAGCATAATAAACGGCTCTGATT
AAATTCTGAAGTTGCCATCAATGCTCAGGACGAATATTGCATGATCTATCATAAATTATAAGGAGGCACTCAAAATGAGTACAAAAGATTTTAACTTGG
ATTTGGTATCTGTTTCGAAGAAAGATTCAGGTGCATCACCACGCATTACAAGTATTTCGCTATGTACACCCGGTTGTAAAACAGGAGCTCTGATGGGT
TGTAACATGAAAACAGCAACTTGTCATTGTAGTATTCACGTAAGCAAATAACCAAATCAAAGGATAGTATTTTGTTAGTTCAGACATGGATACTATCCT
ATTTTTATAAGTTATTTAGGGTTGCTAAATAGCTTATAAAAATAAAGAGAGGAAAAAACATGATAAAAAGTTCATTTAAAGCTCAACCGTTTTTAGTAA
GAAATACAATTTTATCTCCAAACGATAAACGGAGTTTTACTGAATATACTCAAGTCATTGAGACTGTAAGTAAAAATAAAGTTTTTTTGGAACAGTTAC
TACTAGCTAATCCTAAACTCTATGATGTTATGCAGAAATATAATGCTGGTCTGTTAAAGAAGAAAAGGGTTAAAAAATTATTTGAATCTATTTACAAGT
ATTATAAGAGAAGTTATTTACGATCAACTCCATTTGGATTATTTAGTGAAACTTCAATTGGTGTTTTTTCGAAAAGTTCACAGTACAAGTTAATGGGAA
AGACTACAAAGGGTATAAGATTGGATACTCAGTGGTTGATTCGCCTAGTTCATAAAATGGAAGTAGATTTCTCAAAAAAGTTATCATTTACTAGAAATA
ATGCAAATTATAAGTTTGGAGATCGAGTTTTTCAAGTTTATACCATAAATAGTAGTGAGCTTGAAGAAGTAAATATTAAATATACGAATGTTTATCAAAT
TATTTCTGAATTTTGTGAGAATGACTATCAAAAATATGAAGATATTTGTGAAACTGTAACGCTTTGCTATGGAGACGAATATAGAGAACTATCGGAACA
ATATCTTGGCAGTCTGATAGTTAATCATTATTTGATCTCTAATTTACAAAAAGATTTGTTGTCAGATTTTTCTTGGAACACTTTTTTGACTAAAGTTGAA
GCAATAGATGAAGATAAAAAATATATAATTCCTCTGAAAAAAGTTCAAAAGTTTATTCAAGAATACTCAGAAATAGAAATTGGTGAAGGTATTGAGAA
ACTGAAAGAAATATATCAGGAAATGTCACAAATTCTTGAGAATGATAATTATATTCAAATTGATTTAATTAGTGATAGTGAAATAAATTTTGATGTTAAA
CAAAAGCAACAATTAGAACATTTAGCTGAGTTTTTAGGAAATACGACAAAATCTGTAAGAAGAACATATTTGGATGACTATAAGGATAAATTTATCGA
AAAATATGGTGTAGATCAAGAAGTACAAATAACAGAATTATTTGATTCTACATTTGGCATAGGAGCTCCATATAATTATAATCATCCTCGAAATGACTTT
TATGAGTCCGAACCGAGTACTCTATACTATTCAGAAGAGGAGAGAGAAAAGTACCTCAGCATGTATGTAGAAGCCGTTAAAAATCATAATGTAATTAA
TCTTGACGACTTAGAGTCTCATTATCAAAAAATGGACTTAGAAAAGAAAAGTGAACTTCAAGGGTTAGAATTATTTTTGAATTTGGCAAAGGAGTAT
GAAAAAGATATTTTTATTTTAGGGGATATCGTTGGAAATAATAATTTGGGAGGGGCATCAGGTAGATTTTCTGCACTCTCTCCGGAGTTAACAAGTTAT
CATAGAACGATAGTAGATTCTGTCGAAAGAGAAAATGAGAATAAAGAAATTACATCGTGTGAAATAGTATTTCTTCCAGAAAATATCAGACATGCTAA
CGTTATGCATACATCAATTATGAGGAGGAAAGTACTTCCATTTTTTACAAGTACAAGTCACAATGAAGTTCTGTTAACTAATATCTATATTGGAATAGAC
GAAAAAGAAAAATTTTATGCACGAGACATTTCAACTCAAGAGGTATTGAAATTCTACATTACAAGCATGTACAATAAAACGTTATTCAGTAATGAGCT
AAGATTTCTTTACGAAATTTCATTAGATGACAAGTTTGGTAATTTACCTTGGGAACTTATTTACAGAGACTTTGATTATATTCCACGTTTAGTATTTGAC
GAAATAGTAATATCTCCTGCTAAATGGAAAATTTGGGGAAGGGATGTAAATAGTAAGATGACAATAAGAGAACTTATTCAAAGCAAAGAAATTCCCA
AAGAGTTTTATATTGTCAATGGAGATAATAAAGTTTATTTATCACAGGAAAACCCATTGGATATGGAAATTTTAGAGTCGGCGATAAAGAAGAGCTCA
AAAAGAAAAGATTTTATAGAGCTACAAGAATATTTTGAAGATGAAAATATCATAAATAAAGGAGAAAAGGGGAGAGTTGCCGATGTTGTAGTGCCTT
TTATTAGAACGAGAGCATTAGGTAATGAAGGGAGAGCATTTATAAGAGAGAAAAGAGTTTCGGTTGAACGGCGTGAAAAATTGCCCTTTAACGAGT
GGCTTTATCTAAAGTTGTACATTTCTATAAATCGTCAAAATGAATTTTTACTGTCGTATCTTCCAGATATTCAGAAAATAGTAGCAAACCTGGGTGGAA
ATCTATTCTTCCTAAGATATACTGATCCTAAACCACATATTAGATTGCGTATAAAATGTTCAGATTTATTTTTAGCTTACGGATCTATTCTTGAAATCTTAA
AAAGGAGTCGGAAAAATAGGATAATGTCAACTTTTGATATTTCTATTTATGATCAAGAAGTAGAAAGATATGGTGGATTTGATACTTTAGAGTTATCCG
AAGCAATATTTTGTGCCGATTCTAAAATTATTCCAAATTTGCTTACATTGATAAAAGATACTAATAATGATTGGAAAGTCGATGATGTATCAATCTTGGT
GAATTATTTATATCTGAAATGCTTCTTTCAGAATGATAACAAAAAGATTCTTAATTTTTTGAATTTAGTTAGTACTAAAAAGGTTAAAGAAAATGTCAAT
GAAAAGATTGAACATTATCTTAAGCTTCTGAAAGTTAATAATCTAGGTGACCAAATTTTTTATGACAAGAATTTTAAAGAATTAAAGCATGCCATAAAA
AATTTATTTTTAAAAATGATAGCTCAAGATTTTGAACTTCAGAAAGTTTATTCAATTATTGACAGTATCATTCATGTCCATAATAACCGACTAATTGGTAT
TGAACGAGATAAAGAGAAATTAATTTATTACACACTTCAAAGGTTGTTTGTTTCGGAAGAATACATGAAATGAGGACTAATAGATGGATGAAGTGAA
AGAATTCACATCAAAACAATTTTTTAATACTTTACTTACTCTTCCAAGCACCTTGAAGTTAATTTTTCAGTTGGAAAAACGTTATGCAATTTATTTAATT
GTGCTAAATGCTATCACAGCTTTTGTTCCGTTGGCTAGTCTTTTTATTTATCAAGATTTAATAAACTCTGTGCTAGGTTCAGGGAGACATCTTATCAATA
TTATTATCATCTATTTTATTGTTCAAGTGATAACAACAGTTCTGGGACAGCTGGAAAGTTATGTTAGTGGAAAATTTGATATGCGACTTTCTTACAGTAT
CAATATGCGCCTCATGAGGACTACCTCATCTCTTGAATTAAGTGATTATGAGCAGGCTGATATGTATAATATCATAGAAAAAGTTACTCAAGACAGCAC
TTACAAGCCTTTTCAGCTATTTAATGCTATCATTGTTGTGCTTTCATCGTTTATCTCATTGTTATCTAGTCTATTTTTTATTGGAACATGGAACATTGGGGT
AGCAATTTTACTCCTTATTGTTCCAGTATTATCTTTGGTACTTTTTCTCAGAGTGGGACAATTAGAGTTTTTAATCCAGTGGCAGAGAGCAAGTTCTGA
AAGAGAAACATGGTATATTGTATATTTATTGACTCATGATTTTTCATTTAAAGAAATCAAGTTAAATAATATTAGCAATTACTTCATTCATAAATTTGGAA
AATTAAAGAAAGGATTTATCAACCAAGATTTAGCTATTGCTCGTAAGAAGACATATTTCAATATTTTTCTTGATTTCATTTTGAATTTGATAAATATTCTT
ACGATATTTGCTATGATCCTTTCGGTAAGAGCAGGAAAACTTCTTATAGGTAATTTGGTAAGTCTCATACAAGCTATTTCTAAAATCAATACTTATTCTC
AAACAATGATTCAAAATATTTACATCATTTATAATACTAGTTTGTTTATGGAACAACTTTTTGAGTTTTTAAAGAGAGAAAGTGTAGTTCACAAAAAAA
TAGAAGATACTGAAATATGCAATCAACATATAGGAACTGTTAAAGTAATTAATTTATCATATGTTTACCCTAATTCGAATGCCTTTGCACTAAAGAATAT
CAATTTATCCTTTGAAAAAGGAGAATTAACTGCTATTGTAGGAAAAAATGGTTCAGGGAAAAGTACACTAGTAAAGATAATTTCAGGATTATATCAAC
CAACTATGGGAATAATCCAATACGACAAAATGAGAAGTAGTTTGATGCCTGAGGAGTTTTATCAGAAAAACATATCGGTGCTGTTCCAAGATTTTGTG
AAGTATGAGTTAACGATAAGAGAGAATATAGGATTGAGTGATTTGTCTTCTCAATGGGAAGATGAGAAAATTATTAAAGTACTAGATAATTTAGGACTC
GATTTTTTGAAAACTAATAATCAATATGTACTTGATACGCAGTTAGGAAATTGGTTTCAAGAAGGGCATCAACTTTCAGGAGGTCAGTGGCAAAAAAT
TGCATTAGCAAGGACATTCTTTAAGAAAGCTTCAATTTATATTTTAGATGAACCAAGTGCTGCACTCGATCCTGTAGCTGAAAAAGAAATATTTGATTA
TTTTGTTGCTCTTTCGGAAAATAATATTTCAATTTTCATTTCTCATAGTTTGAATGCTGCCAGAAAAGCAAATAAAATCGTGGTTATGAAAGATGGACA
GGTCGAAGATGTTGGAAGTCATGATGTCCTTCTGAGAAGATGTCAATACTATCAAGAACTTTATTATTCAGAGCAATATGAGGATAATGATGAATAAAA
AAAATATAAAAAGAAATGTTGAAAAAATTATTGCTCAATGGGATGAGAGAACTAGAAAAAATAAAGAAAACTTCGATTTCGGAGAGTTGACTCTCTC
TACAGGATTGCCTGGTATAATTTTAATGTTAGCGGAGTTAAAAAATAAAGATAACTCAAAGATATATCAGAAAAAGATAGACAATTATATTGAATATATT
GTTAGCAAACTTTCAACATATGGGCTTTTAACAGGATCACTTTATTCGGGAGCAGCTGGCATTGCATTAAGTATCCTACATTTACGAGAAGATGACGA
AAAATATAAGAATCTTCTTGATAGCCTAAATAGATATATCGAATATTTCGTCAGAGAAAAAATTGAAGGATTTAATTTGGAAAACATTACTCCTCCTGAT
TATGACGTGATTGAAGGTTTATCTGGGATACTTTCCTATCTATTATTAATCAACGACGAGCAATATGATGATTTGAAAATACTCATTATCAATTTTTTATC
AAATCTGACTAAAGAAAACAATGGACTAATATCGCTTTACATCAAATCGGAGAATCAGATGTCTCAATCAGAAAGTGAGATGTATCCACTAGGCTGTT
TGAATATGGGATTAGCACATGGACTTGCTGGAGTGGGCTGTATCTTAGCTTATGCCCACATAAAAGGATATAGTAATGAAGCCTCGTTGTCAGCTTTGC
AAAAAATTATTTTTATTTATGAAAAGTTTGAACTTGAAAGGAAAAAACAGTTTCTATGGAAAGATGGACTTGTAGCAGATGAATTAAAAAAAGAGAA
AGTAATTAGGGAAGCAAGTTTCATTAGAGATGCATGGTGCTATGGAGGTCCAGGTATTAGTCTGCTATACTTATACGGAGGATTAGCACTGGATAATGA
CTATTTTGTAGATAAAGCAGAAAAAATATTAGAGTCAGCTATGCAAAGGAAACTTGGTATTGATTCATATATGATTTGCCATGGCTATTCTGGTTTAATA
GAAATTTGTTCTTTATTTAAGCGGCTATTAAATACAAAAAAGTTTGATTCATACATGGAAGAATTTAATGTTAATAGTGAGCAAATTCTTGAAGAATAC
GGAGATGAAAGTGGCACGGGTTTTCTTGAAGGAATAAGTGGCTGTATACTGGTATTATCGAAATTTGAATATTCAATCAATTTTACTTATTGGAGACAA
GCACTGTTACTTTTTGACGATTTTTTGAAAGGAGGGAAGAGGAAATGAGAAGATATTTAATACTTATTGTGGCCTTAATAGGGATAACAGGTTTATCA
GGGTGTTATCAAACAAGTCATAAAAAGGTGAGGTTTGACGAAGGAAGTTATACTAATTTTATTTATGATAATAAATCGTATTTCGTAACTGATAAGGAG
ATTCCTCAGGAGAACGTTAACAATTCCAAAGTAAAATTTTATAAGCTGTTGATTGTTGACATGAAAAGTGAGAAACTTTTATCAAGTAGCAACAAAA
ATAGTGTGACTTTGGTCTTAAATAATATTTATGAGGCTTCTGACAAGTCGCTATGTATGGGTATTAACGACAGATACTATAAGATACTTCCAGAAAGTGA
TAAGGGGGCGGTCAAAGCTTTGAGATTACAAAACTTTGATGTGACAAGCGATATTTCTGATGATAATTTTGTTATTGATAAAAATGATTCACGAAAAA
TTGACTATATGGGAAATATTTACAGTATATCGGACACCACCGTATCTGATGAAGAATTGGGAGAATATCAGGATGTTTTAGCTGAAGTACGTGTGTTTG

ATTCAGTTAGTGGCAAAAGTATCCCGAGGTCTGAATGGGGGAGAATTGATAAGGATGGTTCAAATTCCAAACAGAGTAGGACGGAATGGGATTATGG
CGAAATCCATTCTATTAGAGGAAAATCTCTTACTGAAGCATTTGCCGTTGAGATAAATGATGATTTTAAGCTTGCAACGAAGGTAGGAAACTAGAGTG
AAAAAAATACTAGGTTTCCTTTTTATCGTTTGTTCGTTGGGTTTATCAGCAACTGTGCATGGGGAGACAACAAATTCACAACAGTTACTCTCAAATAA
TATTAATACGGAATTAATTAATCATAATTCTAATGCAATTTTATCTTCAACAGAGGGATCAACGACTGATTCGATTAATCTAGGGGCGCAGTCACCTGCA
GTAAAATCGACAACAAGGACTGAATTGGATGTAACTGGTGCTGCTAAAACTTTATTACAGACATCAGCTGTTCAAAAAGAAATGAAAGTTTCGTTGC
AAGAAACTCAAGTTAGTTCTGAATTCAGTAAGAGAGATAGCGTTACAAATAAAGAAGCAGTTCCAGTATCTAAGGATGAGCTACTTGAGCAAAGTG
AAGTAGTCGTTTCAACATCATCGATTCAAAAAAATAAAATCCTCGATAATAAGAAGAAAAGAGCTAACTTCGTTACTTCCTCTCCGCTTATTAAGGAA
AAACCATCAAATTCTAAAGATGCATCTGGTGTAATTGATAATTCTGCTTCTCCTCTATCTTATCGTAAAGCTAAGGAAGTGGTATCTCTTAGACAACCTT
TAAAAAATCAAAAAGTAGAGGCACAACCTCTATTGATAAGTAATTCTTCTGAAAAGAAAGCAAGTGTTTATACAAATTCACATGATTTTTGGGATTAT
CAGTGGGATATGAAATATGTGACAAATAATGGAGAAAGCTATGCGCTCTACCAGCCCTCAAAGAAAATTTCTGTTGGAATTATTGATTCAGGAATCAT
GGAAGAACATCCTGATTTGTCAAATAGTTTAGGAAATTATTTTAAAAATCTTGTTCCTAAGGGAGGGTTTGATAATGAAGAACCTGATGAAACTGGAA
ATCCAAGTGATATTGTCGACAAAATGGGACACGGGACGGAAGTCGCAGGTCAGATTACAGCAAATGGTAATATTTTAGGAGTAGCACCAGGGATTAC
TGTAAATATATACAGAGTATTTGGTGAAAATCTTTCGAAATCGGAATGGGTAGCTAGAGCAATAAGAAGAGCTGCGGATGATGGGAACAAGGTCATC
AATATAAGTGCTGGACAGTATCTTATGATTTCAGGATCGTATGATGATGGAACAAATGATTATCAAGAGTATCTTAATTATAAGTCAGCAATAAATTATG
CAACAGCAAAAGGAAGTATTGTTGTCGCAGCTCTTGGTAATGATAGTTTAAACATACAAGATAACCAAACAATGATAAACTTTCTTAAGCGTTTCAGA
AGTATAAAGGTTCCTGGAAAAGTTGTAGATGCACCGAGTGTATTTGAGGATGTAATAGCCGTAGGTGGAATAGATGGTTATGGTAATATTTCTGATTTT
AGTAATATTGGAGCGGATGCAATTTATGCTCCTGCTGGCACAACGGCCAATTTTAAAAAATATGGGCAAGATAAATTTGTCAGTCAGGGTTATTATTTG
AAAGATTGGCTTTTTACAACTACTAATACTGGCTGGTACCAATATGTTTATGGCAACTCATTTGCTACTCCTAAAGTATCTGGGGCACTGGCATTAGTA
GTTGATAAATATGGAATAAAGAATCCTAACCAACTAAAAAGGTTTCTTCTAATGAATTCTCCAGAAGTTAATGGGAATAGAGTATTGAATATTGTTGAT
TTATTGAATGGGAAAAATAAAGCTTTTAGCTTAGATACAGATAAAGGTCAGGATGATGCTATTAACCATAAATCGATGGAGAATCTTAAAGAGTCTAG
GGATACAATGAAACAGGAACAAGATAAAGAAATTCAAAGAAATACAAATAACAATTTTTCTATCAAAAATGATTTTCATAACATTTCAAAAGAAGTA
ATTTCAGTTGATTATAATATTAATCAAAAAATGGCTAATAATCGAAATTCGAGAGGTGCTGTTTCTGTACGAAGTCAAGAAATTTTACCTGTTACTGGA
GATGGAGAAGATTTTTTACCGGCTTTAGGTATAGTGTGTATCTCAATCCTTGGTATATTGAAAAGAAAGACTAAAAATTGATAGGCTGAAAAAGGGTG
GGAATCCACTCTTTTTCTTTTTTCAATCCTTGACTTCCGCTTGAAATCACTTGAGCATTCTACAATTACCGGCTTTAGGTATAGTGTGTATCTTGAGTAC
TAAACAATCGGAGGTAAAGTGGTGTATAAAATTTTAATAGTTGATGATGATCAGGAAATTTTAAAATTAATGAAGACAGCATTAGAAATGAGAAACTA
TGAAGTTGCGACGCATCAAAACATTTCACTTCCCTTGGATATTACTGATTTTCAGGGATTTGATTTGATTTT

Annexe 1 : séquence génomique au format fasta du fragment d'intérêt du génome de *Lactococcus lactis*

```
tar -xvf scan_for_matches.tgz
# dézippe le dossier
gcc -O -o scan_for_matches ggpunit.c scan_for_matches.c
# compile le programme
./run_tests tmp
# lance le programme avec un fichier test (tmp)
diff tmp test_output
# vérifie que rien n'est retourné et donc que le programme fonctionne correctement

scan_for_matches matrice_file.txt < seq1_L.lactis.txt > output_file.txt
# exécute le programme avec la matrice poids position du RBS et la séquence d'intérêt
wc -l output_file.txt
# compte le nombre de ligne et donc retourne le nombre de séquence RBS identifiées (multiplié par 2)

scan_for_matches sigma_matrice_file.txt < seq1_L.lactis.txt > output_sigma_file.txt
# exécute le programme avec la matrice poids position du site du facteur sigma et la séquence d'intérêt

scan_for_matches rho_pattern_file.txt < seq1_L.lactis.txt > output_rho_file.txt
# exécute le programme avec le pattern du site de terminaison rho-dépendant et la séquence d'intérêt
```

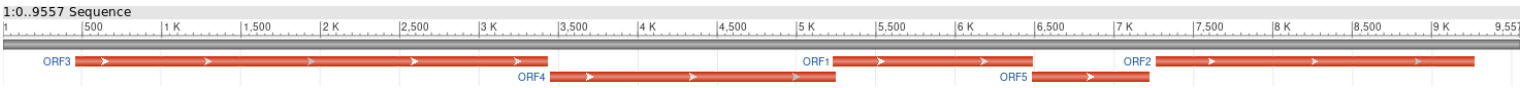
Annexe 2 : commande terminal (bash) utilisée pour analyser le fragment génomique d'intérêt avec le programme ScanForMatches

```
{(-23,-53,20,-33),(-22,-34,20,-46),(14,-46,-7,-15),(-27,-52,19,-17),(-4,-17,14,-10)}>44 5...12 DTG
# matrice_file.txt
# matrice poids position utilisée pour rechercher les RBS dans notre séquence d'intérêt
# les poids sont attribués respectivement à A, C, G et T
```

