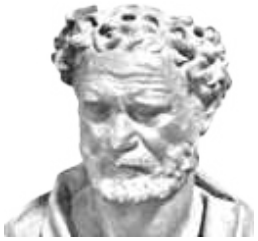


ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΜΟΡΙΑΚΗΣ ΒΙΟΛΟΓΙΑΣ ΚΑΙ ΓΕΝΕΤΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Pinda: Ένα πρόγραμμα εντοπισμού γονιδιακών διπλασιασμών

Δημήτριος - Γεώργιος Κοντόπουλος

Επιβλέπων:
Δρ. Νικόλαος Μ. Γλυκός
Επίκουρος Καθηγητής Υπολογιστικής και Δομικής Βιολογίας
Τμήμα Μοριακής Βιολογίας και Γενετικής
Δημοκρίτειο Πανεπιστήμιο Θράκης

Αλεξανδρούπολη

Ιούλιος 2012

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Pinda:
Ένα πρόγραμμα εντοπισμού γονιδιακών διπλασιασμών

Δημήτριος - Γεώργιος Κοντόπουλος

ΑΕΜ: 833

Επιβλέπων:

Δρ. Νικόλαος Μ. Γλυκός, Επίκουρος Καθηγητής Υπολογιστικής και Δομικής Βιολογίας, Τμήμα Μοριακής Βιολογίας και Γενετικής, Δημοκρίτειο Πανεπιστήμιο Θράκης

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Νικόλαο Γλυκό για την καθοδήγηση και την υποστήριξη που μου παρείχε αλλά και τις γνώσεις που μου μετέδωσε καθ' όλη τη διάρκεια εκπόνησης αυτής της διπλωματικής εργασίας. Ακόμη περισσότερο γιατί με εμπιστεύτηκε και μου έδωσε την ευκαιρία να σχεδιάσω το δικό μου project, παρά το γεγονός ότι διέφερε αρκετά από την κύρια κατεύθυνση της συγκεκριμένης ερευνητικής ομάδας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου για την πολύπλευρη υποστήριξή τους σε όλη τη διάρκεια των σπουδών μου αλλά και σε κάθε επιλογή μου.

Τέλος, δε θα μπορούσα να ξεχάσω τους φίλους μου, που στάθηκαν στο πλευρό μου σε καλές αλλά και δύσκολες στιγμές, αποτελώντας τη δεύτερη οικογένειά μου.

Περιεχόμενα

0.1 Περίληψη 10
0.2 Abstract 11

1 Εισαγωγή **13**
1.1 Στόχος εργασίας 13
1.2 Γονιδιακός Διπλασιασμός και μοντέλα 14
1.3 Στοίχιση αλληλουχιών 16
1.4 Προδιορισμός εξελικτικής απόστασης 17
1.5 Φυλογενετικά δένδρα 18
1.6 Προγράμματα χαρτογράφησης και αλληλούχησης γονιδιωμάτων 21
1.7 Gene Ontology Project 21

2 Μέθοδος επίλυσης **23**
2.1 Γλώσσες Προγραμματισμού 23
2.1.1 Perl 23
2.1.2 JavaScript 24
2.1.3 R 24
2.2 Common Gateway Interface 25
2.3 Διάγραμμα Ροής 26
2.4 Περιγραφή του αλγορίθμου Pinda 26
2.4.1 Καταχώρηση αλληλουχίας 26
2.4.2 Εύρεση συγγενικών αλληλουχιών 27
2.4.3 Στοίχιση και σχεδιασμός δενδρογράμματος 27
2.4.4 Υπολογισμός Επιπέδου Βεβαιότητας 28
2.4.5 Σύγκριση Γονιδιακών Οντολογιών 29
2.4.6 Αποστολή αποτελεσμάτων 29
2.4.7 Queue management 29
2.5 Υλοποίηση 29
2.6 Άδεια Χρήσης 30

3 Παραδείγματα χρήσης **33**
3.1 LSIII, NLNTP και Etxs στο θαλάσσιο φίδι Laticauda semifasciata 33
3.1.1 Βιβλιογραφικά δεδομένα 33
3.1.2 Στιγμιότυπα της εφαρμογής 36
3.2 LWS-1 και LWS-2 στο Zebrafish 41
3.2.1 Βιβλιογραφικά δεδομένα 41
3.2.2 Στιγμιότυπα της εφαρμογής 42
3.3 Απαιτήσεις φυσικής μνήμης και χρονισμός 46

4 Πηγαίος Κώδικας **49**
4.1 Προαπαιτούμενα 49
4.2 Pinda.cgi 51
4.3 Pinda_exec.pl 71
4.4 Pinda.R 124

5 Επίλογος **127**

Βιβλιογραφία **129**

6 Παράρτημα - Βοηθητικά Scripts **133**
6.1 db_temp_check.sh 133
6.2 db_update.sh 134
6.3 remove_temp.sh 136

0.1 Περίληψη

Η επανάσταση της Γονιδιωματικής και των σύγχρονων τεχνικών αλληλούχησης οδήγησε στην εμφάνιση ενός μεγάλου αριθμού γενετικών τόπων και υποτιθέμενων γονιδιακών προϊόντων, των οποίων οι λειτουργίες δεν έχουν προσδιοριστεί. Μια μέθοδος για τον καθορισμό πιθανών λειτουργιών για κάποιο μη χαρακτηρισμένο γονίδιο είναι η διαλεύκανση του εξελικτικού του παρελθόντος, συμπεριλαμβανομένων και των γονιδιακών διπλασιασμών.

Οι γονιδιακοί διπλασιασμοί διαδραματίζουν έναν ιδιαίτερα σημαντικό ρόλο στην εξέλιξη των γονιδιωμάτων και στην προσαρμογή των ειδών στις μεταβαλλόμενες περιβαλλοντικές συνθήκες. Μέσω αυτών συντελείται η γέννηση πρωτεϊνικών οικογενειών, δηλαδή μορίων που φέρουν εξελικτική συγγένεια και επομένως κάποιο βαθμό αλληλουχικής -και πιθανόν δομικής- ομοιότητας.

Στα πλαίσια λοιπόν αυτής της Διπλωματικής Εργασίας ασχοληθήκαμε με την ανάπτυξη ενός υπολογιστικού προγράμματος που στόχο έχει τη διευκόλυνση του εντοπισμού πιθανών ενδοειδικών γονιδιακών διπλασιασμών για ένα μόριο και της περαιτέρω ανάλυσης αυτών. Το πρόγραμμα Pinda εφαρμόζει ένα πρωτόκολλο στο οποίο συμμετέχουν λογισμικά εντοπισμού συγγενικών μορίων, στοίχισης και σχηματισμού δενδρογράμματος. Επιπλέον, τα αποτελέσματα του προγράμματος ιεραρχούνται με βάση τα δεδομένα αποστάσεων που προκύπτουν από τα φυλογενετικά δένδρα, σε συνδυασμό με τη σύγκριση γνωστών οντολογιών, όπου αυτό είναι δυνατό.

Τέλος, η αξιοπιστία του προγράμματος επιβεβαιώθηκε ύστερα από έλεγχο των αποτελεσμάτων του για πρωτεΐνες και γενετικούς τόπους, των οποίων οι διπλασιασμοί ήταν ήδη γνωστοί. Σε όλες τις περιπτώσεις, τα δεδομένα που παράχθηκαν ήταν σε πλήρη ταύτιση με τη σχετική βιβλιογραφία.

0.2 Abstract

The revolution of Genomics and modern sequencing methods led to the appearance of a great number of genomic loci and putative gene products, whose functions have not been determined. A method for inferring the likely functions of an uncharacterized gene is the elucidation of its evolutionary past, including any gene duplications.

Gene duplications play a notably important role in the evolution of genomes and the adaptation of species towards ever changing environmental conditions. They are responsible for the birth of protein families, which comprise molecules bearing evolutionary relationship and therefore a degree of sequence -and possibly structural- similarity.

So, in the terms of this final year thesis we developed a computer program, aiming to facilitate the discovery of possible intraspecies gene duplications of a molecule and their further analysis. The Pinda service applies a protocol, joining software for molecule similarity search, alignment and dendrogram creation. Furthermore, its results are being ranked, according to the distance data which are being derived from the phylogenetic trees, combined with comparison of known ontologies, wherever that is possible.

Finally, the program's reliability has been confirmed, after validating its results against proteins and genetic loci, whose duplications were already known. In every case, generated data were in full accordance with relevant literature.

Κεφάλαιο 1

Εισαγωγή

1.1 Στόχος εργασίας

Ως σήμερα, σύμφωνα με όσα γνωρίζουμε, δεν υπήρχε οποιοδήποτε υπολογιστικό εργαλείο που θα διευκόλυνε τη σύγκριση μιας αλληλουχίας με άλλες στο ίδιο είδος για τον πιθανό εντοπισμό γεγονότων διπλασιασμού. Ο ερευνητής θα έπρεπε να εντοπίσει παρόμοιες αλληλουχίες (μέσω κάποιου εργαλείου BLAST*), να τις στοιχίσει με κάποιο κατάλληλο πρόγραμμα και να χρησιμοποιήσει τη στοίχιση για το σχεδιασμό δενδρογράμματος. Εκτός αυτών, θα έπρεπε να καταβάλει προσπάθεια για την ανάλυση της τοπολογίας του δένδρου και τον υπολογισμό κάποιου επιπέδου βεβαιότητας για τα υποψήφια γεγονότα διπλασιασμού. Εάν μάλιστα τα τελευταία βήματα δεν είναι αυτοματοποιημένα, είναι πιθανό η ανάλυση να εμπεριέχει κάποιο βαθμό υποκειμενισμού. Στόχος λοιπόν της Διπλωματικής Εργασίας αποτελεί η ανάπτυξη ενός φιλικού προς το χρήστη αλλά και αξιόπιστου υπολογιστικού προγράμματος για την αυτοματοποίηση όσο το δυνατόν περισσότερο της χρονοβόρας αυτής διαδικασίας.

Για την επίτευξη του παραπάνω στόχου δημιουργήθηκε το διαδικτυακό υπολογιστικό πρόγραμμα Pinda (**P**ipeline for **In**traspecies **D**uplication **A**nalysis). Διευκολύνει την ανάλυση με την αυτοματοποίηση των απαραίτητων διαδικασιών για την ανάλυση συγγενικών αλληλουχιών ενός είδους και φιλοδοξεί να αποτελέσει μια δημοφιλή πλατφόρμα για τη διαλεύκανση γεγονότων διπλασιασμού.

1.2 Γονιδιακός Διπλασιασμός και μοντέλα

> *Seen in the light of evolution, biology is, perhaps, intellectually the most satisfying and inspiring science. Without that light it becomes a pile of sundry facts -- some of them interesting or curious but making no meaningful picture as a whole.*
>
> **Theodosius Dobzhansky**

Μια από τις χρωμοσωμικές μεταλλαγές είναι ο Διπλασιασμός (*duplication*), κατά τον οποίο διπλασιάζεται ένα τμήμα του χρωμοσώματος. Διπλασιασμοί προκύπτουν ως λάθη κατά τον ομόλογο ανασυνδυασμό, ύστερα από συμβάν ρετρομετάθεσης ή ως αποτέλεσμα διπλασιασμού ολόκληρου του χρωμοσώματος. Τα διπλασιασμένα τμήματα ποικίλλουν ως προς το μέγεθος, ενώ μπορεί να βρίσκονται σε διαδοχική διάταξη ή σε τελείως απομακρυσμένες περιοχές του χρωμοσώματος. Όταν η περιοχή που διπλασιάζεται, περιέχει τουλάχιστον ένα γονίδιο, στο οποίο και επικεντρώνεται η μελέτη, τότε χρησιμοποιείται συνηθέστερα ο όρος "**γονιδιακός διπλασιασμός**". Γονίδια που προέρχονται από διπλασιασμούς συχνά εμφανίζουν παρόμοια αλληλουχία και λειτουργία, μπορούν όμως πιθανώς να εκφράζονται σε διαφορετικούς κυτταρικούς τύπους ή σε διαφορετικές χρονικές περιόδους. Τέτοια γονίδια ονομάζονται παράλογα λόγω της παράλληλης εξέλιξής τους εντός του ίδιου γονιδιώματος και συγκροτούν πολυγονιδιακές οικογένειες. Αντίστοιχα, ομόλογα γονίδια που αποκλίνουν λόγω ειδογένεσης ονομάζονται ορθόλογα και κατανέμονται σε διαφορετικά είδη. Μετά από ένα συμβάν γονιδιακού διπλασιασμού, το αντίγραφο θα ακολουθήσει μια από τις παρακάτω εξελικτικές πορείες. [1],[2],[3],[4],[5],[6],[7]

I. Δημιουργία ψευδογονιδίου

Στο δεύτερο αντίγραφο του γονιδίου κατά κανόνα δεν υφίσταται εξελικτική πίεση, εφόσον το πρώτο παραμένει φυσιολογικό. Συντελείται λοιπόν ατέρμονη συσσώρευση μεταλλαγών χωρίς την ύπαρξη πίεσης επιλογής, φαινόμενο που ονομάζεται γενετική απόκλιση. Οι μεταλλάξεις πολύ πιθανόν να μετατρέψουν το ένα αντίγραφο σε ψευδογονίδιο, που είτε δεν εκφράζεται, είτε δεν επιτελεί κάποια λειτουργία. Ενδεικτικά, ο χρόνος μετατροπής ενός γονιδίου σε ψευδογονίδιο ύστερα από ένα γονιδιακό διπλασιασμό εκτιμάται σε περίπου 4 εκατομμύρια χρόνια. [1],[2],[7],[8]

II. Συντήρηση της λειτουργίας

Σε κάποιες περιπτώσεις, η ύπαρξη ενός επιπλέον γονιδίου με την ίδια λειτουρ-

γία αποτελεί σημαντικό εξελικτικό πλεονέκτημα, καθώς αυξάνονται τα επίπεδα της αντίστοιχης πρωτεΐνης ή RNA. Κάτι τέτοιο παρατηρείται κυρίως σε γονίδια που εκφράζονται σε υψηλό βαθμό και των οποίων τα προϊόντα απαιτούνται συνεχώς και σε μεγάλες ποσότητες, όπως τα rRNAs και οι ιστόνες. [2],[7]

*III. Κατανομή λειτουργιών μεταξύ των αντιγράφων (**Subfunctionalization**)*

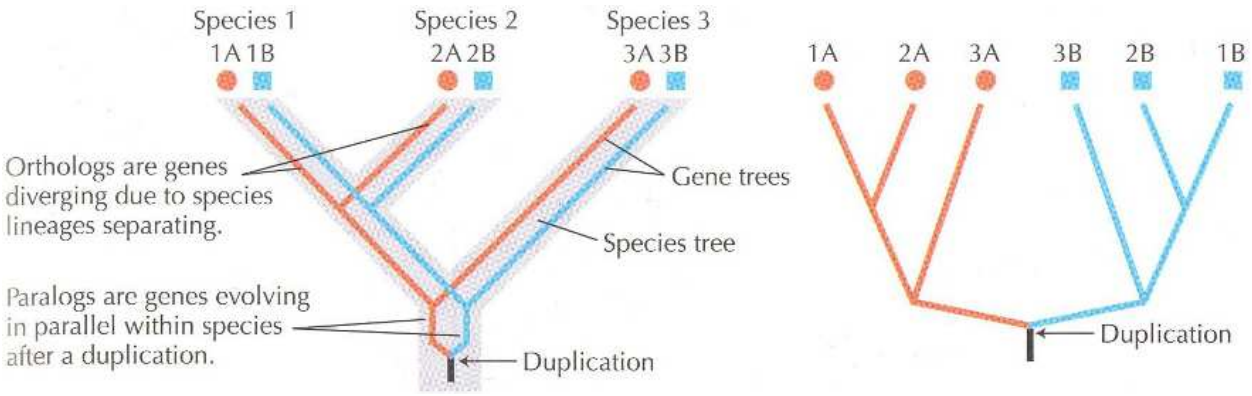
Η ύπαρξη δύο γονιδίων με την ίδια λειτουργία στο γονιδίωμα σπανίως μπορεί να είναι σταθερή, εκτός από τις περιπτώσεις που προαναφέρθηκαν. Αντίθετα, γονίδια με μικρές διαφορές μεταξύ τους υποστηρίζονται περισσότερο από το γονιδίωμα, μέσω του φαινομένου του Subfunctionalization. Σε αυτή την περίπτωση, εφόσον το αρχικό γονίδιο είχε παραπάνω από μία λειτουργίες, τότε η επιλογή οδηγεί σε καταμερισμό των λειτουργιών του προγονικού στα δύο αντίγραφα. Παράλληλα, σχεδόν αμέσως μετά το γεγονός διπλασιασμού, κατά κανόνα παρατηρούνται μεταβολές στα επίπεδα έκφρασης των δύο γονιδίων. [2],[7],[9]

Ένα χαρακτηριστικό παράδειγμα αποτελεί η οικογένεια των α-κρυσταλλινών του φακού. Σε αυτήν ανήκουν πολλές, συγγενικές μεταξύ τους πρωτεΐνες, οι οποίες φέρουν διάφορους μεταβολικούς και ρυθμιστικούς ρόλους, ακόμα και σε περιοχές του σώματος εκτός του φακού. Θεωρείται ότι τα μέλη της οικογένειας προέκυψαν από ένα προγονικό γονίδιο, το οποίο είχε αποκτήσει διακριτές λειτουργίες που μοιράστηκαν ύστερα από ένα γεγονός διπλασιασμού. [7]

*IV. Εμφάνιση νέων λειτουργιών (**Neofunctionalization**)*

Ένα από τα σημαντικότερα αποτελέσματα ενός γονιδιακού διπλασιασμού είναι η εμφάνιση νέων λειτουργιών από το ένα αντίγραφο. Σπάνια, κάποιες μεταλλαγές μπορεί να τροποποιήσουν τη λειτουργία του γονιδιακού προϊόντος, δίνοντας στον οργανισμό ένα εξελικτικό πλεονέκτημα, οπότε το γονίδιο προοδευτικά γίνεται απαραίτητο για τον οργανισμό. Για να πραγματοποιηθεί αυτό, προϋποτίθεται η ύπαρξη μιας περιόδου ουδέτερης εξέλιξης. [2],[7],[9],[10],[11]

Οι γονιδιακοί διπλασιασμοί διαδραματίζουν σημαίνοντα ρόλο στην εξέλιξη των γονιδιωμάτων, καθώς αυξάνουν την πλαστικότητά τους, παρέχοντας επιπλέον ευκαιρίες στους οργανισμούς για προσαρμογή σε μεταβαλλόμενες περιβαλλοντικές συνθήκες. Επίσης, συμμετέχουν στη γένεση πολύπλοκων μονοπατιών ρύθμισης, ενώ ακόμα ευθύνονται για την εμφάνιση αποκλειστικών χαρακτηριστικών στα διάφορα είδη οργανισμών. [2],[7]



Γονιδιακοί διπλασιασμοί και ειδογένεση. Απεικόνιση μέσω εξελικτικών δένδρων ειδών (αριστερά) και γονιδίων (δεξιά).

Αναπαράγεται άνευ αδείας από:
Barton,Nicholas H., Briggs,Derek E. G., Eisen,Jonathan A., Goldstein,David B., Patel,Nipam H. **Evolution**, Cold Spring Harbor Laboratory Press, New York, United States of America, 2007, σελ. 127

1.3 Στοίχιση αλληλουχιών

Το πρώτο βήμα για κάθε ανάλυση συγκριτικής γονιδιωματικής είναι η στοίχιση των αλληλουχιών, είτε πρωτεϊνικών, είτε νουκλεοτιδικών. Αυτές παρατίθενται η μία κάτω από την άλλη, ενώ οι επιμέρους χαρακτήρες αντιστοιχίζονται μεταξύ τους. Όμοιοι και -αν πρόκειται για πρωτεΐνες- παραπλήσιοι χαρακτήρες τοποθετούνται ο ένας κάτω από τον άλλο, ενώ ανόμοιοι είτε τοποθετούνται στην ίδια στήλη είτε στοιχίζονται με κενά. Το αποτέλεσμα της στοίχισης καθορίζεται από το σύστημα βαθμονόμησης (scoring system), καθώς και από τον τύπο της στοίχισης που επιχειρεί ο αλγόριθμος. Οι δύο τύποι στοίχισης είναι η ολική (global) και η τοπική (local). Στην πρώτη περίπτωση ο αλγόριθμος προσπαθεί να στοιχίσει όσο το δυνατόν περισσότερους χαρακτήρες μεταξύ των αλληλουχιών, ενώ στη δεύτερη δημιουργούνται νησίδες με παρόμοιους χαρακτήρες, οι οποίες περιβάλλονται από κενά. [9],[12]

Η ομοιότητα σε επίπεδο αλληλουχίας θεωρείται ως το αποτέλεσμα λειτουργικής, δομικής ή εξελικτικής σχέσης μεταξύ των συγκρινόμενων αλληλουχιών. Ανόμοιοι χαρακτήρες σε μια στήλη αναπαριστούν φαινόμενα σημειακών μεταλλάξεων, ενώ τα κενά αντικατοπτρίζουν γεγονότα εισαγωγής ή έλλειψης καταλοίπων σε τουλάχιστον μία από τις αλληλουχίες. Τα τελευταία ονομάζονται γι' αυτό το λόγο και indels. Αντίθετα, παρόμοιοι χαρακτήρες που τοποθετούνται σε μια στήλη φέρουν ομολογία θέσης (positional homology) σε επίπεδο καταλοίπων. Περιοχές της στοίχισης με μεγάλη ομοιότητα αναδεικνύουν υψηλό βαθμό συντήρησης της αλληλουχίας, γεγονός που μπορεί να υποδεικνύει δομική ή λειτουργική σημαντικότητα για τη συγκεκριμένη περιοχή. [9],[12],[13]

Καθώς η ποιότητα της στοίχισης επηρεάζει άμεσα τα φυλογενετικά δένδρα που θα προκύψουν, είναι πολύ σημαντική η αξιολόγηση των στηλών της. "Κακοστοιχισμένες" περιοχές συνηθώς δημιουργούν "θόρυβο" στην ανάλυση, γι' αυτό και συχνά αφαιρούνται χειροκίνητα (**masking**). Αυτή η διαδικασία όμως εμπεριέχει υποκειμενικότητα, ενώ δεν είναι καθόλου πρακτική σε στοιχίσεις μεγάλης κλίμακας. Αν από την άλλη απορριφθούν όλες οι στήλες που περιέχουν κενά, χάνεται σημαντικό κομμάτι της πληροφορίας, χωρίς να βελτιώνονται τα δένδρα που προκύπτουν. Μια σύγχρονη αντιμετώπιση ενέχει την εφαρμογή προγραμμάτων που αξιολογούν τη στοίχιση σε επίπεδο στήλης, μέσω διαφόρων αλγορίθμων. [13]

A

```
Protein  1   DEMGLGKK----R--VESTR
Protein  2   DEMGVGKRGH-----LESRK
Protein  3   DDMGLGRWK---R--VESTE
Protein  4   DDMGLGHKKK-----VDSTK
Mask         11111111000000011111
```

B

```
Protein  1   DEMGLGKKVESTR
Protein  2   DEMGVGKRLESRK
Protein  3   DDMGLGRKVESTE
Protein  4   DDMGLGHKVDSTK
```

A: Πριν εφαρμοστεί masking. B: Μετά την εφαρμογή masking.

Αναπαράγεται άνευ αδείας από:
http://evolution-textbook.org/content/free/figures/ch27.html

1.4 Προδιορισμός εξελικτικής απόστασης

Η εξελικτική απόσταση (d) υπολογίζεται ως ο αριθμός νουκλεοτιδικών ή αμινοξικών αντικαταστάσεων ανά γενετική θέση μεταξύ δυο ομόλογων αλληλουχιών. Ως μέγεθος, είναι ιδιαίτερα σημαντική για τη μελέτη της μοριακής εξέλιξης και χρησιμοποιείται στο σχεδιασμό φυλογενετικών δέντρων και γονιδιακών οικογενειών. Επιπλέον, μέσω της εξελικτικής απόστασης μπορεί να διερευνηθεί το μοτίβο και ο μηχανισμός της μοριακής εξέλιξης και να προσδιοριστούν προσεγγιστικά γεγονότα διπλασιασμού γονιδίων/γονιδιωμάτων και γεγονότα ειδογένεσης. Για την εκτίμηση της εξελικτικής απόστασης d μεταξύ αλληλουχιών DNA, χρησιμοποιούνται κυρίως στοχαστικά Μαρκοβιανά μοντέλα. [12]

Μια κλασική μέθοδος για τον προσδιορισμό της εξελικτικής απόκλισης μεταξύ δύο ομόλογων πρωτεϊνών είναι ο υπολογισμός του λόγου διαφορετικών αμινοξέων (p) ανά-

μεσα στις δύο αλληλουχίες, γνωστή και ως **p-distance**. Ωστόσο ο p δεν είναι αυστηρά ανάλογος με το χρόνο απόκλισης (t), καθώς πολλαπλές αμινοξικές αντικαταστάσεις μπορούν να λάβουν χώρα στην ίδια θέση. Αντίθετα, μια πιο ακριβής μέθοδος βασίζεται στη διαδικασία Poisson, σύμφωνα με την οποία η πιθανότητα να μη γίνει καμία αμινοξική αντικατάσταση σε διάστημα t χρόνων σε μία θέση της αλληλουχίας ισούται με e^{-ut}, όπου u ο εξελικτικός ρυθμός. Επομένως, η πιθανότητα καμίας αμινοξικής αντικατάστασης σε κανέναν από τους δύο ομόλογους γενετικούς τόπους (q) ισούται με e^{-2ut} και μπορεί να εκτιμηθεί από την εξίσωση $q = 1 - p$. Ως λογική συνέχεια των προηγουμένων, ο αναμενόμενος αριθμός αμινοξικών αντικαταστάσεων ανά θέση των δύο αλληλουχιών, δηλαδή η εξελικτική απόσταση $d = 2ut$ δίνεται από την εξίσωση

$$\boldsymbol{d = -ln(1 - p)}$$

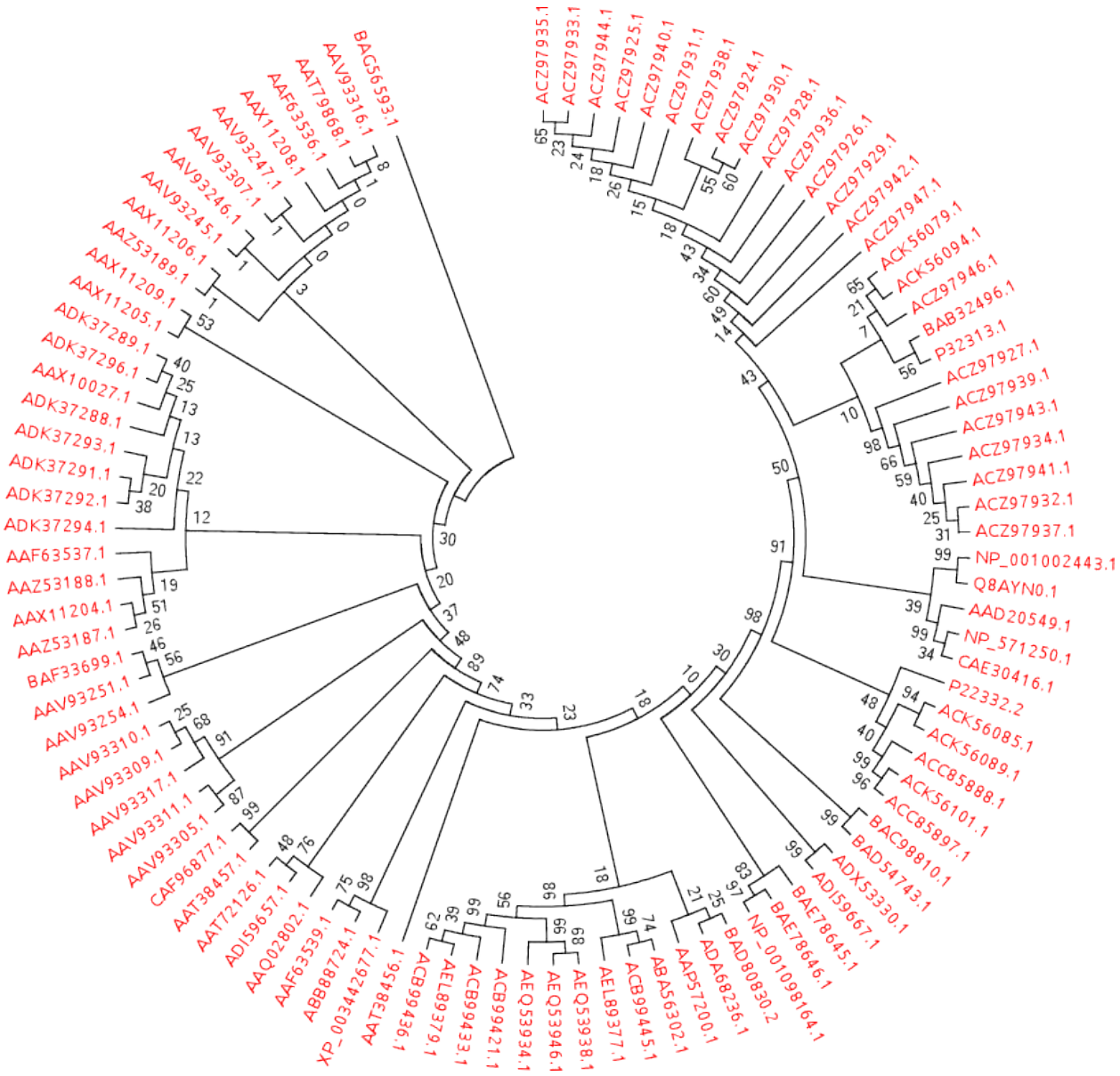
Ας σημειωθεί εδώ ότι η απόσταση Poisson είναι προσεγγιστική, καθώς αντίστροφες και παράλληλες μεταλλάξεις δε λαμβάνονται υπ' όψιν. Παρ' όλα αυτά, τα αποτελέσματα αυτών των μεταλλάξεων είναι γενικά αμελητέα, εκτός και αν ο p είναι μεγάλος. [12]