{(-22,-29,-25,16),(-18,-25,-21,16),(-28,-17,18,-11),(12,-9,-19,-6), (-2,12,-15,-7),(9,-13,-10,-1)}> 30
16...35 {(-32,-29,-45,17),(17,-45,-45,-28),(-3,-9,-21,11),(14,-7,-19,-21),
(14,-7,-19,-20),(-28,-35,-45,17)}> 30
sigma_matrice_file.txt
matrice poids position utilisée pour rechercher les sites de fixation du facteur σ dans notre séquence d'intérêt
les poids sont attribués respectivement à A, C, G et T

r1={AU,UA,GC,CG,GU,UG,GA,AG}
p1 = 3...5 p2 = 3...10 3...9 r1~p2 ~p1 TTTTTT[1,0,0]
rho_pattern_file.txt
pattern utilisé pour rechercher les terminateurs rho-dépendants dans notre séquence d'intérêt

Annexe 3 : matrices et pattern utilisés avec l'outil ScanForMatches



Annexe 4 : résultats obtenus avec ORFfinder et le fragment génomique d'intérêt

GENEMARK PREDICTIONS

Sequence: seq_L_lactis_lactis
Sequence file: seq.fna
Sequence length: 9557
GC Content: 31.57%
Window length: 120
Window step: 12
Threshold value: 0.500

Matrix: Lactococcus_lactis_II1403
Matrix author: -
Matrix order: 4

List of Open reading frames predicted as CDSs, shown with alternate starts
(regions from start to stop codon w/ coding function >0.50)

Left end	Right end	DNA Strand	Coding Frame	Avg Prob	Start Prob
	455	3436 direct	fr 2	0.55	0.02
	623	3436 direct	fr 2	0.56	0.03
	785	3436 direct	fr 2	0.57	0.03
	848	3436 direct	fr 2	0.57	0.05
	1310	3436 direct	fr 2	0.53	0.02
	3447	5249 direct	fr 3	0.63	0.75
	3456	5249 direct	fr 3	0.63	0.10
	3561	5249 direct	fr 3	0.65	0.37
	3630	5249 direct	fr 3	0.63	0.14
	3687	5249 direct	fr 3	0.63	0.31
	5230	6486 direct	fr 1	0.67	0.09
	5239	6486 direct	fr 1	0.68	0.90
	5242	6486 direct	fr 1	0.68	0.73

5374	6486	direct	fr 1	0.71	0.00
5653	6486	direct	fr 1	0.69	0.21
5806	6486	direct	fr 1	0.69	0.92
5827	6486	direct	fr 1	0.69	0.84
6483	7220	direct	fr 3	0.53	0.02
6507	7220	direct	fr 3	0.54	0.63
6564	7220	direct	fr 3	0.58	0.01
6696	7220	direct	fr 3	0.71	0.35
6738	7220	direct	fr 3	0.72	0.25
6789	7220	direct	fr 3	0.75	0.50
6873	7220	direct	fr 3	0.78	0.18
6939	7220	direct	fr 3	0.73	0.03
7222	9270	direct	fr 1	0.89	0.15
7279	9270	direct	fr 1	0.91	0.75
7504	9270	direct	fr 1	0.91	0.00
7792	9270	direct	fr 1	0.90	0.00
9406	9558	direct	fr 1	0.51	0.45
9409	9558	direct	fr 1	0.57	0.64

List of Regions of interest
(regions from stop to stop codon w/ a signal in between)

LEnd	REnd	Strand	Frame
425	3436	direct	fr 2
3444	5249	direct	fr 3
5185	6486	direct	fr 1
6453	7220	direct	fr 3
7210	9270	direct	fr 1

ABOUT THE MATRIX USED:

Training set derived by GeneMarkS, 4.27 September 2014
Tue Sep 23 15:01:07 2014

Annexe 5.1: résultats obtenus avec GeneMark et le fragment génomique d'intérêt (seuil à 0,5). Les éléments en surbrillance indiquent les différences observées avec les résultats obtenus pour un seuil à 0,4 (Annexe 5.2). Le graphe associé est renseigné dans les documents complémentaires (document1).

GENEMARK PREDICTIONS

Sequence: seq_L_lactis_lactis
Sequence file: seq.fna
Sequence length: 9557
GC Content: 31.57%
Window length: 120
Window step: 12
Threshold value: 0.400

Matrix: Lactococcus_lactis_Il1403
Matrix author: -
Matrix order: 4

List of Open reading frames predicted as CDSs, shown with alternate starts
(regions from start to stop codon w/ coding function >0.40)

Left end	Right end	DNA Strand	Coding Frame	Avg Prob	Start Prob
	455	3436 direct	fr 2	0.55	0.02
	623	3436 direct	fr 2	0.56	0.03
	785	3436 direct	fr 2	0.57	0.03
	848	3436 direct	fr 2	0.57	0.05
	1310	3436 direct	fr 2	0.53	0.02
	3447	5249 direct	fr 3	0.63	0.75
	3456	5249 direct	fr 3	0.63	0.10
	3561	5249 direct	fr 3	0.65	0.37
	3630	5249 direct	fr 3	0.63	0.14
	3687	5249 direct	fr 3	0.63	0.31
	5230	6486 direct	fr 1	0.67	0.09
	5239	6486 direct	fr 1	0.68	0.90
	5242	6486 direct	fr 1	0.68	0.73
	5374	6486 direct	fr 1	0.71	0.00
	5653	6486 direct	fr 1	0.69	0.21
	5806	6486 direct	fr 1	0.69	0.92
	5827	6486 direct	fr 1	0.69	0.84
	6483	7220 direct	fr 3	0.53	0.02
	6507	7220 direct	fr 3	0.54	0.63
	6564	7220 direct	fr 3	0.58	0.01
	6696	7220 direct	fr 3	0.71	0.35
	6738	7220 direct	fr 3	0.72	0.25
	6789	7220 direct	fr 3	0.75	0.50
	6873	7220 direct	fr 3	0.78	0.18
	7222	9270 direct	fr 1	0.89	0.15
	7279	9270 direct	fr 1	0.91	0.75
	7504	9270 direct	fr 1	0.91	0.00
	7792	9270 direct	fr 1	0.90	0.00
	9406	9558 direct	fr 1	0.51	0.45
	9409	9558 direct	fr 1	0.57	0.64

List of Regions of interest
(regions from stop to stop codon w/ a signal in between)

LEnd	REnd	Strand	Frame
	425	3436 direct	fr 2
	3444	5249 direct	fr 3
	5185	6486 direct	fr 1
	6453	7220 direct	fr 3
	7210	9270 direct	fr 1
	9388	9558 direct	fr 1

ABOUT THE MATRIX USED:

Training set derived by GeneMarkS, 4.27 September 2014
Tue Sep 23 15:01:07 2014

Annexe 5.2: résultats obtenus avec GeneMark et le fragment génomique d'intérêt (seuil à 0,4). Les éléments en surbrillance indiquent les différences observées avec les résultats obtenus pour un seuil à 0,5 (Annexe 5.1). Le graphe associé est renseigné dans les documents complémentaires (document2).

GeneMark.hmm PROKARYOTIC (Version 3.42)
Date: Fri Nov 29 04:52:54 2024
Sequence file name: seq.fna
Model file name:
/home/genemark/parameters/prokaryotic/Lactococcus_lactis_Il1403/GeneMark_hmm_combined.mod
RBS: true
Model information: Lactococcus_lactis_Il1403

FASTA definition line: seq_L_lactis_lactis
Predicted genes

Gene	Strand	LeftEnd	RightEnd	Length	Gene	Class
	#					
	1	+	174	347	174	2
	2	+	455	3436	2982	2
	3	+	3447	5249	1803	2
	4	+	5242	6486	1245	2
	5	+	6483	7220	738	2
	6	+	7222	9270	2049	1
	7	+	9475	>9555	81	1

Annexe 6 : résultats obtenus avec GeneMarkHMM et le fragment génomique d'intérêt

>seq_L_lactis_lactis:[161,176]
GGAGG CACTCAAA ATG
>seq_L_lactis_lactis:[282,298]
GGAGC TCTGATGGG TTG
>seq_L_lactis_lactis:[1802,1814]
GGGGA TATCG TTG
>seq_L_lactis_lactis:[2317,2335]
GGATG TAAATAGTAAG ATG
>seq_L_lactis_lactis:[2531,2549]
GGAGA AAAGGGGAGAG TTG
>seq_L_lactis_lactis:[4581,4599]
GGAGA ATTAACTGCTA TTG
>seq_L_lactis_lactis:[5497,5512]
GGAGC AGCTGGCA TTG
>seq_L_lactis_lactis:[5611,5625]
GAAGG ATTTAAT TTG
>seq_L_lactis_lactis:[5796,5808]
GGAGA ATCAG ATG
>seq_L_lactis_lactis:[6340,6352]
GGAGA TGAAA GTG
>seq_L_lactis_lactis:[6367,6379]
GAAGG AATAA GTG
>seq_L_lactis_lactis:[6469,6485]
GGAGG GAAGAGGAA ATG
>seq_L_lactis_lactis:[6836,6854]
GGGGG CGGTCAAAGCT TTG
>seq_L_lactis_lactis:[6990,7005]
GGAGA ATATCAGG ATG
>seq_L_lactis_lactis:[7067,7086]
GGGGA GAATTGATAAGG ATG
>seq_L_lactis_lactis:[7205,7224]
GAAGG TAGGAAACTAGA GTG
>seq_L_lactis_lactis:[7449,7462]
GGATG TAACTG GTG
>seq_L_lactis_lactis:[7593,7606]
GGATG AGCTAC TTG
>seq_L_lactis_lactis:[7942,7954]
GGAGA AAGCT ATG
>seq_L_lactis_lactis:[8071,8086]

GGAGG GTTTGATA ATG
>seq_L_lactis_lactis:[8547,8566]
GGATG TAATAGCCGTAG GTG
>seq_L_lactis_lactis:[8608,8626]
GGAGC GGATGCAATTT ATG
>seq_L_lactis_lactis:[9397,9411]
GGAGG TAAAGTG GTG

Annexe 7.1 : résultats obtenus avec ScanForMatches et le fragment génomique d'intérêt pour la recherche des RBS

>seq_L_lactis_lactis:[62,103]
TTGGTA ATATAACGCGAGCATAATAAACGGCTCTGA TTAAAT
>seq_L_lactis_lactis:[111,151]
TTGCCA TCAATGCTCAGGACGAATATTGCATGATC TATCAT
>seq_L_lactis_lactis:[272,304]
TTGTAA AACAGGAGCTCTGATGGGTTG TAACAT
>seq_L_lactis_lactis:[368,395]
TTGTTA GTTCAGACATGGATAC TATCCT
>seq_L_lactis_lactis:[399,435]
TTTATA AGTTATTTAGGGTTGCTAAATAGCT TATAAA
>seq_L_lactis_lactis:[470,504]
TTTAAA GCTCAACCGTTTTTAGTAAGAAA TACAAT
>seq_L_lactis_lactis:[530,570]
TTTACT GAATATACTCAAGTCATTGAGACTGTAAG TAAAAA
>seq_L_lactis_lactis:[581,612]
TTGGAA CAGTTACTACTAGCTAATCC TAAACT
>seq_L_lactis_lactis:[675,716]
TTGAAT CTATTTACAAGTATTATAAGAGAAGTTATT TACGAT
>seq_L_lactis_lactis:[769,812]
TTCACA GTACAAGTTAATGGGAAAGACTACAAAGGGTA TAAGAT
>seq_L_lactis_lactis:[927,971]
TTTATA CCATAAATAGTAGTGAGCTTGAAGAAGTAAATA TTAAAT
>seq_L_lactis_lactis:[1054,1096]
TTGCTA TGGAGACGAATATAGAGAACTATCGGAACAA TATCTT
>seq_L_lactis_lactis:[1148,1185]
TTGTCA GATTTTTCTTGGAACACTTTTTTGAC TAAAGT
>seq_L_lactis_lactis:[1323,1356]
TTGAGA ATGATAATTATATTCAAATTGA TTTAAT
>seq_L_lactis_lactis:[1527,1561]
TTGATT CTACATTTGGCATAGGAGCTCCA TATAAT
>seq_L_lactis_lactis:[1578,1613]
ATGACT TTTATGAGTCCGAACCGAGTACTC TATACT
>seq_L_lactis_lactis:[1716,1754]
TGGACT TAGAAAAGAAAAGTGAACTTCAAGGGT TAGAAT
>seq_L_lactis_lactis:[1766,1812]
TTGGCA AAGGAGTATGAAAAAGATATTTTTATTTTAGGGGA TATCGT
>seq_L_lactis_lactis:[2021,2061]
TTTACA AGTACAAGTCACAATGAAGTTCTGTTAAC TAATAT
>seq_L_lactis_lactis:[2123,2152]
TTGAAA TTCTACATTACAAGCATG TACAAT
>seq_L_lactis_lactis:[2211,2257]
TTGGTA ATTTACCTTGGGAACTTATTTACAGAGACTTTGAT TATATT
>seq_L_lactis_lactis:[2300,2334]
TGGAAA ATTTGGGGAAGGGATGTAAATAG TAAGAT
>seq_L_lactis_lactis:[2352,2386]
TTCAAA GCAAAGAAATTCCCAAAGAGTTT TATATT
>seq_L_lactis_lactis:[2483,2523]
TTTATA GAGCTACAAGAATATTTTGAAGATGAAAA TATCAT
>seq_L_lactis_lactis:[2648,2678]
TTGCCC TTTAACGAGTGGCTTTATC TAAAGT
>seq_L_lactis_lactis:[2758,2785]

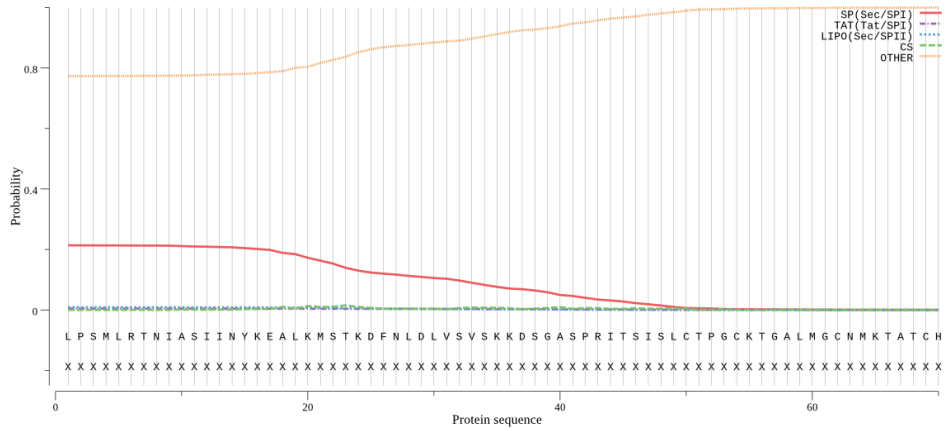
TGGAAA TCTATTCTTCCTAAGA TATACT
>seq_L_lactis_lactis:[3002,3031]
TTGCTT ACATTGATAAAAGATACT AATAAT
>seq_L_lactis_lactis:[3162,3205]
ATGAAA AGATTGAACATTATCTTAAGCTTCTGAAAGTT AATAAT
>seq_L_lactis_lactis:[3225,3263]
ATGACA AGAATTTTAAAGAATTAAAGCATGCCA TAAAAA
>seq_L_lactis_lactis:[3294,3333]
TTGAAC TTCAGAAAGTTTATTCAATTATTGACAG TATCAT
>seq_L_lactis_lactis:[3513,3559]
TTGAAG TTAATTTTTCAGTTGGAAAAACGTTATGCAATTTA TTTAAT
>seq_L_lactis_lactis:[3591,3622]
TTGGCT AGTCTTTTTATTTATCAAGA TTTAAT
>seq_L_lactis_lactis:[3712,3758]
TGGAAA GTTATGTTAGTGGAAAATTTGATATGCGACTTTCT TACAGT
>seq_L_lactis_lactis:[3793,3830]
TTGAAT TAAGTGATTATGAGCAGGCTGATATG TATAAT
>seq_L_lactis_lactis:[3968,3996]
TTTACT CCTTATTGTTCCAGTAT TATCTT
>seq_L_lactis_lactis:[4089,4127]
TTGACT CATGATTTTTCATTTAAAGAAATCAAG TTAAAT
>seq_L_lactis_lactis:[4231,4261]
TTGATT TCATTTTGAATTTGATAAA TATTCT
>seq_L_lactis_lactis:[4317,4348]
TTGGTA AGTCTCATACAAGCTATTTC TAAAAT
>seq_L_lactis_lactis:[4597,4630]
TTGTAG GAAAAAATGGTTCAGGGAAAAG TACACT
>seq_L_lactis_lactis:[4791,4837]
TTGAGT GATTTGTCTTCTCAATGGGAAGATGAGAAAATTAT TAAAGT
>seq_L_lactis_lactis:[4954,4997]
TTGCAT TAGCAAGGACATTCTTTAAGAAAGCTTCAATT TATATT
>seq_L_lactis_lactis:[5047,5078]
TTGATT ATTTTGTTGCTCTTTCGGAA AATAAT
>seq_L_lactis_lactis:[5086,5131]
TTTTCA TTTCTCATAGTTTGAATGCTGCCAGAAAAGCAAA TAAAAT
>seq_L_lactis_lactis:[5164,5204]
TTGGAA GTCATGATGTCCTTCTGAGAAGATGTCAA TACTAT
>seq_L_lactis_lactis:[5338,5369]
TTGACT CTCTCTACAGGATTGCCTGG TATAAT
>seq_L_lactis_lactis:[5657,5692]
TTGAAG GTTTATCTGGGATACTTTCCTATC TATTAT
>seq_L_lactis_lactis:[5845,5891]
TTGAAT ATGGGATTAGCACATGGACTTGCTGGAGTGGGCTG TATCTT
>seq_L_lactis_lactis:[6001,6034]
TGGAAA GATGGACTTGTAGCAGATGAAT TAAAAA
>seq_L_lactis_lactis:[6194,6231]
TTGGTA TTGATTCATATATGATTTGCCATGGC TATTCT
>seq_L_lactis_lactis:[6239,6273]
TAGAAA TTTGTTCTTTATTTAAGCGGCTA TTAAAT
>seq_L_lactis_lactis:[6344,6389]
ATGAAA GTGGCACGGGTTTTCTTGAAGGAATAAGTGGCTG TATACT
>seq_L_lactis_lactis:[6463,6499]
TTGAAA GGAGGGAAGAGGAAATGAGAAGATA TTTAAT
>seq_L_lactis_lactis:[6571,6605]
TTGACG AAGGAAGTTATACTAATTTTATT TATGAT
>seq_L_lactis_lactis:[6772,6812]
CTGACA AGTCGCTATGTATGGGTATTAACGACAGA TACTAT
>seq_L_lactis_lactis:[6873,6916]
GTGACA AGCGATATTTCTGATGATAATTTTGTTATTGA TAAAAA
>seq_L_lactis_lactis:[7748,7779]
TTGATA ATTCTGCTTCTCCTCTATCT TATCGT
>seq_L_lactis_lactis:[8026,8057]
TTGTCA AATAGTTTAGGAAATTATTT TAAAAA
>seq_L_lactis_lactis:[8313,8346]
TGGACA GTATCTTATGATTTCAGGATCG TATGAT
>seq_L_lactis_lactis:[8448,8482]

TTTAAA CATACAAGATAACCAAACAATGA TAAACT
>seq_L_lactis_lactis:[8543,8588]
TTGAGG ATGTAATAGCCGTAGGTGGAATAGATGGTTATGG TAATAT
>seq_L_lactis_lactis:[8650,8694]
TTTAAA AAATATGGGCAAGATAAATTTGTCAGTCAGGGT TATTAT
>seq_L_lactis_lactis:[8798,8836]
TTGATA AATATGGAATAAAGAATCCTAACCAAC TAAAAA
>seq_L_lactis_lactis:[9066,9108]
TTTTCA TAACATTTCAAAAGAAGTAATTTCAGTTGAT TATAAT
>seq_L_lactis_lactis:[9204,9248]
TTTACC GGCTTTAGGTATAGTGTGTATCTCAATCCTTGG TATATT
>seq_L_lactis_lactis:[9307,9353]
TTTTCA ATCCTTGACTTCCGCTTGAAATCACTTGAGCATTC TACAAT
>seq_L_lactis_lactis:[9380,9407]
TTGAGT ACTAAACAATCGGAGG TAAAGT

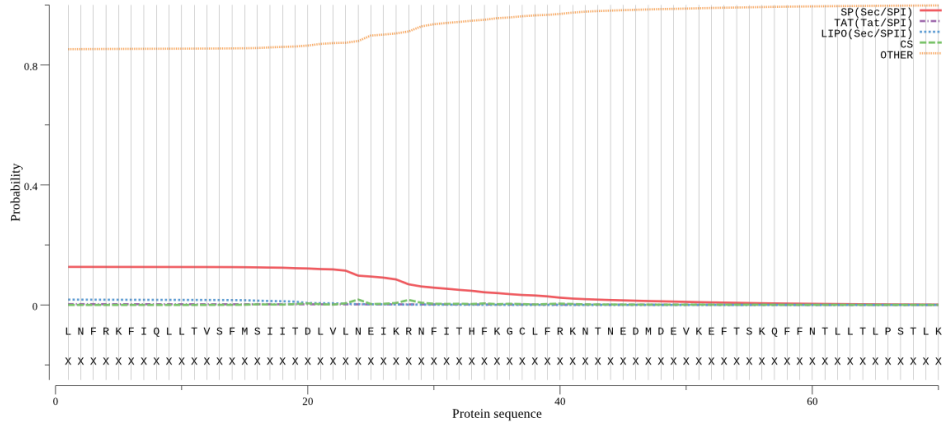
Annexe 7.2 : résultats obtenus avec ScanForMatches et le fragment génomique d'intérêt pour la recherche des sites de fixation du facteur sigma (seuil supérieur à 30)

>seq_L_lactis_lactis:[29,59]
AAC CCCGG ATGAGAATT CCGGG GTT TTTTTC
>seq_L_lactis_lactis:[557,582]
ACT GTA AGTAAAAA TAA AGT TTTTTT
>seq_L_lactis_lactis:[1771,1800]
AAA GGAGT ATGAAAAA GATAT TTT TATTTT
>seq_L_lactis_lactis:[3986,4009]
AGT ATT ATCTTT GGT ACT TTTTCT
>seq_L_lactis_lactis:[4404,4424]
TTG TTT ATG GAA CAA CTTTTT
>seq_L_lactis_lactis:[6683,6717]
GTT GATTGTT GACATGAAA AGTGAGA AAC TTTTAT
>seq_L_lactis_lactis:[7224,7248]
GAA AAA AATACTA GGT TTC CTTTTT
>seq_L_lactis_lactis:[8162,8183]
TTA CAG CAAA TGG TAA TATTTT
>seq_L_lactis_lactis:[8670,8696]
TAA ATT TGTCAGTCA GGG TTA TTATTT
>seq_L_lactis_lactis:[9277,9310]
GAA AAAGGGT GGGAATCC ACTCTTT TTC TTTTTT

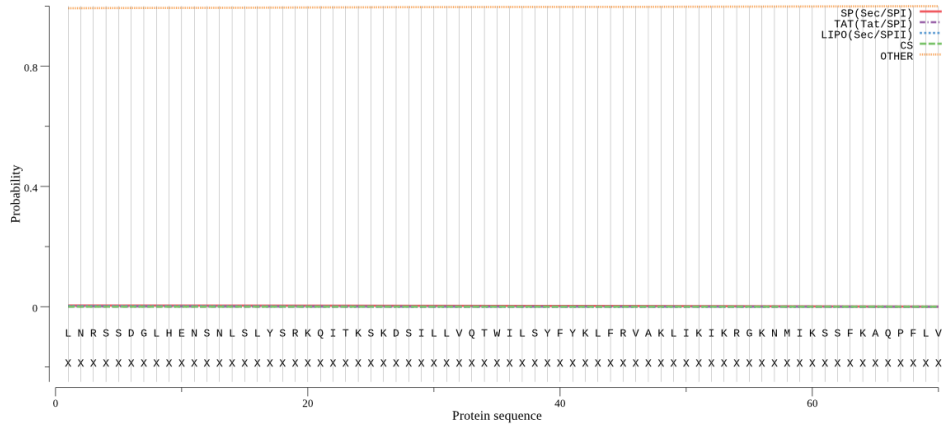
Annexe 7.3 : résultats obtenus avec ScanForMatches et le fragment génomique d'intérêt pour la recherche des sites de terminaison rho-dépendant



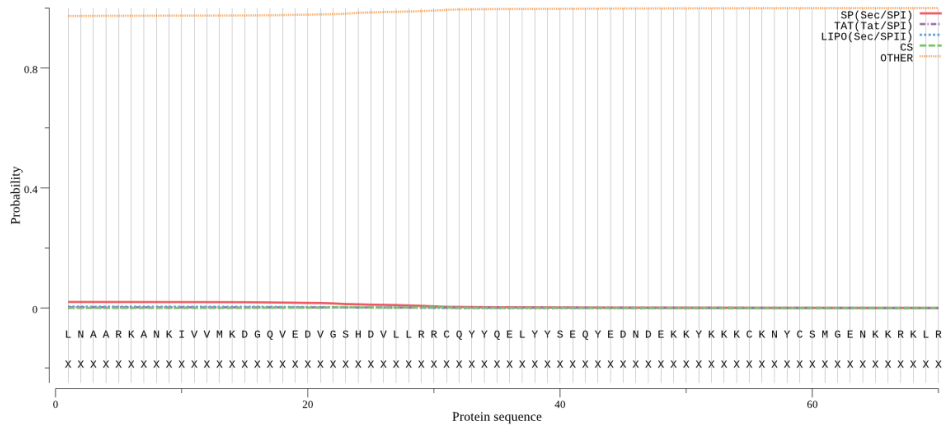
ORF0



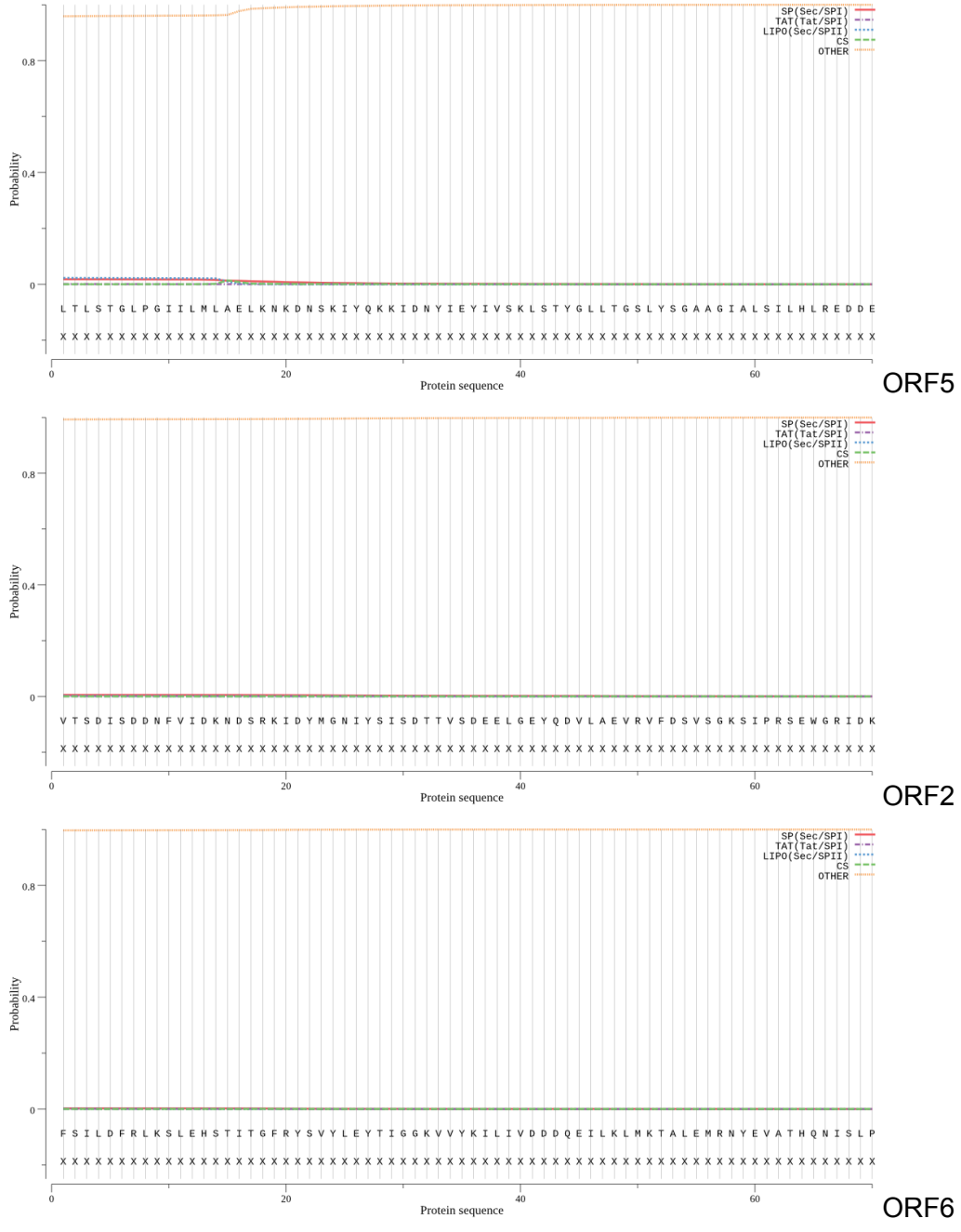
ORF3



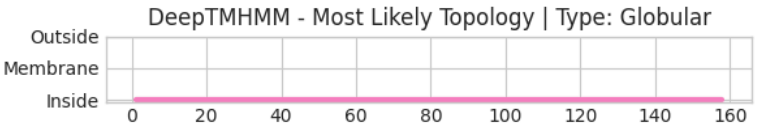
ORF4

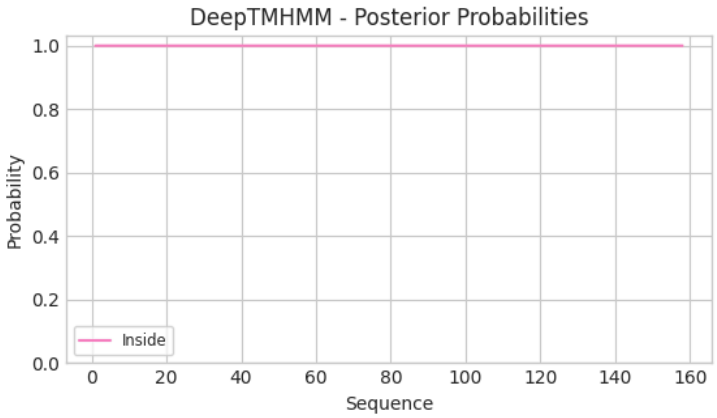


ORF1

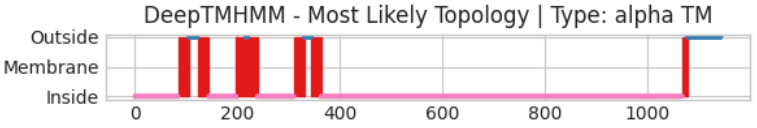


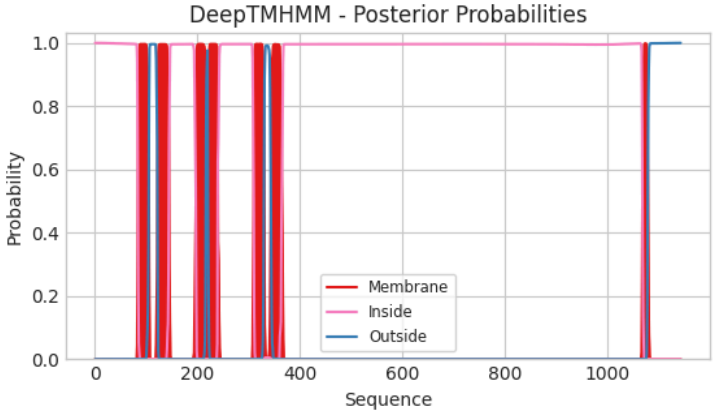
Annexe 8 : résultats obtenus avec SignalIP et les séquences protéiques identifiées



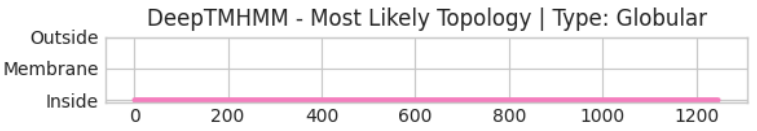


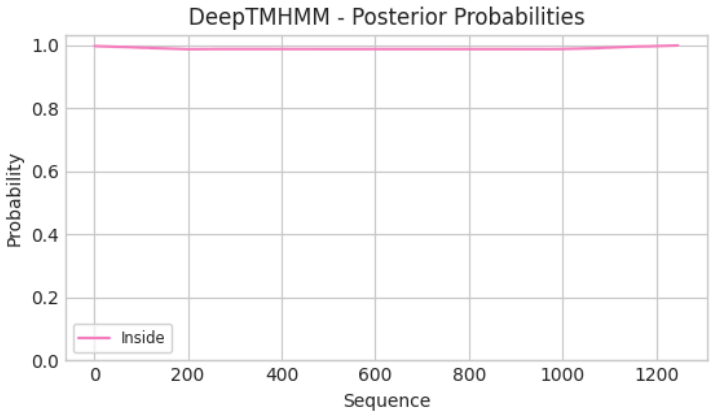
ORF0



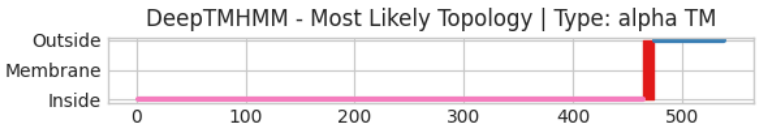


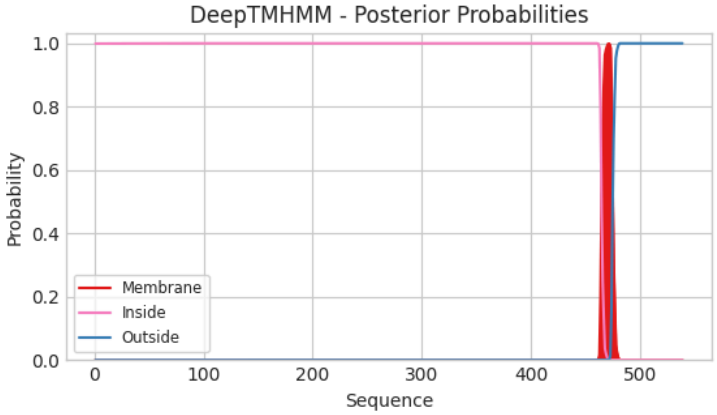
ORF3



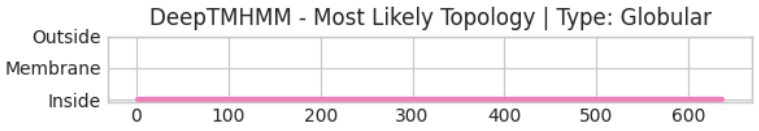


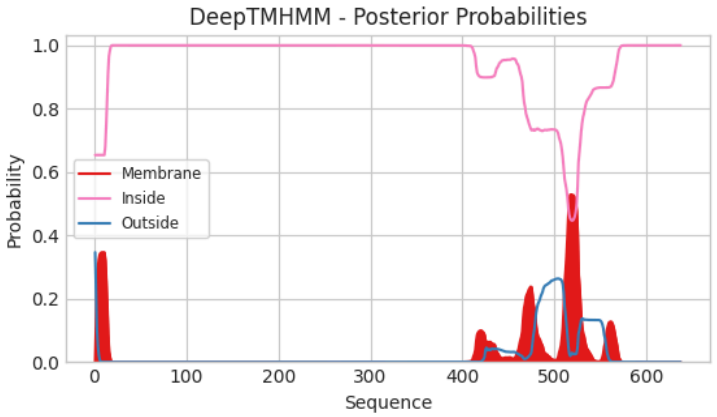
ORF4



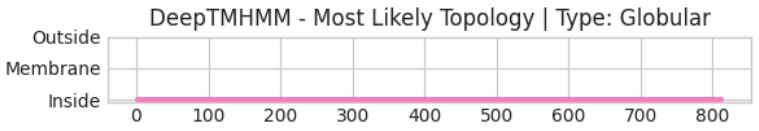


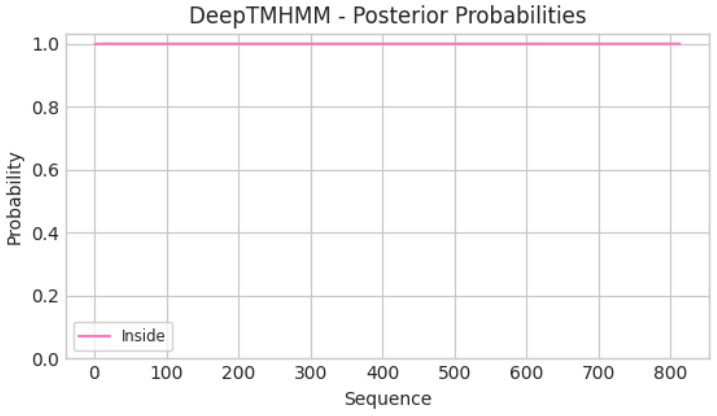
ORF1



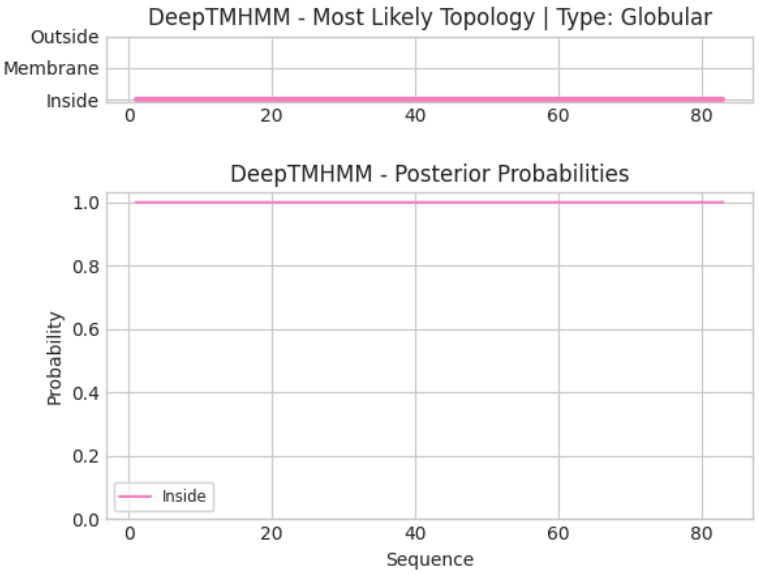


ORF5





ORF2



ORF6

Annexe 9 : résultats obtenus avec DeepTMHMM et les séquences protéiques identifiées

DOCUMENTS COMPLÉMENTAIRES

Document 1 : graphe obtenu avec GeneMark sur la séquence d'intérêt (seuil à 0,5)

Document 2 : graphe obtenu avec GeneMark sur la séquence d'intérêt (seuil à 0,4)

Document 3 : graphe obtenu avec GeneMarkHMM sur la séquence d'intérêt

Document 4 : code python pour la conversion des fichiers en format GFF

Document 5 : résultats obtenus après conversion des fichiers de sortie à l'aide du code fourni (document complémentaire 4)

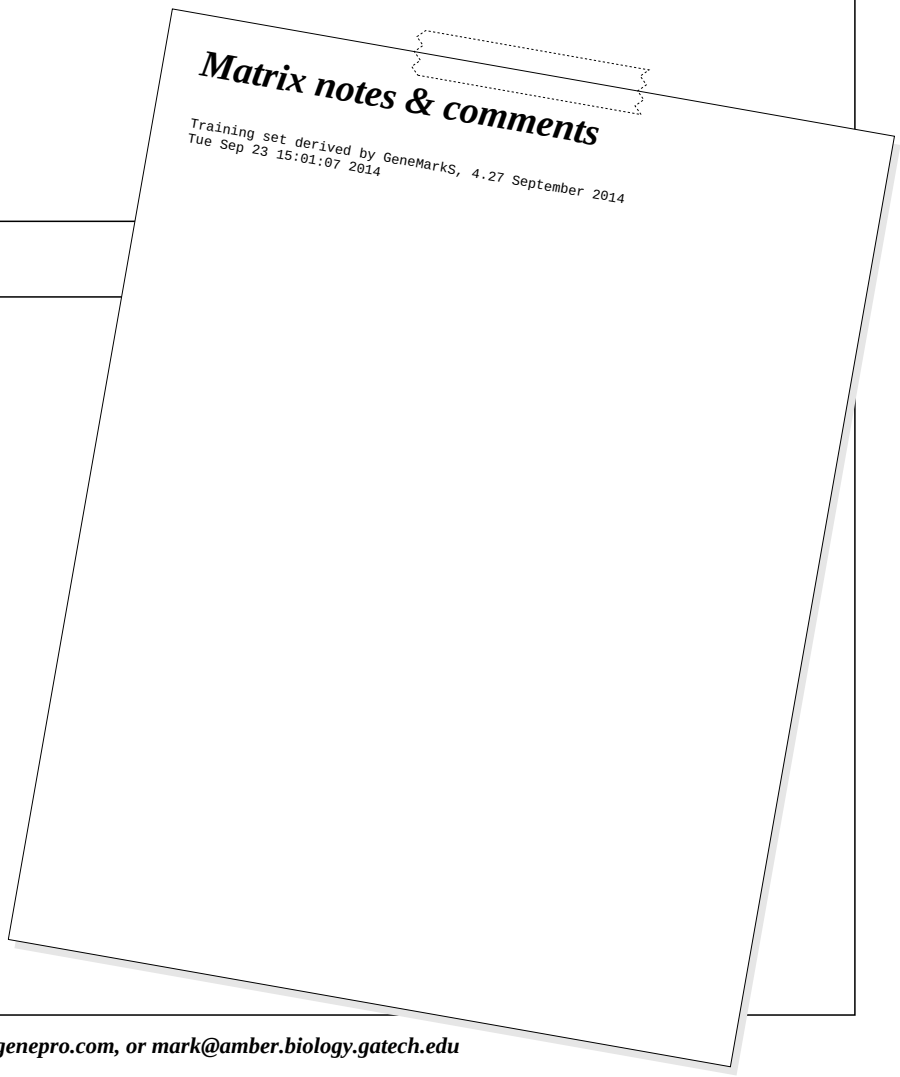
Document 1 : graphe obtenu avec GeneMark sur la séquence d’intérêt (seuil à 0,5)