1.5 Φυλογενετικά δένδρα

Οι εξελικτικές σχέσεις μεταξύ γονιδίων συνήθως παρουσιάζονται με μια δενδροειδή μορφή, είτε με ρίζα, είτε χωρίς. Ως τοπολογία του δένδρου ορίζεται το μοτίβο διακλαδώσεων που εμφανίζεται. Η χρήση μοριακών δεδομένων για το σχεδιασμό φυλογενετικών δένδρων ξεκινάει από τους Cavalli-Sforza και Edwars (1967) και τους Fitch και Margoliash (1967). Για την επιλογή της πιο πιθανής τοπολογίας, έχουν αναπτυχθεί διάφορες μέθοδοι που χωρίζονται σε 4 κατηγορίες: (1) μέθοδοι απόστασης, (2) μέθοδοι φειδωλότητας, (3) μέθοδοι μέγιστης πιθανότητας και (4) Bayesian μέθοδοι. Οι πολύπλοκοι μαθηματικοί υπολογισμοί που απαιτούνται για τις δύο τελευταίες κατηγορίες είναι ιδιαίτερα απαιτητικοί σε υπολογιστικούς πόρους, όμως τα αποτελέσματα που παράγουν θεωρούνται πιο αξιόπιστα. Ωστόσο, στα πλαίσια μιας γενικής εκτίμησης των εξελικτικών σχέσεων μεταξύ συγκεκριμένων γενετικών τόπων, ακόμα και οι μέθοδοι απόστασης κρίνονται ικανοποιητικοί, ειδικά λόγω των χαμηλών απαιτήσεων σε υπολογιστικούς πόρους. [12]

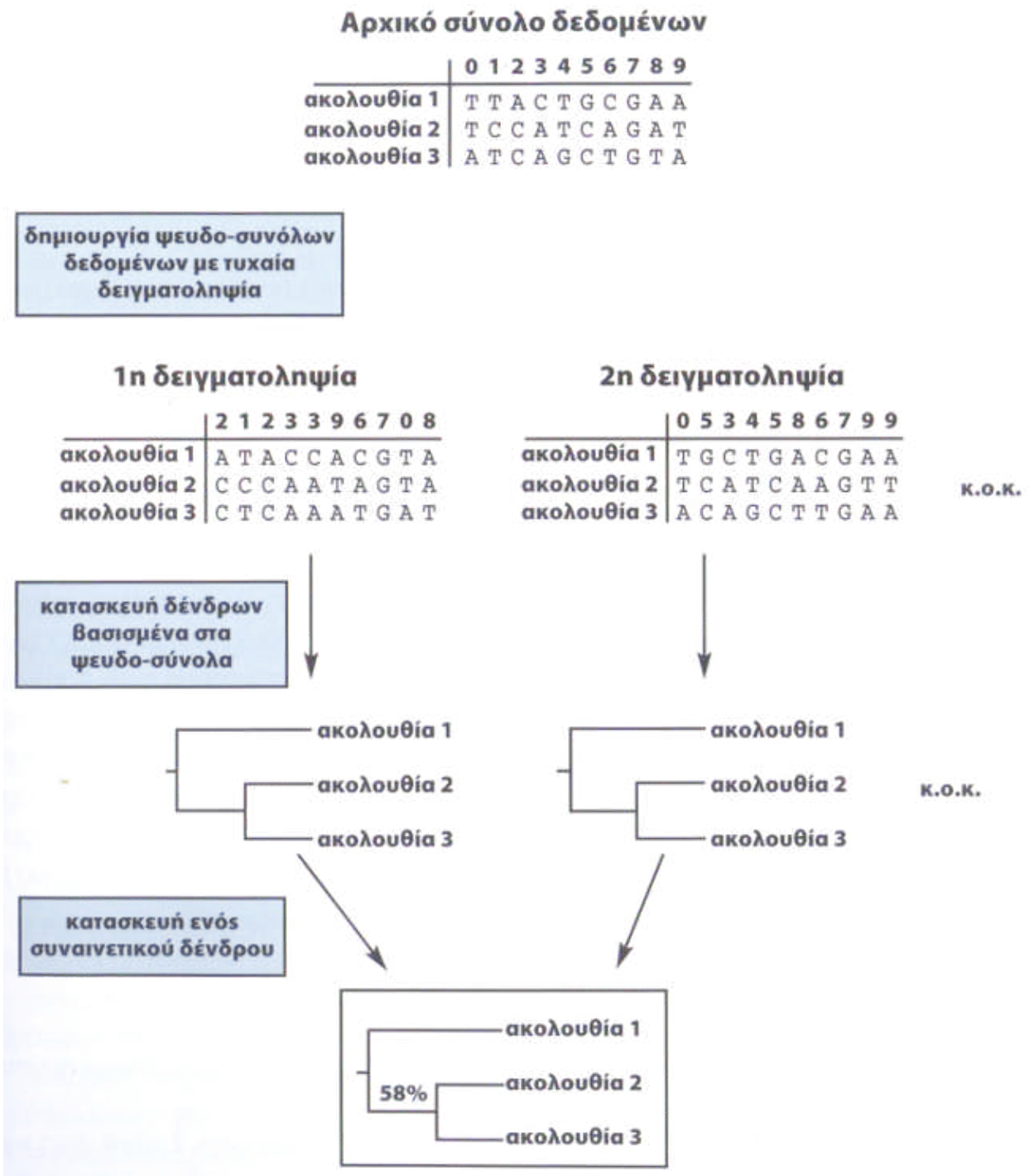
Ένας αλγόριθμος που ανήκει στις μεθόδους απόστασης και χρησιμοποιείται ευρέως είναι ο αλγόριθμος Neighbor - Joining. Δημοσιεύθηκε το 1987 από τους Saitou και Nei. Η μέθοδος αυτή δεν εξετάζει όλες τις πιθανές τοπολογίες αλλά σε κάθε στάδιο εφαρμόζεται η αρχή της ελάχιστης εξέλιξης. Δεν προϋποθέτει την ύπαρξη ενός μοριακού ρολογιού, αλλά απαιτεί τα δεδομένα να φέρουν την προσθετική ιδιότητα για τα μήκη

των κλάδων. Το αποτέλεσμα είναι ένα φυλογενετικό δένδρο χωρίς ρίζα. [12],[14]



Φυλογενετικό δένδρο χωρίς ρίζα για 100 πρωτεϊνικές αλληλουχίες γονιδίων οψίνης από διάφορα είδη. Ο αλγόριθμος που χρησιμοποιήθηκε είναι ο Neighbor - Joining.

Για να αξιολογηθεί η στατιστική σημαντικότητα του φυλογενετικού δένδρου εφαρμόζεται η διαδικασία bootstrapping.[15],[16] Δημιουργούνται σύνολα με τυχαία επιλογή στηλών από το αρχικό σύνολο και για κάθε σύνολο κατασκευάζεται ένα νέο δένδρο με τις ίδιες ακριβώς παραμέτρους με το αρχικό. Το ποσοστό επανεμφάνισης του κάθε κόμβου του αρχικού δένδρου στα νέα δένδρα είναι η τιμή bootstrap για τον κόμβο. Εάν η τιμή bootstrap είναι μεγαλύτερη του 70%, τότε ο κόμβος είναι σωστός, με πιθανότητα 95%. [9]



Η διαδικασία bootstrapping.

Αναπαράγεται άνευ αδείας από:

*Κοσσίδα,Σοφία, **Βιοπληροφορική: Δυνατότητες και προοπτικές**, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, Ελλάδα, 2008, σελ. 115*

1.6 Προγράμματα χαρτογράφησης και αλληλούχησης γονιδιωμάτων

Η αλληλούχηση του ανθρώπινου γονιδιωμάτος (Human Genome Poject) προτάθηκε το 1986, ξεκίνησε τον Οκτώβριο του 1990 και ολοκληρώθηκε πλήρως το 2003. Στόχοι της ήταν η ταυτοποίηση όλων των ανθρώπινων γονιδίων, ο προσδιορισμός της πλήρους αλληλουχίας DNA, η αποθήκευση και ο χειρισμός των δεδομένων και τέλος η διαχείριση των ηθικών και κοινωνικών ζητημάτων που πιθανόν να ανέκυπταν. [3]

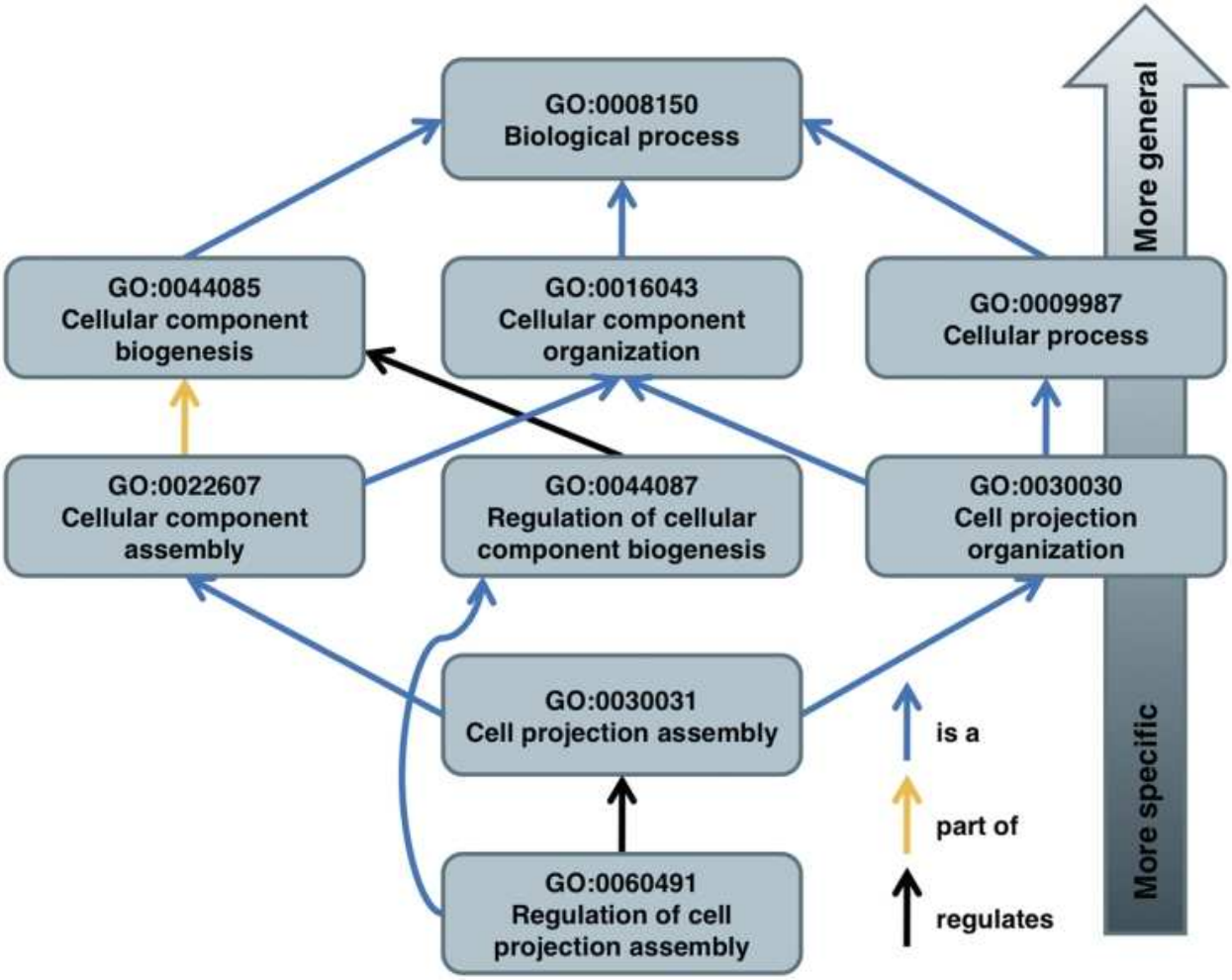
Σήμερα, έχουν αλληλουχηθεί πλήρως τα γονιδιώματα πολλών οργανισμών, ενώ ο ρυθμός κατάθεσης νέων συνεχώς αυξάνεται. Κάποια παραδείγματα τέτοιων οργανισμών είναι η *Escherichia coli*, ο *Saccharomyces cerevisiae*, η *Drosophila melanogaster* και η *Arabidopsis thaliana*. Στα πλαίσια της μελέτης της δομής και της λειτουργίας των νέων γονιδιωμάτων, είναι χρήσιμη η συγκριτική ανάλυση γονιδίων εντός του ίδιου είδους για τον εντοπισμό διπλασιασμένων γονιδίων, τα οποία μπορεί να έχουν παρόμοια λειτουργία. Γονίδια που προκύπτουν από γονιδιακό διπλασιασμό συχνά ρυθμίζουν τη λειτουργία μονοπατιών με διαφορετικό τρόπο ή ενεργοποιούνται σε διαφορετικές χρονικές στιγμές υπό την εμφάνιση διαφορετικών ερεθισμάτων. Επομένως, μετά την -έστω και μερική- αλληλούχηση νέων γονιδιωμάτων, μπορεί να διενεργηθεί έλεγχος για γεγονότα διπλασιασμού, ώστε να κατανοηθούν σταδιακά οι πολύπλοκες λειτουργίες των αντίστοιχων οργανισμών. [3]

1.7 Gene Ontology Project

Το Gene Ontology Project ξεκίνησε το 1998 από μια ομάδα ερευνητών που μελετούσε το γονιδίωμα της *Drosophila melanogaster*, του *Mus musculus* και του *Saccharomyces cerevisiae*. Στην πορεία προστέθηκαν αρκετές βάσεις δεδομένων για κάθε οργανισμό - μοντέλο, συνεισφέροντας στην ανάπτυξη του Προγράμματος. [17]

Αποτελεί ένα λεξικό όρων, οι οποίοι κατηγοριοποιούνται σε τρεις μη επικαλυπτόμενες ομάδες: **(1)** Μοριακή Λειτουργία (MF), **(2)** Βιολογική Διαδικασία (BP) και **(3)** Κυτταρικό Διαμέρισμα (CC). Κάθε οντολογία διαθέτει ένα αλφαριθμητικό αναγνωριστικό και περιγράφει ένα συγκεκριμένο χαρακτηριστικό ενός γονιδίου ή ενός γονιδιακού προϊόντος. Τα δεδομένα των οντολογιών προέρχονται -κυρίως αυτόματα και χωρίς επιμέλεια- είτε από την επιστημονική βιβλιογραφία, είτε από κάποια βάση δεδομένων, είτε από υπολογιστικές ενδείξεις. Ο αριθμός των οντολογιών το 2000 έφτανε μόλις τις 5.000, ενώ το 2010 ξεπέρασε τις 20.000. Το πρόγραμμα GO μπορεί να χρησιμοποιηθεί για τον εντοπισμό γονιδίων με παρόμοιες λειτουργίες μέσα στον ίδιο ή

και μεταξύ διαφορετικών οργανισμών, παρέχοντας έτσι τις απαραίτητες πληροφορίες για την έναρξη μιας φυλογενετικής ανάλυσης. [17],[18]



Ένα παράδειγμα οργάνωσης των οντολογιών μεταξύ τους.

Αναπαράγεται άνευ αδείας από:
du Plessis,Luis, Škunca,Nives, Dessimoz,Christophe,
The what, where, how and why of gene ontology—a primer for bioinformaticians,
Briefings in Bioinformatics, 2011; 12(6), σελ. 723-735

Κεφάλαιο 2

Μέθοδος επίλυσης

Για τον επιτυχή και γρήγορο προσδιορισμό γονιδίων που προκύπτουν από γονιδιακό διπλασιασμό μέσα σε ένα είδος, αποφασίστηκε η δημιουργία του Pinda. Αποτελείται κυρίως από τρία υποπρογράμματα, ο πηγαίος κώδικας των οποίων είναι γραμμένος στις γλώσσες προγραμματισμού Perl, JavaScript και R.

2.1 Γλώσσες Προγραμματισμού

> *I simultaneously believe that languages are wonderful and awful. You have to hold both of those. Ugly things can be beautiful. And beautiful can get ugly very fast. You know, take Lisp. You know, it's the most beautiful language in the world. At least up until Haskell came along. But, you know, every program in Lisp is just ugly. I don't figure how that works.*
>
> **Larry Wall**

2.1.1 Perl

Η γλώσσα προγραμματισμού **Perl** δημιουργήθηκε το 1987 από το γλωσσολόγο, συγγραφέα και προγραμματιστή, Larry Wall. Η Perl είναι μια ισχυρή, υψηλού επιπέδου γλώσσα προγραμματισμού που κατατάσσεται στις διερμηνεύσιμες (***interpreted***). Ανήκει στο Ελεύθερο Λογισμικό υπό την άδεια GPL, υλοποιήθηκε με τη γλώσσα προγραμματισμού C και μπορεί να χρησιμοποιηθεί σε διάφορες πλατφόρμες λογισμικού. Όπως σε κάθε διερμηνεύσιμη γλώσσα, τα προγράμματα που γράφονται στην Perl υστερούν σημαντικά σε ταχύτητα εκτέλεσης σε σχέση με αλγόριθμους που υλοποιούνται σε κάποια μεταγλωττίσιμη (***compiled***) γλώσσα, όπως η C. Ειδικά όταν πρόκειται για

μαθηματικούς υπολογισμούς, η υλοποίηση σε C μπορεί να είναι ακόμα και **65** φορές ταχύτερη, σε σχέση με την αντίστοιχη υλοποίηση σε Perl![19] Ωστόσο, η Perl χρησιμοποιείται κατά κόρον σε διάφορες εφαρμογές χειρισμού αλληλουχιών στη Βιοπληροφορική, λόγω των διευκολύνσεων που προσφέρει στο χειρισμό κειμένου και ειδικά των κανονικών εκφράσεων (***regular expressions***). Επιπλέον, ένα μεγάλο πλεονέκτημα που φέρει η Perl όσον αφορά τη Βιοπληροφορική είναι η BioPerl, μια συλλογή τμημάτων κώδικα (***modules***) σχετικών με Βιοπληροφορική, γεγονός που διευκολύνει τη συγγραφή αντίστοιχων προγραμμάτων. Η BioPerl είναι ένα ενεργό project, διατίθεται υπό την ελεύθερη άδεια GPL και διαδραμάτισε σημαίνοντα ρόλο στο Πρόγραμμα Χαρτογράφησης του Ανθρώπινου Γονιδιώματος. [20]

2.1.2 JavaScript

Η γλώσσα προγραμματισμού **JavaScript** δημιουργήθηκε από τον προγραμματιστή Brendan Eich το 1995. Προέκυψε από τη συνεργασία της Netscape Communications Corporation, στην οποία εργαζόταν ο Eich, και της Sun Microsystems, Inc. Η JavaScript είναι μια διερμηνεύσιμη γλώσσα προγραμματισμού και χρησιμοποιείται κυρίως για την παραγωγή δυναμικού περιεχομένου σε ιστοσελίδες, μέσω εκτέλεσης κώδικα στην πλευρά του πελάτη (***client-side scripting***). Επομένως, ένας κώδικας γραμμένος σε JavaScript μπορεί να εκτελεστεί ανεξαρτήτως πλατφόρμας λογισμικού, στη συντριπτική πλειοψηφία των φυλλομετρητών ιστού. [21]

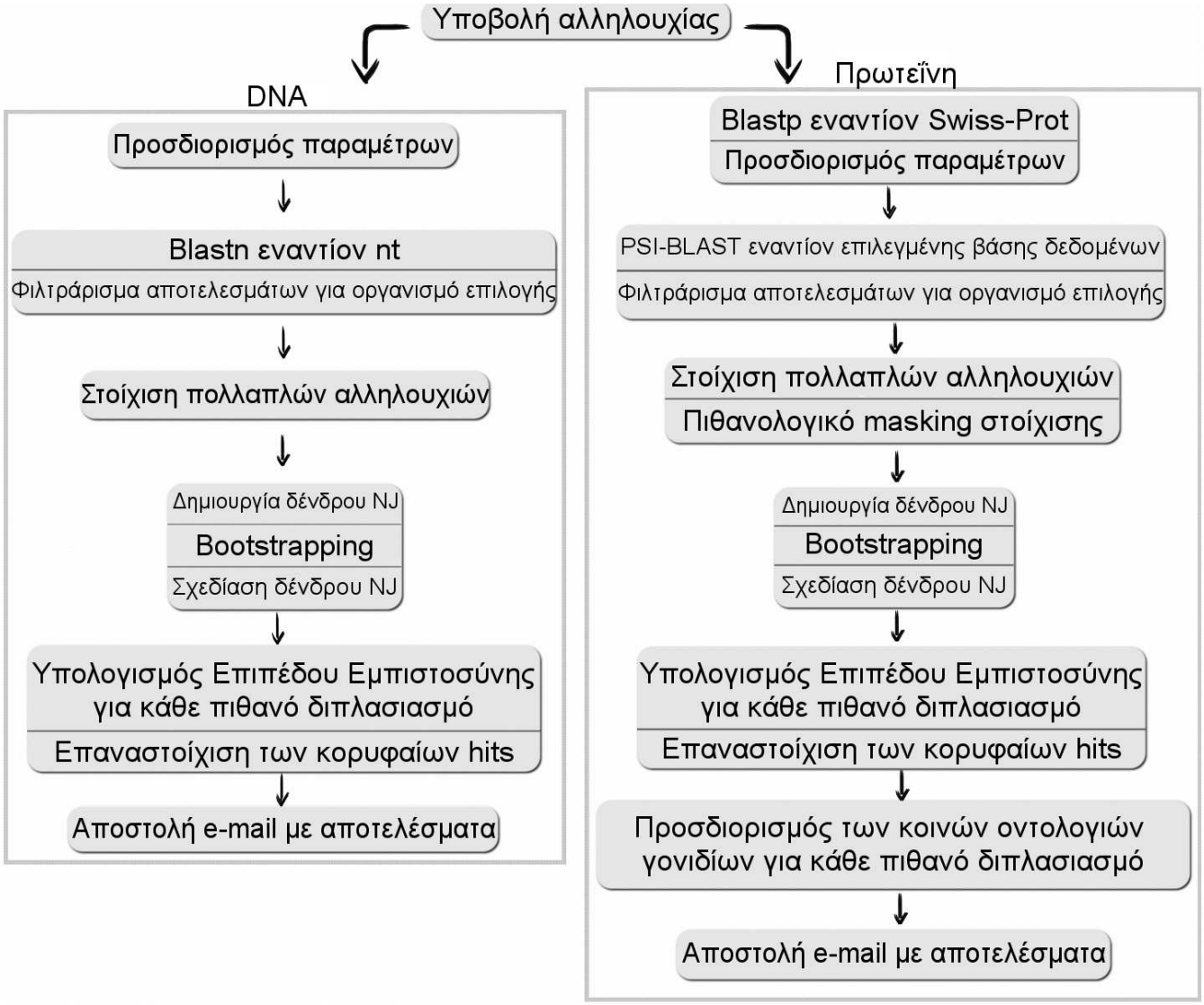
2.1.3 R

Η **R** δημιουργήθηκε το 1993 από τους Ross Ihaka και Robert Gentleman στο Πανεπιστήμιο Auckland της Νέας Ζηλανδίας, ενώ συνεχίζει να αναπτύσσεται ακόμα και σήμερα από την R Development Core Team. Είναι ταυτόχρονα μια ισχυρή και ευέλικτη γλώσσα προγραμματισμού, αλλά και ένα περιβάλλον για στατιστικές αναλύσεις και γραφικά. Ανήκει στο Ελεύθερο Λογισμικό υπό την άδεια GPL και ο πηγαίος κώδικας του περιβάλλοντός της είναι γραμμένος σε C, Fortran και R. Ένα πλεονέκτημα της γλώσσας είναι η πληθώρα έτοιμων πακέτων για διάφορα είδη αναλύσεων, συμπεριλαμβανομένων και αναλύσεων Εξέλιξης και Οικολογίας, οι οποίες παρέχονται από πακέτα όπως τα ape (***A**nalysis of **P**hylogenetics and **E**volution*) και ade4 (***A**nalyses des **D**onnées **E**cologiques: méthodes **E**xploratoires et **E**uclidiennes en sciences de l'**E**nvironnement*). Άλλα πλεονεκτήματά της αποτελούν η δυνατότητα αποθήκευσης και ταχύτατου χειρισμού μεγάλου όγκου δεδομένων, αλλά και ο εύκολος σχεδιασμός διαγραμμάτων υψηλής ποιότητας. [22],[23],[24]

2.2 Common Gateway Interface

Όσον αφορά τον τύπο του προγράμματος, επιλέχθηκε το μοντέλο πελάτη - εξυπηρετητή (***client-server***), οπότε θα εκτελείται χρησιμοποιώντας τους υπολογιστικούς πόρους του εξυπηρετητή, απαλλάσσοντας τον τελικό χρήστη από την ανάγκη για ισχυρό hardware ή τις δυσκολίες εγκατάστασης και συντήρησης. Για τη διαβίβαση πληροφοριών μεταξύ της εφαρμογής και του εξυπηρετητή, επιλέχθηκε το πρότυπο διεπαφής CGI, το οποίο υποστηρίζει δυναμικό περιεχόμενο και είναι ένας εύκολος και κοινός τρόπος για έναν εξυπηρετητή να περάσει την αίτηση ενός χρήστη σε κάποιο πρόγραμμα και να του επιστρέψει τα δεδομένα που προκύπτουν. Έτσι, ο χρήστης του προγράμματος θα δύναται μέσω ενός φυλλομετρητή ιστού (π.χ Mozilla Firefox, Google Chrome, Midori) να εισαγάγει τα δεδομένα του στο πρόγραμμα, από το οποίο θα λάβει και τα αποτελέσματα. Το γεγονός αυτό διασφαλίζει και τη φορητότητα του προγράμματος, αφού μπορεί να χρησιμοποιηθεί από ποικίλα λειτουργικά συστήματα μέσω σχεδόν οποιουδήποτε φυλλομετρητή, παρακάμπτοντας διάφορα προβλήματα συμβατότητας. [20]

2.3 Διάγραμμα Ροής



2.4 Περιγραφή του αλγορίθμου Pinda

2.4.1 Καταχώρηση αλληλουχίας

Περνώντας στον τρόπο λειτουργίας του προγράμματος, το Pinda αρχικά δέχεται την πλήρη νουκλεοτιδική ή πρωτεϊνική αλληλουχία του γονιδίου ή του γενετικού τόπου γενικότερα, μέσω μιας φόρμας HTML.

Στη συνέχεια, για πρωτεϊνικές αλληλουχίες το Pinda εκτελεί BLASTP[25] εναντίον της SwissProt και από τα αποτελέσματά του δημιουργεί μια λίστα οργανισμών που περιέχουν την ίδια ή πολύ κοντινές αλληλουχίες. Αν υπάρχει δήλωση οργανισμού μέσω

μορφής FASTA, τότε το πρόγραμμα προτείνει τον αντίστοιχο οργανισμό στο χρήστη, αφήνοντάς του ωστόσο τη δυνατότητα να επιβεβαιώσει ή να κάνει κάποια άλλη επιλογή οργανισμού. Η παραπάνω διαδικασία αποσκοπεί στον περιορισμό σφαλμάτων που μπορεί να προκληθούν από αλληλουχίες που είναι όμοιες μεταξύ ειδών ή διαφέρουν πολύ λίγο. Ο χρήστης πραγματοποιεί την τελική επιλογή του είδους, εντός του οποίου θα αναζητηθούν γονιδιακοί διπλασιασμοί. Οι βάσεις δεδομένων που μπορούν να επιλεγούν για την ανάλυση είναι είτε η Swiss-Prot από μόνη της, είτε η Swiss-Prot και η TrEMBL, οι οποίες μαζί συνιστούν την UniProt.[26] Υπάρχουν επίσης οι επιλογές για απενεργοποίηση της διαδικασίας filtering περιοχών χαμηλής πολυπλοκότητας και απενεργοποίηση της διαδικασίας masking της στοίχισης.

Εναλλακτικά, αν η αλληλουχία είναι νουκλεοτιδική και λόγω έλλειψης υπολογιστικών πόρων, καλείται ο χρήστης να δηλώσει τον οργανισμό - πηγή, χωρίς να παρέχονται προτάσεις από το πρόγραμμα. Η βάση δεδομένων που θα χρησιμοποιηθεί στην ανάλυση είναι η nt (GENBANK/EMBL/DDBJ + RefSeq), ενώ δίνεται και η επιλογή απενεργοποίησης της διαδικασίας filtering περιοχών χαμηλής πολυπλοκότητας.

Η διαδικασία υποβολής της αλληλουχίας ολοκληρώνεται με το πρόγραμμα να ανακοινώνει στο χρήστη τη σειρά προτεραιότητας της εργασίας του και τον εκτιμώμενο χρόνο ολοκλήρωσής της.

2.4.2 Εύρεση συγγενικών αλληλουχιών

Ακολούθως, εφόσον η αρχική αλληλουχία είναι πρωτεϊνική, διενεργείται PSI-BLAST, τα αποτελέσματα του οποίου φιλτράρονται ως προς τον οργανισμό - είδος επιλογής. Ο λόγος που επιλέχθηκε το PSI-BLAST έναντι των άλλων μορφών του αλγορίθμου BLAST είναι το γεγονός ότι μπορεί να εντοπίσει απομακρυσμένους εξελικτικά συγγενείς για την αλληλουχία που δόθηκε. Μετά την πρώτη έρευνα στις βάσεις δεδομένων, σχηματίζεται ένα μοτίβο από τα αποτελέσματα που προκύπτουν, το οποίο χρησιμοποιείται για την προσθήκη περισσότερων αποτελεσμάτων, μέχρι τα δεδομένα να συγκλίνουν χωρίς περαιτέρω αλλαγές ή μέχρι ένα μέγιστο όριο 50 κύκλων. [25]

Στην περίπτωση νουκλεοτιδικών αλληλουχιών, εκτελείται BLASTN[28] και τα αποτελέσματα αυτού φιλτράρονται, όπως παραπάνω.

2.4.3 Στοίχιση και σχεδιασμός δενδρογράμματος

Ακολουθεί πολλαπλή στοίχιση των αλληλουχιών που προκύπτουν με τα προγράμματα Clustal Omega[29] ή Kalign2[30] για πρωτεΐνες ή DNA αντίστοιχα. Επιπλέον, στην περίπτωση των πρωτεϊνών και εφόσον τα αποτελέσματα είναι κάτω των 150, χρησι-

μοποιείται το πρόγραμμα ZORRO[31] για την αξιολόγηση των στηλών της στοίχισης. Σε κάθε στήλη της στοίχισης αποδίδεται ένα σκορ από 0 έως 10. Στήλες με σκορ 0.4 ή χαμηλότερο απορρίπτονται από τη στοίχιση, ώστε να βελτιωθεί η ακρίβεια του επικείμενου δενδρογράμματος, χωρίς την απώλεια σημαντικής πληροφορίας.

Έπεται ο σχεδιασμός Neighbor Joining[14] Δένδρου και bootstrapping[9],[15],[16] 1000 επαναλήψεων με το ClustalW[32]. Ο αριθμός των επαναλήψεων είναι σχετικά μεγάλος, ώστε η τοπολογία του δένδρου να φέρει αρκετά μεγάλη στατιστική βεβαιότητα. Το δέντρο απεικονίζεται από το **Pinda.R** μέσω του πακέτου ape[23].

2.4.4 Υπολογισμός Επιπέδου Βεβαιότητας

Για να αξιολογηθεί η πιθανότητα να υπάρχει όντως σχέση διπλασιασμού μεταξύ της αρχικής αλληλουχίας και της κάθε αλληλουχίας - αποτελέσματος, το Pinda επεξεργάζεται το δένδρο που παράγεται -με τη συνδρομή των πακέτων ape[23] και ade4[24]-, εξάγοντας μια τιμή **Confidence Value** για κάθε αποτέλεσμα. Η πρώτη σκέψη για την τιμή Confidence Value ήταν να υπολογιστεί ως το γινόμενο των τιμών bootstrap κατά μήκος μιας διαδρομής μεταξύ της αλληλουχίας εισόδου και κάθε άλλης αλληλουχίας του δένδρου:

$$Confidence = \prod bootstrap(seq_{input} \rightarrow \ seq_{resulting}) \tag{2.1}$$

Σύμφωνα με την παραπάνω εξίσωση, όσο μειώνεται το γινόμενο των τιμών bootstrap κατά μήκος της διαδρομής, τόσο μικρότερη είναι η πιθανότητα άμεσου διπλασιασμού και το αντίστροφο. Ωστόσο, η εξίσωση 2.1 φέρει το μειονέκτημα ότι αγνοεί τελείως την εξελικτική απόσταση μεταξύ των αλληλουχιών. Έτσι, αλληλουχίες με παρόμοιο γινόμενο τιμών bootstrap, αλλά με μεγάλη εξελικτική απόσταση, φέρουν περίπου ίδια τιμή Confidence Value, γεγονός μη βιολογικώς ορθό. Αναπροσαρμόζοντας την εξίσωση και με το σκεπτικό ότι όσο αυξάνεται η εξελικτική απόσταση, τόσο μικρότερη πρέπει να είναι η πιθανότητα άμεσου διπλασιασμού, προέκυψε η εξίσωση 2.2:

$$\boxed{Confidence = \frac{\prod bootstrap(seq_{input} \rightarrow \ seq_{resulting})}{\sum distance(seq_{input} \rightarrow \ seq_{resulting})}} \tag{2.2}$$

Η τιμή **Confidence** για μια αλληλουχία - αποτέλεσμα (resulting) ισούται με το πηλίκο του γινομένου των τιμών bootstrap, ξεκινώντας από τα input και resulting tips μέχρι τον πρώτο κοινό τους κόμβο ή τη ρίζα του δένδρου, προς το άθροισμα των αποστάσεων μεταξύ της input αλληλουχίας και της resulting αλληλουχίας. Με άλλα λόγια, ισούται με το **λόγο του γινομένου των τιμών bootstrap προς το άθροισμα των αποστάσεων, κατά μήκος της διαδρομής input sequence tip -> resulting sequence tip**. Η

φυσική σημασία της εξίσωσης δηλώνει πως αλληλουχίες με μεγάλη πιθανότητα άμεσου διπλασιασμού θα βρίσκονται σχετικά κοντά σε ένα δενδρόγραμμα, ενώ οι τιμές bootstrap της αντίστοιχης περιοχής του δενδρογράμματος θα είναι υψηλές. Αποτελέσματα με χαμηλό γινόμενο τιμών boostrap υποδεικνύουν χαμηλή πιθανότητα άμεσης εξελικτικής σχέσης μεταξύ των δύο αλληλουχιών. Αντίστοιχα, αλληλουχίες που χωρίζονται από μεγάλη εξελικτική απόσταση είναι λιγότερο πιθανό να είναι αποτέλεσμα άμεσου διπλασιασμού. Το γεγονός αυτό δεν αποκλείει η μία από τις δύο αλληλουχίες να είναι προϊόν διπλασιασμού ενός προγονικού γονιδίου που με περαιτέρω διπλασιασμούς οδήγησε στην άλλη αλληλουχία.

Από το σετ των τιμών Confidence διενεργείται ένα Z-τεστ, ενώ οι τιμές Z χρησιμοποιούνται για τον υπολογισμό του Επιπέδου Βεβαιότητας για κάθε αποτέλεσμα. Από τις τιμές του Επιπέδου Βεβαιότητας δημιουργείται ένας πίνακας των αποτελεσμάτων, ταξινομημένος κατά φθίνουσα σειρά. Όσα αποτελέσματα σημειώνουν **Επίπεδο Βεβαιότητας τουλάχιστον 50%**, επαναστοιχίζονται με την αρχική, δημιουργώντας έτσι μια νέα στοίχιση με τις πιο πιθανές υποψήφιες αλληλουχίες διπλασιασμού.

2.4.5 Σύγκριση Γονιδιακών Οντολογιών

Αν η εισαχθείσα αλληλουχία είναι πρωτεϊνική, τότε ακολουθεί ένα βήμα ακόμα στο οποίο γίνεται σύγκριση των Γονιδιακών Οντολογιών μεταξύ της αλληλουχίας - στόχου και κάθε άλλης αλληλουχίας. Σημειώνεται ο αριθμός και η περιγραφή των κοινών οντολογιών, καθώς και αυτών που δεν χαρακτηρίζουν την αλληλουχία - στόχο.

2.4.6 Αποστολή αποτελεσμάτων

Τέλος, το πρόγραμμα αποστέλλει τα αποτελέσματα της εργασίας, δηλαδή τον πίνακα, τις στοιχίσεις και τα δενδρογράμματα μέσω ηλεκτρονικού ταχυδρομείου. Καθώς οι εργασίες που λαμβάνει το Pinda μπορεί να διαρκέσουν κάποιες φορές έως και αρκετές ώρες, η αποστολή e-mail προφυλάσσει τόσο τον client όσο και τον server από άσκοπη παραμονή στη σελίδα του Pinda, μέχρι την ολοκλήρωση της εκάστοτε εργασίας.

2.4.7 Queue management

Μετά την καταχώρηση της αλληλουχίας στη φόρμα του Pinda, το πρόγραμμα αποστέλλει τις πληροφορίες της εργασίας στην εφαρμογή SLURM[27], η οποία ευθύνεται για το queue management των εργασιών. Με απλά λόγια, φροντίζει ώστε οι εργασίες

να εκτελούνται με σειρά προτεραιότητας, η μία μετά την άλλη, ενώ ακόμα καταγράφει τα σφάλματα που τυχόν προκύπτουν.

2.5 Υλοποίηση

Ο πηγαίος κώδικας του Pinda αριθμεί συνολικά 3.946 γραμμές κώδικα σε Perl και 51 γραμμές κώδικα R. Εντός του κώδικα της Perl περιλαμβάνονται επίσης 22 γραμμές κώδικα JavaScript. Επιπλέον, για την υποστήριξη του προγράμματος μέσω της αυτόματης διαγραφής προσωρινών αρχείων και της ανανέωσης των βάσεων δεδομένων, γράφτηκαν τρία Bash scripts, μεγέθους 213 γραμμών συνολικά. Ο κώδικας του Pinda παρατίθεται στο Κεφάλαιο 4, ενώ αυτός των βοηθητικών scripts στο Παράρτημα (Κεφάλαιο 6), στο τέλος αυτής της εργασίας.

Το Pinda είναι εγκατεστημένο στον server "Orion", ο οποίος ανήκει στην ομάδα Δομικής και Υπολογιστικής Βιολογίας του Επίκουρου Καθηγητή, Δρ. Νικολάου Γλυκού στο Τμήμα Μοριακής Βιολογίας και Γενετικής του Δημοκριτείου Πανεπιστημίου Θράκης. Ο διαδικτυακός τόπος του Pinda είναι ο `http://orion.mbg.duth.gr/Pinda`, ενώ οι οδηγίες χρήσης του προγράμματος (σε μορφή html) είναι διαθέσιμες μέσω του `http://orion.mbg.duth.gr/Pinda/documentation.html`.

2.6 Άδεια Χρήσης

Ολοκληρώνοντας τη σύντομη περιγραφή της υλοποίησης του αλγορίθμου, αξίζει να επισημανθεί πως ο πηγαίος κώδικας του προγράμματος είναι διαθέσιμος σε αποθετήριο GitHub στη διεύθυνση `https://github.com/dgkontopoulos/Pinda/`. Η άδεια

χρήσης του προγράμματος είναι η **GNU Affero General Public License**, μια ελεύθερη άδεια που επιτρέπει την εκτέλεση του προγράμματος για οποιοδήποτε λόγο, την τροποποίησή του και το διαμοιρασμό αντιγράφων της αρχικής του έκδοσης ή εκδόσεων με προσωπικές τροποποιήσεις. Επίσης, η AGPL υποχρεώνει την παροχή του πηγαίου κώδικα προγραμμάτων που τρέχουν μέσω εξυπηρετητή και παραγώγων τους, ώστε όποιες αλλαγές γίνουν από τρίτους χρήστες να μπορούν να ενσωματωθούν για τη βελτίωση του αρχικού προγράμματος. Οι λόγοι χρήσης μιας ελεύθερης άδειας, πέραν των ιδεολογικών, είναι καθαρά πρακτικοί και λειτουργικοί. Ο κώδικας του προγράμματος είναι στη διάθεση όποιου τον επιθυμεί και όχι μόνο του δημιουργού, γεγονός που διευκολύνει τον εντοπισμό και τη διόρθωση προβλημάτων, την επιτάχυνση της εκτέλεσής του και πιθανώς την εμφάνιση νέων λειτουργιών. [33]

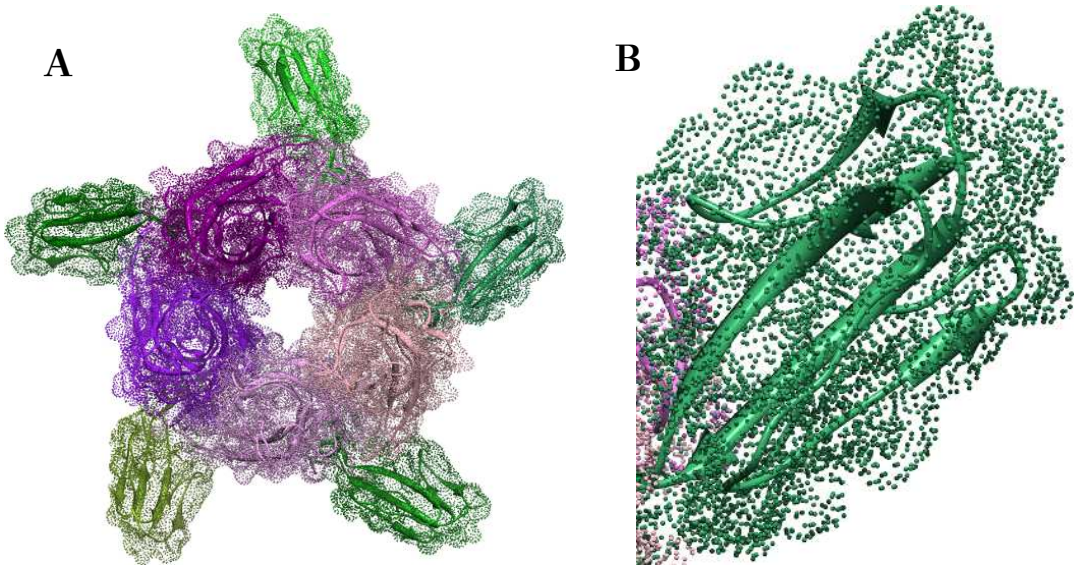
Κεφάλαιο 3

Παραδείγματα χρήσης

3.1 LSIII, NLNTP και Etxs στο θαλάσσιο φίδι Laticauda semifasciata

3.1.1 Βιβλιογραφικά δεδομένα

Το δηλητήριο των φιδιών εμπεριέχει διάφορες τοξικές πρωτεΐνες, εκ των οποίων οι περισσότερο μελετημένες είναι οι α-νευροτοξίνες. Διαθέτουν χαρακτηριστική τρισδιάστατη δομή, γνωστή και ως "τριών δακτύλων". Κατατάσσονται σε δύο ομάδες, τις νευροτοξίνες μακριάς και βραχείας αλυσίδας. Οι τοξίνες αυτές συγγενεύουν δομικά και θεωρούνται ότι έχουν προκύψει μέσω διπλασιασμών από έναν κοινό πρόγονο. [34]

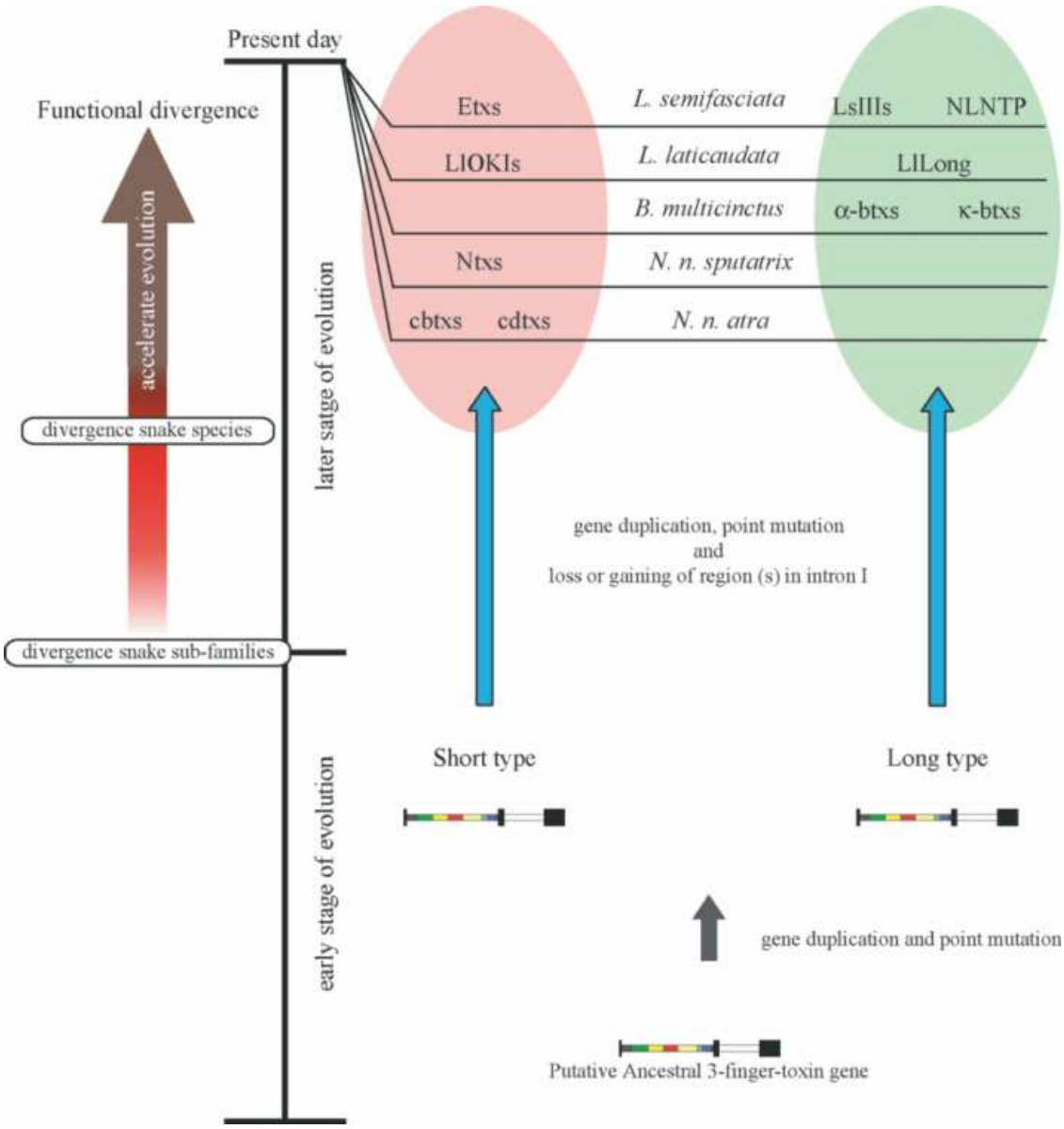


A: Κρυσταλλική δομή συμπλόκου 5 α-νευροτοξινών μακριάς αλυσίδας (με αποχρώσεις του πράσινου) με την πενταμερή πρωτεΐνη πρόσδεσης ακετυλοχολίνης (με αποχρώσεις του μωβ). **B**: Διακρίνεται η χαρακτηριστική δομή τριών δακτύλων των νευροτοξινών. (PDB ID: **1YI5**)

Στις νευροτοξίνες βραχείας αλυσίδας ανήκουν οι erabutoxins (Etxa, Etxb, Etxc), οι οποίες αποτελούν το κυριότερο συστατικό του δηλητηρίου του θαλάσσιου φιδιού *Laticauda semifasciata*. Αντίθετα, το γονίδιο LSIII ευθύνεται για τη σύνθεση μιας α-νευροτοξίνης μακριάς αλυσίδας. Στο γονιδίωμα της *Laticauda semifasciata* έχει επίσης εντοπιστεί και ένα ψευδογονίδιο, αντίγραφο του LSIII, το οποίο ονομάστηκε NLNTP (*Novel Long chain NeuroToxin Pseudogene*). [34]

Το θαλάσσιο φίδι Black-banded sea krait (*Laticauda semifasciata*).

Αναπαράγεται ελεύθερα από:
http://commons.wikimedia.org/wiki/File:Laticauda_semifasciata.jpg

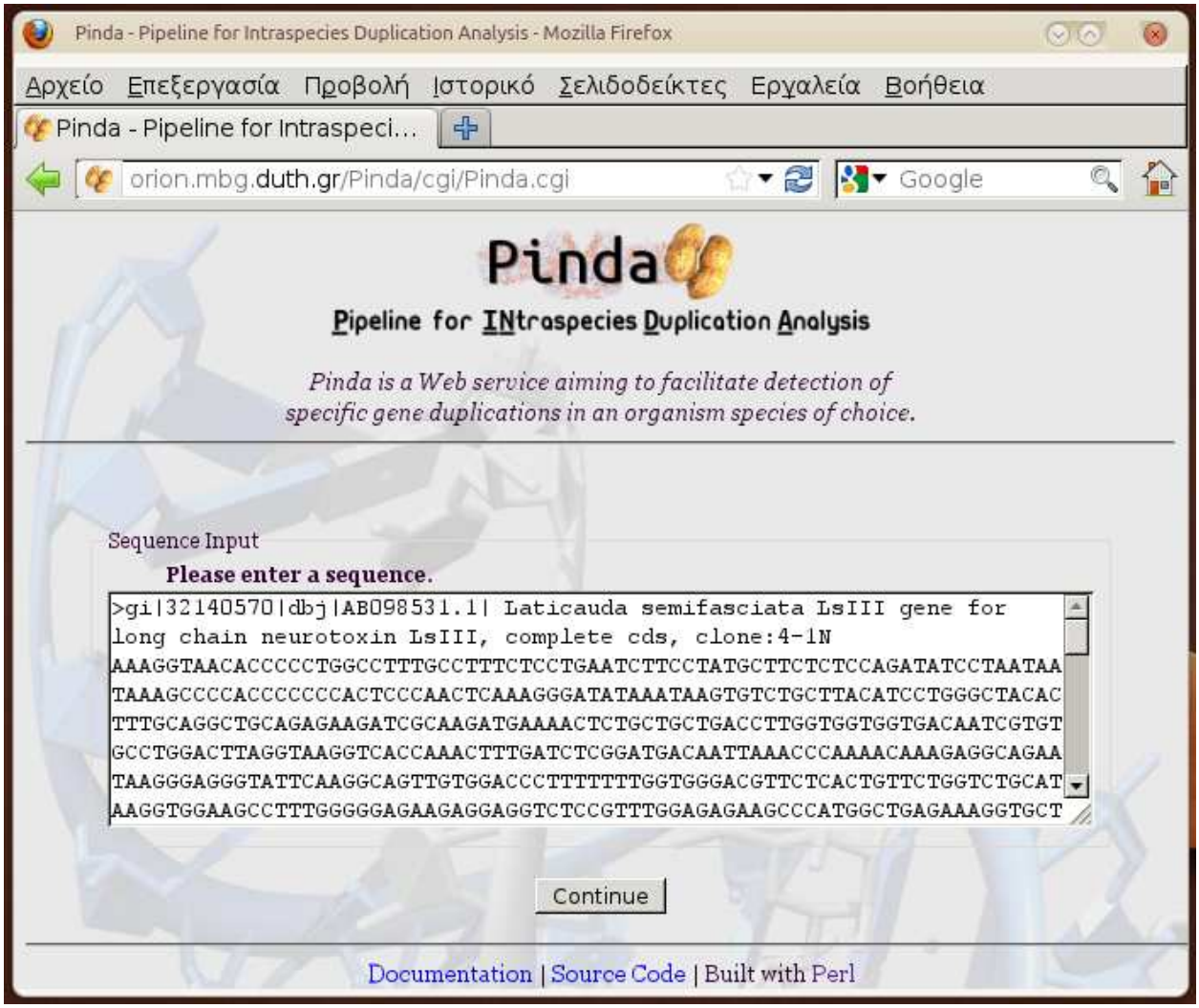


Υποθετικό μοντέλο εξέλιξης των γονιδίων τοξινών με δομή τριών δακτύλων στα φίδια.