GeneMark

Version 2.5p (09.08.06)

PROGRAM INFORMATION

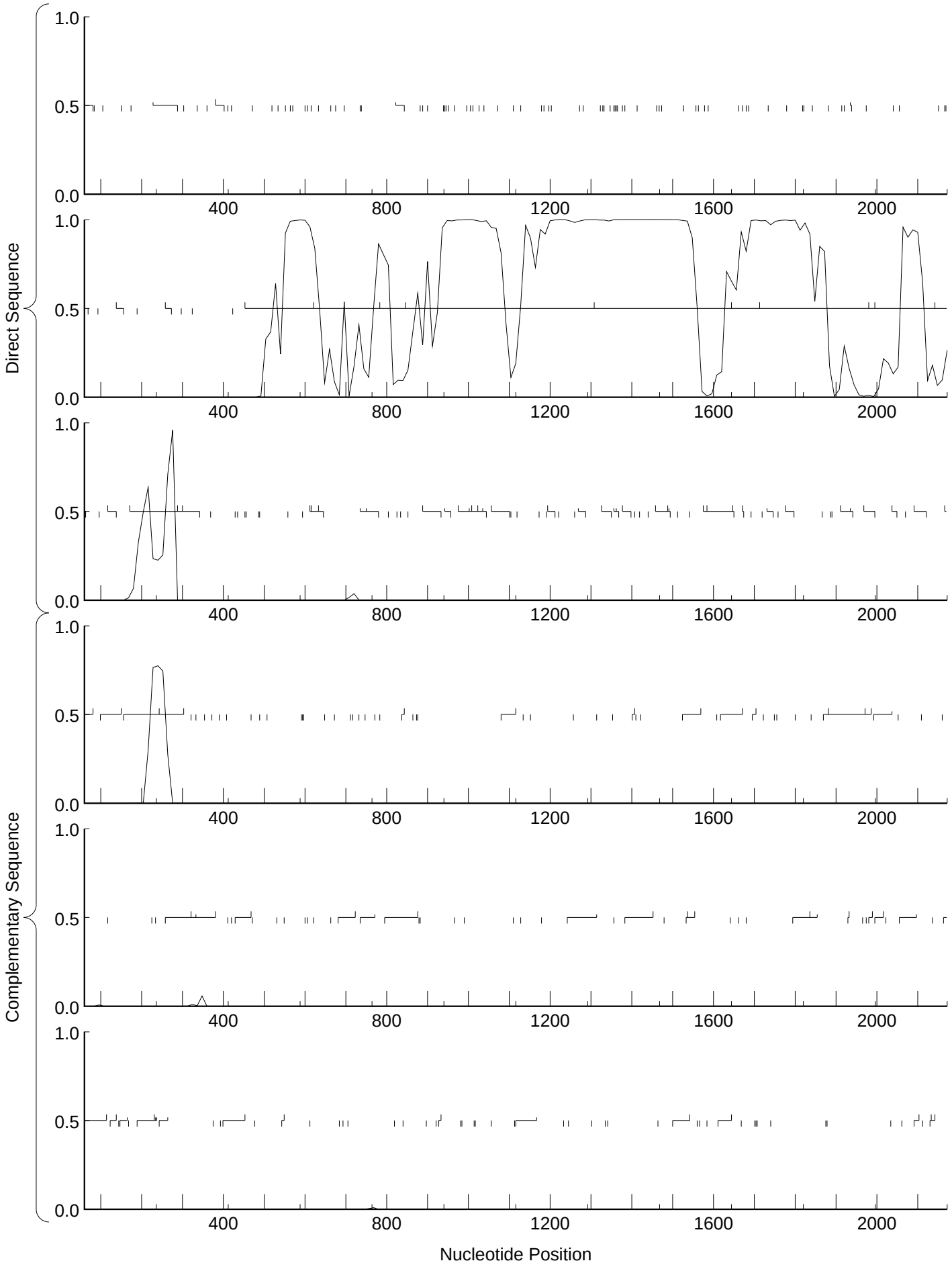
```
Sequence         :  seq_L_lactis_lactis
Analysis Date    :  11/29/24 at 4:45:35
Pages            :  6
Sequence Length  :  9557 bp
GC Content       :   31.57%

Window Length    :  120 bp
Window Step      :  12 bp
Threshold Value  :  0.400

PS-Version       :  1.2

GeneMark Options :  PostScript graph,
                    Mark ORFs / splice sites,
                    List ORFs,
                    List regions and/or splice sites,
```

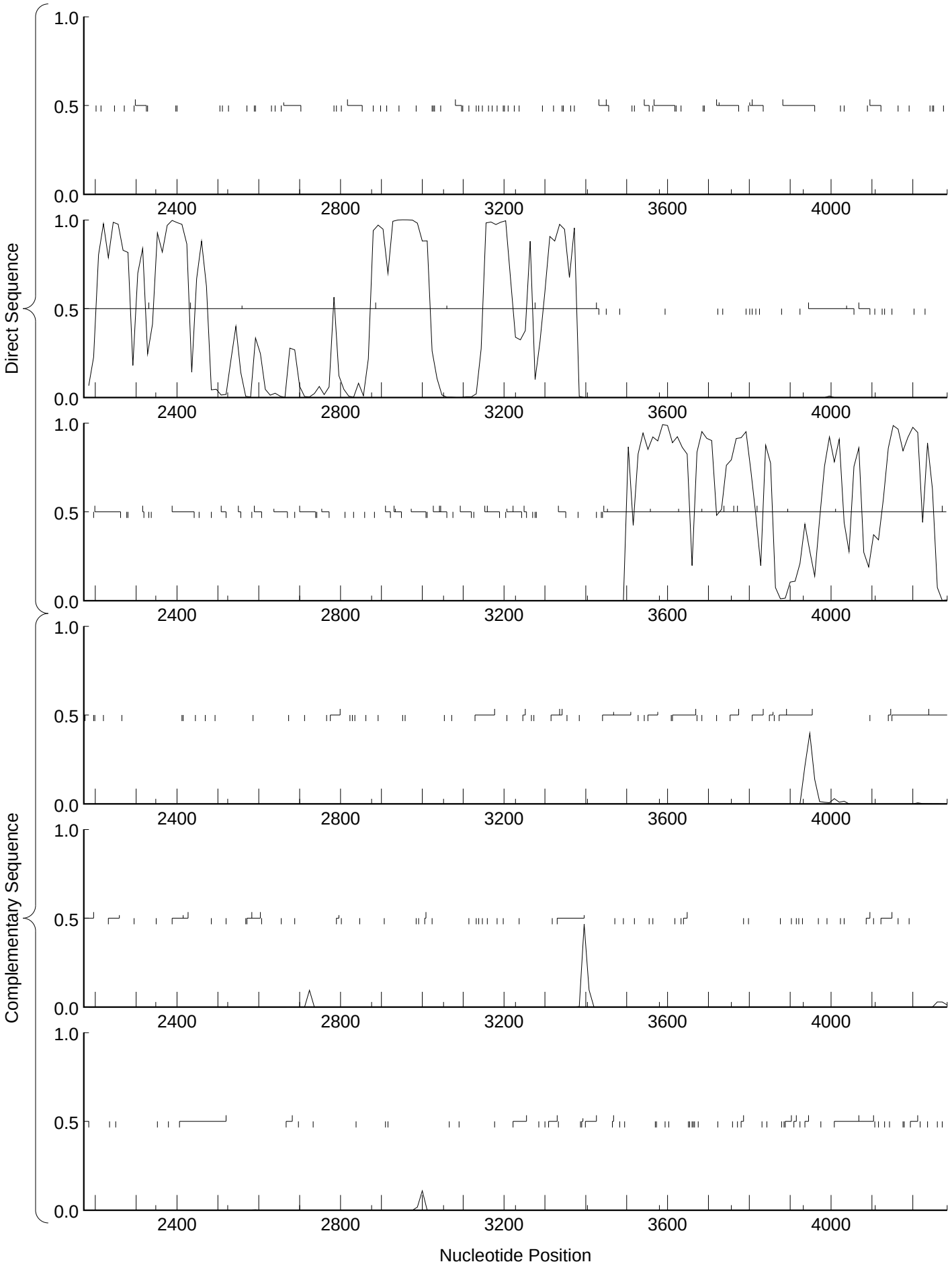


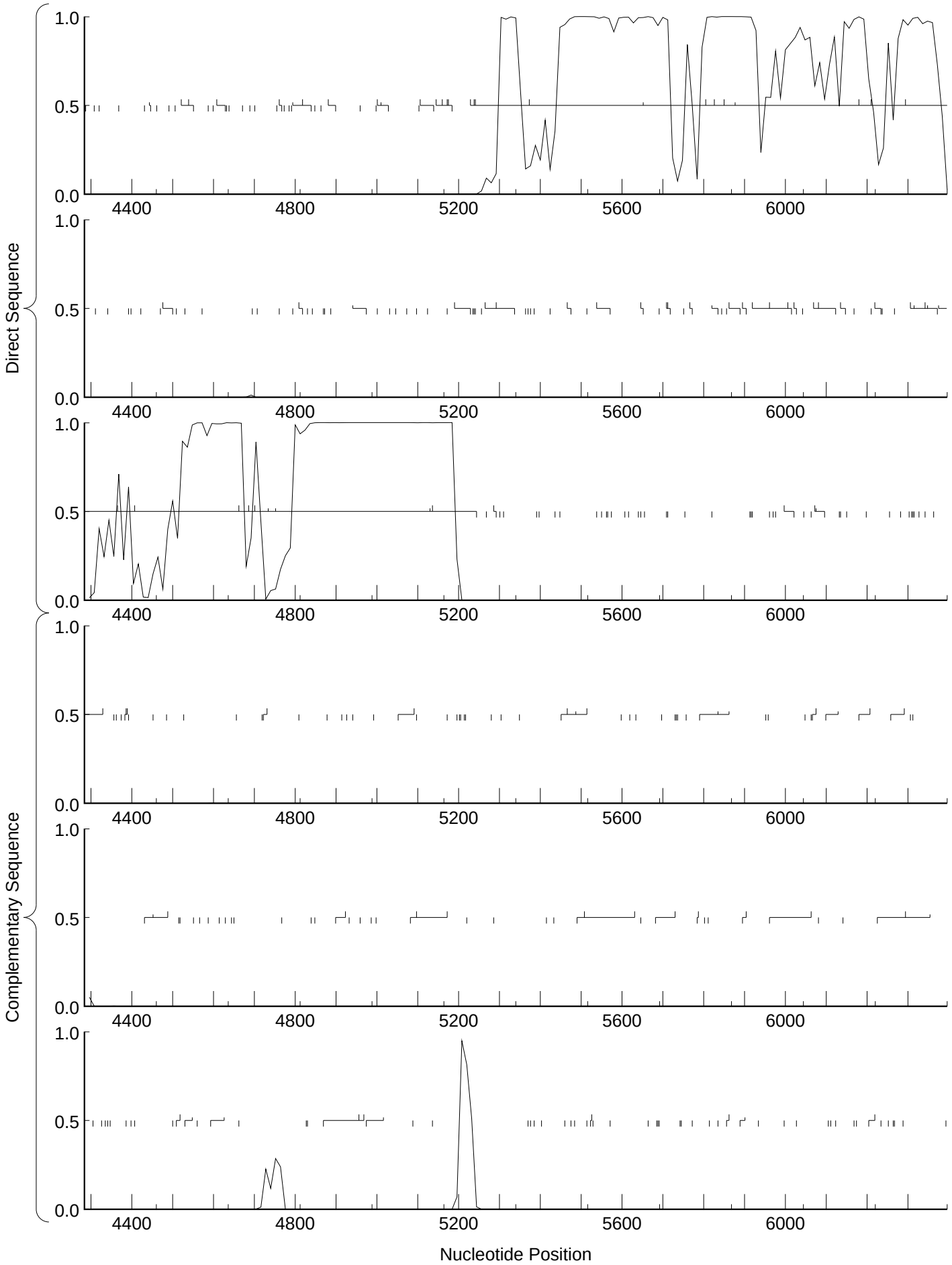
MATRIX INFORMATION

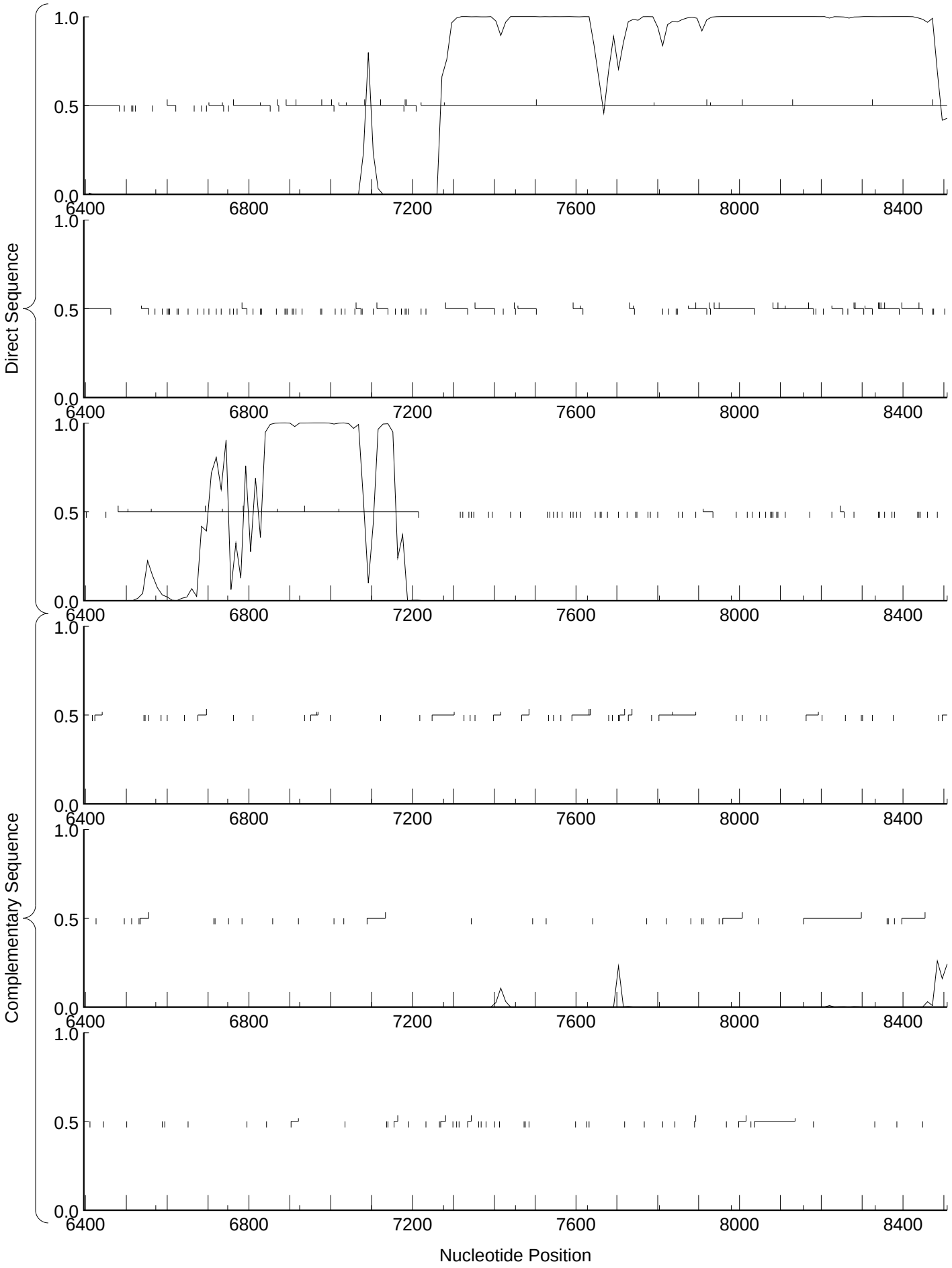
```
Matrix     :  Lactococcus_lactis_Il1403
Author     :  -

Order      :  4
```

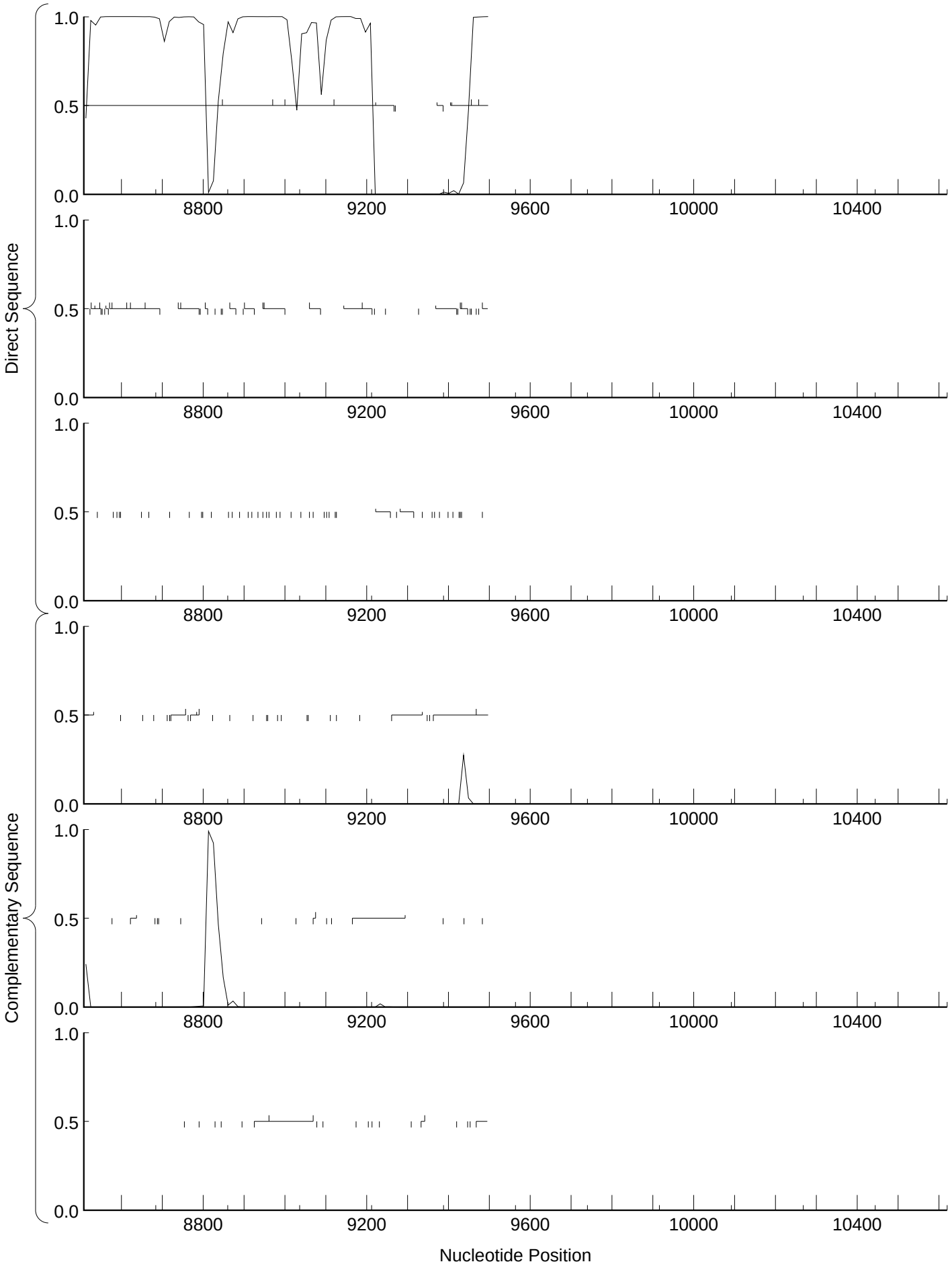

```
  Send questions / comments to:
      Dr. M. Borodovsky
      Georgia Institute of Technology
      School of Biology
      Atlanta, GA  30332-0230
```


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
400
800
1200
1600
2000
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
2400
2800
3200
3600
4000
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
4400
4800
5200
5600
6000
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
6400
6800
7200
7600
8000
8400
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
8800
9200
9600
10000
10400
Nucleotide Position

Document 2 : graphe obtenu avec GeneMark sur la séquence d’intérêt (seuil à 0,4)