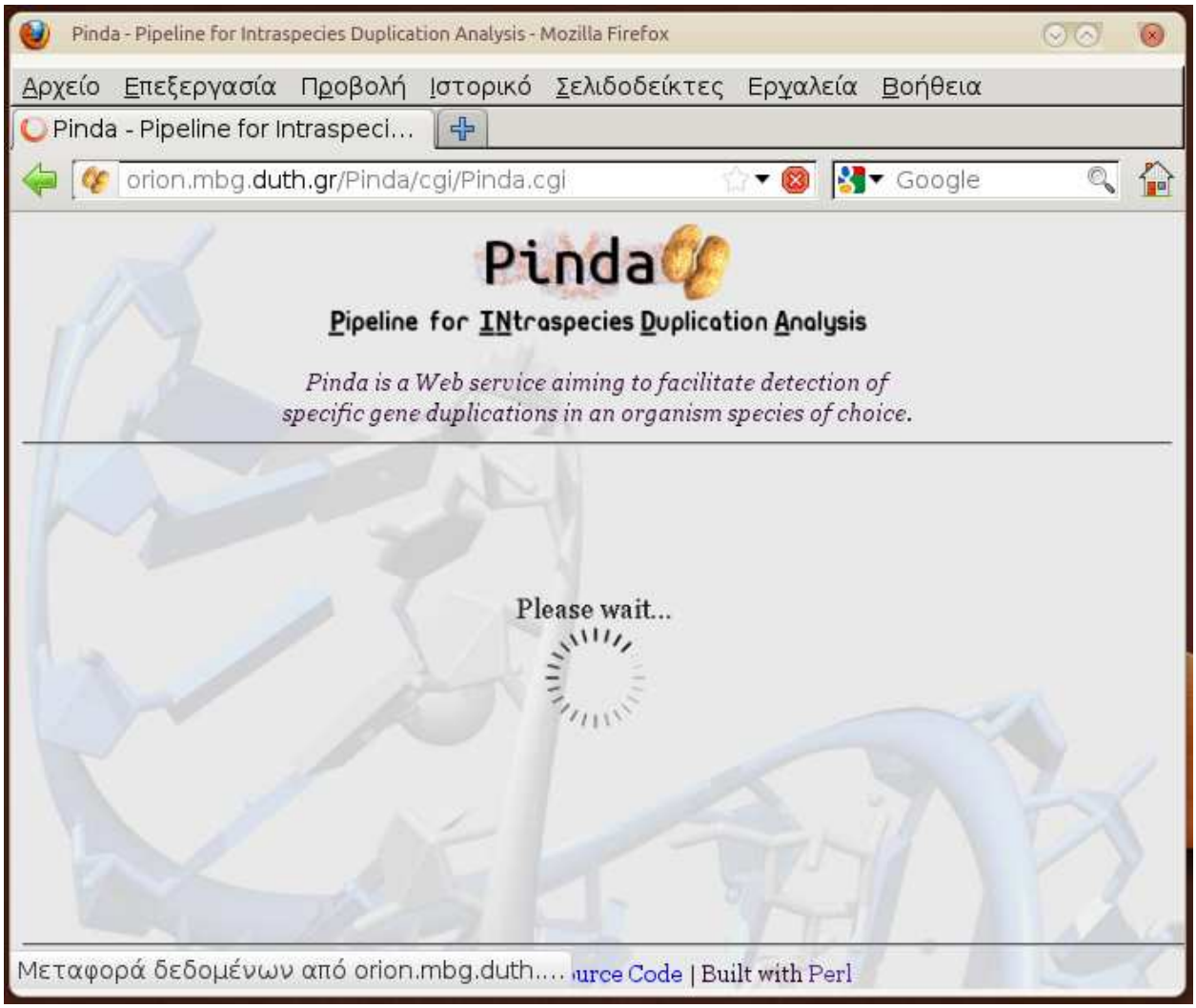
Αναπαράγεται άνευ αδείας από:

Fujimi,T. J., Nakajyo,T., Nishimura,E., Ogura,E., Tsuchiya,T., Tamiya,T.
Molecular evolution and diversification of snake toxin genes, revealed by analysis of intron sequences,
Gene, 2003; 313, σελ. 111-118

3.1.2 Στιγμιότυπα της εφαρμογής



Για το παράδειγμα αυτό ελήφθη η νουκλεοτιδική αλληλουχία του γονιδίου LSIII του *Laticauda semifasciata* από τη βάση δεδομένων Nucleotide του NCBI, η οποία και εισήχθη στην αρχική σελίδα του Pinda.



Αφού πατήθηκε το κουμπί "Submit", το πρόγραμμα άρχισε να φορτώνει την επόμενη σελίδα, ενώ παράλληλα εμφανιζόταν σχετική εικόνα με τις λέξεις "*Please wait*".

Στη συνέχεια εμφανίστηκαν οι επιλογές για την ανάλυση των διπλασιασμών. Αφού ορίστηκε ως οργανισμός η "Laticauda semifasciata", πατήθηκε το κουμπί "Submit".

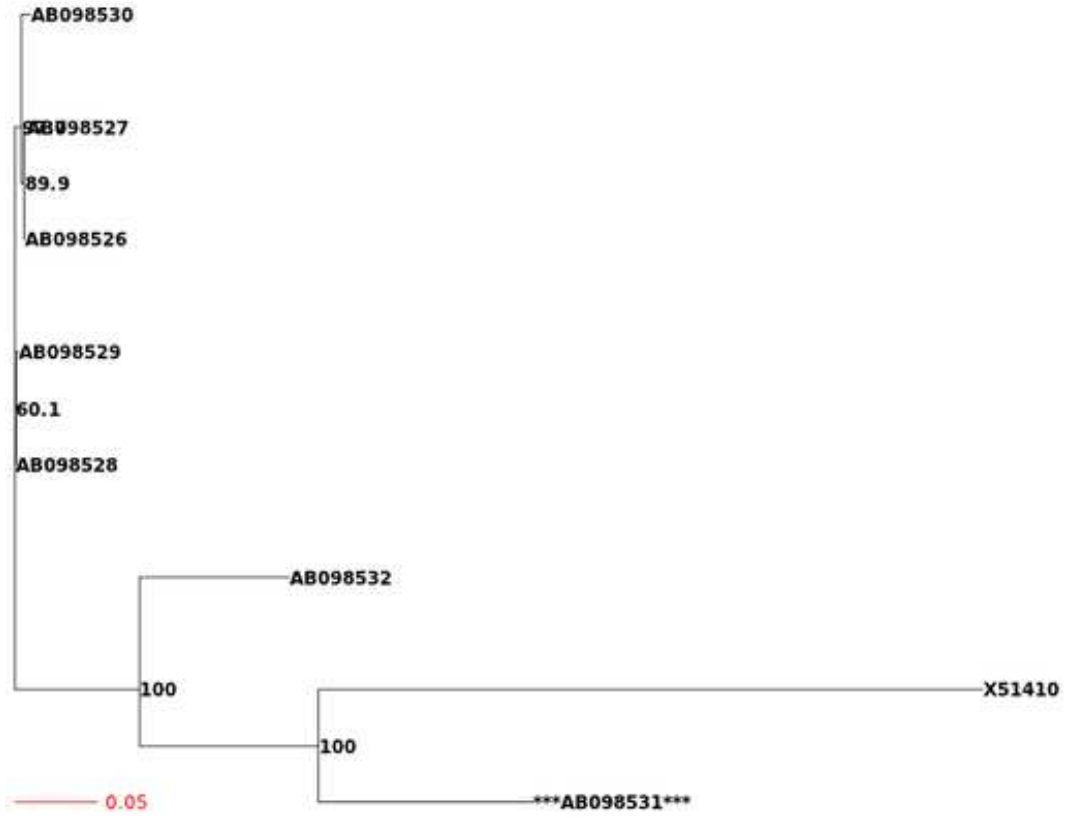
Το Pinda προώθησε τα δεδομένα της εργασίας στο σύστημα queue management, ενημερώνοντας το χρήστη για τον αριθμό προτεραιότητας της εργασίας και για τον εκτιμώμενο χρόνο ολοκλήρωσής της.

The nt database is partially non-redundant.
You are advised to pay extra attention when interpreting the results.

This sequence has been identified as "AB098531" from "Laticauda semifasciata".

Alignment of all hits | Alignment of top hits

Possible duplications of AB098531.	Z Value	Level Of Confidence
AB098532	1.14	74.7%
AB098530	1.12	73.5%
AB098526	0.57	
AB098527	0.55	
AB098528	-1.10	
AB098529	-1.12	
X51410	-1.17	



NJ trees produced in .ph/.phb format

Temporary files from this job, including alignments, .ph/.phb trees and plotted trees, will be **DELETED** after ten days.

Τα τελικά αποτελέσματα του προγράμματος ελήφθησαν μέσω e-mail. Η αρχική αλληλουχία φαίνεται στον τίτλο του πίνακα, αλλά και στο δένδρο, σημειωμένη με τρία άστρα (*) εκατέρωθεν. Οι καταχωρήσεις AB098531 (*αρχική*), AB098532 (*Level of Confidence 74.7%*) και AB098530 (*Level of Confidence 73.5%*) αντιστοιχούν στα γονίδια LSIII, NLNTP (ψευδογονίδιο)

και Etxc του *Laticauda semifasciata* αντίστοιχα, τα οποία όπως αναφέρθηκε παραπάνω έχουν προκύψει μέσω γονιδιακών διπλασιασμών. Στα υπόλοιπα αποτελέσματα συναντάμε άλλη μια καταχώρηση του Etxc και δύο καταχωρήσεις για καθένα από τα Etxa και Etxb. Η περιστασιακή ύπαρξη πολλαπλών καταχωρήσεων για την ίδια κωδική περιοχή (CDS) είναι και ο λόγος που η βάση nt χαρακτηρίζεται ως "partially non-redundant". Συμπερασματικά, το αποτέλεσμα του προγράμματος είναι ορθό, καθώς συμφωνεί με τη βιβλιογραφία.

Ο χρόνος που απαιτήθηκε για την ολοκλήρωση αυτής της εργασίας ήταν **3.5 λεπτά** σε έναν υπολογιστή με επεξεργαστή Intel Core i3-2120 CPU στα 3.30GHz, μνήμη RAM 8 GB και Λειτουργικό Σύστημα Ubuntu GNU/Linux 11.10 Oneiric Ocelot x86_64 server.

3.2 LWS-1 και LWS-2 στο Zebrafish

3.2.1 Βιβλιογραφικά δεδομένα

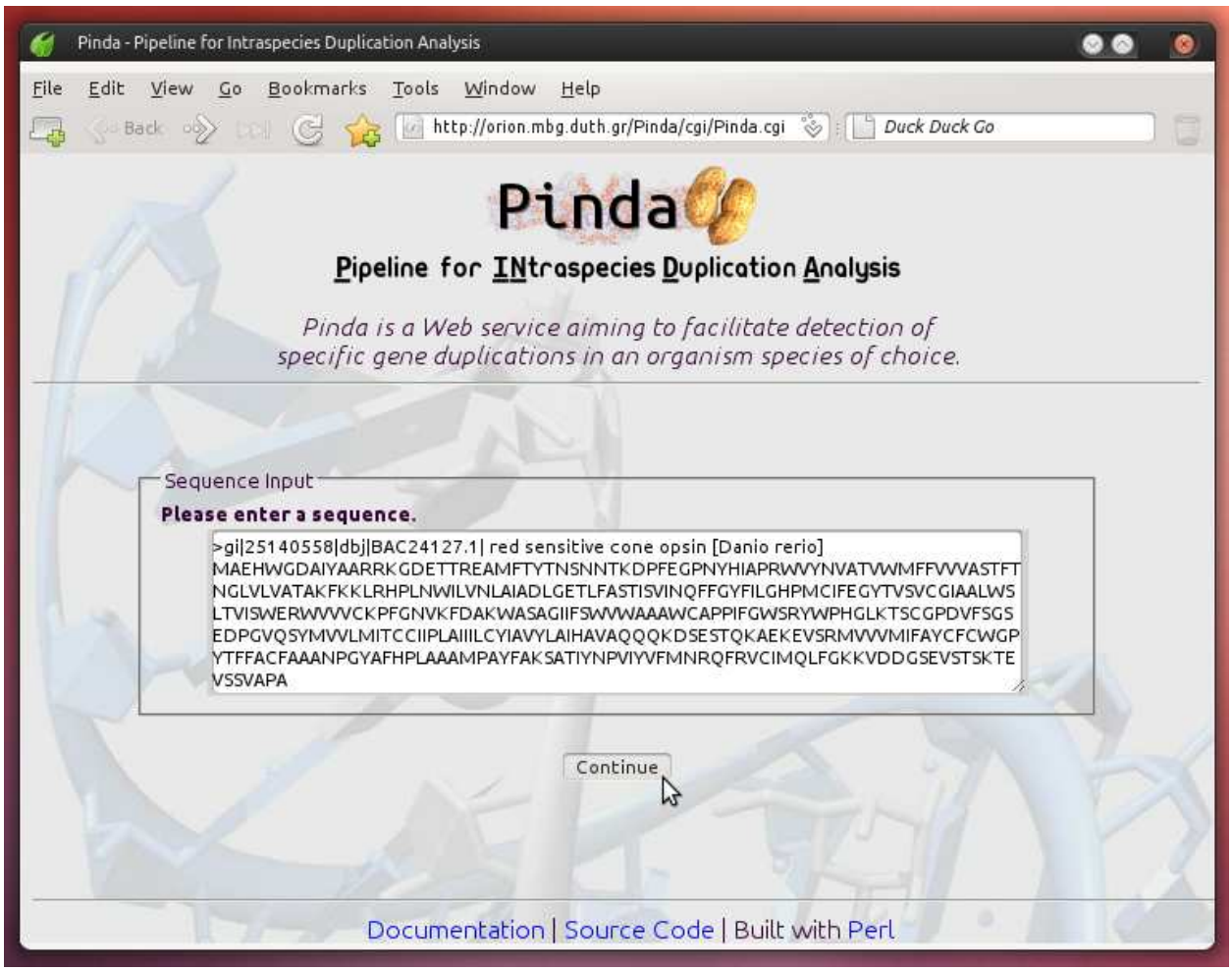
Η έγχρωμη όραση των σπονδυλωτών οφείλεται στην παρουσία πολλαπλών τάξεων κυττάρων του αμφιβληστροειδούς χιτώνα, των κωνίων, καθεμία εκ των οποίων χαρακτηρίζεται από διαφορετικό εύρος απορρόφησης φωτός. Το εύρος απορρόφησης καθορίζεται από την οπτική χρωστική του κυττάρου, η οποία αποτελείται από το σύμπλοκο μιας οψίνης και ενός χρωμοφόρου. Οι οψίνες είναι φωτοευαίσθητοι μεμβρανικοί υποδοχείς, συζευγμένοι με G-πρωτεΐνες. Στα σπονδυλωτά υπάρχουν πέντε τύποι οπτικών οψινών, οι οποίες και αναγνωρίζουν διαφορετικό μήκος κύματος φωτός. [35]

Στο Zebrafish (*Danio rerio*) τα γονίδια LWS-1 και LWS-2, τα οποία αναγνωρίζουν το κόκκινο χρώμα, βρίσκονται σε διαδοχική διάταξη και εκφράζονται στα κύτταρα του μακρού μέλους των διπλών κωνίων. Τα γονίδια των οψινών είναι ένα καλό παράδειγμα γονιδιακών διπλασιασμών για τη μελέτη των μηχανισμών του subfunctionalization και neofunctionalization. [35]

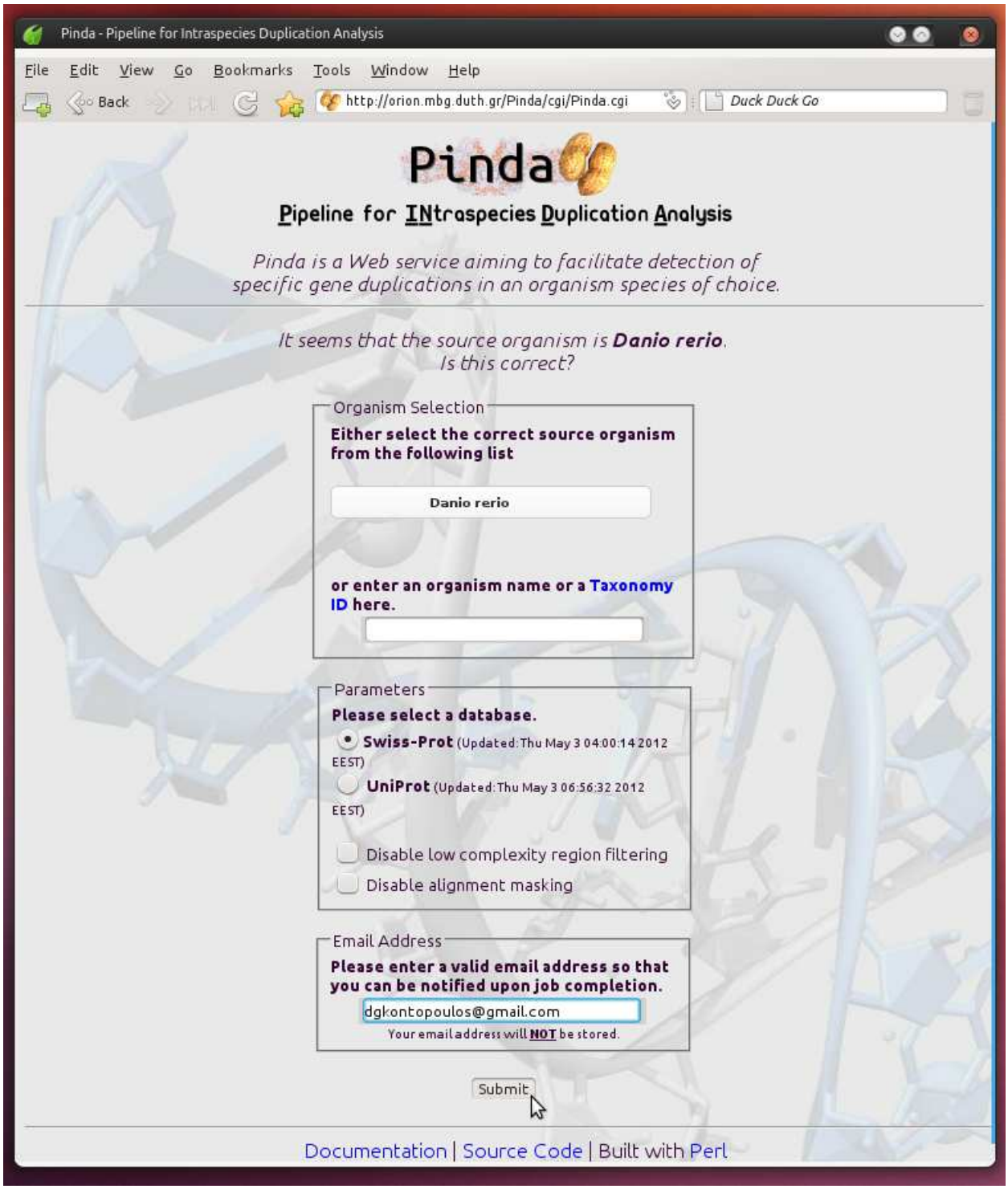
Το ψάρι Zebrafish (*Danio rerio*).
Αναπαράγεται ελεύθερα από:
`http://commons.wikimedia.org/wiki/File:Danio_rerio_port.jpg#`

Τα γονίδια LWS-1 (opn1lw1) και LWS-2 (opn1lw2) στο χρωμόσωμα 11 του γονιδιώματος του Zebrafish.

3.2.2 Στιγμιότυπα της εφαρμογής



Για το παράδειγμα αυτό ελήφθη η αμινοξική αλληλουχία του γονιδίου LWS-1 (opn1lw1) του Zebrafish από τη βάση δεδομένων UniProt, η οποία και εισήχθη στην αρχική σελίδα του Pinda. Αφού πατήθηκε το κουμπί "Submit", το πρόγραμμα ξεκίνησε τη λειτουργία του, εκτελώντας BLASTP για την αλληλουχία στόχο. Μέχρι την ολοκλήρωση του BLASTP, εμφανιζόταν σχετική εικόνα με τις λέξεις "*Please wait*".

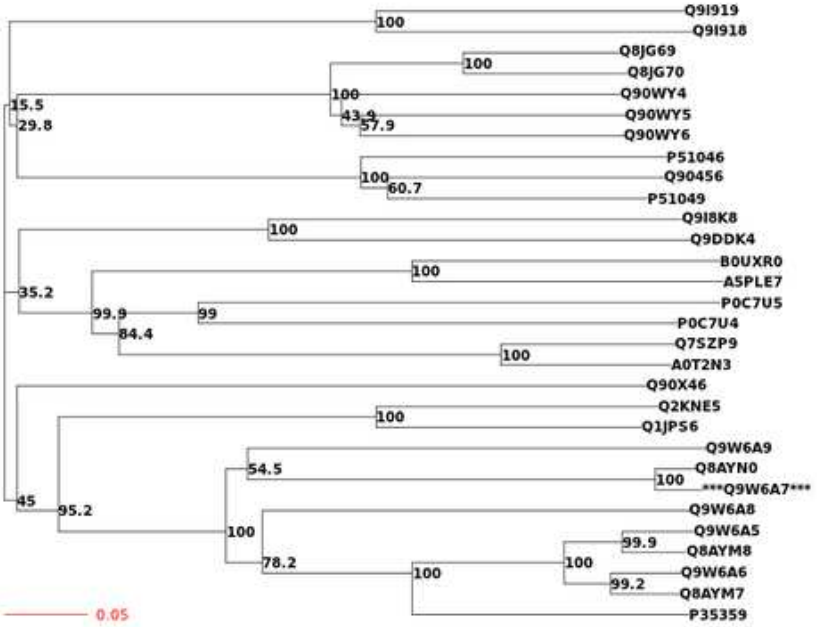


Μόλις ολοκληρώθηκε το BLASTP, εμφανίστηκε μια λίστα οργανισμών από τους οποίους προέκυψαν αποτελέσματα για τη συγκεκριμένη αλληλουχία, ταξινομημένη ως προς την τιμή e-value και κάποιες επιλογές για την ανάλυση. Αφού επιλέχθηκε ο οργανισμός Danio rerio, πατήθηκε το κουμπί "Submit", οπότε το Pinda προώθησε τα δεδομένα της εργασίας στο σύστημα queue management.

This sequence has been identified as "Q9W6A7" from "Danio rerio".

Alignment of all hits | Masked Alignment | Alignment of top hits

Possible duplications of Q9W6A7.	Z Value	Level Of Confidence	GOs in common (out of 6)	GOs not found in Q9W6A7
Q8AYN0	5.27	100.0%	4	2
Q9W6A9	0.01		4	2
Q8AYM8	-0.05		4	2
Q8AYM7	-0.05		4	2
Q9W6A6	-0.05		4	2
P35359	-0.05		3	3
Q9W6A8	-0.05		4	3
Q9W6A5	-0.06		4	3
Q1JPS6	-0.07		3	2
Q2KNE5	-0.07		4	2
Q90X46	-0.19		2	3
Q9I8K8	-0.24		1	7
Q9DDK4	-0.24		1	5
B0UXR0	-0.24		1	2
A5PLE7	-0.24		1	2
A0T2N3	-0.25		1	5
Q7SZP9	-0.25		1	4
P0C7U4	-0.25		1	3
P0C7U5	-0.25		1	3
Q9I919	-0.26		2	1
Q9I918	-0.26		2	1
Q8JG69	-0.27		0	3
Q8JG70	-0.27		0	3
P51046	-0.27		1	3
P51049	-0.27		1	3
Q90456	-0.27		1	3
Q90WY4	-0.27		0	3
Q90WY6	-0.27		1	3
Q90WY5	-0.27		1	3



NJ trees produced in .ph/.phb format

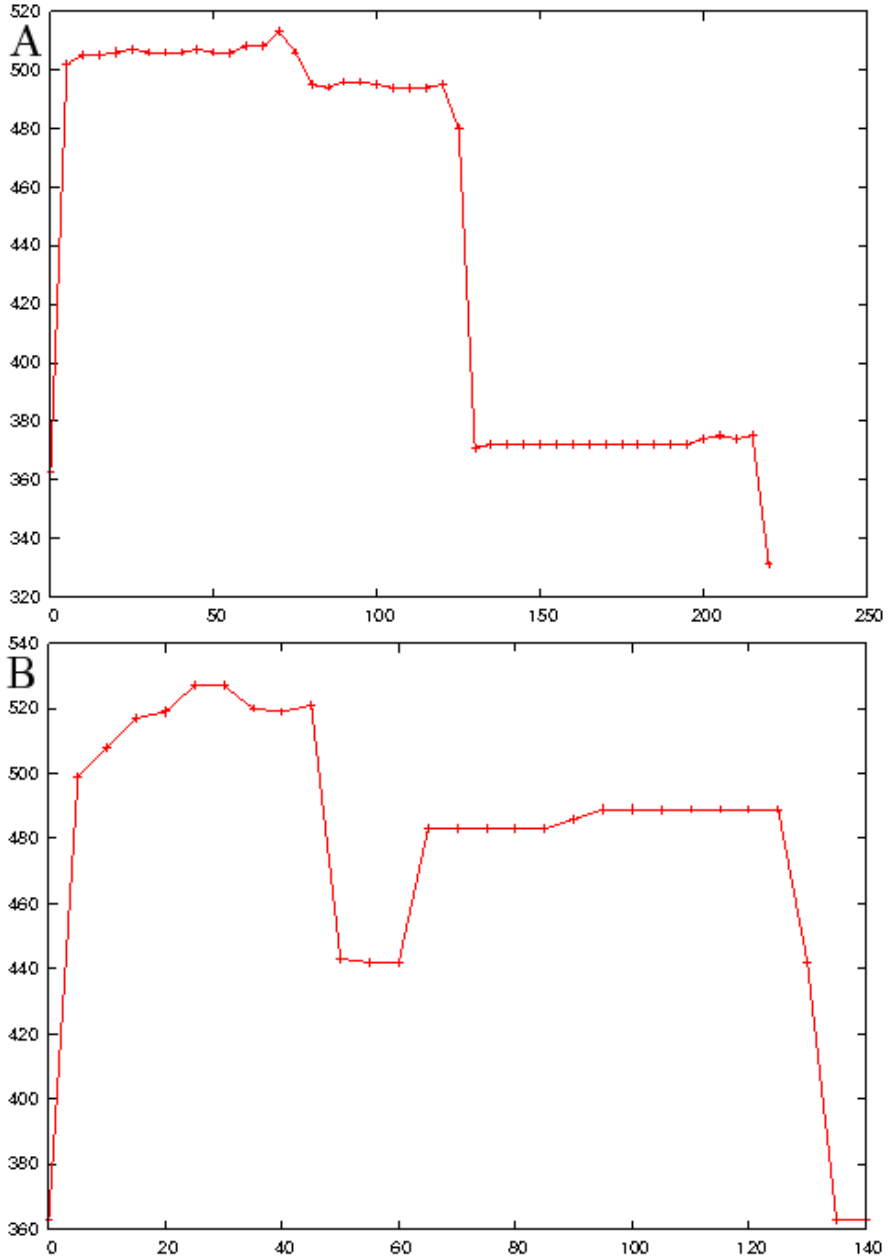
Temporary files from this job, including alignments, .ph/.phb trees and plotted trees, will be **DELETED** after ten days.

Τα τελικά αποτελέσματα του προγράμματος ελήφθησαν μέσω e-mail. Η αρχική αλληλουχία φαίνεται στον τίτλο του πίνακα, αλλά και στο δένδρο, σημειωμένη με τρία άστρα (*) εκατέρωθεν. Οι καταχωρήσεις Q9W6A7 (*αρχική*) και Q8AYN0 (*Level of Confidence 100%*) αντιστοιχούν στα γονίδια LWS-1 και LWS-2 του Zebrafish αντίστοιχα, τα οποία όπως αναφέρθηκε παραπάνω έχουν προκύψει μέσω άμεσου γονιδιακού διπλασιασμού. Τα υπόλοιπα αποτελέσματα αναφέρονται σε άλλες οψίνες ή γενικότερα υποδοχείς συζευγμένους με G-πρωτεΐνες του Zebrafish, όπως ο υποδοχέας απελίνης και ο υποδοχέας-1 της 1-φωσφορικής σφιγγοσίνης. Συμπερασματικά, το αποτέλεσμα του προγράμματος είναι ορθό, καθώς συμφωνεί με τη βιβλιογραφία.

Ο χρόνος που απαιτήθηκε για την ολοκλήρωση αυτής της εργασίας ήταν **2.5 λεπτά** σε έναν υπολογιστή με επεξεργαστή Intel Core i3-2120 CPU στα 3.30GHz, μνήμη RAM 8 GB και Λειτουργικό Σύστημα Ubuntu GNU/Linux 11.10 Oneiric Ocelot x86_64 server.

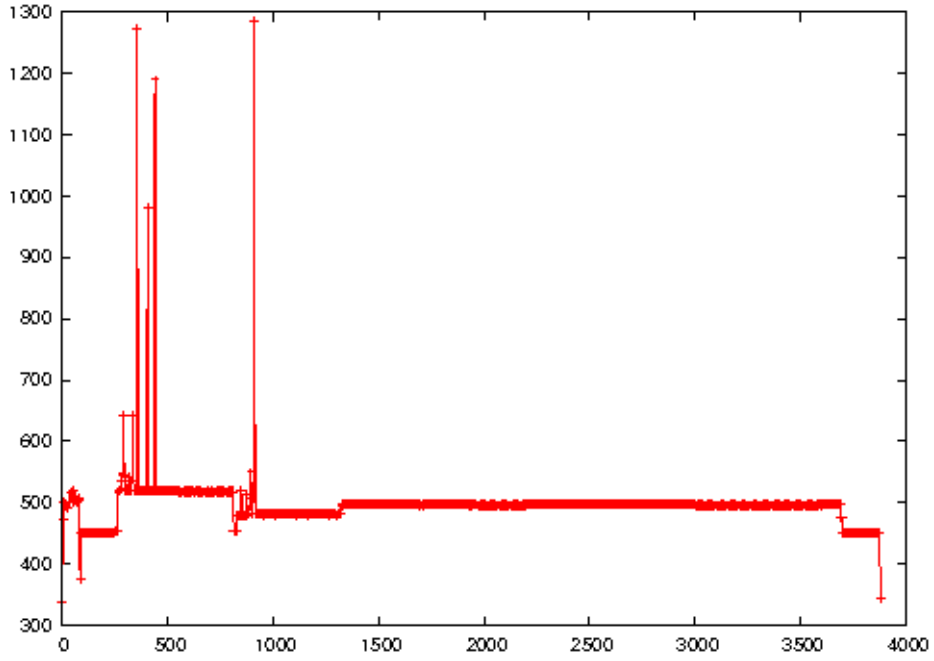
3.3 Απαιτήσεις φυσικής μνήμης και χρονισμός

Οι απαιτήσεις φυσικής μνήμης ποικίλλουν ανάλογα με το είδος της αρχικής αλληλουχίας (πρωτεϊνική ή νουκλεοτιδική), το μήκος της και το πλήθος των αποτελεσμάτων. Ενδεικτικά παρατίθενται τα γραφήματα χρησιμοποίησης φυσικής μνήμης στο server, για τα παραδείγματα των Ενοτήτων 3.1 και 3.2.