GeneMark

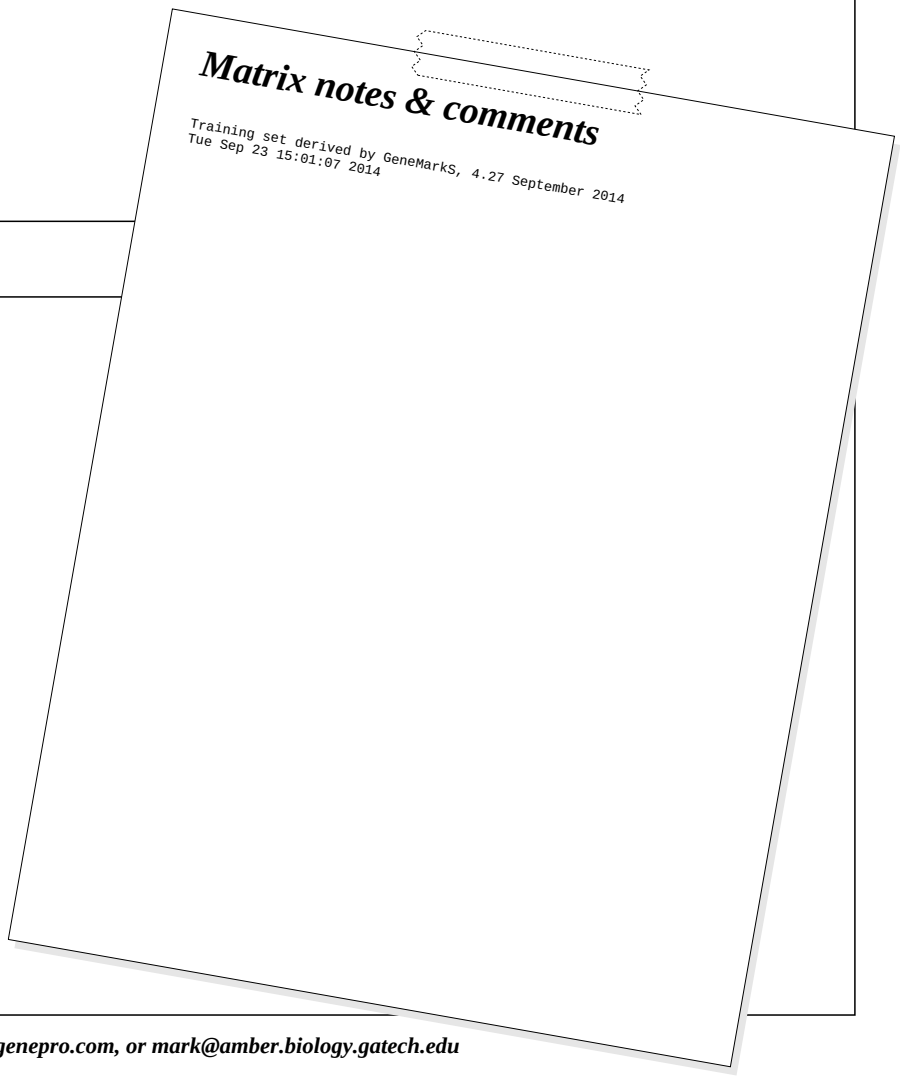
Version 2.5p (09.08.06)

PROGRAM INFORMATION

```
Sequence         :  seq_L_lactis_lactis
Analysis Date    :  11/29/24 at 4:40:54
Pages            :  6
Sequence Length  :  9557 bp
GC Content       :   31.57%

Window Length    :  120 bp
Window Step      :  12 bp
Threshold Value  :  0.500

PS-Version       :  1.2

GeneMark Options :  PostScript graph,
                    Mark ORFs / splice sites,
                    List ORFs,
                    List regions and/or splice sites,
```

MATRIX INFORMATION

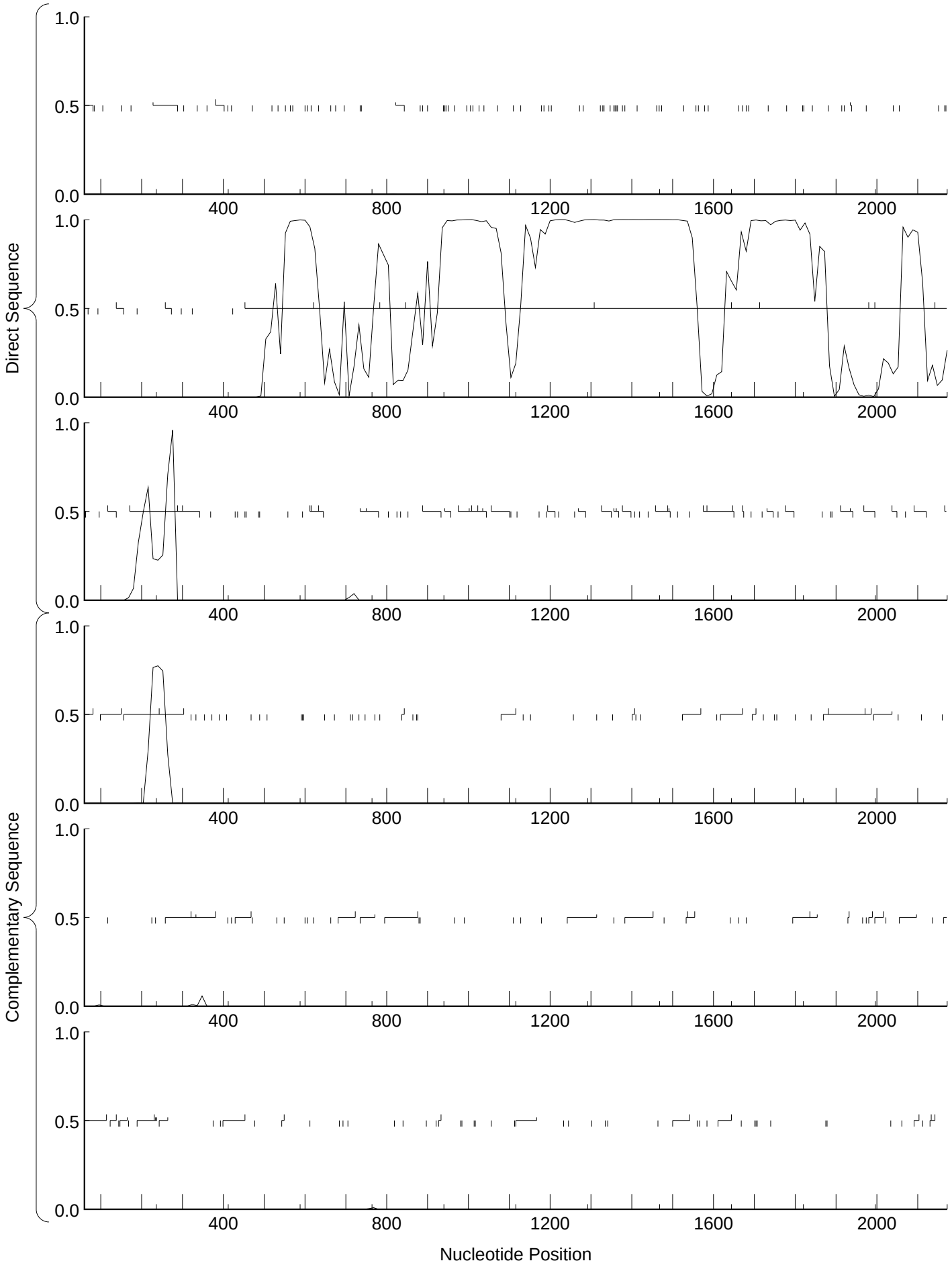
```
Matrix     :  Lactococcus_lactis_Il1403
Author     :  -

Order      :  4


  Send questions / comments to:
      Dr. M. Borodovsky
      Georgia Institute of Technology
      School of Biology
      Atlanta, GA  30332-0230
```

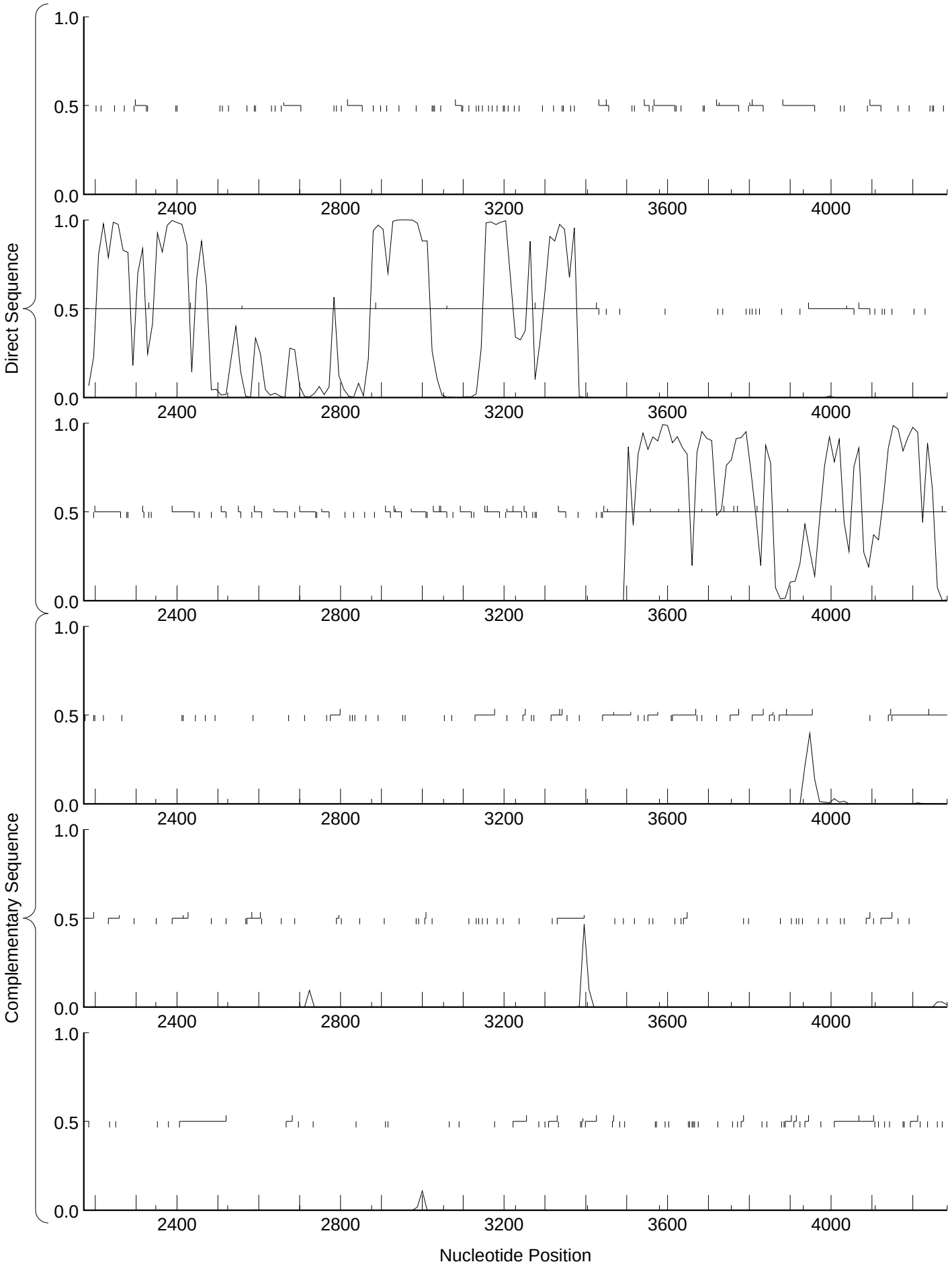


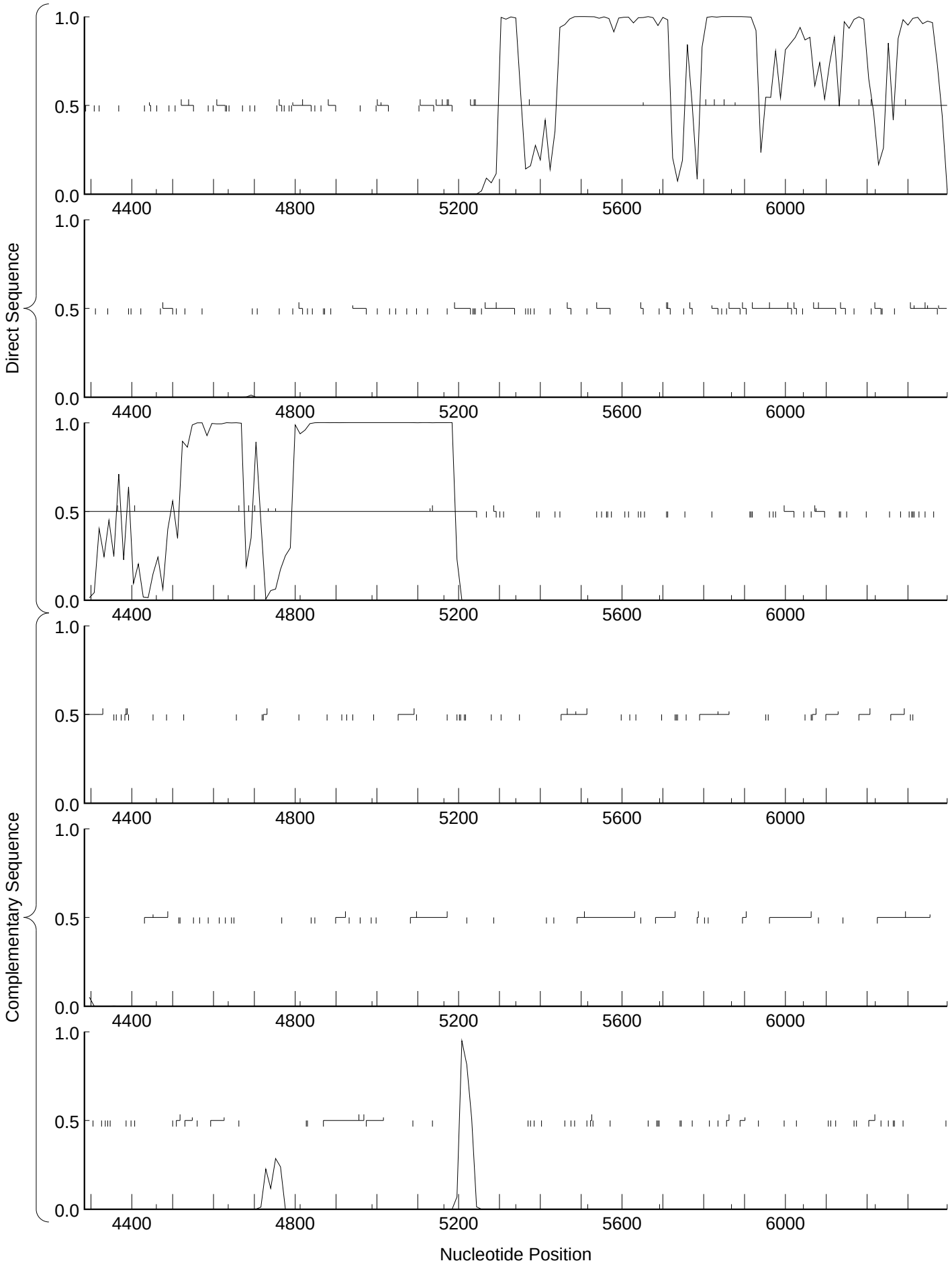
Matrix notes & comments

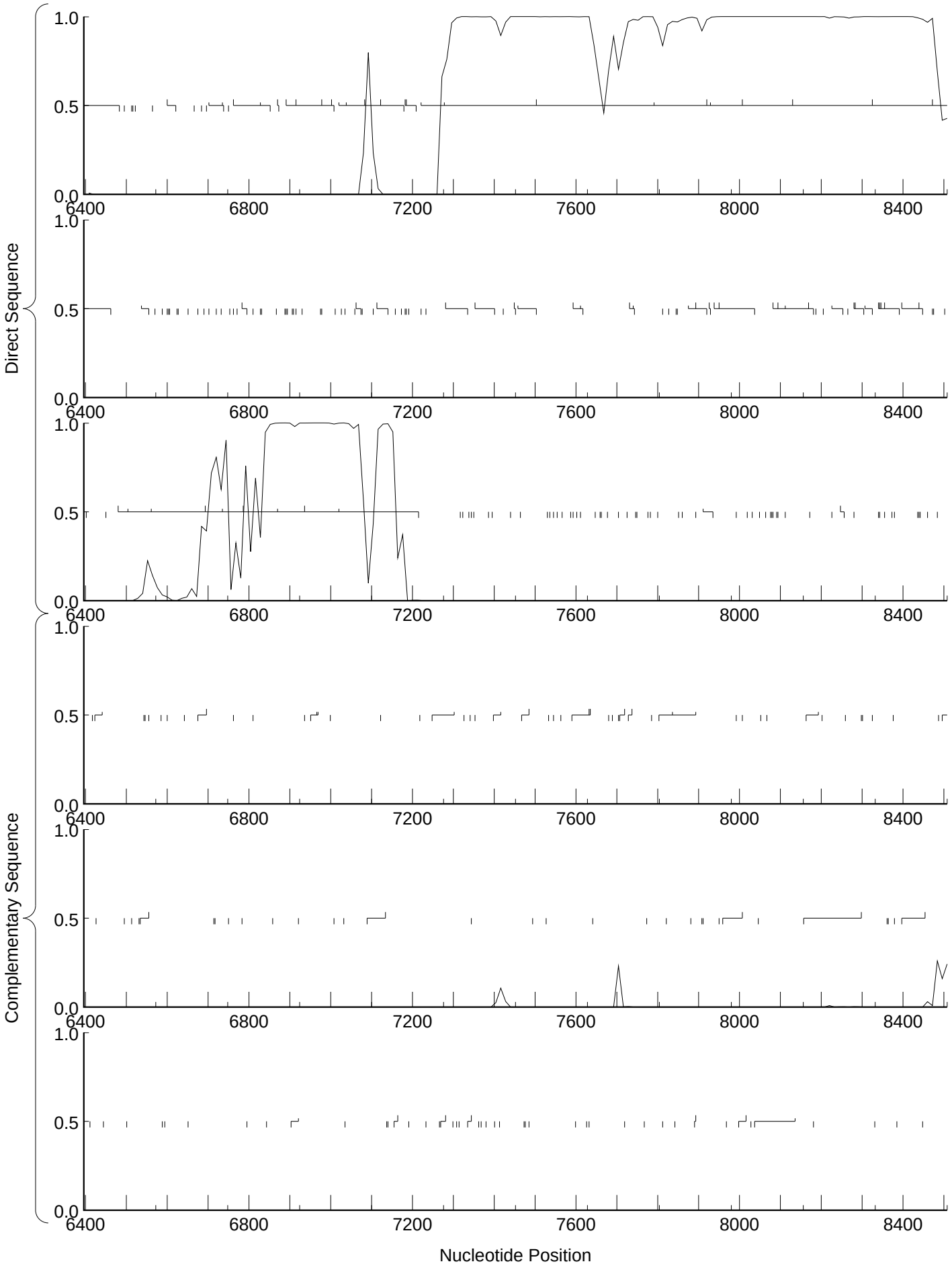
Training set derived by GeneMarkS, 4.27 September 2014
Tue Sep 23 15:01:07 2014

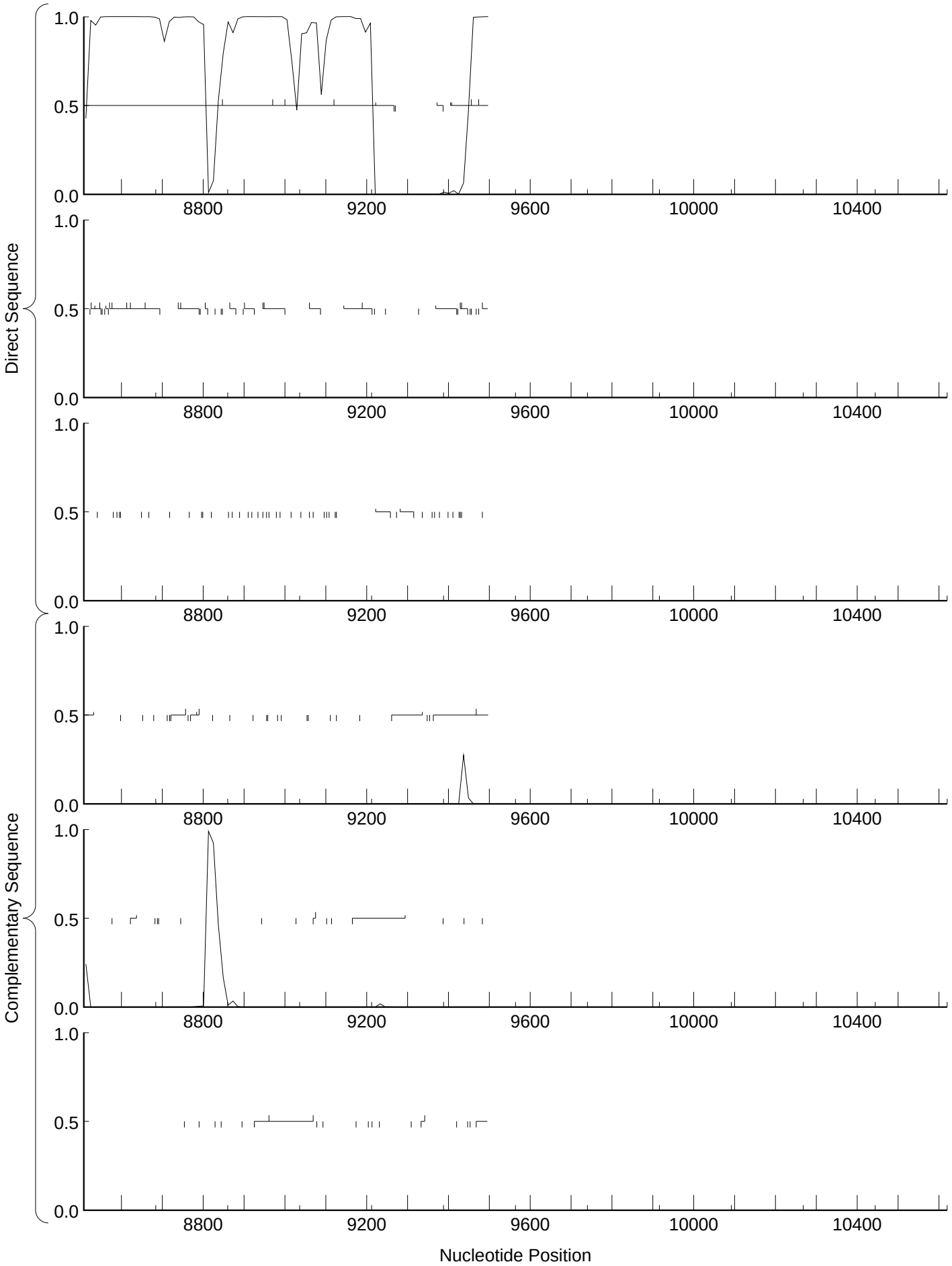
For help or information, send electronic mail to: custserv@genepro.com, or mark@amber.biology.gatech.edu