Γραφήματα κατανάλωσης φυσικής μνήμης **(A)** στο παράδειγμα της νουκλεοτιδικής αλληλουχίας AB098531 της *Laticauda semifasciata* και **(B)** στο παράδειγμα της αμινοξικής αλληλουχίας Q9W6A7 του *Danio rerio*. Ο οριζόντιος άξονας αντιπροσωπεύει το χρόνο (σε δευτερόλεπτα) και ο κατακόρυφος την κατανάλωση μνήμης (σε MB).

Αξίζει να γίνει σύγκριση με το γράφημα κατανάλωσης μνήμης από ένα άλλο παράδειγμα, αμινοξικής αλληλουχίας με 472 αποτελέσματα.



Γράφημα κατανάλωσης φυσικής μνήμης για την αμινοξική αλληλουχία P31749 του *Homo sapiens*. Ο οριζόντιος άξονας αντιπροσωπεύει το χρόνο (σε δευτερόλεπτα) και ο κατακόρυφος την κατανάλωση μνήμης (σε MB).

Με βάση τα παραπάνω, συμπεραίνουμε πως όταν τα αποτελέσματα είναι σχετικά λίγα, τότε η κατανάλωση μνήμης δεν ξεπερνά τα 530 περίπου megabytes. Καθώς αυξάνεται ο αριθμός των αποτελεσμάτων, αυξάνεται και η κατανάλωση μνήμης από το πρόγραμμα, ειδικά σε περιπτώσεις ανάλυσης πρωτεϊνικών αλληλουχιών, όπου παρατηρήθηκε ένα μέγιστο περίπου 1.300 megabytes. Λαμβάνοντας υπ' όψιν ότι σε κατάσταση αδράνειας του server χρησιμοποιούνται περίπου 360 megabytes φυσικής μνήμης, γίνεται αντιληπτό ότι το πρόγραμμα δεν είναι ιδιαίτερα απαιτητικό.

Ο χρόνος λειτουργίας του προγράμματος ποικίλλει εξίσου, σε βαθμό που εξαρτάται από το περιεχόμενο της ανάλυσης, όπως αναφέρθηκε και παραπάνω. Στα παραδείγματα της *Laticauda semifasciata* και του *Danio rerio*, ο απαιτούμενος χρόνος για την ολοκλήρωση της ανάλυσης ήταν 3.5 (Ενότητα 3.1) και 2.5 λεπτά (Ενότητα 3.2) αντίστοιχα. Ωστόσο, έχουν παρατηρηθεί περιπτώσεις όπου η ανάλυση μπορεί να διαρκέσει από κάποια δευτερόλεπτα έως και αρκετές ώρες. Σύμφωνα με τα στατιστικά στοιχεία που συγκεντρώνονται από το πρόγραμμα, ο μέσος όρος του χρόνου ολοκλήρωσης μιας ανάλυσης προσδιορίζεται στα 30 λεπτά περίπου, τόσο για αμινοξικές, όσο και για νουκλεοτιδικές αλληλουχίες.

Κεφάλαιο 4

Πηγαίος Κώδικας

Στο κεφάλαιο αυτό παρατίθεται ο πηγαίος κώδικας των προγραμμάτων που απαρτίζουν το Pinda. Το **Pinda.cgi** ευθύνεται για τη δημιουργία του περιβάλλοντος CGI, τη δημιουργία της ιστοσελίδας και την αποστολή των δεδομένων του χρήστη στο **Pinda_exec.pl**. Το τελευταίο αναλαμβάνει το μεγαλύτερο μέρος της επεξεργασίας των δεδομένων. Για την απεικόνιση του δένδρου και την εξαγωγή δεδομένων από αυτό ευθύνεται το **Pinda.R**.

4.1 Προαπαιτούμενα

- Λειτουργικό Σύστημα **GNU/Linux**. (*Στην περίπτωσή μας χρησιμοποιήθηκε η διανομή Ubuntu 11.10 Oneiric Ocelot x86_64.*)
- Εφαρμογή **Server**. (*Στην περίπτωσή μας χρησιμοποιήθηκε ο lighttpd με την έκδοση 1.4.28.*)
- Έκδοση της **Perl** v5.12.4 ή νεότερη.
- Modules της Perl:

  -**Bio**::**AlignIO** (έκδοση 1.006901 ή νεότερη).

  -**Bio**::**DB**::**Taxonomy** (έκδοση 1.006901 ή νεότερη).

  -**Bio**::**Perl** (έκδοση 1.006901 ή νεότερη).

  -**Bio**::**Root**::**IO** (έκδοση 1.006901 ή νεότερη).

  -**Bio**::**Search**::**HSP**::**GenericHSP** (έκδοση 1.006901 ή νεότερη).

  -**Bio**::**SearchIO** (έκδοση 1.006901 ή νεότερη).

  -**Bio**::**Search**::**Iteration**::**GenericIteration** (έκδοση 1.006901 ή νεότερη).

-**Bio::Search::Result::BlastResult** (έκδοση 1.006901 ή νεότερη).

-**Bio::SeqIO** (έκδοση 1.006901 ή νεότερη).

-**Bio::Tools::Run::StandAloneBlastPlus** (έκδοση 1.006901 ή νεότερη).

-**Bio::Tools::Run::StandAloneBlastPlus::BlastMethods** (έκδοση 1.006901 ή νεότερη).

-**Bio::TreeIO** (έκδοση 1.006901 ή νεότερη).

-**Bio::Tree::TreeFunctionsI** (έκδοση 1.006901 ή νεότερη).

-**Bio::Tree::TreeI** (έκδοση 1.006901 ή νεότερη).

-**CGI** (έκδοση 3.59 ή νεότερη).

-**Data::Validate::Email** (έκδοση 0.04 ή νεότερη).

-**File::stat** (έκδοση 1.05 ή νεότερη).

-**FreezeThaw** (έκδοση 0.5001 ή νεότερη).

-**List::MoreUtils** (έκδοση 0.33 ή νεότερη).

-**LWP::Simple** (έκδοση 6.00 ή νεότερη).

-**Math::Cephes** (έκδοση 0.47 ή νεότερη).

-**MIME::Lite** (έκδοση 3.028 ή νεότερη).

-**Statistics::Basic** (έκδοση 1.6607 ή νεότερη).

-**Sys::CPU** (έκδοση 0.52 ή νεότερη).

-**Time::localtime** (έκδοση 1.02 ή νεότερη).

- Έκδοση της **R** 2.13.1 ή νεότερη.
- Πακέτα της R:

  -**ape** (έκδοση 3.0-1 ή νεότερη).

  -**ade4** (έκδοση 1.4-17 ή νεότερη).

- Βάσεις Δεδομένων: **SwissProt**, **UniProt** (*SwissProt* + *TrEMBL*), **nt** (*GENBANK/EMBL/DDBJ* + *RefSeq*).
- Πακέτο Εφαρμογής **BLAST+**.
- Εφαρμογή **Clustal Omega** με έκδοση 1.0.3 ή νεότερη.
- Εφαρμογή **ClustalW** με έκδοση 2.1 ή νεότερη.

- Εφαρμογή **Kalign** με έκδοση 2.04 ή νεότερη.
- Εφαρμογή **sreformat**.
- Εφαρμογή **zip** με έκδοση 3.0-4 ή νεότερη.
- Εφαρμογή **ZORRO** και η προαπαιτούμενή της **FastTree** με έκδοση 2.1.4.
- Εφαρμογή **slurm-llnl** με έκδοση 2.2.7-1.
- Σύνδεση στο **Διαδίκτυο**.

4.2 Pinda.cgi

Το Pinda.cgi είναι υπεύθυνο για τη δημιουργία του περιβάλλοντος αλληλεπίδρασης με το χρήστη. Παραλαμβάνει τις εργασίες και τις παραπέμπει στο Pinda_exec.pl, μέσω του συστήματος queue management.

```
1  #!/usr/bin/env perl
2
3  ####################
4  #$VERSION = '0.02';#
5  ####################
6
7  =head1 NAME
8
9  PINDA - Pipeline for INtraspecies Duplication Analysis
10
11 =head1 DESCRIPTION
12
13 A Web service aiming to facilitate detection of specific gene
14 duplications in an organism species of choice.
15
16 =head1 AVAILABILITY
17
18 Pinda is located at http://orion.mbg.duth.gr/Pinda/cgi/Pinda.cgi,
19 whereas the Source Code can be obtained at
20 https://github.com/dgkontopoulos/Pinda.
21
22 =head1 USAGE
23
24 At Pinda's starting page, you may enter a Protein or DNA sequence. After
25 clicking the "Continue" button, Pinda will take you to the next page,
26 where you may choose the organism and the database within which the
27 duplication analysis will take place. You also have to enter your email
28 address, so that you can be notified after the job is complete. Past
29 this point, your task has entered the queue. You will receive its
30 results via email, including a possible duplications table, a multiple
31 sequence alignment and a dendrogram.
32
33 =head1 FLOWCHART
```

```
34
35  http://orion.mbg.duth.gr/Pinda/flowchart.png
36
37  =head1 AUTHOR
38
39  Pinda has been developed by Dimitrios - Georgios Kontopoulos as his
40  final year project, under the supervision of Prof. Nicholas M. Glykos
41  at the Department of Molecular Biology and Genetics of Democritus
42  University of Thrace, Greece.
43
44  =head1 LICENSE
45
46  This program is free software: you can redistribute it and/or modify
47  it under the terms of the GNU Affero General Public License as
48  published by the Free Software Foundation, either version 3 of the
49  License, or (at your option) any later version.
50
51  For more information, see http://www.gnu.org/licenses/.
52
53  =cut
54
55  use Bio::DB::Taxonomy;
56  use Bio::Search::HSP::GenericHSP;
57  use Bio::Tools::Run::StandAloneBlastPlus::BlastMethods;
58  use CGI qw(:standard);
59  use Data::Validate::Email qw(is_email);
60  use File::stat;
61  use FreezeThaw qw(freeze thaw);
62  use List::MoreUtils qw(uniq);
63  use LWP::Simple qw(get);
64  use Sys::CPU;
65  use Time::localtime;
66
67  use feature qw(say);
68
69  use strict;
70  use warnings;
71
72  open STDERR, '>', '/tmp/err' or die $!;
73
74  ###################################
75  #Initializing the CGI environment.#
76  ###################################
77  my $query = CGI->new;
78  print $query->header;
79  print <<"ENDHTML";
80  <!DOCTYPE html
81          PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
82           "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
83  <html xmlns="http://www.w3.org/1999/xhtml" lang="en-US" xml:lang="en-US">
84  <head>
85  <meta name="title" content="Pinda - Pipeline for Intraspecies Duplication Analysis" />
86  <title>Pinda - Pipeline for Intraspecies Duplication Analysis</title>
87  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
88  <link rel="stylesheet" href="../css/Pinda.css" type="text/css" media="screen">
```

```
<meta name="description" content="Pinda is a Web service aiming to facilitate detection of
specific gene duplications in an organism species of choice." />
<link rel="image_src" href="http://orion.mbg.duth.gr/Pinda/pindalogo.png" />

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js"
type="text/javascript" charset="utf-8"></script>

<script src="../js/jquery.dropkick.1-0.1.js" type="text/javascript"
charset="utf-8"></script><script type="text/javascript" charset="utf-8">
\$(function () {
    \$('.default').dropkick();
});
</script>

<script type="text/javascript">

  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'UA-31909639-1']);
  _gaq.push(['_trackPageview']);

  (function() {
    var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') +
    '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
  })();

</script>

</head>
<body background='../background.jpg'>
<LINK REL='SHORTCUT ICON' HREF='../pinda.ico'>
<center><a href='http://orion.mbg.duth.gr/Pinda/cgi/Pinda.cgi'>
<img src='http://orion.mbg.duth.gr/Pinda/pindalogo.png' width=354
height=74 alt='Pipeline for INtraspecies Duplication Analysis'></a><br><br>
<font size=3 face='Georgia' color='330033'>
<i>Pinda is a Web service aiming to facilitate detection of<br>
specific gene duplications in an organism species of choice.</i>
</font>
<p style='width: 500px; text-align:center;margin-bottom:1px;margin-top:1px'>
<hr/>
</p>
</center>
<div style="position:fixed;bottom:0;width:100%">
<div style="width:300;margin:0px auto;">
<hr /><center>
<font size='3' face='Georgia' color='330033'>
<a href='http://orion.mbg.duth.gr/Pinda/documentation.html'>Documentation</a>
|
<a href="https://github.com/dgkontopoulos/Pinda/">Source Code</a>
| Built with <a href="http://www.perl.org/">Perl</a></font></center>
</div></div>
ENDHTML

################################
```

```
144 #Starting page with input box.#
145 ################################
146 if ( !$query->param )
147 {
148     ##########################################
149     #Get the timestamp for the database files.#
150     ##########################################
151     print <<"ENDHTML";
152     <center>
153     <br><br>
154     <form id = 'submit' method = 'post' action=http://orion.mbg.duth.gr/Pinda/cgi/Pinda.cgi>
155     <fieldset class="fieldset-auto-width"><legend><font size='2'
156     face='Georgia' color='330033'>Sequence Input</font></legend>
157     <font size='2' face='Georgia' color='330033'>
158     <p style='width: 570px; text-align:left;margin-bottom:1px;margin-top:1px'>
159     <b>Please enter a sequence.</b><br>
160     </font><center>
161     <textarea name='sequence' rows=7 cols=70 style='font-family:courier new'
162     ></textarea></center></fieldset><br><br>
163     <input id='submit' TYPE='submit' value=' Continue '></form></center>
164 
165     </body></html>
166 ENDHTML
167 }
168 
169 ######################################
170 #After the original input is given...#
171 ######################################
172 elsif ( !$query->param('button') && !$query->param('dropdown') )
173 {
174     my $SWISSPROT = '/usr/local/databases/Swissprot/uniprot_sprot.fasta';
175     my $UNIPROT   = '/usr/local/databases/UniProt/UniProt.fasta';
176     my $NT        = '/usr/local/databases/nt/nt.fasta';
177     my $list      = 0;
178     my $list2     = 0;
179     my $number    = 0;
180 
181     my (
182         $db,        $dbfetch,    $hit,      $hit_check,
183         $input_hit, $match_line, $org,      $organism,
184         $orghash,   $tmp_fh,     @organism, %organisms
185     );
186     print '<center>';
187     my $string = $query->param('sequence');
188 
189     ######################
190     #Database selection.#
191     ######################
192     my $string2;
193     if ( $string =~ /^>.*\n/ )
194     {
195         $string2 = $';
196     }
197     else
198     {
```

```
199         $string2 = $string;
200     }
201     chomp $string2;
202     do
203     {
204         $string2 =~ s/\n//;
205         $string2 =~ s/\s//;
206     } while ( $string2 =~ /\n/ || $string2 =~ /\s/ );
207     if ( $string2 =~ /^[A|C|G|T]+$/i )
208     {
209         $db = $NT;
210     }
211     elsif ( $string2 =~
212         /^[R|H|K|D|E|S|T|N|Q|C|G|P|A|V|I|L|M|F|Y|W|X|U|O|B|Z|J]+$/i )
213     {
214         $db = $SWISSPROT;
215     }
216     else
217     {
218         print <<"ENDHTML";
219         <br><br><br><br><br>
220         <font size='3' face='Georgia' color='330033'>
221         <b>ERROR!</b><br>This sequence looks neither like an amino acid one nor
222         like a nucleotide one.<br><br>
223         Please <a href='javascript:history.go(-1)'><u>go back</u></a> and enter
224         a Protein or DNA sequence.
225         </font>
226 ENDHTML
227         exit;
228     }
229
230     my $timezone = `date`;
231     if ( $timezone =~ /EEST/ )
232     {
233         $timezone = 'EEST';
234     }
235     else
236     {
237         $timezone = 'EET';
238     }
239
240     ############################
241     #Handling empty sequences#
242     ############################
243     if ( $string =~ /^\s*$/ || $string !~ /\n?\w+/ )
244     {
245         print <<"ENDHTML";
246         <br><br><br><br><br>
247         <font size='3' face='Georgia' color='330033'>
248         <b>ERROR!</b> No sequence entered!<br><br>
249         Please <a href='javascript:history.go(-1)'><u>go back</u></a> and enter
250         a sequence.
251         </font>
252 ENDHTML
253         exit;
```

```
254     }
255     ######################################################
256     #Making sure that sequences will be in FASTA format.#
257     ######################################################
258     else
259     {
260         if ( $string =~ /(>)/ )
261         {
262             if ( $' =~ /(>)/ )
263             {
264                 $string = '>' . $`;
265             }
266         }
267         elsif ( $string !~ /^>/ )
268         {
269             $string = ">\n" . $string;
270         }
271         ############################################
272         #Handling organism declaration in proteins.#
273         ############################################
274         if (   ( $string =~ /OS\=(\w+)\s+(\w+)/ )
275             || ( $string =~ /\[(\w+)\s+(\w+)/ ) )
276         {
277             $organism = $1 . q{ } . $2;
278             if (   ( $' =~ /^$/ )
279                 || ( $' =~ /^\s+$/ ) )
280             {
281                 print <<"ENDHTML";
282                 <br><br><br><br><br><br>
283                 <font size='3' face='Georgia' color='330033'>
284                 <b>ERROR!</b> No sequence entered!<br><br>
285                 Please <a href='javascript:history.go(-1)'><u>go back</u></a>
286                 and enter a sequence.
287                 </font>
288 ENDHTML
289                 exit;
290             }
291             if ( $string =~ /[|](\w{6,})[|]/ )
292             {
293                 $hit_check = $1;
294                 if ( $hit_check =~ /\d/ && $hit_check =~ /\D/ )
295                 {
296                     $input_hit = $hit_check;
297                 }
298             }
299         }
300     }
301     print <<"ENDHTML";
302     <div id="loading" class='unhidden'>
303     <br><br><br><br><br><br>
304     <center><img src="../loading.gif"></center><br>
305     </div>
306 ENDHTML
307
308     ########################################
```

```
309     #Get process id and set BLAST parameters#
310     ########################################
311     my $prid = time . $$;
312     my $tmp  = '../tmps/blast/' . $prid . '.tmp';
313     my $out  = '../outs/blast/' . $prid . '.tmp';
314     open $tmp_fh, '>', $tmp or die $!;
315     print {$tmp_fh} "$string";
316     close $tmp_fh or die $!;
317     my $query_line;
318 
319     if ( $string =~ />.*\n/ )
320     {
321         $query_line = $';
322         do
323         {
324             $query_line =~ s/\n//;
325             $query_line =~ s/\s//;
326         } while ( $query_line =~ /\n/ || $query_line =~ /\s/ );
327     }
328 
329     ############################
330     #Avoiding browser timeout.#
331     ############################
332     my $timeout = 60;
333     $SIG{ALRM} = sub { say q{.}; alarm $timeout; };
334     alarm $timeout;
335 
336     ##############################
337     #Get the number of cpu cores.#
338     ##############################
339     my $cpu_n = Sys::CPU::cpu_count();
340 
341     ######################
342     #BLASTP for Proteins.#
343     ######################
344     if ( $db !~ /nt[.]fasta/ )
345     {
346         say '<!--';
347         system(
348 "blastp -query $tmp -db $db -evalue 0.1 -num_threads $cpu_n -out $out -seg yes"
349           ) == 0
350           or die $?;
351 
352         my $blast = Bio::SearchIO->new(
353             -format => 'blast',
354             -file   => "$out"
355         );
356 
357         my $hit_old = 0;
358 
359         ############################
360         #Start parsing BLAST output.#
361         ############################
362         while ( my $result = $blast->next_result )
363         {
```

```
364             while ( my $hit = $result->next_hit )
365             {
366                 if ( $hit->description =~ /OS\=\w+\s+\w+/ )
367                 {
368                     if ( $hit->accession =~ /tr[|](\w+)[|]/ )
369                     {
370                         my $ac = $1;
371                         local $/ = undef;
372                         open my $out_fh, '<', $out or die $!;
373                         while ( my $readline = <$out_fh> )
374                         {
375                             if ( $readline =~ />tr[|]$ac[|]/ )
376                             {
377                                 $readline = $';
378                                 if ( $readline =~ /OS\=(\w+\s+\w+)\s+/ )
379                                 {
380                                     $org = $1;
381                                     if ( $org =~ /\n/ )
382                                     {
383                                         $org = $` . $';
384                                     }
385                                 }
386                             }
387                         }
388                         close $out_fh or die $!;
389                     }
390                     else
391                     {
392                         my $ac = $hit->accession;
393                         local $/ = undef;
394                         open my $out_fh, '<', $out or die $!;
395                         while ( my $readline = <$out_fh> )
396                         {
397                             if ( $readline =~ />sp[|]$ac[|]/ )
398                             {
399                                 $readline = $';
400                                 if ( $readline =~ /OS\=(\w+\s+\w+)\s+/ )
401                                 {
402                                     $org = $1;
403                                     if ( $org =~ /\n/ )
404                                     {
405                                         $org = $` . $';
406                                     }
407                                 }
408                             }
409                         }
410                     }
411                     ######################################
412                     #Populate the organism dropdown list.#
413                     ######################################
414                     $organism[$list] = $org;
415                     $list++;
416 
417                     ############################################################
418                     #Populate a hash with the first result from every organism.#
```

```
                    ###########################################################
                    while ( my $hsp = $hit->next_hsp )
                    {
                        my $match_line = $hsp->hit_string;
                        do
                        {
                            $match_line =~ s/\n//;
                            $match_line =~ s/\s//;
                          } while ( $match_line =~ /\n/
                            || $match_line =~ /\s/ );
                        if ( $match_line eq $query_line )
                        {
                            if ( !( defined $organisms{$org} ) )
                            {
                                if ( $hit->accession =~ /tr[|](\w+)[|]/ )
                                {
                                    $organisms{$org} = $1;
                                }
                                else
                                {
                                    $organisms{$org} = $hit->accession;
                                }
                            }
                        }
                        else
                        {
                            $organisms{$org} = q{};
                        }
                    }
                }
            }
        }