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
400
800
1200
1600
2000
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
2400
2800
3200
3600
4000
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
4400
4800
5200
5600
6000
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
6400
6800
7200
7600
8000
8400
Nucleotide Position


Direct Sequence
Complementary Sequence
1.0
0.5
0.0
8800
9200
9600
10000
10400
Nucleotide Position

Document 3 : graphe obtenu avec GeneMarkHMM sur la séquence d’intérêt

GeneMark

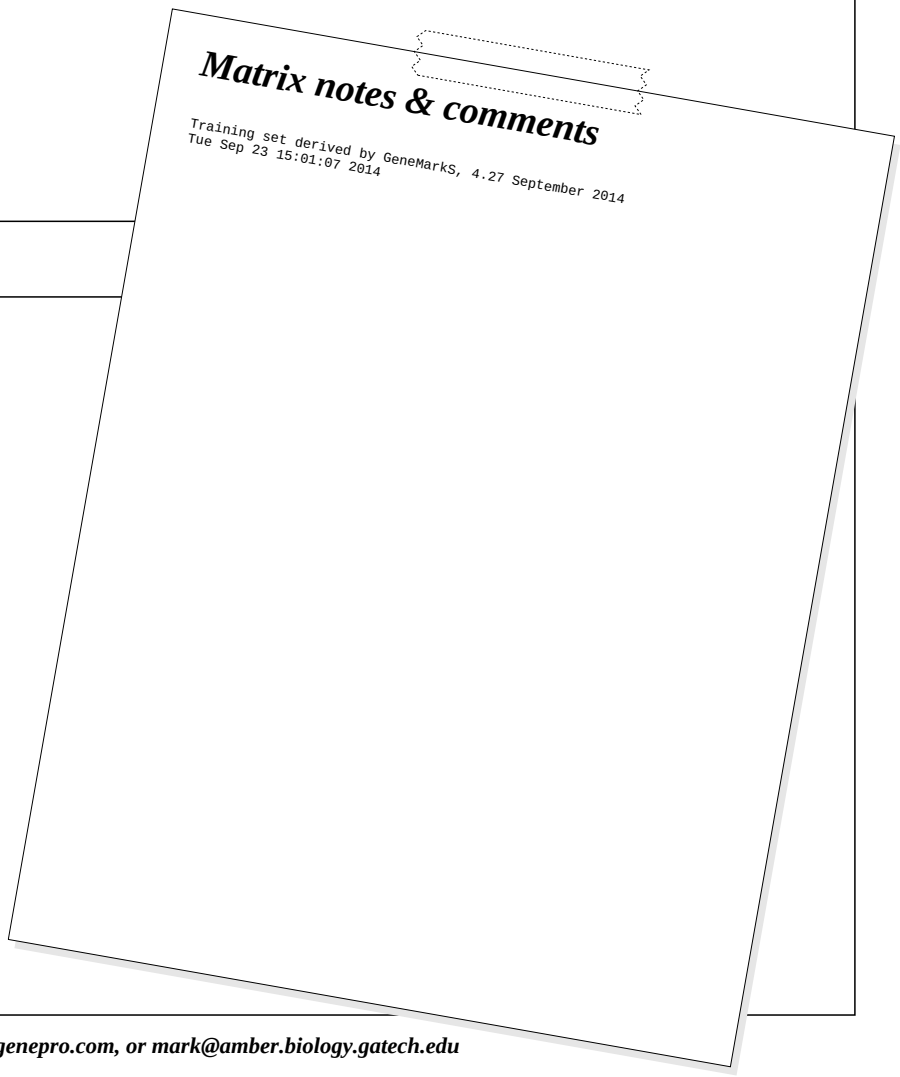
Version 2.5p (09.08.06)

PROGRAM INFORMATION

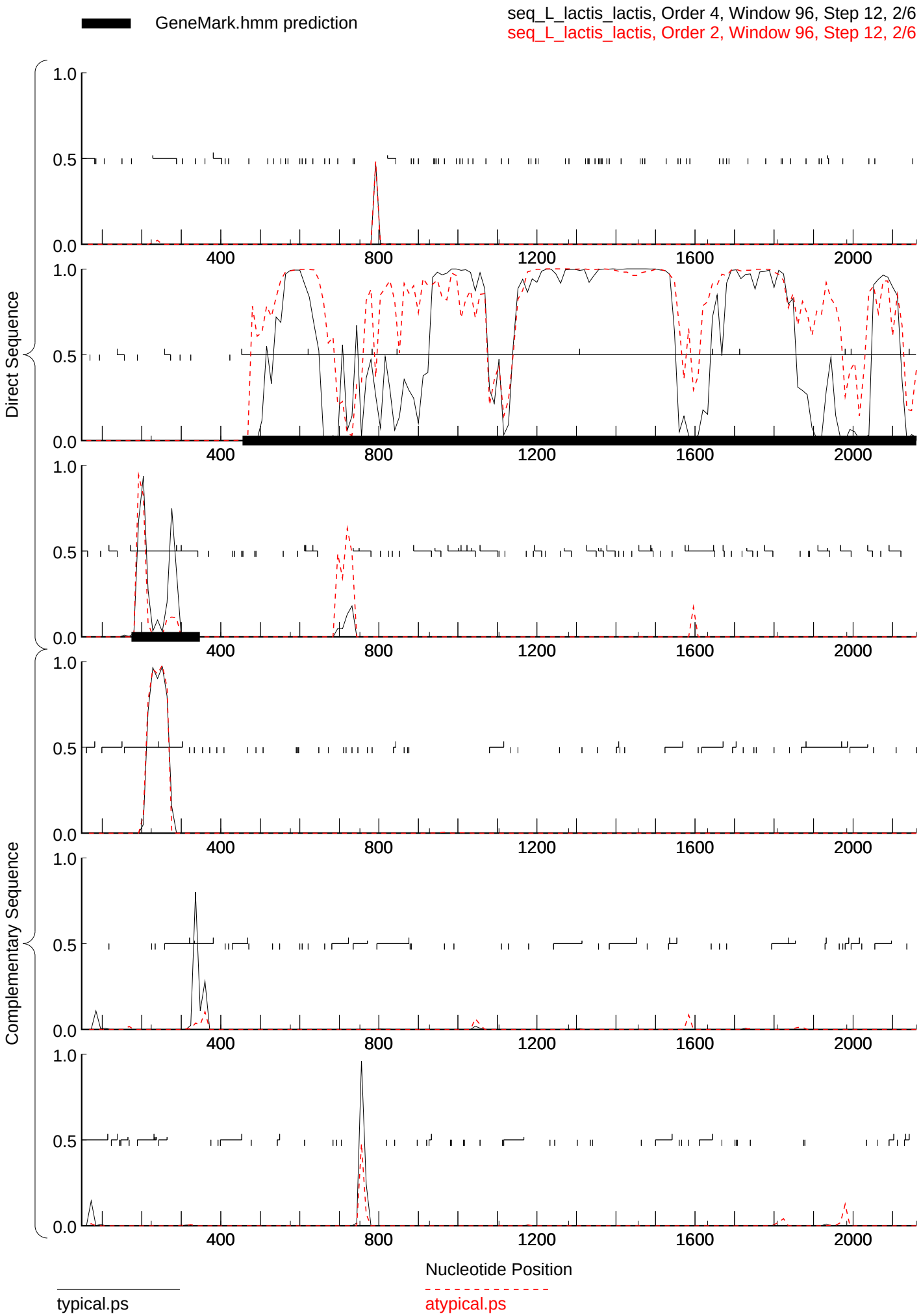
```
Sequence         :  seq_L_lactis_lactis
Analysis Date    :  11/29/24 at 4:52:54
Pages            :  6
Sequence Length  :  9557 bp
GC Content       :   31.57%

Window Length    :  96 bp
Window Step      :  12 bp
Threshold Value  :  0.500

PS-Version       :  1.2

GeneMark Options :  PostScript graph,
                    Mark ORFs / splice sites,
                    List ORFs,
                    List regions and/or splice sites,
```

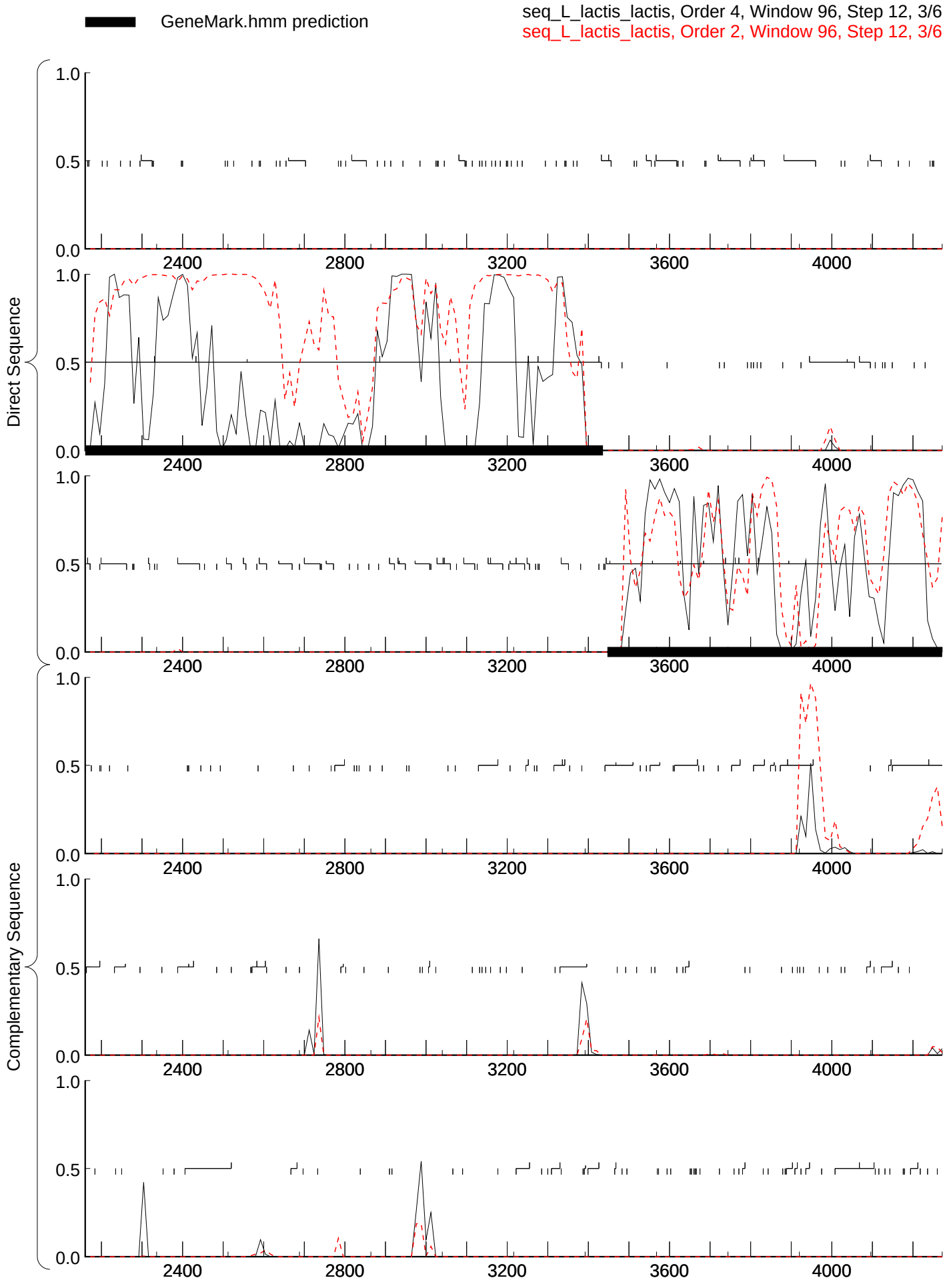
MATRIX INFORMATION

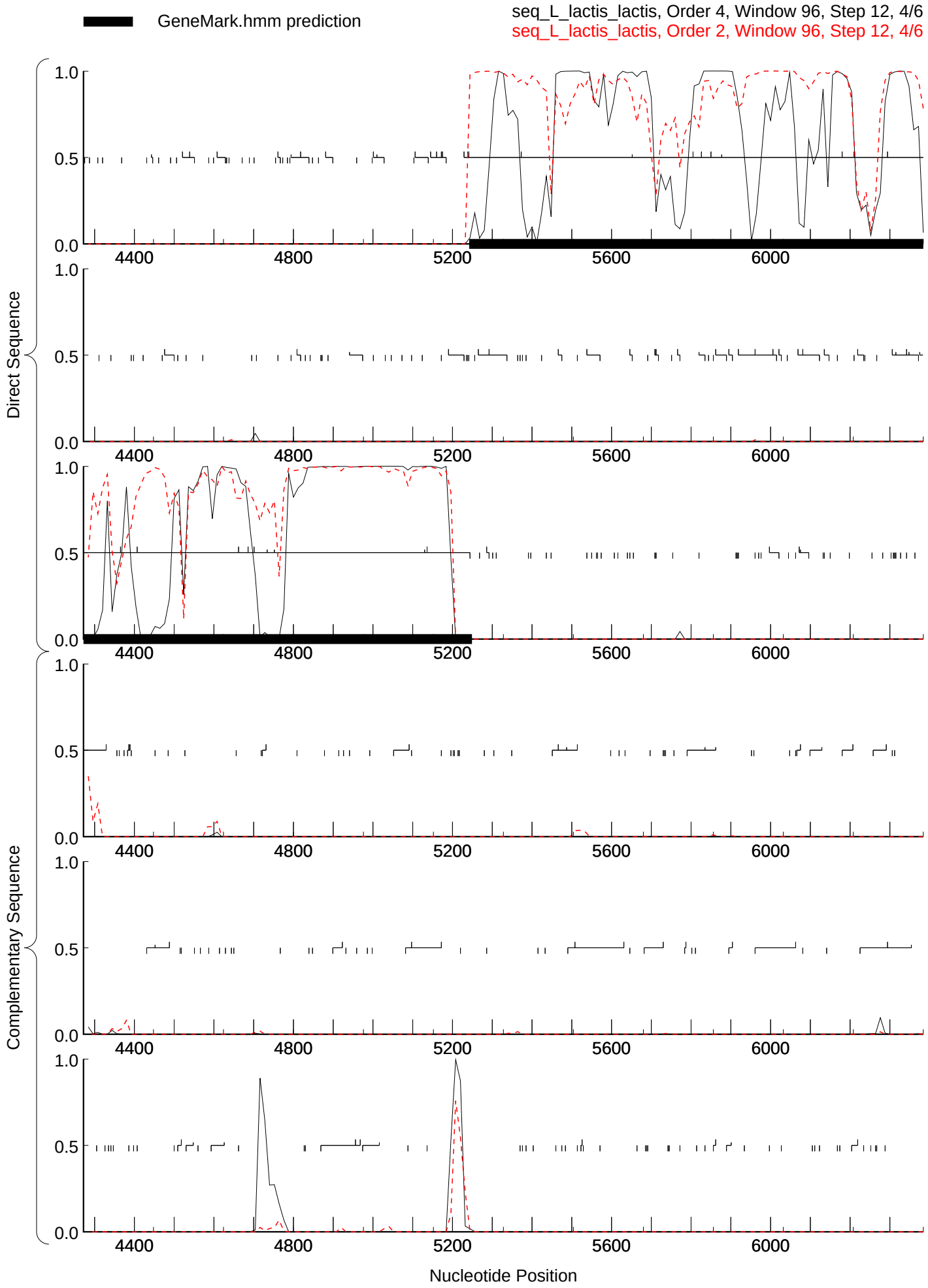
```
Matrix     :  Lactococcus_lactis_Il1403
Author     :  -

Order      :  4


  Send questions / comments to:
      Dr. M. Borodovsky
      Georgia Institute of Technology
      School of Biology
      Atlanta, GA  30332-0230
```

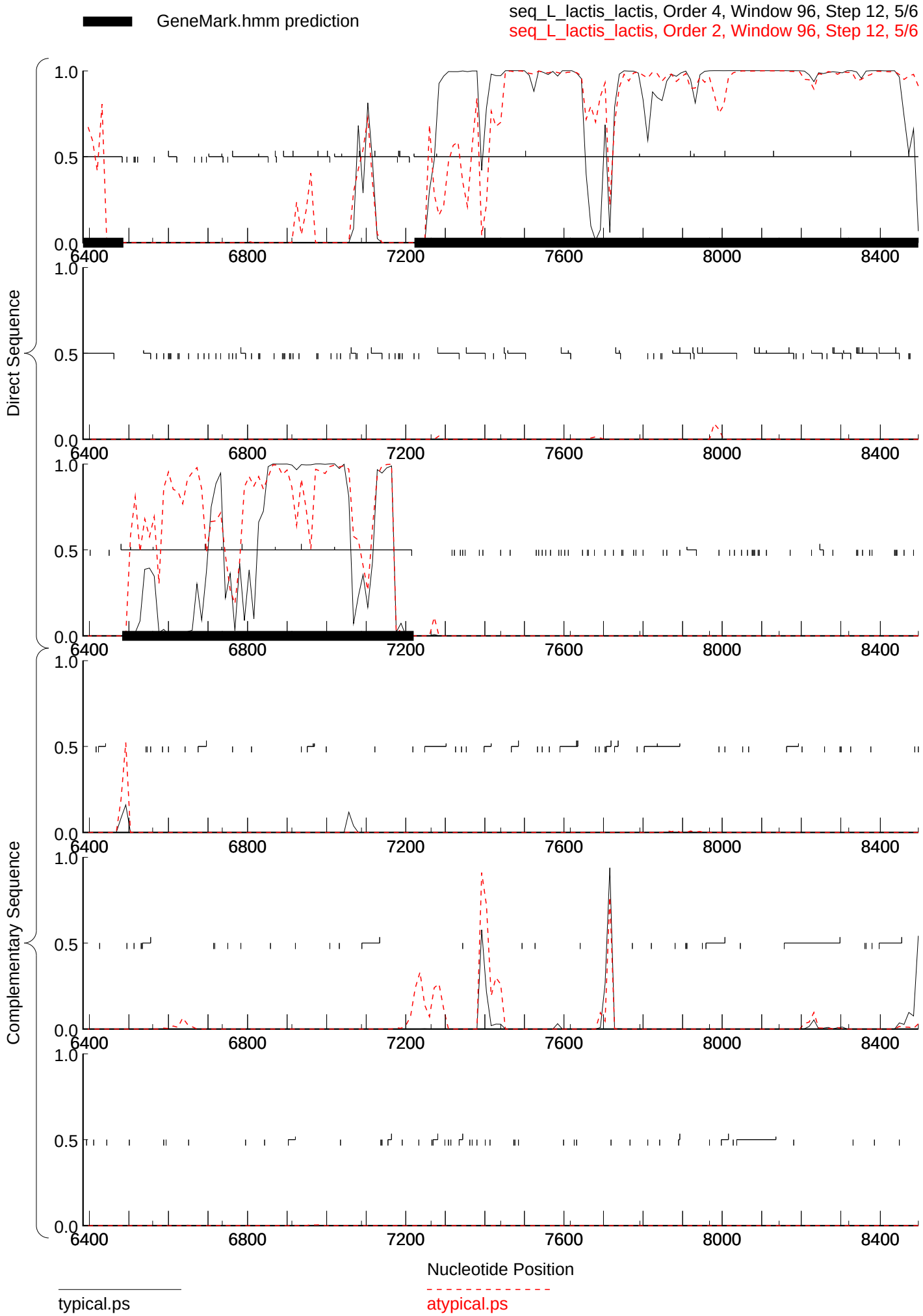


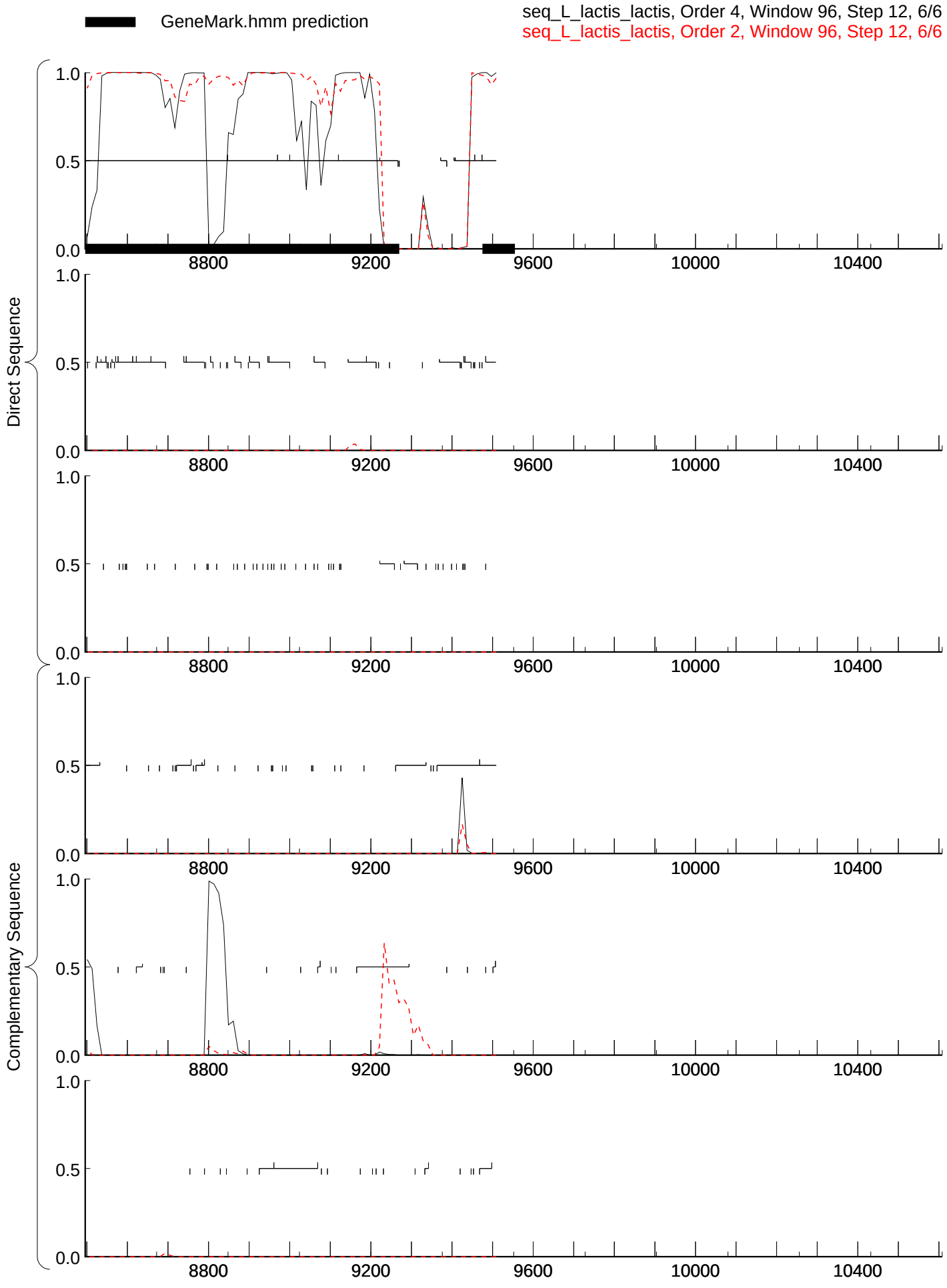
For help or information, send electronic mail to: custserv@genepro.com, or mark@amber.biology.gatech.edu


GeneMark.hmm prediction
seq_L_lactis_lactis, Order 4, Window 96, Step 12, 2/6
seq_L_lactis_lactis, Order 2, Window 96, Step 12, 2/6
Direct Sequence
Complementary Sequence
1.0
0.5
0.0
400
800
1200
1600
2000
Nucleotide Position
typical.ps
atypical.ps


GeneMark.hmm prediction
seq_L_lactis_lactis, Order 4, Window 96, Step 12, 3/6
seq_L_lactis_lactis, Order 2, Window 96, Step 12, 3/6
Direct Sequence
Complementary Sequence
1.0
0.5
0.0
2400
2800
3200
3600
4000
Nucleotide Position
typical.ps
atypical.ps


GeneMark.hmm prediction
seq_L_lactis_lactis, Order 4, Window 96, Step 12, 4/6
seq_L_lactis_lactis, Order 2, Window 96, Step 12, 4/6
Direct Sequence
Complementary Sequence
1.0
0.5
0.0
4400
4800
5200
5600
6000
Nucleotide Position
typical.ps
atypical.ps


GeneMark.hmm prediction
seq_L_lactis_lactis, Order 4, Window 96, Step 12, 5/6
seq_L_lactis_lactis, Order 2, Window 96, Step 12, 5/6
Direct Sequence
Complementary Sequence
1.0
0.5
0.0
6400
6800
7200
7600
8000
8400
Nucleotide Position
typical.ps
atypical.ps


GeneMark.hmm prediction
seq_L_lactis_lactis, Order 4, Window 96, Step 12, 6/6
seq_L_lactis_lactis, Order 2, Window 96, Step 12, 6/6
Direct Sequence
Complementary Sequence
1.0
0.5
0.0
8800
9200
9600
10000
10400
Nucleotide Position
typical.ps
atypical.ps

Document 4 : code python pour la conversion des fichiers en format GFF

- convert_to_GFF.py

programme principal permettant la lecture du fichier et l'écriture d'un fichier de sortie

- search_motif.py

module complémentaire permettant la recherche de motif spécifique afin de créer l'en-tête du fichier de sortie

- search_data.py

module complémentaire recherchant les différentes informations du fichier d'entrée et permettant la conversion au format GFF

- man.py

module complémentaire recensant les arguments, les erreurs possibles et les bonnes pratiques du programme (manuel d'utilisation)

filename : convert_to_GFF.py

```
#!/usr/bin/env python
import sys
import search_motif
import search_data
import man

'''
ERREURS possibles
'''
def Error_arg():
    """
    Displays an error message for invalid or insufficient arguments and
exits the program.

    Raises:
        SystemExit: The program exits immediately after displaying the
error message.
    """
    quit("\nERR_ARG ! Bad arguments given ... \nusage: convert_to_GFF.py
[file type : SFM, GENM, GENMH] [file path] [only if SFM : RBS/PROM/TERM]
[name output file (Default=output.GFF)]\n")

def Error_type():
    """
    Displays an error message for an invalid or unsupported file type and
exits the program.

    Raises:
        SystemExit: The program exits immediately after displaying the
error message.
    """
    quit("\nERR_TYPE ! Wrong type given ... \nusage: convert_to_GFF.py [file
type : SFM, GENM, GENMH] [file path] [only if SFM : RBS/PROM/TERM] [name
output file (Default=output.GFF)]\n")


'''
INITIALISATION du programme
'''
def initialise()->list[str]:
    '''
    Initializes the script parameters by retrieving command-line arguments.

    - Displays the name of the script being executed.
    - Checks for the presence of a category (RBS, Prom, or Term), a type
(GeneMark/GeneMarkHMM/ScanForMatches), an input file (required),
      and an output file (optional).
    - Returns a list containing the categorie, the type, the input file and
```

```
the output file.

    Expected usage:
        script.py [input_file] [output_file (optional)] [category
(RBS/Prom/Term)] [type (GENM/GENMH/SFM)]

    If the output file is not specified, "output.GFF" is used as the
default.
    If insufficient arguments are provided, an error message is displayed,
and the program terminates.

    Returns:
        list: [filename, fileout, categorie, type]
    '''
    if len(sys.argv) > 1 and (sys.argv[1] == "--help" or sys.argv[1] ==
"-h"):
        man.help()

    try :
        print(sys.argv[0],"is running ... ")
        type = sys.argv[1]
        filename = sys.argv[2]
        type = type.upper()

        if type=="SFM":
            try :
                categorie = sys.argv[3]
                categorie = categorie.upper()
                if categorie≠"RBS" and categorie≠"PROM" and
categorie≠"TERM":
                    Error_arg()
                try:
                    fileout = sys.argv[4]
                except:
                    fileout="output.GFF"
            except:
                Error_arg()
        else:
            categorie="None"
            try:
                fileout = sys.argv[3]
            except:
                fileout="output.GFF"
    except:
        Error_arg()

    return [filename, fileout, categorie, type]
```

```
'''
LECTURE du fichier input
'''
def lire(files:list[str])→str:
    """
    Reads the contents of a file.

    - Retrieves the file name from the first element of the input list.
    - Opens the file in read mode and reads its entire content.

    Args:
        files (list): A list where the first element is the name of the
file to be read.

    Returns:
        str: The content of the file as a string.
    """
    filename = files[0]
    print("Reading file", filename)
    with open(filename, "r") as input_file:
        input = input_file.read()
    return input


'''
TYPE du fichier input
'''
def ScanForMatches(files:list[str], input:str):
    """
    Writes the processed data from ScanForMatches file into an output file
in GFF format.
    """
    fileout=files[1]
    category = files[2]
    if category=="RBS":
        RBS=True
    else:
        RBS=False
    try:
        Date=search_motif.Date()
        Info = search_data.Create_Tab_SFM(input)
        taille = Info[0]
        DataFrame = Info[1]
    except:
        Error_type()

    with open(fileout, "w") as output_file:
```

```
        output_file.write("##gff-version 2\n##source-version
ScanForMatches")
        output_file.write("\n##date: "+Date[0]+" "+Date[1]+" "+Date[2]+"
"+Date[3]+":"+Date[4]+":"+Date[5]+" "+Date[6])
        output_file.write("\n# Sequence file name: -\n# Model file name:
-")
        output_file.write("\n# RBS: "+str(RBS))
        output_file.write("\n# Model information: -\n\n")

        for index in range (taille):

output_file.write(DataFrame[index][0]+"\tScanForMatches\t"+category+"\t"+Da
taFrame[index][1]+"\t"+DataFrame[index][2]+"\t.\t+\t.\tnote \"
"+DataFrame[index][3]+"\"\n")

def GeneMark(files:list[str], input:str):
    """
    Writes the processed data from GeneMark file into an output file in GFF
format.
    """
    fileout=files[1]
    try:
        SourceVersion = search_motif.SourceVersion_GM(input)
        Date = search_motif.Date()
        Seqfilename = search_motif.Seqfilename_GM(input)
        ModelInformation = search_motif.ModelInformation_GM(input)
        Seq = search_data.Seq_GM(input)
        Info = search_data.Create_Tab_GM(input)
        taille = Info[0]
        DataFrame = Info[1]
    except:
        Error_type()

    with open(fileout, "w") as output_file:
        output_file.write("##gff-version 2\n##source-version GeneMark
"+SourceVersion)
        output_file.write("\n##date: "+Date[0]+" "+Date[1]+" "+Date[2]+"
"+Date[3]+":"+Date[4]+":"+Date[5]+" "+Date[6])
        output_file.write("\n# "+Seqfilename)
        output_file.write("\n# Model file name: -")
        output_file.write("\n# RBS: -")
        output_file.write("\n# Model information:
"+ModelInformation+"\n\n")

        for index in range (taille):

output_file.write(Seq+"\tGeneMark\t"+DataFrame[index][6]+"\t"+str(DataFrame
[index][1])+"\t"+str(DataFrame[index][2])+"\t?\t"+DataFrame[index][0]+"\t.\
```

```
t"+str(DataFrame[index][7])+"\n")
        output_file.write("\n")

def GeneMarkHMM(files:list[str], input:str):
    """
    Writes the processed data from GeneMarkHMM file into an output file in
GFF format.
    """
    fileout=files[1]
    try:
        SourceVersion = search_motif.SourceVersion_GMH(input)
        Date = search_motif.Date_GMH(input)
        Seqfilename = search_motif.Seqfilename_GMH(input)
        Model = search_motif.Model_GMH(input)
        RBS = search_motif.RBS_GMH(input)
        ModelInformation = search_motif.ModelInformation_GMH(input)
        Seq = search_data.Seq_GMH(input)
        Info = search_data.Create_Tab_GMH(input)
        taille = Info[0]
        DataFrame = Info[1]
    except:
        Error_type()

    with open(fileout, "w") as output_file:
        output_file.write("##gff-version 2\n##source-version
"+SourceVersion[0]+" "+SourceVersion[1])
        output_file.write("\n##date:"+Date)
        output_file.write("\n# "+Seqfilename)
        output_file.write("\n# "+Model)
        output_file.write("\n# "+RBS)
        output_file.write("\n# "+ModelInformation+"\n\n")

        for index in range (taille):

output_file.write(Seq+"_"+str((index+1))+"\tGeneMark.hmm\tCDS\t"+DataFrame[
index][1]+"\t"+DataFrame[index][2]+"\t?\t"+DataFrame[index][0]+"\t.\tgene
GMH_CDS_"+str((index+1))+"\n")


'''
ECRITURE du fichier output
'''
def ecrire(files:list[str], input:str):
    """
    Writes the processed data into an output file in GFF format.

    - Retrieves the output file name from the second element of the `files`
list.
```

```
    - Extracts metadata and sequence data from the input using helper
functions (module search_motif and search_data).
    - Formats the data according to the GFF version 2 specification.
    - Writes the formatted data to the specified output file.

    Args:
        files (list): A list containing file names, where the second
element is the output file name
                    and the third element is the category.
        input (str): The content of the input file to be processed and
written to the output.

    Raises:
        SystemExit: If an error occurs while extracting data, an error
message is displayed,
                   and the program terminates.

    Output:
        The output file is written with the following structure:
        - Metadata: GFF version, source version, date, sequence file name,
model information.
        - Sequence data in GFF format, including start and end positions,
strand, and other details.
    """
    type=files[3]
    fileout=files[1]
    print("Writting", fileout)
    if type=="GENM":
        GeneMark(files, input)
    elif type=="GENMH":
        GeneMarkHMM(files, input)
    else:
        ScanForMatches(files, input)


if __name__ == "__main__":
    files=initialise()
    input=lire(files)
    ecrire(files, input)
    print("Work done !")
```

filename : search_motif.py

```
#!/usr/bin/env python
import re
import datetime

def SourceVersion_GMH(input:str)→tuple[str,str]:
    """
    Extracts the source and version information from the input data using a
regular expression.

    - Searches the input string for a pattern that matches a source name
(24 characters),
      followed by an additional string, and then a version number (format:
X.XXX).
    - If a match is found, it returns the source name and version, else
returns empty strings
      for both source and version.

    Args:
        input (str): The input data to search for the source and version
information.

    Returns:
        list: A list containing two elements:
            - The source name as a string.
            - The version number as a string.
    """
    motif_sourceversion =
re.compile(r"(?P<source>.{24})(?P<other>.{10})(?P<version>\d{1,2}\.\d{1,3})
")
    matchSourceversion = re.search(motif_sourceversion, input)

    if matchSourceversion is not None:
        source=matchSourceversion.group('source')
        version=matchSourceversion.group('version')
    else:
        source=""
        version=""

    return (source, version)

def Date_GMH(input:str)→str:
    """
    Extracts the date information from the input data using a regular
expression.

    - Searches the input string for the pattern "Date: <date>", where the
date is a 24-25 character long string.
    - If a match is found, it returns the extracted date, else returns an
```

```
empty string.

    Args:
        input (str): The input data to search for the date information.

    Returns:
        str: The extracted date as a string, or an empty string if no date
is found.
    """
    motif_date = re.compile(r"(Date:)(?P<date>.{24,25})")
    matchDate = re.search(motif_date, input)

    if matchDate is not None:
        date=matchDate.group('date')
    else:
        date=""

    return date

def Seqfilename_GMH(input:str)→str:
    """
    Extracts the sequence file name from the input data using a regular
expression.

    - Searches the input string for a pattern matching "Sequence file name:
<filename>.fna".
    - If a match is found, it returns the sequence file name, else returns
an empty string.

    Args:
        input (str): The input data to search for the sequence file name.

    Returns:
        str: The extracted sequence file name as a string (e.g.,
"example.fna"), or an empty string if no match is found.
    """
    motif_Seqfilename =
re.compile(r"(?P<Seqfilename>Sequence\sfile\sname:.{1,}\.fna)")
    matchSeqfilename = re.search(motif_Seqfilename, input)

    if matchSeqfilename is not None:
        Seqfilename=matchSeqfilename.group('Seqfilename')
    else:
        Seqfilename=""

    return Seqfilename

def Model_GMH(input:str)→str:
```

```
    """
    Extracts the model file name from the input data using a regular
expression.

    - Searches the input string for a pattern matching "Model file name:
<filename>".
    - If a match is found, it returns the model file name, else returns an
empty string.

    Args:
        input (str): The input data to search for the model file name.

    Returns:
        str: The extracted model file name as a string, or an empty string
if no match is found.
    """
    motif_Model = re.compile(r"(?P<Model>Model\sfile\sname:.{1,})")
    matchModel = re.search(motif_Model, input)

    if matchModel is not None:
        Model=matchModel.group('Model')
    else:
        Model=""

    return Model

def RBS_GMH(input:str)→str:
    """
    Extracts the RBS (Ribosome Binding Site) information from the input
data using a regular expression.

    - Searches the input string for a pattern matching "RBS: <sequence>",
where <sequence> is "true" or "false".
    - If a match is found, it returns true or false, else returns an empty
string.

    Args:
        input (str): The input data to search for the RBS information.

    Returns:
        str: The extracted RBS information as a string, or an empty string
if no match is found.
    """
    motif_RBS = re.compile(r"(?P<RBS>RBS:\s\w{4,5})")
    matchRBS = re.search(motif_RBS, input)

    if matchRBS is not None:
        RBS=matchRBS.group('RBS')
```

```
    else:
        RBS=""

    return RBS

def ModelInformation_GMH(input:str)→str:
    """
    Extracts the model information from the input data using a regular
expression.

    - Searches the input string for a pattern matching "Model information:
<information>".
    - If a match is found, it returns the model information, else returns
an empty string.

    Args:
        input (str): The input data to search for the model information.

    Returns:
        str: The extracted model information as a string, or an empty
string if no match is found.
    """
    motif_ModelInformation =
re.compile(r"(?P<ModelInformation>Model\sinformation:.{1,})")
    matchModelInformation = re.search(motif_ModelInformation, input)

    if matchModelInformation is not None:
        ModelInformation=matchModelInformation.group('ModelInformation')
    else:
        ModelInformation=""

    return ModelInformation

def SourceVersion_GM(input:str)→str:
    """
    Extracts the source version information from the input data using a
regular expression.

    - Searches the input string for a pattern matching "Training set
derived by <text> <version>", where the version
      is in the format X.XXX.
    - If a match is found, it returns the extracted version number, else
returns an empty string.

    Args:
        input (str): The input data to search for the source version
information.
```

```
    Returns:
        str: The extracted version number as a string, or an empty string
if no match is found.
    """
    motif_sourceversion =
re.compile(r"Training\sset\sderived\sby.{1,}(?P<version>\d{1,2}\.\d{1,3})")
    matchSourceversion = re.search(motif_sourceversion, input)

    if matchSourceversion is not None:
        version=matchSourceversion.group('version')
    else:
        version=""

    return version

def Seqfilename_GM(input:str)→str:
    """
    Extracts the sequence file name from the input data using a regular
expression.

    - Searches the input string for a pattern matching "Sequence file:
<filename>.fna".
    - If a match is found, it returns the sequence file name, else returns
an empty string.

    Args:
        input (str): The input data to search for the sequence file name.

    Returns:
        str: The extracted sequence file name as a string (e.g.,
"example.fna"), or an empty string if no match is found.
    """
    motif_Seqfilename =
re.compile(r"(?P<Seqfilename>Sequence\sfile:.{1,}\.fna)")
    matchSeqfilename = re.search(motif_Seqfilename, input)

    if matchSeqfilename is not None:
        Seqfilename=matchSeqfilename.group('Seqfilename')
    else:
        Seqfilename=""

    return Seqfilename

def ModelInformation_GM(input:str)→str:
    """
    Extracts the model information from the input data using a regular
expression.
```

```
    - Searches the input string for a pattern matching "Matrix: <model
information>".
    - If a match is found, it returns the model information, else returns
an empty string.

    Args:
        input : The input data to search for the model information.

    Returns:
        The extracted model information as a string, or an empty string if
no match is found.
    """
    motif_ModelInformation =
re.compile(r"Matrix:\s(?P<ModelInformation>.{1,})")
    matchModelInformation = re.search(motif_ModelInformation, input)

    if matchModelInformation is not None:
        ModelInformation=matchModelInformation.group('ModelInformation')
    else:
        ModelInformation=""

    return ModelInformation

def Date()→list[str]:
    """
    Retrieves the current date and time in a specific formatted structure.
    - Uses the current system date and time to generate a formatted string.
    - Returns the formatted date as a list in the order: [weekday, month,
day, hour, minute, second, year].
    The weekday and month names are abbreviated (e.g., "Mon", "Jan").

    Returns:
        A list containing the formatted date and time as strings:
            - [weekday, month, day, hour, minute, second, year]
            For example: ["Mon", "Dec", "08", "15", "45", "30", "2024"]
    """
    date = datetime.datetime.today() #year, month, day, hour, min, sec
    wkday = date.isoweekday()
    semaine = {1: "Mon", 2: "Tue", 3: "Wed", 4: "Thu", 5: "Fri", 6: "Sat",
7: "Sun"}
    annee = {1: "Jan", 2: "Feb", 3: "Mar", 4: "Apr", 5: "May", 6: "Jun", 7:
"Jul", 8: "Aug", 9: "Sep", 10: "Oct", 11: "Nov", 12: "Dec"}
    date = str(date)
    y_m = date.split("-") # y=[0], m=[1]
    d = y_m[2].split(" ") # d=[0]
    h_mn = d[1].split(":") # h=[0], mn=[1]
    s = h_mn[2].split(".") # s=[0]
```

```
    LineDate = [semaine[int(wkday)], annee[int(y_m[1])], d[0], h_mn[0],
h_mn[1], s[0], y_m[0]]
    return LineDate
```