        ################################
        #Show each organism only once.#
        ################################
        my @organism2 = uniq(@organism);
        my $mikos     = @organism2;
        ######################################
        #Pass the hash to the next CGI call.#
        ######################################
        $orghash = freeze %organisms;
        alarm 0;
        say '-->';
        if ( defined $organism )
        {
            print <<"ENDHTML";
        <font size='3' face='Georgia' color='330033'>
        <i>It seems that the source organism is <b>$organism</b>.
        <br>Is this correct?</i><br><br></font>
ENDHTML
        }
        ################################
        #Creation of the dropdown list.#
        ################################
        print <<"ENDHTML";
```

```
474     <form id = 'submit' method = 'post' action=http://orion.mbg.duth.gr/Pinda/cgi/Pinda.cgi>
475     <fieldset class="fieldset-auto-width"><legend><font size='2' face='Georgia' color='330033'>
476     Organism Selection </font></legend>
477 ENDHTML
478 
479         if ( $mikos == 0 )
480         {
481             print <<"ENDHTML";
482                 <font size='2' face='Georgia' color='330033'>
483                 <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
484                 <center>
485                 No organism could be identified.<br><br>
486                 <b>Please enter an organism name or a
487                 <a href=http://www.ncbi.nlm.nih.gov/Taxonomy/>Taxonomy ID</a> here.
488                 </b><br>
489                 <center><input type='text' name='organism2' size="30" maxlength="60">
490                 </fieldset>
491 ENDHTML
492         }
493         else
494         {
495             say <<"ENDHTML";
496         <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
497     <font size='2' face='Georgia' color='330033'>
498     <b>Either select the correct source organism from the following list
499     </b></font><br><br></p>
500     <p style='width: 270px; text-align:right;margin-bottom:1px;margin-top:1px'>
501     <center><select name='organism' tabindex='1' class='default'>
502     <option value=''>---Select an organism---</option>
503 ENDHTML
504 
505             for ( 0 .. $mikos - 1 )
506             {
507                 say "<option value='$organism2[$_]'>$organism2[$_]</option>";
508             }
509             print <<"ENDHTML";
510                 </select></center></p>
511                 <br><br><br><font size='2' face='Georgia' color='330033'>
512                 <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
513                 <b>or enter an organism name or a
514                 <a href=http://www.ncbi.nlm.nih.gov/Taxonomy/>Taxonomy ID</a>
515                 here.</b><br>
516                 <center><input type='text' name='organism2' size="30" maxlength="60">
517                 </input></center></fieldset>
518 ENDHTML
519         }
520         my $sw_timestamp =
521           ctime( stat('/usr/local/databases/Swissprot/')->mtime );
522         my $uni_timestamp =
523           ctime( stat('/usr/local/databases/UniProt/')->mtime );
524         print <<"ENDHTML";
525         <br><br><fieldset class="fieldset-auto-width"><legend>
526          Parameters </legend>
527                 <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
528                 <b>Please select a database.</b><br>
```

```
529                 <font size='2'>
530         <input type="radio" name="db" value="Swiss-Prot" checked><b>Swiss-Prot</b>
531         </font><font size='1'>(Updated: $sw_timestamp $timezone)</font><br>
532         <font size='2'>
533         <input type="radio" name="db" value="UniProt"> <b>UniProt</b>
534         </font><font size='1'>(Updated: $uni_timestamp $timezone)</font>
535         <br><br><input type=checkbox name='lcr_filtering' value='1'>
536                 Disable low complexity region filtering
537         <br><input type=checkbox name='masking' value='1'>
538                 Disable alignment masking
539                 </fieldset><br>
540 ENDHTML
541     }
542     else
543     {
544         print <<"ENDHTML";
545     <form id = 'submit' method = 'post' action=http://orion.mbg.duth.gr/Pinda/cgi/Pinda.cgi>
546     <fieldset class="fieldset-auto-width"><legend><font size='2' face='Georgia' color='330033'>
547     Organism Selection </font></legend>
548         <font size='2' face='Georgia' color='330033'>
549         <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
550         <center>
551         <b>Please enter an organism name or a
552         <a href=http://www.ncbi.nlm.nih.gov/Taxonomy/>Taxonomy ID</a>
553         here.</b><br>
554         <center><input type='text' name='organism2' size="30" maxlength="60">
555         </fieldset>
556 ENDHTML
557         my $nt_timestamp = ctime( stat('/usr/local/databases/nt/')->mtime );
558         print <<"ENDHTML";
559         <br><br><fieldset class="fieldset-auto-width"><legend>
560          Parameters </legend>
561                 <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
562                 <b>Database</b><br>
563                 <font size='2'>
564         <input type="radio" name="db" value="nt" checked><b>nt</b>
565         </font><font size='1'>(Updated: $nt_timestamp $timezone)</font><br>
566         <br><input type=checkbox name='lcr_filtering' value='1'>
567                 Disable low complexity region filtering
568         </fieldset><br>
569 
570 ENDHTML
571     }
572 
573     print <<"ENDHTML";
574     <br><fieldset class="fieldset-auto-width"><legend> Email Address </legend>
575     <p style='width: 270px; text-align:left;margin-bottom:1px;margin-top:1px'>
576     <font size='2' face='Georgia' color='330033'><b>Please enter a valid email
577     address so that you can be notified upon job completion.</b></font><br></p>
578     <center><input type='text' name='email' size="30" maxlength="60"></input>
579     <br><font size='1'>Your email address will <b><u>NOT</u></b> be stored.
580     </fieldset></font>
581     <br><br><input type=submit name='dropdown' value='Submit'></form>
582     <br><br><br><br></p>
583 ENDHTML
```

```
584
585     if ( $db !~ /nt[.]fasta/ )
586     {
587         print <<"ENDHTML";
588     <input type=hidden name='prid' value='$prid'>
589     <input type=hidden name='db' value='$db'>
590     <input type=hidden name='organisms' value='$orghash'>
591 ENDHTML
592     }
593     else
594     {
595         print <<"ENDHTML";
596                 <input type=hidden name='prid' value='$prid'>
597                 <input type=hidden name='db' value='$db'>
598 ENDHTML
599     }
600
601     print <<"ENDHTML";
602     <script type="text/javascript">
603     document.getElementById("loading").className = "hidden";
604     </script>
605
606     </body>
607     </html>
608 ENDHTML
609 }
610 else
611 {
612     my $email = $query->param('email');
613     if ( !is_email($email) )
614     {
615         print <<"ENDHTML";
616         <br><br><br><br><br><br>
617         <center><font size='3' face='Georgia' color='330033'>
618         <b>ERROR!</b> The email address does not appear to be valid!<br><br>
619         Please <a href='javascript:history.go(-1)'><u>go back</u></a> and enter
620         a valid email address.</center></font>
621 ENDHTML
622         exit;
623     }
624
625     my ( $organism, $note );
626     if ( defined $query->param('organism') && $query->param('organism') ne q{} )
627     {
628         $organism = $query->param('organism');
629     }
630     elsif ( $query->param('organism2') ne q{} )
631     {
632         my $org_temp = $query->param('organism2');
633         if ( $org_temp =~ /\d+/ )
634         {
635             my $dbfetch = get(
636 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=taxonomy&id=$org_temp&style=raw"
637             );
638             if ( $dbfetch =~ /SCIENTIFIC NAME\s+[:] (\w.+)\n/ )
```

```
639             {
640                 $organism = $1;
641                 if ( $dbfetch !~ /RANK\s+[:] species/ )
642                 {
643                     $note = 1;
644                 }
645             }
646             else
647             {
648                 print <<"ENDHTML";
649                             <br><br><br><br><br>
650                             <center><font size='3' face='Georgia' color='330033'>
651                             <b>ERROR!</b> This Taxonomy ID is not valid!<br><br>
652                             Please <a href='javascript:history.go(-1)'><u>go back</u></a> and
653                             select an organism or enter a valid Taxonomy ID.</center></font>
654 ENDHTML
655                 exit;
656             }
657         }
658         else
659         {
660             my $db      = Bio::DB::Taxonomy->new( -source => 'entrez' );
661             my $taxonid = $db->get_taxonid($org_temp);
662             my $dbfetch = get(
663 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=taxonomy&id=$taxonid&style=raw"
664             );
665             if ( $dbfetch =~ /SCIENTIFIC NAME\s+[:] (\w.+\n)/ )
666             {
667                 if ( $dbfetch !~ /RANK\s+[:] species/ )
668                 {
669                     $note = 2;
670                 }
671             }
672             elsif ( !defined $taxonid )
673             {
674                 $note = 2;
675             }
676             $org_temp =~ s/\s+$//;
677             $organism = $org_temp;
678         }
679     }
680     else
681     {
682         print <<"ENDHTML";
683         <br><br><br><br><br>
684         <center><font size='3' face='Georgia' color='330033'>
685         <b>ERROR!</b> No organism specified!<br><br>
686         Please <a href='javascript:history.go(-1)'><u>go back</u></a> and
687         specify the source organism.</center></font>
688 ENDHTML
689         exit;
690     }
691     my $database = $query->param('db');
692     my $one;
693     if ( $database !~ /nt/ )
```

```
694     {
695         my %organisms = thaw $query->param('organisms');
696         if ( defined $organisms{$organism} && $organisms{$organism} ne q{} )
697         {
698             $one = $organisms{$organism};
699         }
700         else
701         {
702             $one = 'QUERY';
703         }
704     }
705     else
706     {
707         $one = 'QUERY';
708     }
709     my $prid = $query->param('prid');
710     if ( `ls /var/www/Pinda/job_files` =~ /$prid/ )
711     {
712         print <<"ENDHTML";
713         <br><br><br><br><br>
714         <center><font size='3' face='Georgia' color='330033'>
715         <b>ERROR!</b> You have already submitted this sequence to Pinda.<br><br>
716         If you want to resubmit it with different parameters,<br>
717         please submit it again to the
718         <a href='http://orion.mbg.duth.gr/Pinda'>starting page</a>.</center></font>
719 ENDHTML
720         exit;
721     }
722     my $lcr_filtering;
723     if ( defined $query->param('lcr_filtering')
724         && $query->param('lcr_filtering') == 1 )
725     {
726         $lcr_filtering = '0';
727     }
728     else
729     {
730         $lcr_filtering = '1';
731     }
732     my $masking;
733     if ( defined $query->param('masking')
734         && $query->param('masking') == 1 )
735     {
736         $masking = '0';
737     }
738     else
739     {
740         $masking = '1';
741     }
742
743     my $job =
744         "nice -n +19 ../Pinda_exec.pl $email " . q{"}
745       . $organism . q{"}
746       . " $prid $database $lcr_filtering $one $masking";
747     my $job_temp = "/var/www/Pinda/job_files/$prid";
748     open my $job_fh, '>', $job_temp;
```

```
749     print {$job_fh} <<"END";
750 #!/bin/sh
751 cd /var/www/Pinda/slurm_errors/
752 END
753     print {$job_fh} $job;
754     close $job_fh;
755     my $job_temp_pl = $job_temp . '.pl';
756     open my $job_pl_fh, '>', $job_temp_pl;
757     print {$job_pl_fh} <<"END";
758 #!/usr/bin/perl -w
759 use MIME::Lite;
760 system("chmod +x $job_temp");
761 open my \$job_fh, '<', "$job_temp";
762 my \$email;
763 my \$db;
764 my \$organism;
765 my \$prid;
766 while ( my \$line = <\$job_fh> )
767 {
768         if ( \$line =~ /Pinda_exec.pl (.+) ["]/ )
769         {
770                 \$email = \$1;
771         }
772         if ( \$line =~ /nt[.]fasta/ )
773         {
774                 \$db = "nt.fasta";
775         }
776         elsif ( \$line =~ /Swiss/ )
777         {
778                 \$db = "Swiss-prot";
779         }
780         else
781         {
782                 \$db = "UniProt";
783         }
784         if ( \$line =~ /["](.+)["]\\s(\\d+)\\s/ )
785         {
786                 \$organism = \$1;
787                 \$prid = \$2;
788         }
789 }
790 close \$job_fh;
791 system("$job_temp");
792 if ( \$? != 0 )
793 {
794         my \$error_data = "Pinda has run into an error while processing your job.";
795         \$error_data .= "\\nWe have been notified and are looking into it.";
796         send_email(\$email, \$error_data);
797         restore_job_count(\$db);
798         open my \$input_fh, '<', "/var/www/Pinda/tmps/blast/\$prid.tmp";
799         \$/ = undef;
800         my \$whole = <\$input_fh>;
801         close \$input_fh;
802         my \$program_output = `cd /var/www/Pinda/slurm_errors && ls -1t|tail -1`;
803         my \$output = `cd /var/www/Pinda/slurm_errors && cat < \$program_output`;
```

```
804         my \$error = "/tmp/error_Pinda";
805         open my \$error_fh, '>', \$error;
806         print {\$error_fh} "<b>Pinda has run into an error.</b>\\n\\n";
807         print {\$error_fh} "<b>Database:</b> \$db\\n\\n";
808         print {\$error_fh} "<b>Organism:</b> \$organism\\n\\n";
809         print {\$error_fh} "<b>Input:</b> \$whole\\n\\n";
810         print {\$error_fh} "<b>Pinda's Output:</b> \$output\\n\\n";
811         print {\$error_fh} "<b>Email of Submitter:</b> \$email\\n\\n";
812         close \$error_fh;
813         system("mail Pinda -s 'Pinda Error' < /tmp/error_Pinda");
814         system("rm /tmp/error_Pinda");
815         system("rm /var/www/Pinda/slurm_errors/\$program_output");
816 }
817 system("rm $job_temp");
818 system("rm $job_temp_pl");
819
820
821 sub send_email
822 {
823     my \$msg = MIME::Lite->new(
824         Subject => "Pinda ERROR",
825         From    => "Pinda\\\@orion.mbg.duth.gr",
826         To      => "\$_[0]",
827         Type    => 'text/html',
828         Data    => \$_[1]
829     );
830     \$msg->send();
831     return 0;
832 }
833
834 sub restore_job_count
835 {
836         my \$job_counting = "/var/www/Pinda/running_jobs";
837         my \$protein_jobs;
838         my \$dna_jobs;
839         open my \$job_counting_fh, '<', \$job_counting;
840         local \$/ = "\\n";
841         while ( my \$line = <\$job_counting_fh> )
842         {
843                 if ( \$line =~ /Protein[:]\\s(\\d+)/ )
844                 {
845                         \$protein_jobs = \$1;
846                 }
847                 elsif ( \$line =~ /DNA[:]\\s(\\d+)/ )
848                 {
849                         \$dna_jobs = \$1;
850                 }
851         }
852         close \$job_counting_fh;
853         if ( \$_[0] =~ /nt[.]fasta/ )
854         {
855                 \$dna_jobs--;
856         }
857         else
858         {
```

```
859                 \$protein_jobs--;
860         }
861         open \$job_counting_fh, '>', \$job_counting;
862         say {\$job_counting_fh} "Protein: \$protein_jobs";
863         say {\$job_counting_fh} "DNA: \$dna_jobs";
864         close \$job_counting_fh;
865         return 0;
866 }
867
868 END
869     close $job_pl_fh;
870     system
871 "cd /var/www/Pinda/slurm_errors/ && sbatch --mail-type=FAIL $job_temp_pl > /dev/null";
872
873     my $job_counting = '../running_jobs';
874     my $protein_jobs;
875     my $dna_jobs;
876     open my $job_counting_fh, '<', $job_counting;
877     local $/ = "\n";
878     while ( my $line = <$job_counting_fh> )
879     {
880         if ( $line =~ /Protein[:] (\d+)/ )
881         {
882             $protein_jobs = $1;
883         }
884         elsif ( $line =~ /DNA[:] (\d+)/ )
885         {
886             $dna_jobs = $1;
887         }
888     }
889     close $job_counting_fh;
890     if ( $database =~ /nt/ )
891     {
892         $dna_jobs++;
893     }
894     else
895     {
896         $protein_jobs++;
897     }
898     open $job_counting_fh, '>', $job_counting;
899     print {$job_counting_fh} "Protein: $protein_jobs\n";
900     print {$job_counting_fh} "DNA: $dna_jobs\n";
901     close $job_counting_fh;
902     my $rank = $protein_jobs + $dna_jobs;
903
904     my ( $pr_time, $dn_time );
905     my $job_average = '/var/www/Pinda/job_times';
906     open my $job_average_fh, '<', $job_average;
907     local $/ = "\n";
908     while ( my $line = <$job_average_fh> )
909     {
910         if ( $line =~ /Protein Jobs[:] \d+ Average Time[:] (\d+)/ )
911         {
912             $pr_time = $1;
913         }
```

```
914         elsif ( $line =~ /DNA Jobs[:] \d+ Average Time[:] (\d+)/ )
915         {
916             $dn_time = $1;
917         }
918     }
919     close $job_average_fh;
920     my $estimated_time;
921     $estimated_time =
922       ( ( $protein_jobs * $pr_time ) + ( $dna_jobs * $dn_time ) ) * 2;
923
924     print <<"ENDHTML";
925     <p style='width: 470px; text-align:center;margin-bottom:1px;margin-top:1px'>
926     <font size='3' face='Georgia' color='330033'>
927     <center><br><br><br>
928     <b>Your job has entered the queue with a rank of
929     <font size=5>#$rank</font>.
930     <br>Results will be sent to you via email.</b></p></center>
931     </p></font><font color='black'>
932 ENDHTML
933
934     if ( defined $note && $note == 1 )
935     {
936         print <<"ENDHTML";
937                 <br><center><font size='2' face='Georgia' color='330033'>
938                 <b>Note:</b> This Taxonomy ID does not belong to a species.
939                 <br>We hope you know what you're doing...</font></center>
940                 <font color='black'>
941 ENDHTML
942     }
943     elsif ( defined $note && $note == 2 )
944     {
945         print <<"ENDHTML";
946                 <br><center><font size='2' face='Georgia' color='330033'>
947                 <b>Note:</b> The name that you entered does not correspond to one exact species.
948                 <br>We hope you know what you're doing...</font></center>
949                 <font color='black'>
950 ENDHTML
951     }
952
953     job_timer($estimated_time);
954     print <<"ENDHTML";
955     <br><br><br><center><font size='2'>
956     [<a href="http://orion.mbg.duth.gr/Pinda">Return to submission form</a>]
957     <br><br><br>
958     <fieldset class="fieldset-auto-width"; style="background-color: #FFFFCC; width:300px"><legend>
959     Please, do <b>NOT</b> use your browser's back button to resubmit this
960     sequence with other parameters.
961     <br><br>Rather, submit it again via the <a href="http://orion.mbg.duth.gr/Pinda">
962     starting page</a>.</font>
963     </legend></fieldset>
964     <br><br><br></center>
965 ENDHTML
966 }
967
968 sub job_timer
```

```
969 {
970     my ( $hours, $minutes );
971     if ( $_[0] > 3600 )
972     {
973         $hours = $_[0] / 3600;
974         $_[0] %= 3600;
975     }
976     if ( $_[0] > 60 )
977     {
978         $minutes = $_[0] / 60;
979         $_[0] %= 60;
980     }
981     print
982 '<br><br><font size="2" face="Courier New"><center>Estimated time until job completion: about ';
983     if ( defined $hours )
984     {
985         if ( $hours >= 2 )
986         {
987             if ( defined $minutes && $minutes > 30 )
988             {
989                 $hours += 1;
990                 printf '<b>%.0f hours</b>.', $hours;
991             }
992             elsif ( defined $minutes && $minutes <= 30 )
993             {
994                 $hours += 0.5;
995                 printf '<b>%.1f hours</b>.', $hours;
996             }
997             else
998             {
999                 printf '<b>%.0f hours</b>.', $hours;
1000            }
1001        }
1002        else
1003        {
1004            if ( defined $minutes && $minutes > 30 )
1005            {
1006                printf '<b>2 hours</b>.';
1007            }
1008            elsif ( defined $minutes && $minutes <= 30 )
1009            {
1010                printf '<b>1.5 hours</b>.';
1011            }
1012            else
1013            {
1014                printf '<b>1 hour</b>.';
1015            }
1016        }
1017    }
1018    else
1019    {
1020        if ( $minutes > 30 )
1021        {
1022            printf '<b>1 hour</b>.';
1023        }
```

```
1024            else
1025            {
1026                printf '<b>0.5 hours</b>.';
1027            }
1028        }
1029        print '</font></center>';
1030        return 0;
1031    }
```