filename : search_data.py

```
#!/usr/bin/env python
import re

'''
Allow to convert GeneMarkHMM LST file to GFF file
'''
def Seq_GMH(input:str)→str:
    """
    Extracts the sequence definition line from the input data using a
regular expression.

    - Searches the input string for a line matching the pattern "FASTA
definition line: <sequence>".
    - If a match is found, extracts and returns the sequence portion, else
returns an empty string.

    Args:
        input (str): The input data to search for the sequence definition
line.

    Returns:
        str: The extracted sequence definition line, or an empty string if
no match is found.
    """
    motif_Seq =
re.compile(r"(?P<Other>FASTA\sdefinition\sline:\s)(?P<Seq>.{1,})")
    matchSeq = re.search(motif_Seq, input)

    if matchSeq is not None:
        Seq=matchSeq.group('Seq')
    else:
        Seq=""

    return Seq

def Create_Tab_GMH(input:str)→list:
    """
    Parses input data to create a dictionary of sequence features with
their strand, left, and right endpoints.

    - Iteratively searches the input for lines matching the pattern:
      "<number> <strand> <left-end> <right-end>".
    - Stores each match as a list in a dictionary, where keys are
sequential indices (starting from 0).
    - Continues searching for matches until no further lines are found.

    Args:
        input (str): The input data to parse for sequence features.
```

```
    Returns:
        list: A list containing:
            - The total number of features found (int).
            - A dictionary (dict) with sequential indices as keys and
feature details as values,
              where each value is a list: [strand, left-end, right-end].
    """
    tab, n = {}, 1

    motifTab =
re.compile(re.escape(str(n))+r"\s{1,}(?P<Strand>[+,-])\s{1,}(?P<LeftEnd><{0
,1}\d{1,10})\s{1,}(?P<RightEnd>>{0,1}\d{1,10})")
    matchLine = re.search(motifTab, input)
    if matchLine is not None:
        Line = [matchLine.group("Strand"), matchLine.group("LeftEnd"),
matchLine.group("RightEnd")]

    while matchLine is not None :
        tab[n-1]=Line
        n+=1
        motifTab =
re.compile(re.escape(str(n))+r"\s{1,}(?P<Strand>[+,-])\s{1,}(?P<LeftEnd><{0
,1}\d{1,10})\s{1,}(?P<RightEnd>>{0,1}\d{1,10})")
        matchLine = re.search(motifTab, input)
        if matchLine is not None:
            Line = [matchLine.group("Strand"), matchLine.group("LeftEnd"),
matchLine.group("RightEnd")]

    return [n-1,tab]


'''
Allow to convert GeneMark file to GFF file
'''
def Seq_GM(input:str)→str:
    """
    Extracts the sequence information from the input data using a regular
expression.

    - Searches the input string for a line matching the pattern "Sequence:
<sequence>".
    - If a match is found, extracts and returns the sequence portion, else
returns an empty string.

    Args:
        input (str): The input data to search for the sequence information.
```

```
    Returns:
        str: The extracted sequence, or an empty string if no match is
found.
    """
    motif_Seq = re.compile(r"(?P<Other>Sequence:\s)(?P<Seq>.{1,})")
    matchSeq = re.search(motif_Seq, input)

    if matchSeq is not None:
        Seq=matchSeq.group('Seq')
    else:
        Seq=""

    return Seq

def Create_Tab_GM(input:str)→list:
    """
    Parses GeneMark input data to create a structured table of gene
information.

    - Identifies features such as strand, left and right ends, probability,
and sequence type (CDS/ATG).
    - Iteratively matches data lines using a regular expression and stores
information in a dictionary.
    - Handles duplicate features, selecting the one with the highest
probability and marking it as an "ATG".
    - Generates annotations for each feature, adding gene-specific details.
    - Creates a separate dictionary for features marked as "ATG".

    Args:
        input (str): The GeneMark output data to parse.

    Returns:
        list: A list containing:
            - The total number of valid features (`taille`) as an integer.
            - A dictionary (`tab`) with indices as keys and feature details
as values.
              Each value is a list with the following structure:
              [strand, left-end, right-end, CDS index, sub-index,
probability, feature type (CDS/ATG), annotation].
    """
    tab, taille, index, i = [], 0, 0, 0
    Line = ["0-Strand", "1-LeftEnd", "2-RightEnd", "3-IndexCDS", "4-Index",
"5-Proba", "6-CDS/ATG", "7-note"]
    motifTab =
re.compile(r"\s{1,}(?P<LeftEnd><{0,1}\d{1,10})\s{1,}(?P<RightEnd>>{0,1}\d{1
,10})\s{1,}(?P<Strand>\w{6,10})\s{1,}(?P<Other>fr\s\d\s{1,}\d.\d{2})\s{1,}(
?P<Proba>.{4})")
    matchLine = re.finditer(motifTab, input)
```

```
    if matchLine is not None :
        for element in matchLine:
            if element.group("Strand")=="direct":
                strand="+"
            else:
                strand="-"
            if element.group("LeftEnd")≠Line[1] and
element.group("RightEnd")≠Line[2]:
                index+=1
                i=0
            i+=1
            Line = [strand, element.group("LeftEnd"),
element.group("RightEnd"), index, i, element.group("Proba"), "CDS", "."]
            if float(element.group("Proba"))≤1 or
element.group("Proba")==".....":
                tab.append(Line)
                taille+=1

    proba, positionATG, position = 0, 0, 0
    while position<taille:
        try:
            if tab[position][3]==tab[position+1][3] :
                if float(tab[position][5])>proba:
                    proba=float(tab[position][5])
                    positionATG=position
            else:
                if float(tab[position][5])>proba:
                    positionATG=position
                tab[positionATG][6]="ATG"
                positionATG=position
                proba=0
        except:
            if float(tab[position][5])>proba:
                positionATG=position
            tab[positionATG][6]="ATG"
        position+=1

    position=0
    while position<taille:
        if tab[position][6]=="ATG":
            new_line = tab[position].copy()
            tab.insert(position+1, new_line)
            tab[position+1][6]="CDS"
            taille+=1
            tab[position][7]="."
            tab[position][2]=int(tab[position][1])+2
        else:
```

```
            tab[position][7]="gene
GM_CDS_"+str(tab[position][3])+"."+str(tab[position][4])
        position+=1

    return [taille,tab]


'''
Allow to convert ScanForMatches file to GFF file
'''
def Create_Tab_SFM(input:str)→list:
    """
    Parses Sequence Feature Model (ScanForMatches) data and creates a
structured table of sequence features.

    - Uses a regular expression to match sequence features in the input
data, extracting the sequence name,
      left and right endpoints, and associated notes. Stores each feature
as a list.
    - Creates a dictionary where each key is a feature index, and each
value is the corresponding feature details.

    Args:
        input (str): The Sequence Feature Model data to parse.

    Returns:
        list: A list containing:
            - The total number of features (`n`) as an integer.
            - A dictionary (`tab`) where each key is an index and each
value is a list with the structure:
              [sequence name, left-end, right-end, note].
    """
    tab, n = {}, 0

    motifTab =
re.compile(r">(?P<Seq>.{1,}):\[(?P<LeftEnd>\d{1,}),(?P<RightEnd>\d{1,})\]\n
(?P<Note>[^>\n]{1,})")
    matchLine = re.finditer(motifTab, input)
    if matchLine is not None:
        for element in matchLine:
            tab[n] = [element.group("Seq"), element.group("LeftEnd"),
element.group("RightEnd"), element.group("Note")]
            n+=1

    return [n,tab]
```

filename : man.py

```
#!/usr/bin/env python
import sys

def pause():
    """
    Pauses the program execution and waits for user input to proceed or
quit.
    - Displays a prompt asking the user to press Enter to continue or 'q'
to quit.
    - Removes the prompt from the terminal after the user inputs a
response.
    - If the user enters 'q' (case insensitive), the program terminates.
    - If the user presses any key, the program continues execution.

    Raises:
        SystemExit: If the user enters 'q', the program exits.
    """
    response = input("\nPress any key to continue or 'q' to quit: ")
    sys.stdout.write("\033[F\033[K") # \033[F : Déplace le curseur à la
ligne précédente et \033[K : Efface la ligne.
    if response.lower() == 'q':
        quit()

def help():
    """
    Displays a paginated help message, pausing for user input between
sections.

    Raises:
        SystemExit: at the end of the function, the program exits.
    """
    print(
    "="*40,
    "User Manual for 'convert_to_GFF.py'".center(10),
    "="*40,
    "\n\n",
    "-"*40,
    "Purpose".center(10),
    "-"*40,"\n",
    "The 'convert_to_GFF.py' script converts gene sequence data from three
possible formats:\n\
        -GeneMark (GENM)\n\
        -GeneMarkHMM (GENMH)\n\
        -ScanForMatches (SFM)\n\
    into the standard GFF (General Feature Format) file format. This
facilitates the integration of gene annotations into various bioinformatics
tools.\n")
```

```
    pause()

    print(
    "-"*40,
    "Features".center(10),
    "-"*40,"\n",
    "Supported input types:\n\
        GENM : GeneMark format\n\
        GENMH: GeneMarkHMM format\n\
        SFM: ScanForMatches format with an optional category (e.g., 'RBS',
'PROM', 'TERM')\n\
    Automatic metadata extraction:\n\
        Extracts key details like source version, model information,
sequence file name and date\n\
    Error Handling:\n\
        Validates input arguments and file formats with informative error
messages.\n\
    Output Format:\n\
        Produces a well-structured GFF v2 file.\n")

    pause()

    print(
    "-"*40,
    "Usage".center(10),
    "-"*40,"\n",
    "python convert_to_GFF.py [file type: SFM, GENM, GENMH] [input file
path] [category (only for SFM: RBS/PROM/TERM)] [output file name
(optional)]\n")

    pause()

    print(
    "-"*40,
    "Arguments".center(10),
    "-"*40,"\n",
    "file type (Required):\n\
        Defines the type of the input file: 'SFM', 'GENM', or 'GENMH'.\n\
    input file path (Required):\n\
        Path to the input file containing the gene sequence data.\n\
    category (Optional; only for SFM):\n\
        Specifies the category for SFM files: 'RBS', 'PROM', or 'TERM'.\n\
    output file name (Optional):\n\
        Name of the output GFF file. If not provided, defaults to
'output.GFF'.\n")

    pause()
```

```
    print(
    "-"*40,
    "Examples".center(10),
    "-"*40,"\n",
    "Convert GeneMark file:\n\
        'python convert_to_GFF.py GENM input_file.lst output.GFF'\n\
    Convert GeneMarkHMM file:\n\
        'python convert_to_GFF.py GENMH input_file.lst output.GFF'\n\
    Convert ScanForMatches file with category:\n\
        'python convert_to_GFF.py SFM input_file.sfm RBS output.GFF'\n")

    pause()

    print(
    "-"*40,
    "Output".center(10),
    "-"*40,"\n",
    "The script generates a GFF v2 file containing:\n\
        Metadata:\n\
            GFF version\n\
            Source version and date\n\
            Sequence file name, model file name, and RBS status\n\
            Model information\n\
        Annotations:\n\
            Gene locations with features like start, end, strand, and type
(CDS/ATG)\n\
            Notes for each gene\n")

    pause()

    print(
    "-"*40,
    "Error Messages".center(10),
    "-"*40,"\n",
    "Argument Errors:\n\
        If required arguments are missing or invalid:\n\
            'ERR_ARG ! Bad arguments given ... \n\
            usage: convert_to_GFF.py [file type : SFM, GENM, GENMH] [file
path] [only if SFM : RBS/PROM/TERM] [name output file
(Default=output.GFF)]'\n\
    File Format Errors:\n\
        If the input file format is incorrect or unsupported:\n\
            'ERR_TYPE ! Wrong type given ... \n\
            usage: convert_to_GFF.py [file type : SFM, GENM, GENMH] [file
path] [only if SFM : RBS/PROM/TERM] [name output file
(Default=output.GFF)]'\n")

    pause()
```

```
    print(
    "-"*40,
    "How It Works".center(10),
    "-"*40,"\n",
    "Initialization:\n\
        Parses and validates the command-line arguments.\n\
        Determines input and output file names and types.\n\
    File Reading:\n\
        Reads the input file content into memory.\n\
    Data Processing:\n\
        Extracts metadata and gene annotations using functions from
'search_data.py' and 'search_motif.py'.\n\
        Processes each input format separately to handle its specific
structure.\n\
    Output Writing:\n\
        Converts processed data into GFF format and writes it to the
specified output file.\n")

    pause()

    print(
    "-"*40,
    "Additional Modules".center(10),
    "-"*40,"\n",
    "'search_data.py':\n\
        Provides utilities for parsing GeneMark and ScanForMatches data.\n\
        Key functions: 'Seq_GMH', 'Seq_GM', 'Create_Tab_GMH',
'Create_Tab_GM', 'Create_Tab_SFM'.\n\
    'search_motif.py':\n\
        Extracts metadata like version, date, and model information.\n\
        Key functions: 'SourceVersion_GMH', 'Date_GMH', 'Model_GMH',
'RBS_GMH', 'ModelInformation_GMH'.\n")

    pause()

    print(
    "-"*40,
    "Contact".center(10),
    "-"*40,"\n",
    "For further inquiries or support, please contact the developer :
RODRIGUES Camille-Astrid (camille-astrid.rodrigues@univ-tlse3.fr).\n")

    quit()
```

Document 5 : résultats obtenus après conversion des fichiers de sortie à l’aide du code fourni (document complémentaire 4)

- output_GENMH.GFF

fichier de sortie obtenu après conversion du fichier input_GENMH.LSF (résultats de l'étude menée avec GeneMarkHMM) avec le programme convert_to_GFF.py

- output_GENM.GFF

fichier de sortie obtenu après conversion du fichier input_GENM.txt (résultats de l'étude menée avec GeneMark) avec le programme convert_to_GFF.py

- output_SFM.GFF

fichier de sortie obtenu après conversion du fichier input_SFM.txt (résultats de l'étude menée avec ScanForMatches) avec le programme convert_to_GFF.py

filename : output_GENMH.GFF

```
##gff-version 2
##source-version GeneMark.hmm PROKARYOTIC 3.42
##date: Sun Dec  1 13:49:12 2024
# Sequence file name: seq.fna
# Model file name:
/home/genemark/parameters/prokaryotic/Lactococcus_lactis_Il1403/GeneMark_hm
m_combined.mod
# RBS: true
# Model information: Lactococcus_lactis_Il1403

seq_L_lactis_lactis_1   GeneMark.hmm    CDS     174     347     ?       +
.       gene GMH_CDS_1
seq_L_lactis_lactis_2   GeneMark.hmm    CDS     455     3436    ?       +
.       gene GMH_CDS_2
seq_L_lactis_lactis_3   GeneMark.hmm    CDS     3447    5249    ?       +
.       gene GMH_CDS_3
seq_L_lactis_lactis_4   GeneMark.hmm    CDS     5242    6486    ?       +
.       gene GMH_CDS_4
seq_L_lactis_lactis_5   GeneMark.hmm    CDS     6483    7220    ?       +
.       gene GMH_CDS_5
seq_L_lactis_lactis_6   GeneMark.hmm    CDS     7222    9270    ?       +
.       gene GMH_CDS_6
seq_L_lactis_lactis_7   GeneMark.hmm    CDS     9475    >9555   ?       +
.       gene GMH_CDS_7
```

filename : output_GENM.GFF

```
##gff-version 2
##source-version GeneMark 4.27
##date: Wed Jan 01 13:44:31 2025
# Sequence file: seq.fna
# Model file name: -
# RBS: -
# Model information: Lactococcus_lactis_Il1403

seq_L_lactis_lactis     GeneMark        CDS     455     3436    ?       +
.       gene GM_CDS_1.1
seq_L_lactis_lactis     GeneMark        CDS     623     3436    ?       +
.       gene GM_CDS_1.2
seq_L_lactis_lactis     GeneMark        CDS     785     3436    ?       +
.       gene GM_CDS_1.3
seq_L_lactis_lactis     GeneMark        ATG     848     850     ?       +
.       .
seq_L_lactis_lactis     GeneMark        CDS     848     3436    ?       +
.       gene GM_CDS_1.4
seq_L_lactis_lactis     GeneMark        CDS     1310    3436    ?       +
.       gene GM_CDS_1.5
seq_L_lactis_lactis     GeneMark        ATG     3447    3449    ?       +
.       .
seq_L_lactis_lactis     GeneMark        CDS     3447    5249    ?       +
.       gene GM_CDS_2.1
seq_L_lactis_lactis     GeneMark        CDS     3456    5249    ?       +
.       gene GM_CDS_2.2
seq_L_lactis_lactis     GeneMark        CDS     3561    5249    ?       +
.       gene GM_CDS_2.3
seq_L_lactis_lactis     GeneMark        CDS     3630    5249    ?       +
.       gene GM_CDS_2.4
seq_L_lactis_lactis     GeneMark        CDS     3687    5249    ?       +
.       gene GM_CDS_2.5
seq_L_lactis_lactis     GeneMark        CDS     5230    6486    ?       +
.       gene GM_CDS_3.1
seq_L_lactis_lactis     GeneMark        CDS     5239    6486    ?       +
.       gene GM_CDS_3.2
seq_L_lactis_lactis     GeneMark        CDS     5242    6486    ?       +
.       gene GM_CDS_3.3
seq_L_lactis_lactis     GeneMark        CDS     5374    6486    ?       +
.       gene GM_CDS_3.4
seq_L_lactis_lactis     GeneMark        CDS     5653    6486    ?       +
.       gene GM_CDS_3.5
seq_L_lactis_lactis     GeneMark        ATG     5806    5808    ?       +
.       .
seq_L_lactis_lactis     GeneMark        CDS     5806    6486    ?       +
.       gene GM_CDS_3.6
seq_L_lactis_lactis     GeneMark        CDS     5827    6486    ?       +
.       gene GM_CDS_3.7
```

```
seq_L_lactis_lactis     GeneMark        CDS     6483    7220    ?       +
.       gene GM_CDS_4.1
seq_L_lactis_lactis     GeneMark        ATG     6507    6509    ?       +
.       .
seq_L_lactis_lactis     GeneMark        CDS     6507    7220    ?       +
.       gene GM_CDS_4.2
seq_L_lactis_lactis     GeneMark        CDS     6564    7220    ?       +
.       gene GM_CDS_4.3
seq_L_lactis_lactis     GeneMark        CDS     6696    7220    ?       +
.       gene GM_CDS_4.4
seq_L_lactis_lactis     GeneMark        CDS     6738    7220    ?       +
.       gene GM_CDS_4.5
seq_L_lactis_lactis     GeneMark        CDS     6789    7220    ?       +
.       gene GM_CDS_4.6
seq_L_lactis_lactis     GeneMark        CDS     6873    7220    ?       +
.       gene GM_CDS_4.7
seq_L_lactis_lactis     GeneMark        CDS     6939    7220    ?       +
.       gene GM_CDS_4.8
seq_L_lactis_lactis     GeneMark        CDS     7222    9270    ?       +
.       gene GM_CDS_5.1
seq_L_lactis_lactis     GeneMark        ATG     7279    7281    ?       +
.       .
seq_L_lactis_lactis     GeneMark        CDS     7279    9270    ?       +
.       gene GM_CDS_5.2
seq_L_lactis_lactis     GeneMark        CDS     7504    9270    ?       +
.       gene GM_CDS_5.3
seq_L_lactis_lactis     GeneMark        CDS     7792    9270    ?       +
.       gene GM_CDS_5.4
seq_L_lactis_lactis     GeneMark        CDS     9406    9558    ?       +
.       gene GM_CDS_6.1
seq_L_lactis_lactis     GeneMark        ATG     9409    9411    ?       +
.       .
seq_L_lactis_lactis     GeneMark        CDS     9409    9558    ?       +
.       gene GM_CDS_6.2
```

filename : output_SFM.GFF

```
##gff-version 2
##source-version ScanForMatches
##date: Tue Dec 10 22:30:59 2024
# Sequence file name: -
# Model file name: -
# RBS: True
# Model information: -

seq_L_lactis_lactis     ScanForMatches  RBS     161     176     .       +
.       note " GGAGG CACTCAAA ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     282     298     .       +
.       note " GGAGC TCTGATGGG TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     1802    1814    .       +
.       note " GGGGA TATCG TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     2317    2335    .       +
.       note " GGATG TAAATAGTAAG ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     2531    2549    .       +
.       note " GGAGA AAAGGGGAGAG TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     4581    4599    .       +
.       note " GGAGA ATTAACTGCTA TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     5497    5512    .       +
.       note " GGAGC AGCTGGCA TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     5611    5625    .       +
.       note " GAAGG ATTTAAT TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     5796    5808    .       +
.       note " GGAGA ATCAG ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     6340    6352    .       +
.       note " GGAGA TGAAA GTG "
seq_L_lactis_lactis     ScanForMatches  RBS     6367    6379    .       +
.       note " GAAGG AATAA GTG "
seq_L_lactis_lactis     ScanForMatches  RBS     6469    6485    .       +
.       note " GGAGG GAAGAGGAA ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     6836    6854    .       +
.       note " GGGGG CGGTCAAAGCT TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     6990    7005    .       +
.       note " GGAGA ATATCAGG ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     7067    7086    .       +
.       note " GGGGA GAATTGATAAGG ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     7205    7224    .       +
.       note " GAAGG TAGGAAACTAGA GTG "
seq_L_lactis_lactis     ScanForMatches  RBS     7449    7462    .       +
.       note " GGATG TAACTG GTG "
seq_L_lactis_lactis     ScanForMatches  RBS     7593    7606    .       +
.       note " GGATG AGCTAC TTG "
seq_L_lactis_lactis     ScanForMatches  RBS     7942    7954    .       +
.       note " GGAGA AAGCT ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     8071    8086    .       +
.       note " GGAGG GTTTGATA ATG "
```

```
seq_L_lactis_lactis     ScanForMatches  RBS     8547    8566    .       +
.       note " GGATG TAATAGCCGTAG GTG "
seq_L_lactis_lactis     ScanForMatches  RBS     8608    8626    .       +
.       note " GGAGC GGATGCAATTT ATG "
seq_L_lactis_lactis     ScanForMatches  RBS     9397    9411    .       +
.       note " GGAGG TAAAGTG GTG "
```