4.3 Pinda_exec.pl

Το Pinda_exec.pl αποτελεί το κυριότερο μέρος της ανάλυσης του προγράμματος. Συντονίζει τη λειτουργία τρίτων προγραμμάτων, από τη στοίχιση μέχρι το σχηματισμό του δενδρογράμματος και τη σύγκριση των οντολογιών.

```
#!/usr/bin/env perl

####################
#$VERSION = '0.02';#
####################

=head1 NAME

PINDA - Pipeline for INtraspecies Duplication Analysis

=head1 AUTHOR

Pinda has been developed by Dimitrios - Georgios Kontopoulos as his
final year project, under the supervision of Prof. Nicholas M. Glykos
at the Department of Molecular Biology and Genetics of Democritus
University of Thrace, Greece.

=head1 LICENSE

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as
published by the Free Software Foundation, either version 3 of the
License, or (at your option) any later version.

For more information, see http://www.gnu.org/licenses/.

=cut

use Bio::AlignIO;
use Bio::Perl;
use Bio::Root::IO;
use Bio::SearchIO;
use Bio::Search::Iteration::GenericIteration;
use Bio::Search::Result::BlastResult;
use Bio::SeqIO;
use Bio::Tools::Run::StandAloneBlastPlus;
use Bio::Tools::Run::StandAloneBlastPlus::BlastMethods;
use Bio::TreeIO;
use Bio::Tree::TreeFunctionsI;
use Bio::Tree::TreeI;
use File::stat;
use FreezeThaw qw(freeze thaw);
use List::MoreUtils qw(uniq);
use LWP::Simple qw(get);
use Math::Cephes qw(:all);
use MIME::Lite;
use Statistics::Basic qw(:all nofill);
use Sys::CPU;
```

```
49  use Time::localtime;
50
51  use feature qw(say);
52
53  use bignum;
54  use strict;
55  use warnings;
56
57  open STDERR, '>', '/tmp/err' or die $!;
58
59  ########################################
60  #On to the serious computation phase...#
61  ########################################
62  my $seq2counter      = 0;
63  my $hit_old          = 0;
64  my $iteration_number = 0;
65  my $resnum           = 0;
66  my $resnum2          = 0;
67  my $sequences_number = 0;
68  my $tdcounter        = 0;
69
70  my (
71      $acnumber,     $dbfetch,        $hit,        $i,
72      $line,         $match_line,     $number,     $one_gos,
73      $org1,         $out,            $out_fh,     $sequence,
74      $sequences_fh, $starting_point, $tdbg,       @accession,
75      @candidate,    @cand_sans,      @sequences2, @realign,
76      @reslines,     @seq,            @seq2,       %common,
77      %hsp,          %hsp_pos,        %hsp_seq,    %textcommon,
78      %textncommon,  %texts,          %texts2
79  );
80  my $start_timer = time;
81
82  my $email         = $ARGV[0];
83  my $organism      = $ARGV[1];
84  my $prid          = $ARGV[2];
85  my $db            = $ARGV[3];
86  my $lcr_filtering = $ARGV[4];
87  my $one           = $ARGV[5];
88  my $masking       = $ARGV[6];
89
90  my $slurm_queue = `squeue`;
91  my $slurm_id;
92  my $squeue_line;
93  {
94      local $/ = "\n";
95      open my $squeue_fh, '<', \$slurm_queue;
96      while ( $squeue_line = <$squeue_fh> )
97      {
98          if ( $squeue_line =~ /1 orion/ )
99          {
100             if ( $squeue_line =~ /\s+(\d+)\s+/ )
101             {
102                 $slurm_id = $1;
103                 last;
```

```
104             }
105         }
106     }
107     close $squeue_fh;
108 }
109
110 my $tmp = '../tmps/blast/' . $prid . '.tmp';
111 my $query_line;
112 if ( $one eq 'QUERY' )
113 {
114     open my $tmp_fh, '<', $tmp;
115     local $/ = undef;
116     while ( my $line = <$tmp_fh> )
117     {
118         if ( $line =~ />.*\n/ )
119         {
120             $query_line = $';
121             $query_line = uc $query_line;
122             last;
123         }
124         elsif ( $line !~ />/ )
125         {
126             $query_line = $line;
127             $query_line = uc $query_line;
128             last;
129         }
130     }
131     close $tmp_fh;
132 }
133 my $query_length = length $query_line;
134 my $SWISSPROT    = '/usr/local/databases/Swissprot/uniprot_sprot.fasta';
135 my $UNIPROT      = '/usr/local/databases/UniProt/UniProt.fasta';
136 my $NT           = '/usr/local/databases/nt/nt.fasta';
137 my $database     = $db;
138
139 if ( $db eq 'Swiss-Prot' )
140 {
141     $db = $SWISSPROT;
142 }
143 elsif ( $db eq 'UniProt' )
144 {
145     $db = $UNIPROT;
146 }
147 elsif ( $db eq 'nt' )
148 {
149     $db = $NT;
150 }
151
152 my $input = '/var/www/Pinda/tmps/blast/' . $prid . '.tmp';
153 open my $input_fh, '<', $input or die $!;
154 my $line_input;
155 {
156     local $/ = "\n";
157     while ( $line = <$input_fh> )
158     {
```

```
159         $line_input .= $line . '<br>';
160     }
161     if ( $line_input =~ /<br>$/ )
162     {
163         $line_input = $`;
164     }
165 }
166 close $input_fh;
167 chomp $line_input;
168 $line_input =~ s/\n//g;
169
170 my $email_data = <<"EMAIL_END";
171 <center><br>
172 <a href='http://orion.mbg.duth.gr/Pinda/cgi/Pinda.cgi'>
173 <img src='http://orion.mbg.duth.gr/Pinda/pindalogo.png'></a>
174 </center><br><br><br><hr />
175 <b>Input Sequence:<br><font face='Courier New'>$line_input
176 </font><br></b></b></b><hr />
177 <b>Organism:</b> $organism<br><hr />
178 <b>Database:</b> $database<hr /><br><center>
179 EMAIL_END
180
181 if ( $db =~ /nt[.]fasta/ )
182 {
183     $email_data .= <<"EMAIL_END";
184         <font size='2'>
185         <img src='http://orion.mbg.duth.gr/Pinda/caution.png' width=39
186         height=35><b><br>
187         The nt database is partially non-redundant.<br>
188         You are advised to pay extra attention when interpreting the results.
189         </b></font><br>
190 EMAIL_END
191 }
192
193 ##############################
194 #Get the number of cpu cores.#
195 ##############################
196 my $cpu_n = Sys::CPU::cpu_count();
197 ####################
198 #E-value threshold.#
199 ####################
200 if ( $db !~ /nt[.]fasta/ )
201 {
202     my $e_th = '0.00000000001';
203
204     $out = '../outs/psiblast/' . $prid . '.tmp';
205
206     if ( $lcr_filtering == 1 )
207     {
208         system
209 "psiblast -query $tmp -num_alignments 7000 -num_iterations 50 -evalue $e_th -db $db
210 -num_threads $cpu_n -out $out -seg yes";
211     }
212     else
213     {
```

```
214         system
215 "psiblast -query $tmp -num_alignments 7000 -num_iterations 50 -evalue $e_th -db $db
216 -num_threads $cpu_n -out $out -seg no";
217     }
218     ###########################################################################
219     #Shorten the PSI-BLAST output file. We only need the last iteration.#
220     ###########################################################################
221     open $out_fh, '<', $out or die $!;
222     while ( $line = <$out_fh> )
223     {
224         if ( $line =~ /Results from round (\d+)/ )
225         {
226             $resnum = $1;
227         }
228     }
229     close $out_fh or die $!;
230     open $out_fh, '<', $out or die $!;
231     while ( $line = <$out_fh> )
232     {
233         if ( $line =~ /Results from round $resnum/ )
234         {
235             while ( $line = <$out_fh> )
236             {
237                 $reslines[$resnum2] = $line;
238                 $resnum2++;
239             }
240         }
241     }
242     close $out_fh or die $!;
243     open $out_fh, '>', $out or die $!;
244     foreach my $line (@reslines)
245     {
246         print {$out_fh} $line;
247     }
248     close $out_fh or die $!;
249
250     my $blast = Bio::SearchIO->new(
251         -format => 'blast',
252         -file   => "$out"
253     );
254
255     my $query_found = 0;
256     ###################################
257     #Start parsing PSI-BLAST output.#
258     ###################################
259     while ( my $result = $blast->next_result )
260     {
261         while ( my $it = $result->next_iteration )
262         {
263             $i = 1000.0;
264             my $number = 0;
265             while (( $hit = $it->next_hit_new )
266                 || ( $hit = $it->next_hit_old ) )
267             {
268                 if ( ( $it->number ) > $iteration_number )
```

```
269                 {
270                     $iteration_number = $it->number;
271                 }
272                 if ( $hit->description =~ /(OS\=\w+)\s+(\w+)/ )
273                 {
274                     if ( $hit->accession =~ /tr[|](\w+)[|]/ )
275                     {
276                         my $ac = $1;
277                         local $/ = undef;
278                         open my $out_fh, '<', $out;
279                         while ( my $readline = <$out_fh> )
280                         {
281                             if ( $readline =~ />tr[|]$ac[|]/ )
282                             {
283                                 $readline = $';
284                                 if ( $readline =~ /OS\=(.+\s+.+)\s+/ )
285                                 {
286                                     $org1 = $1;
287                                     if ( $org1 =~ /\n/ )
288                                     {
289                                         $org1 = $` . $';
290                                     }
291                                 }
292                                 else
293                                 {
294                                     undef $org1;
295                                 }
296                             }
297                         }
298                         close $out_fh;
299                     }
300                     else
301                     {
302                         my $ac = $hit->accession;
303                         local $/ = undef;
304                         open my $out_fh, '<', $out;
305                         while ( my $readline = <$out_fh> )
306                         {
307                             if ( $readline =~ />sp[|]$ac[|]/ )
308                             {
309                                 $readline = $';
310                                 if ( $readline =~ /OS\=(.+\s+.+)\s+/ )
311                                 {
312                                     $org1 = $1;
313                                     if ( $org1 =~ /\n/ )
314                                     {
315                                         $org1 = $` . $';
316                                     }
317                                 }
318                                 else
319                                 {
320                                     undef $org1;
321                                 }
322                             }
323                         }
```

```
                    }
                    if ( defined $org1 && $org1 =~ /$organism/i )
                    {
                        if ( $hit->accession =~ /tr[|](\w+)[|]/ )
                        {
                            $accession[$number] = $1;
                            $acnumber = $accession[$number];
                        }
                        else
                        {
                            $accession[$number] = $hit->accession;
                            $acnumber = $accession[$number];
                        }
                        ###################################
                        #Get the full sequence of the hit.#
                        ###################################
                        if ( $hit_old ne $acnumber )
                        {
                            $dbfetch = get(
"http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=uniprotkb&id=$acnumber&format=fasta&style=raw"
                            );
                            if ( !( defined $dbfetch ) )
                            {
                                for ( 0 .. 3 )
                                {
                                    $dbfetch = get(
"http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=uniprotkb&id=$acnumber&format=fasta&style=raw"
                                    );
                                    if ( defined $dbfetch )
                                    {
                                        last;
                                    }
                                }
                            }
                            if ( defined $dbfetch && $dbfetch =~ /\n/ )
                            {
                                $match_line = $';
                                my $seq_length = length $match_line;
                                if (   $seq_length > $query_length / 5
                                    && $seq_length <= 3 * $query_length )
                                {
                                    $seq[$number] =
">$accession[$number] $org1\n$match_line\n\n";
                                    $number++;
                                }
                                elsif ( $seq_length > $query_length / 5 )
                                {
                                    my $hsp_old_length;
                                    while ( my $hsp = $hit->next_hsp )
                                    {
                                        if ( !( defined $hsp_old_length )
                                            || $hsp->length('hit') >
                                            $hsp_old_length )
                                        {
                                            $hsp{ $accession[$number] } =
```

```
379                                         ">$accession[$number] $org1\n"
380                                         . $hsp->hit_string . "\n\n";
381                                     $hsp_seq{ $accession[$number] } =
382                                       $hsp->hit_string;
383                                     $hsp_pos{ $accession[$number] } =
384                                         '('
385                                       . $hsp->start('hit') . ' -> '
386                                       . $hsp->end('hit') . ')';
387                                     $hsp_old_length =
388                                       $hsp->length('hit');
389                                 }
390                             }
391                             $seq[$number] = $hsp{ $accession[$number] };
392                             undef $hsp_old_length;
393                             delete $hsp{ $accession[$number] };
394                             $number++;
395                         }
396                     }
397                     if ( defined $query_line )
398                     {
399                         $query_line =~ s/\n//g;
400                         $query_line =~ s/\s//g;
401                         $match_line =~ s/\n//g;
402                         $match_line =~ s/\s//g;
403                         if ( ( uc $query_line ) eq ( uc $match_line ) )
404                         {
405                             $query_found = '1';
406                             $one         = $acnumber;
407                         }
408                     }
409                 }
410                 $hit_old = $acnumber;
411             }
412         }
413     }
414 }
415 }
416 if ( defined $query_line && $query_found == 0 )
417 {
418     my $number = @seq;
419     $seq[$number] = ">QUERY $organism\n" . $query_line . "\n\n";
420 }
421 }
422 else
423 {
424     $cpu_n--;
425     $out = '../outs/blast/' . $prid . '.tmp';
426     if ( $lcr_filtering == 1 )
427     {
428         system(
429 "blastn -query $tmp -db $db -evalue 0.00000000001 -num_threads $cpu_n -out $out -dust yes"
430           ) == 0
431           or die $?;
432     }
433     else
```

```
434     {
435         system(
436 "blastn -query $tmp -db $db -evalue 0.00000000001 -num_threads $cpu_n -out $out -dust no"
437           ) == 0
438           or die $?;
439     }
440 
441     my $blast = Bio::SearchIO->new(
442         -format => 'blast',
443         -file   => "$out"
444     );
445 
446     my $hit_old = 0;
447     my ( $accession, $org, @organism );
448     my $list        = 0;
449     my $number      = 0;
450     my $query_found = 0;
451     ##############################
452     #Start parsing BLAST output.#
453     ##############################
454     while ( my $result = $blast->next_result )
455     {
456         while ( my $hit = $result->next_hit )
457         {
458             if ( $hit->name =~ /ref[|](.+)[|]/ )
459             {
460                 if ( $1 =~ /[.]\d*/ )
461                 {
462                     $accession = $`;
463                 }
464                 $dbfetch = get(
465 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$accession&style=raw"
466                 );
467                 if ( !( defined $dbfetch ) )
468                 {
469                     for ( 0 .. 3 )
470                     {
471                         $dbfetch = get(
472 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$accession&style=raw"
473                         );
474                         if ( defined $dbfetch )
475                         {
476                             last;
477                         }
478                     }
479                 }
480 
481                 ######################################
482                 #Populate the organism dropdown list.#
483                 ######################################
484                 open my $db_fh, '<', \$dbfetch;
485                 local $/ = "\n";
486                 while ( my $db_line = <$db_fh> )
487                 {
488                     if ( $db_line =~ /ORGANISM\s+(.+\s+.+)/ )
```

```
489                     {
490                         $org = $1;
491                         $organism[$list] = $org;
492                         $list++;
493                         last;
494                     }
495                 }
496                 close $db_fh;
497             }
498             elsif ($hit->name =~ /gb[|](.+)[|]/
499                 || $hit->name =~ /dbj[|](.+)[|]/
500                 || $hit->name =~ /emb[|](.+)[|]/
501                 || $hit->name =~ /tpg[|](.+)[|]/ )
502             {
503                 if ( $1 =~ /[.]\d*/ )
504                 {
505                     $accession = $`;
506                 }
507                 $dbfetch = get(
508 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$accession&style=raw"
509                 );
510                 if ( !( defined $dbfetch ) )
511                 {
512                     for ( 0 .. 3 )
513                     {
514                         $dbfetch = get(
515 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$accession&style=raw"
516                         );
517                         if ( defined $dbfetch )
518                         {
519                             last;
520                         }
521                     }
522                 }
523
524                 ######################################
525                 #Populate the organism dropdown list.#
526                 ######################################
527                 open my $db_fh, '<', \$dbfetch;
528                 local $/ = "\n";
529                 while ( my $db_line = <$db_fh> )
530                 {
531                     if ( $db_line =~ /OS\s+(.+\s+.+)/ )
532                     {
533                         $org = $1;
534                         $organism[$list] = $org;
535                         $list++;
536                         last;
537                     }
538                 }
539             }
540             ####################
541             #Get the sequences.#
542             ####################
543             if ( defined $accession && $org =~ /$organism/i )
```

```
544             {
545                 if ( !( defined $hit_old ) || $hit_old ne $accession )
546                 {
547                     if ( $accession =~ /[_]/ )
548                     {
549                         $dbfetch = get(
550 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$accession&format=fasta&style=raw"
551                         );
552                         if ( !( defined $dbfetch ) )
553                         {
554                             for ( 0 .. 3 )
555                             {
556                                 $dbfetch = get(
557 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$accession&format=fasta&style=raw"
558                                 );
559                                 if ( defined $dbfetch )
560                                 {
561                                     last;
562                                 }
563                             }
564                         }
565                     }
566                     else
567                     {
568                         $dbfetch = get(
569 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$accession&format=fasta&style=raw"
570                         );
571                         if ( !( defined $dbfetch ) )
572                         {
573                             for ( 0 .. 3 )
574                             {
575                                 $dbfetch = get(
576 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$accession&format=fasta&style=raw"
577                                 );
578                                 if ( defined $dbfetch )
579                                 {
580                                     last;
581                                 }
582                             }
583                         }
584                     }
585                     if ( $dbfetch =~ /\n/ )
586                     {
587                         $match_line = $';
588                     }
589                     if ( defined $query_line )
590                     {
591                         $query_line =~ s/\n//g;
592                         $query_line =~ s/\s//g;
593                         $match_line =~ s/\n//g;
594                         $match_line =~ s/\s//g;
595                         if ( ( uc $query_line ) eq ( uc $match_line ) )
596                         {
597                             $query_found = '1';
598                             $one         = $accession;
```

```
599                             $accession   = 'III' . $accession . 'III';
600                         }
601                     }
602
603                     $match_line = uc $match_line;
604                     $match_line =~ tr/\n//d;
605                     my $seq_length = length $match_line;
606                     if (   $seq_length > $query_length / 5
607                         && $seq_length <= 3 * $query_length )
608                     {
609                         $seq[$number] = ">$accession $org\n$match_line\n\n";
610                         $number++;
611                     }
612                     elsif ( $seq_length > $query_length / 5 )
613                     {
614                         my $hsp_old_length;
615                         while ( my $hsp = $hit->next_hsp )
616                         {
617                             if ( !( defined $hsp_old_length )
618                                 || $hsp->length('hit') > $hsp_old_length )
619                             {
620                                 $hsp{$accession} = ">$accession $org\n"
621                                   . $hsp->hit_string . "\n\n";
622                                 $hsp_seq{$accession} = $hsp->hit_string;
623                                 $hsp_pos{$accession} = '('
624                                   . $hsp->start('hit') . ' -> '
625                                   . $hsp->end('hit') . ')';
626                                 $hsp_old_length = $hsp->length('hit');
627                             }
628                         }
629                         $seq[$number] = $hsp{$accession};
630                         $number++;
631                         undef $hsp_old_length;
632                         delete $hsp{$accession};
633                     }
634                 }
635             }
636             $hit_old = $accession;
637         }
638     }
639     if ( defined $query_line && $query_found == 0 )
640     {
641         my $number = @seq;
642         $seq[$number] = ">IIIQUERYIII $organism\n" . $query_line . "\n";
643     }
644
645 }
646
647 ############################
648 #Append sequences to file.#
649 ############################
650 @seq2 = uniq(@seq);
651 my $sequences = '/var/www/Pinda/seq/final_seq/' . $prid . '.tmp';
652 open $sequences_fh, '>', $sequences or die $!;
653
```

```
654 foreach my $sequence (@seq2)
655 {
656     if ( $sequence =~ /$organism/i )
657     {
658         if ( $sequence =~ /(\w+)[.]\d*/ )
659         {
660             $sequence = $` . $1 . $';
661         }
662         print {$sequences_fh} $sequence;
663 
664     }
665 }
666 close $sequences_fh or die $!;
667 if ( $db !~ /nt[.]fasta/ )
668 {
669     open $sequences_fh, '<', $sequences or die $!;
670     my $line2;
671     {
672         local $/ = "\n";
673         while ( $line2 = <$sequences_fh> )
674         {
675             ###################################
676             #Add stars to the input sequence.#
677             ###################################
678             $line2 =~ s/$one/***$one***/;
679             $sequences2[$seq2counter] = $line2;
680             $seq2counter++;
681         }
682     }
683     close $sequences_fh or die $!;
684 
685     open $sequences_fh, '>', $sequences or die $!;
686     foreach my $seq2counter (@sequences2)
687     {
688         print {$sequences_fh} $seq2counter;
689     }
690     close $sequences_fh or die $!;
691 }
692 
693 my $fnaln     = '/var/www/Pinda/results/final_alns/multalign/' . $prid . '.aln';
694 my $ph        = '/var/www/Pinda/results/trees/phs/' . $prid . '.ph';
695 my $phb       = '/var/www/Pinda/results/trees/phbs/' . $prid . '.phb';
696 my $drawntree = '/var/www/Pinda/results/trees/drawn/' . $prid . '.png';
697 my $zip       = '/var/www/Pinda/results/trees/zips/' . $prid . '.zip';
698 
699 open $sequences_fh, '<', $sequences or die $!;
700 my $line3;
701 {
702     local $/ = "\n\n";
703     while ( $line3 = <$sequences_fh> )
704     {
705         if ( $line3 !~ /^\s*$/ )
706         {
707             ###################################
708             #Count the resulting sequences.#
```

```
709             ################################
710             $sequences_number++;
711         }
712     }
713 }
714 close $sequences_fh or die $!;
715 
716 if ( $one ne 'QUERY' )
717 {
718     $email_data .= <<"EMAIL_END";
719         <br>This sequence has been identified as "$one" from "$organism".
720         <br>
721 EMAIL_END
722 }
723 
724 my $email2 = $email;
725 $email2 =~ s/([\@])/\\$1/;
726 if ( $sequences_number > 3 )
727 {
728     ############################################################
729     #Alignment, NJ tree plotting, bootstrapping and parsing.#
730     ############################################################
731     align( $sequences, $fnaln, $db, 1 );
732     if ( $db !~ /nt[.]fasta/ && $masking == 1 && $sequences_number <= 150 )
733     {
734         my $conf_val =
735           '/var/www/Pinda/results/final_alns/multalign/conf/' . $prid . '.tmp';
736         alignment_masking( $fnaln, $conf_val ) == 0 or die $?;
737     }
738     elsif ( $db !~ /nt[.]fasta/ && $masking == 1 && $sequences_number > 150 )
739     {
740         $email_data .= <<"EMAIL_END";
741                 <br><center>The resulting hits were over 150.
742                 <br>Due to computational limits, masking was <b>NOT</b> performed.</center>
743 EMAIL_END
744     }
745     my $fnaln2 = $fnaln . '.fasta';
746     if ( !( -e $fnaln2 ) )
747     {
748         system "cp $fnaln $fnaln2";
749     }
750     system("clustalw -INFILE=$fnaln2 -OUTFILE=$ph -tree") == 0 or die $?;
751     system(
752 "clustalw -INFILE=$fnaln2 -OUTFILE=$phb -bootstrap=1000 -bootlabels=node"
753       ) == 0
754       or die $?;
755     rename "../results/final_alns/multalign/$prid.aln.ph",
756       "../results/trees/phs/$prid.ph"
757       or die $!;
758     rename "../results/final_alns/multalign/$prid.aln.phb",
759       "../results/trees/phbs/$prid.phb"
760       or die $!;
761     tree_manipulation1($phb);
762     ############################################################
763     #Parsing the phb tree for distances and bootstrap values.#
```

```
764     ############################################################
765     system("../Pinda.R -parser $phb > /var/www/Pinda/parsing/$prid.tmp") == 0
766       or die $!;
767     system("../Pinda.R -lengths_1 $phb > /var/www/Pinda/parsing/$prid\_1.tmp")
768       == 0
769       or die $?;
770     system("../Pinda.R -lengths_2 $phb > /var/www/Pinda/parsing/$prid\_2.tmp")
771       == 0
772       or die $?;
773     tree_manipulation2($phb);
774     system("zip -j $zip $ph $phb") == 0 or die $?;
775     ################
776     #Draw the tree.#
777     ################
778     system("../Pinda.R $drawntree $phb") == 0
779       or die $?;
780     system("rm $ph $phb") == 0 or die $?;
781
782     parser( "../parsing/$prid.tmp", "../parsing/$prid\_1.tmp",
783         "../parsing/$prid\_2.tmp" );
784
785     ################################################
786     #Compute level of confidence for duplications.#
787     ################################################
788     compute_probability(@candidate);
789     my $realign_num = @realign;
790     if ( $realign_num >= 1 )
791     {
792         open $sequences_fh, '>', $sequences or die $!;
793         my $one_dbfetch;
794         if ( $db =~ /nt[.]fasta/ )
795         {
796             if ( $one =~ /^\D{2}\_/ )
797             {
798                 $one_dbfetch = get(
799 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$one&format=fasta&style=raw"
800                 );
801                 if ( !( defined $one_dbfetch ) )
802                 {
803                     for ( 0 .. 3 )
804                     {
805                         $one_dbfetch = get(
806 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$one&format=fasta&style=raw"
807                         );
808                         if ( defined $one_dbfetch )
809                         {
810                             last;
811                         }
812                     }
813                 }
814                 $one_dbfetch = uc $one_dbfetch;
815                 if ( $one_dbfetch =~ /\n/ )
816                 {
817                     $one_dbfetch = ">III" . $one . "III\n" . $' . "\n\n";
818                 }
```

```
819             }
820             elsif ( $one eq 'QUERY' )
821             {
822                 $one_dbfetch = ">IIIQUERYIII\n$query_line\n\n";
823             }
824             else
825             {
826                 $one_dbfetch = get(
827 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&format=fasta&style=raw"
828                 );
829                 if ( !( defined $one_dbfetch ) )
830                 {
831                     for ( 0 .. 3 )
832                     {
833                         $one_dbfetch = get(
834 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&format=fasta&style=raw"
835                         );
836                         if ( defined $dbfetch )
837                         {
838                             last;
839                         }
840                     }
841                 }
842                 if ( $one_dbfetch =~ /\n/ )
843                 {
844                     $one_dbfetch = '>III' . $one . "III\n" . $' . "\n\n";
845                 }
846             }
847         }
848         else
849         {
850             if ( $one eq 'QUERY' )
851             {
852                 $one_dbfetch = ">QUERY\n$query_line\n\n";
853             }
854             else
855             {
856                 $one_dbfetch = get("http://www.uniprot.org/uniprot/$one.fasta");
857             }
858         }
859         if ( $db !~ /nt[.]fasta/ )
860         {
861             $one_dbfetch =~ s/$one/***$one***/;
862         }
863         say {$sequences_fh} $one_dbfetch;
864         my $dbfetch;
865         foreach my $reseq (@realign)
866         {
867             if ( defined $hsp_seq{$reseq} && defined $hsp_pos{$reseq} )
868             {
869                 $dbfetch =
870                   ">$reseq $hsp_pos{$reseq}\n" . $hsp_seq{$reseq} . "\n\n";
871             }
872             else
873             {
```

```
874                 if ( $db =~ /nt[.]fasta/ )
875                 {
876                     if ( $reseq =~ /^\D{2}\_/ )
877                     {
878                         $dbfetch = get(
879 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$reseq&format=fasta&style=raw"
880                         );
881                         if ( !( defined $dbfetch ) )
882                         {
883                             for ( 0 .. 3 )
884                             {
885                                 $dbfetch = get(
886 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$reseq&format=fasta&style=raw"
887                                 );
888                                 if ( defined $dbfetch )
889                                 {
890                                     last;
891                                 }
892                             }
893                         }
894                         $dbfetch = uc $dbfetch;
895                         if ( $dbfetch =~ /\n/ )
896                         {
897                             $dbfetch = ">$reseq\n" . $' . "\n\n";
898                         }
899                     }
900                     else
901                     {
902                         $dbfetch = get(
903 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$reseq&format=fasta&style=raw"
904                         );
905                         if ( !( defined $dbfetch ) )
906                         {
907                             for ( 0 .. 3 )
908                             {
909                                 $dbfetch = get(
910 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$reseq&format=fasta&style=raw"
911                                 );
912                                 if ( defined $dbfetch )
913                                 {
914                                     last;
915                                 }
916                             }
917                         }
918                         if ( $dbfetch =~ /\n/ )
919                         {
920                             $dbfetch = ">$reseq\n" . $' . "\n\n";
921                         }
922                     }
923                 }
924                 else
925                 {
926                     $dbfetch =
927                       get("http://www.uniprot.org/uniprot/$reseq.fasta");
928                     if ( !( defined $dbfetch ) )
```

```
929                     {
930                         for ( 0 .. 3 )
931                         {
932                             $dbfetch = get(
933                                 "http://www.uniprot.org/uniprot/$reseq.fasta");
934                             if ( defined $dbfetch )
935                             {
936                                 last;
937                             }
938                         }
939                     }
940                 }
941             }
942             say {$sequences_fh} $dbfetch;
943         }
944         close $sequences_fh or die $!;
945         $fnaln .= '2';
946         align( $sequences, $fnaln, $db );
947     }
948     if ( $db !~ /nt[.]fasta/ )
949     {
950         ##############################
951         #Find common gene ontologies.#
952         ##############################
953         GOs_in_common();
954     }
955     alarm 0;
956     ########################
957     #Final results output.#
958     ########################
959     $email_data .= <<"EMAIL_END";
960                 <center><br><font size='3' face='Georgia' color='330033'>
961      <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln>
962      Alignment of all hits</a>
963 EMAIL_END
964     my $clu_file = "/var/www/Pinda/results/final_alns/multalign/$prid.aln.clu";
965     if ( -e $clu_file )
966     {
967         $email_data .= <<"EMAIL_END";
968         | <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln.clu>
969         Masked Alignment</a>
970 EMAIL_END
971     }
972     if ( $realign_num >= 1 )
973     {
974         $email_data .= <<"EMAIL_END";
975     | <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln2>
976       Alignment of top hits</a>
977 EMAIL_END
978     }
979
980     $email_data .= <<"EMAIL_END";
981         </font>
982         <br><br>
983         <table border='1'>
```

```
984         <tr bgcolor=FFFF66><th><center>Possible duplications of
985 EMAIL_END
986
987     if ( $one ne 'QUERY' )
988     {
989         if ( $db !~ /nt[.]fasta/ )
990         {
991             $email_data .= <<"EMAIL_END";
992             <a href=http://www.uniprot.org/uniprot/$one>
993             $one</a>.</center></th>
994             <th><center>Z Value</center></th>
995             <th><center>Level Of Confidence</center></th>
996             <th>GOs in common (out of $one_gos)</th>
997             <th>GOs not found in $one</th></tr>
998 EMAIL_END
999         }
1000        else
1001        {
1002            if ( $one =~ /^\D{2}\_/ )
1003            {
1004                $email_data .= <<"EMAIL_END";
1005                <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$one&style=raw>
1006                $one</a>.</center></th>
1007                <th><center>Z Value</center></th>
1008                                <th><center>Level Of Confidence</center></th></tr>
1009 EMAIL_END
1010            }
1011            else
1012            {
1013                $email_data .= <<"EMAIL_END";
1014            <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&style=raw>
1015            $one</a>.</center></th>
1016            <th><center>Z Value</center></th>
1017            <th><center>Level Of Confidence</center></th></tr>
1018 EMAIL_END
1019            }
1020        }
1021    }
1022    else
1023    {
1024        if ( $db !~ /nt[.]fasta/ )
1025        {
1026            $email_data .= <<"EMAIL_END";
1027            $one.</center></th>
1028            <th><center>Z Value</center></th>
1029            <th><center>Level Of Confidence</center></th>
1030            <th>GO terms</th></tr>
1031 EMAIL_END
1032        }
1033        else
1034        {
1035            if ( $one =~ /^\D{2}\_/ )
1036            {
1037                $email_data .= <<"EMAIL_END";
1038                $one.</center></th>
```

```
1039                 <th><center>Z Value</center></th>
1040                                 <th><center>Level Of Confidence</center></th></tr>
1041 EMAIL_END
1042             }
1043             else
1044             {
1045                 $email_data .= <<"EMAIL_END";
1046                 $one.</center></th>
1047                 <th><center>Z Value</center></th>
1048                                 <th><center>Level Of Confidence</center></th></tr>
1049 EMAIL_END
1050             }
1051         }
1052     }
1053     ###################################
1054     #Table of Z Values/Probabilities.#
1055     ###################################
1056     my $top_can;
1057     #################
1058     #Input sequence.#
1059     #################
1060     if ( $candidate[0] =~ /(\d?\d?\d?.?\d+e?-?\d*) \w+/ )
1061     {
1062         $top_can = $1;
1063     }
1064
1065     foreach my $can (@candidate)
1066     {
1067         ################################################
1068         #List every sequence, except for the input one.#
1069         ################################################
1070         if (   $can !~ /$starting_point/
1071             && $can =~
1072             /(-?\d?\d?\d?.?\d+e?-?\d*) (\d?\d?\d?.?\d+e?-?\d*) (\w+)/ )
1073         {
1074             my $z_temp    = $1;
1075             my $conf_temp = $2;
1076             my $uni_temp  = $3;
1077             ###################################
1078             #Color the html table alternately.#
1079             ###################################
1080             if ( $tdcounter == 1 )
1081             {
1082                 $tdbg      = 'F8FBFE';
1083                 $tdcounter = 0;
1084             }
1085             else
1086             {
1087                 $tdbg = 'EAF1FB';
1088                 $tdcounter++;
1089             }
1090             if ( $db !~ /nt[.]fasta/ )
1091             {
1092                 if ( defined $hsp_pos{$3} )
1093                 {
```

```
1094                    $email_data .= <<"EMAIL_END";
1095 <tr bgcolor=$tdbg><td><center><a href=http://www.uniprot.org/uniprot/$3>$3</a> $hsp_pos{$3}
1096 </center></td>
1097 EMAIL_END
1098                 }
1099                 else
1100                 {
1101                     $email_data .= <<"EMAIL_END";
1102 <tr bgcolor=$tdbg><td><center><a href=http://www.uniprot.org/uniprot/$3>$3</a>
1103 </center></td>
1104 EMAIL_END
1105                 }
1106             }
1107             else
1108             {
1109                 if ( $uni_temp =~ /^\D{2}\_/ )
1110                 {
1111                     if ( defined $hsp_pos{$uni_temp} )
1112                     {
1113                         $email_data .= <<"EMAIL_END";
1114 <tr bgcolor=$tdbg><td><center>
1115 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>$uni_temp</a>
1116 $hsp_pos{$uni_temp}
1117 </center></td>
1118 EMAIL_END
1119                     }
1120                     else
1121                     {
1122                         $email_data .= <<"EMAIL_END";
1123 <tr bgcolor=$tdbg><td><center>
1124 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>$uni_temp</a>
1125 </center></td>
1126 EMAIL_END
1127                     }
1128                 }
1129                 else
1130                 {
1131                     if ( defined $hsp_pos{$uni_temp} )
1132                     {
1133                         $email_data .= <<"EMAIL_END";
1134 <tr bgcolor=$tdbg><td><center>
1135 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>$uni_temp</a>
1136 $hsp_pos{$uni_temp}
1137 </center></td>
1138 EMAIL_END
1139                     }
1140                     else
1141                     {
1142                         $email_data .= <<"EMAIL_END";
1143 <tr bgcolor=$tdbg><td><center>
1144 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>$uni_temp</a>
1145 </center></td>
1146 EMAIL_END
1147                     }
1148                 }
```

```
1149            }
1150            $email_data .=
1151              sprintf
1152 '<td align=left><center>%5.2f</center></td><td align=left><center>%5.1f%%</center></td>',
1153              $z_temp, $conf_temp;
1154            if ( $one ne 'QUERY' )
1155            {
1156                if ( $db !~ /nt[.]fasta/ )
1157                {
1158                    if ( $common{$uni_temp} =~ /(\w+)\s(\w+)/ )
1159                    {
1160                        $email_data .= <<"EMAIL_END";
1161                                        <td title="$textcommon{$uni_temp}"><center>
1162                                        $1</center></td>
1163                                        <td title="$textncommon{$uni_temp}"><center>
1164                                        $2</center></td></tr>
1165 EMAIL_END
1166                    }
1167                }
1168            }
1169            else
1170            {
1171                if ( $db !~ /nt[.]fasta/ )
1172                {
1173                    if ( $texts{$uni_temp} =~ /(\w+)/ )
1174                    {
1175                        $email_data .= <<"EMAIL_END";
1176                                        <td title="$texts2{$uni_temp}"><center>
1177                                        $1</center></td></tr>
1178 EMAIL_END
1179                    }
1180                }
1181            }
1182        }
1183        elsif ($can !~ /$starting_point/
1184            && $can =~ /(-?\d?\d?\d?.?\d+e?-?\d*) (\w+)/ )
1185        {
1186            my $z_temp   = $1;
1187            my $uni_temp = $2;
1188            ####################################
1189            #Color the html table alternately.#
1190            ####################################
1191            if ( $tdcounter == 1 )
1192            {
1193                $tdbg      = 'F8FBFE';
1194                $tdcounter = 0;
1195            }
1196            else
1197            {
1198                $tdbg = 'EAF1FB';
1199                $tdcounter++;
1200            }
1201            if ( $db !~ /nt[.]fasta/ )
1202            {
1203                if ( defined $hsp_pos{$2} )
```

```
1204                {
1205                    $email_data .= <<"EMAIL_END";
1206 <tr bgcolor=$tdbg><td><center>
1207 <a href=http://www.uniprot.org/uniprot/$2>$2</a> $hsp_pos{$2}
1208 </center></td>
1209 EMAIL_END
1210                }
1211                else
1212                {
1213                    $email_data .= <<"EMAIL_END";
1214 <tr bgcolor=$tdbg><td><center>
1215 <a href=http://www.uniprot.org/uniprot/$2>$2</a>
1216 </center></td>
1217 EMAIL_END
1218                }
1219            }
1220            else
1221            {
1222                if ( $uni_temp =~ /^\D{2}\_/ )
1223                {
1224                    if ( defined $hsp_pos{$uni_temp} )
1225                    {
1226                        $email_data .= <<"EMAIL_END";
1227 <tr bgcolor=$tdbg><td><center>
1228 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>$uni_temp</a>
1229 $hsp_pos{$uni_temp}
1230 </center></td>
1231 EMAIL_END
1232                    }
1233                    else
1234                    {
1235                        $email_data .= <<"EMAIL_END";
1236 <tr bgcolor=$tdbg><td><center>
1237 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>$uni_temp</a>
1238 </center></td>
1239 EMAIL_END
1240                    }
1241                }
1242                else
1243                {
1244                    if ( defined $hsp_pos{$uni_temp} )
1245                    {
1246                        $email_data .= <<"EMAIL_END";
1247 <tr bgcolor=$tdbg><td><center>
1248 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>$uni_temp</a>
1249 $hsp_pos{$uni_temp}
1250 </center></td>
1251 EMAIL_END
1252                    }
1253                    else
1254                    {
1255                        $email_data .= <<"EMAIL_END";
1256 <tr bgcolor=$tdbg><td><center>
1257 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>$uni_temp</a>
1258 </center></td>
```

```
1259  EMAIL_END
1260                      }
1261                  }
1262              }
1263              $email_data .=
1264                sprintf '<td align=left><center>%5.2f</center></td><td></td>',
1265                $z_temp;
1266              if ( $one ne 'QUERY' )
1267              {
1268                  if ( $db !~ /nt[.]fasta/ )
1269                  {
1270                      if ( $common{$uni_temp} =~ /(\w+)\s(\w+)/ )
1271                      {
1272                          $email_data .= <<"EMAIL_END";
1273                                                  <td title="$textcommon{$uni_temp}"><center>
1274                                                  $1</center></td>
1275                                                  <td title="$textncommon{$uni_temp}"><center>
1276                                                  $2</center></td></tr>
1277  EMAIL_END
1278                      }
1279                  }
1280              }
1281              else
1282              {
1283                  if ( $db !~ /nt[.]fasta/ )
1284                  {
1285                      if ( $texts{$uni_temp} =~ /(\w+)/ )
1286                      {
1287                          $email_data .= <<"EMAIL_END";
1288                                                  <td title="$texts2{$uni_temp}"><center>
1289                                                  $1</center></td></tr>
1290  EMAIL_END
1291                      }
1292                  }
1293              }
1294          }
1295      }
1296
1297      ##############
1298      #Drawn tree.#
1299      ##############
1300      $email_data .= <<"EMAIL_END";
1301          </table><img src='http://orion.mbg.duth.gr/Pinda/results/trees/drawn/$prid.png'><br>
1302          <a href=http://orion.mbg.duth.gr/Pinda/results/trees/zips/$prid.zip>
1303          NJ trees produced in .ph/.phb format
1304          </a>
1305  EMAIL_END
1306
1307  }
1308  else
1309  {
1310      if ( $sequences_number == 3 )
1311      {
1312          ##########################################
1313          #Alignment, NJ-tree plotting and parsing.#
```

```
1314          ########################################
1315          align( $sequences, $fnaln, $db, 1 );
1316          if ( $db !~ /nt[.]fasta/ && $masking == 1 )
1317          {
1318              my $conf_val = '/var/www/Pinda/results/final_alns/multalign/conf/'
1319                . $prid . '.tmp';
1320              alignment_masking( $fnaln, $conf_val ) == 0 or die $?;
1321          }
1322          my $fnaln2 = $fnaln . '.fasta';
1323          if ( !( -e $fnaln2 ) )
1324          {
1325              system "cp $fnaln $fnaln2";
1326          }
1327          system("clustalw -INFILE=$fnaln2 -OUTFILE=$ph -tree") == 0 or die $?;
1328          rename "../results/final_alns/multalign/$prid.aln.ph",
1329            "../results/trees/phs/$prid.ph"
1330            or die $!;
1331          tree_manipulation1($ph);
1332          ####################################
1333          #Parsing the ph tree for distances.#
1334          ####################################
1335          system("../Pinda.R -parser $ph > /var/www/Pinda/parsing/$prid.tmp") == 0
1336            or die $?;
1337          system(
1338              "../Pinda.R -lengths_1 $ph > /var/www/Pinda/parsing/$prid\_1.tmp")
1339            == 0
1340            or die $?;
1341          system(
1342              "../Pinda.R -lengths_2 $ph > /var/www/Pinda/parsing/$prid\_2.tmp")
1343            == 0
1344            or die $?;
1345
1346          tree_manipulation2($ph);
1347
1348          system("zip -j $zip $ph") == 0 or die $?;
1349          ################
1350          #Draw the tree.#
1351          ################
1352          system("../Pinda.R $drawntree $ph") == 0
1353            or die $?;
1354          system("rm $ph") == 0 or die $?;
1355
1356          $email_data .= <<"EMAIL_END";
1357                          <font size='3' face='Georgia' color='330033'><br><br>
1358              <center>
1359              The number of sequences for this particular organism is three.
1360              <br><b>The dendrogram cannot be bootstrapped.</b></font>
1361              </center>
1362  EMAIL_END
1363
1364          parser( "../parsing/$prid.tmp", "../parsing/$prid\_1.tmp",
1365              "../parsing/$prid\_2.tmp" );
1366          compute_probability(@candidate);
1367          my $realign_num = @realign;
1368          if ( $realign_num >= 1 )
```

```
1369        {
1370            open $sequences_fh, '>', $sequences or die $!;
1371            my $one_dbfetch;
1372            if ( $db =~ /nt[.]fasta/ )
1373            {
1374                if ( $one =~ /^\D{2}\_/ )
1375                {
1376                    $one_dbfetch = get(
1377 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$one&format=fasta&style=raw"
1378                    );
1379                    if ( !( defined $one_dbfetch ) )
1380                    {
1381                        for ( 0 .. 3 )
1382                        {
1383                            $one_dbfetch = get(
1384 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$one&format=fasta&style=raw"
1385                            );
1386                            if ( defined $one_dbfetch )
1387                            {
1388                                last;
1389                            }
1390                        }
1391                    }
1392                }
1393                elsif ( $one eq 'QUERY' )
1394                {
1395                    $one_dbfetch = ">IIIQUERYIII\n$query_line\n\n";
1396                }
1397                else
1398                {
1399                    $one_dbfetch = get(
1400 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&format=fasta&style=raw"
1401                    );
1402                    if ( !( defined $one_dbfetch ) )
1403                    {
1404                        for ( 0 .. 3 )
1405                        {
1406                            $one_dbfetch = get(
1407 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&format=fasta&style=raw"
1408                            );
1409                            if ( defined $one_dbfetch )
1410                            {
1411                                last;
1412                            }
1413                        }
1414                    }
1415                    if ( !( defined $one_dbfetch ) )
1416                    {
1417                        for ( 0 .. 3 )
1418                        {
1419                            $one_dbfetch = get(
1420 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&format=fasta&style=raw"
1421                            );
1422                            if ( defined $one_dbfetch )
1423                            {
```

```
1424                                last;
1425                            }
1426                        }
1427                    }
1428                }
1429                if ( $one_dbfetch =~ /\n/ )
1430                {
1431                    $one_dbfetch = '>III' . $one . "III\n" . $' . "\n\n";
1432                }
1433            }
1434            else
1435            {
1436                if ( $one eq 'QUERY' )
1437                {
1438                    $one_dbfetch = ">QUERY\n$query_line\n\n";
1439                }
1440                else
1441                {
1442                    $one_dbfetch =
1443                      get("http://www.uniprot.org/uniprot/$one.fasta");
1444                }
1445            }
1446            $one_dbfetch = uc $one_dbfetch;
1447            if ( $db !~ /nt[.]fasta/ )
1448            {
1449                $one_dbfetch =~ s/$one/***$one***/;
1450            }
1451            say {$sequences_fh} $one_dbfetch;
1452            my $dbfetch;
1453            foreach my $reseq (@realign)
1454            {
1455                if ( defined $hsp_seq{$reseq} && defined $hsp_pos{$reseq} )
1456                {
1457                    $dbfetch =
1458                      ">$reseq $hsp_pos{$reseq}\n" . $hsp_seq{$reseq} . "\n\n";
1459                }
1460                else
1461                {
1462                    if ( $db =~ /nt[.]fasta/ )
1463                    {
1464                        if ( $reseq =~ /^\D{2}\_/ )
1465                        {
1466                            $dbfetch = get(
1467 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$reseq&format=fasta&style=raw"
1468                            );
1469                            if ( !( defined $dbfetch ) )
1470                            {
1471                                for ( 0 .. 3 )
1472                                {
1473                                    $dbfetch = get(
1474 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$reseq&format=fasta&style=raw"
1475                                    );
1476                                    if ( defined $dbfetch )
1477                                    {
1478                                        last;
```

```
1479                                    }
1480                                }
1481                            }
1482                            if ( $dbfetch =~ /\n/ )
1483                            {
1484                                $dbfetch = ">$reseq\n" . $' . "\n\n";
1485                            }
1486                        }
1487                        else
1488                        {
1489                            $dbfetch = get(
1490 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$reseq&format=fasta&style=raw"
1491                            );
1492                            if ( !( defined $dbfetch ) )
1493                            {
1494                                for ( 0 .. 3 )
1495                                {
1496                                    $dbfetch = get(
1497 "http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$reseq&format=fasta&style=raw"
1498                                    );
1499                                    if ( defined $dbfetch )
1500                                    {
1501                                        last;
1502                                    }
1503                                }
1504                            }
1505                            if ( $dbfetch =~ /\n/ )
1506                            {
1507                                $dbfetch = ">$reseq\n" . $' . "\n\n";
1508                            }
1509                        }
1510                    }
1511                    else
1512                    {
1513                        $dbfetch =
1514                          get("http://www.uniprot.org/uniprot/$reseq.fasta");
1515                    }
1516                    $dbfetch = uc $dbfetch;
1517                }
1518                say {$sequences_fh} $dbfetch;
1519            }
1520            close $sequences_fh or die $!;
1521            $fnaln .= "2";
1522            align( $sequences, $fnaln, $db );
1523        }
1524        if ( $db !~ /nt[.]fasta/ )
1525        {
1526            ##############################
1527            #Find common gene ontologies.#
1528            ##############################
1529            GOs_in_common();
1530        }
1531
1532        $email_data .= <<"EMAIL_END";
1533                        <center><br><font size='3' face='Georgia' color='330033'>
```

```
1534                <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln>
1535                Alignment of all hits</a>
1536 EMAIL_END
1537         my $clu_file =
1538           "/var/www/Pinda/results/final_alns/multalign/$prid.aln.clu";
1539         if ( -e $clu_file )
1540         {
1541             $email_data .= <<"EMAIL_END";
1542                 | <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln.clu>
1543                 Masked Alignment</a>
1544 EMAIL_END
1545         }
1546         if ( $realign_num >= 1 )
1547         {
1548             $email_data .= <<"EMAIL_END";
1549        | <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln2>
1550        Alignment of top hits</a>
1551 EMAIL_END
1552         }
1553
1554         $email_data .= <<"EMAIL_END";
1555             </font>
1556             <br><br>
1557             <table border='1'>
1558             <tr bgcolor=FFFF66><th><center>Possible duplications of
1559 EMAIL_END
1560
1561         if ( $one ne 'QUERY' )
1562         {
1563             if ( $db !~ /nt[.]fasta/ )
1564             {
1565                 $email_data .= <<"EMAIL_END";
1566             <a href=http://www.uniprot.org/uniprot/$one>
1567             $one</a>.</center></th>
1568             <th><center>Z Value</center></th>
1569             <th><center>Level Of Confidence</center></th>
1570             <th>GOs in common (out of $one_gos)</th>
1571             <th>GOs not found in $one</th></tr>
1572 EMAIL_END
1573             }
1574             else
1575             {
1576                 if ( $one =~ /^\D{2}\_/ )
1577                 {
1578                     $email_data .= <<"EMAIL_END";
1579                 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$one&style=raw>
1580                 $one</a>.</center></th>
1581                 <th><center>Z Value</center></th>
1582                                 <th><center>Level Of Confidence</center></th></tr>
1583 EMAIL_END
1584                 }
1585                 else
1586                 {
1587                     $email_data .= <<"EMAIL_END";
1588             <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$one&style=raw>
```

```
1589            $one</a>.</center></th>
1590            <th><center>Z Value</center></th>
1591            <th><center>Level Of Confidence</center></th></tr>
1592  EMAIL_END
1593                }
1594            }
1595        }
1596        else
1597        {
1598            if ( $db !~ /nt[.]fasta/ )
1599            {
1600                $email_data .= <<"EMAIL_END";
1601            $one.</center></th>
1602            <th><center>Z Value</center></th>
1603            <th><center>Level Of Confidence</center></th>
1604            <th>GO terms</th></tr>
1605  EMAIL_END
1606            }
1607            else
1608            {
1609                if ( $one =~ /^\D{2}\_/ )
1610                {
1611                    $email_data .= <<"EMAIL_END";
1612                $one.</center></th>
1613                <th><center>Z Value</center></th>
1614                                <th><center>Level Of Confidence</center></th></tr>
1615  EMAIL_END
1616                }
1617                else
1618                {
1619                    $email_data .= <<"EMAIL_END";
1620                $one.</center></th>
1621                <th><center>Z Value</center></th>
1622                                <th><center>Level Of Confidence</center></th></tr>
1623  EMAIL_END
1624                }
1625            }
1626        }
1627
1628        my $top_can;
1629        ##################
1630        #Input sequence.#
1631        ##################
1632        if ( $candidate[0] =~ /(\d?\d?\d?.?\d+e?-?\d*) \w+/ )
1633        {
1634            $top_can = $1;
1635        }
1636
1637        foreach my $can (@candidate)
1638        {
1639            if (   $can !~ /$starting_point/
1640                && $can =~
1641                /(-?\d?\d?\d?.?\d+e?-?\d*) (\d?\d?\d?.?\d+e?-?\d*) (\w+)/ )
1642            {
1643                my $z_temp    = $1;
```

```
1644                    my $conf_temp = $2;
1645                    my $uni_temp  = $3;
1646                    ###################################
1647                    #Color the html table alternately.#
1648                    ###################################
1649                    if ( $tdcounter == 1 )
1650                    {
1651                        $tdbg      = 'F8FBFE';
1652                        $tdcounter = 0;
1653                    }
1654                    else
1655                    {
1656                        $tdbg = 'EAF1FB';
1657                        $tdcounter++;
1658                    }
1659                    if ( $db !~ /nt[.]fasta/ )
1660                    {
1661                        if ( defined $hsp_pos{$3} )
1662                        {
1663                            $email_data .= <<"EMAIL_END";
1664    <tr bgcolor=$tdbg><td><center>
1665    <a href=http://www.uniprot.org/uniprot/$3>$3</a> $hsp_pos{$3}
1666    </center></td>
1667    EMAIL_END
1668                        }
1669                        else
1670                        {
1671                            $email_data .= <<"EMAIL_END";
1672    <tr bgcolor=$tdbg><td><center>
1673    <a href=http://www.uniprot.org/uniprot/$3>$3</a>
1674    </center></td>
1675    EMAIL_END
1676                        }
1677                    }
1678                    else
1679                    {
1680                        if ( $uni_temp =~ /^\D{2}\_/ )
1681                        {
1682                            if ( defined $hsp_pos{$uni_temp} )
1683                            {
1684                                $email_data .= <<"EMAIL_END";
1685    <tr bgcolor=$tdbg><td><center>
1686    <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>
1687    $uni_temp</a> $hsp_pos{$uni_temp}
1688    </center></td>
1689    EMAIL_END
1690                            }
1691                            else
1692                            {
1693                                $email_data .= <<"EMAIL_END";
1694    <tr bgcolor=$tdbg><td><center>
1695    <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>
1696    $uni_temp</a>
1697    </center></td>
1698    EMAIL_END
```

```
1699                        }
1700                    }
1701                    else
1702                    {
1703                        if ( defined $hsp_pos{$uni_temp} )
1704                        {
1705                            $email_data .= <<"EMAIL_END";
1706 <tr bgcolor=$tdbg><td><center>
1707 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>
1708 $uni_temp</a> $hsp_pos{$uni_temp}
1709 </center></td>
1710 EMAIL_END
1711                        }
1712                        else
1713                        {
1714                            $email_data .= <<"EMAIL_END";
1715 <tr bgcolor=$tdbg><td><center>
1716 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>
1717 $uni_temp</a>
1718 </center></td>
1719 EMAIL_END
1720                        }
1721                    }
1722                }
1723                $email_data .=
1724                  sprintf
1725 '<td align=left><center>%5.2f</center></td><td align=left><center>%5.1f%%</center></td>',
1726                  $z_temp, $conf_temp;
1727                if ( $one ne 'QUERY' )
1728                {
1729                    if ( $db !~ /nt[.]fasta/ )
1730                    {
1731                        if ( $common{$uni_temp} =~ /(\w+)\s(\w+)/ )
1732                        {
1733                            $email_data .= <<"EMAIL_END";
1734                                        <td title="$textcommon{$uni_temp}">
1735                                        <center>$1</center></td>
1736                                        <td title="$textncommon{$uni_temp}">
1737                                        <center>$2</center></td></tr>
1738 EMAIL_END
1739                        }
1740                    }
1741                }
1742                else
1743                {
1744                    if ( $db !~ /nt[.]fasta/ )
1745                    {
1746                        if ( $texts{$uni_temp} =~ /(\w+)/ )
1747                        {
1748                            $email_data .= <<"EMAIL_END";
1749                                        <td title="$texts2{$uni_temp}">
1750                                        <center>$1</center></td></tr>
1751 EMAIL_END
1752                        }
1753                    }
```

```
1754                 }
1755             }
1756             elsif ($can !~ /$starting_point/
1757                 && $can =~ /(-?\d?\d?\d?.?\d+e?-?\d*) (\w+)/ )
1758             {
1759                 my $z_temp   = $1;
1760                 my $uni_temp = $2;
1761                 ###################################
1762                 #Color the html table alternately.#
1763                 ###################################
1764                 if ( $tdcounter == 1 )
1765                 {
1766                     $tdbg      = 'F8FBFE';
1767                     $tdcounter = 0;
1768                 }
1769                 else
1770                 {
1771                     $tdbg = 'EAF1FB';
1772                     $tdcounter++;
1773                 }
1774                 if ( $db !~ /nt[.]fasta/ )
1775                 {
1776                     if ( defined $hsp_pos{$2} )
1777                     {
1778                         $email_data .= <<"EMAIL_END";
1779 <tr bgcolor=$tdbg><td><center>
1780 <a href=http://www.uniprot.org/uniprot/$2>$2</a> $hsp_pos{$2}
1781 </center></td>
1782 EMAIL_END
1783                     }
1784                     else
1785                     {
1786                         $email_data .= <<"EMAIL_END";
1787 <tr bgcolor=$tdbg><td><center>
1788 <a href=http://www.uniprot.org/uniprot/$2>$2</a>
1789 </center></td>
1790 EMAIL_END
1791                     }
1792                 }
1793                 else
1794                 {
1795                     if ( $uni_temp =~ /^\D{2}\_/ )
1796                     {
1797                         if ( defined $hsp_pos{$uni_temp} )
1798                         {
1799                             $email_data .= <<"EMAIL_END";
1800 <tr bgcolor=$tdbg><td><center>
1801 <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>
1802 $uni_temp</a> $hsp_pos{$uni_temp}
1803 </center></td>
1804 EMAIL_END
1805                         }
1806                         else
1807                         {
1808                             $email_data .= <<"EMAIL_END";
```

```
1809  <tr bgcolor=$tdbg><td><center>
1810  <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=refseqn&id=$uni_temp&style=raw>
1811  $uni_temp</a>
1812  </center></td>
1813  EMAIL_END
1814                          }
1815                      }
1816                      else
1817                      {
1818                          if ( defined $hsp_pos{$uni_temp} )
1819                          {
1820                              $email_data .= <<"EMAIL_END";
1821  <tr bgcolor=$tdbg><td><center>
1822  <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>
1823  $uni_temp</a> $hsp_pos{$uni_temp}
1824  </center></td>
1825  EMAIL_END
1826                          }
1827                          else
1828                          {
1829                              $email_data .= <<"EMAIL_END";
1830  <tr bgcolor=$tdbg><td><center>
1831  <a href=http://www.ebi.ac.uk/Tools/dbfetch/dbfetch?db=embl&id=$uni_temp&style=raw>
1832  $uni_temp</a>
1833  </center></td>
1834  EMAIL_END
1835                          }
1836                      }
1837                  }
1838                  $email_data .=
1839                    sprintf '<td align=left><center>%5.2f</center></td><td></td>',
1840                    $z_temp;
1841                  if ( $one ne 'QUERY' )
1842                  {
1843                      if ( $db !~ /nt[.]fasta/ )
1844                      {
1845                          if ( $common{$uni_temp} =~ /(\w+)\s(\w+)/ )
1846                          {
1847                              $email_data .= <<"EMAIL_END";
1848                                              <td title="$textcommon{$uni_temp}">
1849                                              <center>$1</center></td>
1850                                              <td title="$textncommon{$uni_temp}">
1851                                              <center>$2</center></td></tr>
1852  EMAIL_END
1853                          }
1854                      }
1855                  }
1856                  else
1857                  {
1858                      if ( $db !~ /nt[.]fasta/ )
1859                      {
1860                          if ( $texts{$uni_temp} =~ /(\w+)/ )
1861                          {
1862                              $email_data .= <<"EMAIL_END";
1863                                              <td title="$texts2{$uni_temp}"><center>$1
```

```
1864                                                    </center></td></tr>
1865 EMAIL_END
1866                         }
1867                     }
1868                 }
1869             }
1870         }
1871
1872         $email_data .= <<"EMAIL_END";
1873             </table><center>
1874             <img src='http://orion.mbg.duth.gr/Pinda/results/trees/drawn/$prid.png'>
1875             <br>
1876             <a href=http://orion.mbg.duth.gr/Pinda/results/trees/zips/$prid.zip>
1877             NJ trees produced in .ph/.phb format
1878             </a></center>
1879 EMAIL_END
1880
1881     }
1882     if ( $sequences_number == 2 )
1883     {
1884         #################
1885         #Alignment only.#
1886         #################
1887         align( $sequences, $fnaln, $db );
1888         $email_data .= <<"EMAIL_END";
1889             <center>
1890             <font size='3' face='Georgia' color='330033'><br><br>
1891             Only <b>two</b> similar sequences from <b>$organism</b>
1892             have been identified.
1893             <br><b>Phylogenetic analysis is meaningless therefore.</b>
1894             </font>
1895             </center>
1896 EMAIL_END
1897
1898         $email_data .= <<"EMAIL_END";
1899             <center><br><font size='3' face='Georgia' color='330033'><br>
1900                 <a href=http://orion.mbg.duth.gr/Pinda/results/final_alns/multalign/$prid.aln>
1901                 Alignment</a></font></center>
1902 EMAIL_END
1903     }
1904     if ( $sequences_number <= 1 )
1905     {
1906         alarm 0;
1907         ######################
1908         #Nothing to do here.#
1909         ######################
1910         if ( $db =~ /nt[.]fasta/ )
1911         {
1912             $email_data .= <<"EMAIL_END";
1913             <center>
1914             <font size='3' face='Georgia' color='330033'><br><br>
1915             No similar sequences from <b>$organism</b>
1916             have been identified.
1917             <br><b>Phylogenetic analysis is meaningless therefore.</b>
1918             </font>
```

```
            </center>
EMAIL_END
        }
        else
        {
            $email_data .= <<"EMAIL_END";
            <center>
            <font size='3' face='Georgia' color='330033'><br><br>
            No similar sequences from <b>$organism</b>
            have been identified.
            <br><b>Phylogenetic analysis is meaningless therefore.</b>
            </font>
            </center>
EMAIL_END
        }
    }
}
##############################################
#Calculate how much time it took for the job.#
##############################################
my $end_timer   = time;
my $run_time    = $end_timer - $start_timer;
my $job_average = '/var/www/Pinda/job_times';
my ( $pr_jobs, $pr_time, $dn_jobs, $dn_time );
open my $job_average_fh, '<', $job_average;
local $/ = "\n";
while ( my $line = <$job_average_fh> )
{

    if ( $line =~ /Protein Jobs[:] (\d+) Average Time[:] (\d+)/ )
    {
        $pr_jobs = $1;
        $pr_time = $2;
    }
    elsif ( $line =~ /DNA Jobs[:] (\d+) Average Time[:] (\d+)/ )
    {
        $dn_jobs = $1;
        $dn_time = $2;
    }
}
close $job_average_fh;

if ( $db =~ /nt[.]fasta/ )
{
    $dn_time = ( ( $dn_jobs * $dn_time ) + $run_time ) / ( $dn_jobs + 1 );
    $dn_jobs++;
}
else
{
    $pr_time = ( ( $pr_jobs * $pr_time ) + $run_time ) / ( $pr_jobs + 1 );
    $pr_jobs++;
}
open $job_average_fh, '>', $job_average;
say {$job_average_fh} "Protein Jobs: $pr_jobs Average Time: $pr_time";
say {$job_average_fh} "DNA Jobs: $dn_jobs Average Time: $dn_time";
```

```
1974  close $job_average_fh;
1975
1976  $email_data .= <<"ENDHTML";
1977  <br><br>
1978  <font size='1'>Temporary files from this job, including alignments, .ph/.phb
1979  trees and plotted trees, will be <b>DELETED</b> after ten days.</font>
1980  </center>
1981  </body>
1982  </html>
1983  ENDHTML
1984
1985  ##############
1986  #Send e-mail.#
1987  ##############
1988  send_email( $one, $email );
1989
1990  my $job_counting = '/var/www/Pinda/running_jobs';
1991  my $protein_jobs;
1992  my $dna_jobs;
1993  open my $job_counting_fh, '<', $job_counting;
1994  local $/ = "\n";
1995  while ( my $line = <$job_counting_fh> )
1996  {
1997
1998      if ( $line =~ /Protein[:] (\d+)/ )
1999      {
2000          $protein_jobs = $1;
2001      }
2002      elsif ( $line =~ /DNA[:] (\d+)/ )
2003      {
2004          $dna_jobs = $1;
2005      }
2006  }
2007  close $job_counting_fh;
2008  if ( $db =~ /nt[.]fasta/ )
2009  {
2010      $dna_jobs--;
2011  }
2012  else
2013  {
2014      $protein_jobs--;
2015  }
2016  open $job_counting_fh, '>', $job_counting;
2017  say {$job_counting_fh} "Protein: $protein_jobs";
2018  say {$job_counting_fh} "DNA: $dna_jobs";
2019  close $job_counting_fh;
2020  system "rm /var/www/Pinda/slurm_errors/slurm-$slurm_id.out";
2021
2022  ############################
2023  #Multiple Sequence Alignment#
2024  ############################
2025  sub align
2026  {
2027      my $out = $_[1] . '.fasta';
2028      my $db  = $_[2];
```

```
2029        my $again;
2030        if ( defined $_[3] )
2031        {
2032            $again = $_[3];
2033        }
2034        if ( $db !~ /nt[.]fasta/ && defined $again && $masking == 1 )
2035        {
2036            system
2037    "/usr/local/bin/clustalo -i $_[0] -o $_[1] --outfmt=clu --threads=4 -v --force";
2038            system
2039    "/usr/local/bin/clustalo -i $_[0] -o $out --outfmt=fasta --threads=4 -v --force";
2040        }
2041        elsif (( $db !~ /nt[.]fasta/ && !( defined $again ) )
2042            || ( $db !~ /nt[.]fasta/ && $masking == 0 ) )
2043        {
2044            system
2045    "/usr/local/bin/clustalo -i $_[0] -o $_[1] --outfmt=clu --threads=4 -v --force";
2046        }
2047        else
2048        {
2049            system "/usr/local/bin/kalign -i $_[0] -o $_[1] -f clu -q";
2050            my @sequences2  = ();
2051            my $seq2counter = 0;
2052            open my $fnaln_fh, '<', $fnaln or die $!;
2053            my $line2;
2054            {
2055                local $/ = "\n";
2056                while ( $line2 = <$fnaln_fh> )
2057                {
2058                    ###################################
2059                    #Add stars to the input sequence.#
2060                    ###################################
2061                    my $one_temp = 'III' . $one . 'III';
2062                    $line2 =~ s/$one_temp/***$one***/;
2063
2064                    $sequences2[$seq2counter] = $line2;
2065                    $seq2counter++;
2066                }
2067            }
2068            close $fnaln_fh or die $!;
2069
2070            open $fnaln_fh, '>', $fnaln or die $!;
2071            foreach my $seq2counter (@sequences2)
2072            {
2073                print {$fnaln_fh} $seq2counter;
2074            }
2075            close $fnaln_fh or die $!;
2076
2077            open $fnaln_fh, '<', $fnaln;
2078            local $/ = undef;
2079            my $line = q{};
2080            while ( $line = <$fnaln_fh> )
2081            {
2082                if ( $line =~ /in ClustalW format/ )
2083                {
```

```
2084                 $line = $';
2085                 last;
2086             }
2087         }
2088         close $fnaln_fh;
2089         if ( defined $line )
2090         {
2091             open $fnaln_fh, '>', $fnaln;
2092             print {$fnaln_fh} 'CLUSTAL W (1.83) multiple sequence alignment';
2093             print {$fnaln_fh} $line;
2094             close $fnaln_fh;
2095         }
2096     }
2097     return 0;
2098 }
2099
2100 ###############################################################
2101 #Mask alignments using confidence values generated by ZORRO#
2102 ###############################################################
2103 sub alignment_masking
2104 {
2105     my $input  = $_[0] . '.fasta';
2106     my $output = $_[1];
2107     system "/usr/local/bin/zorro $input > $output";
2108
2109     my $counter  = 0;
2110     my $counter2 = 0;
2111     my @low_conf;
2112
2113     open my $output_fh, '<', $output or die $!;
2114     local $/ = "\n";
2115     while ( my $line = <$output_fh> )
2116     {
2117         if ( $line =~ /(\d+[.]\d+)/ )
2118         {
2119             if ( $1 <= 0.4 )
2120             {
2121                 $low_conf[$counter2] = $counter + 1;
2122                 $counter2++;
2123             }
2124         }
2125         $counter++;
2126     }
2127     close $output_fh;
2128
2129     if ( $counter2 >= 1 && ( $counter2 / $counter ) < 0.75 )
2130     {
2131         open my $input_fh, '<', $input or die $!;
2132         local $/ = undef;
2133         my $newline = <$input_fh>;
2134         close $input_fh;
2135         my $seqc = 0;
2136         my $resc = 0;
2137         my @sequence;
2138
```

```
2139            while ( $newline =~ />[*]?[*]?[*]?\w+/ )
2140            {
2141                if ( $newline =~ /(>.+\s)/ )
2142                {
2143                    $sequence[$seqc][0] = $1;
2144                    if ( $' =~ />/ || $' =~ /\s$/ )
2145                    {
2146                        $resc = 0;
2147                        my $temp = $`;
2148                        $newline = '>' . $';
2149                        $temp =~ s/\n//g;
2150                        foreach my $residue ( split //, $temp )
2151                        {
2152                            $resc++;
2153                            $sequence[$seqc][$resc] = $residue;
2154                        }
2155                    }
2156                    else
2157                    {
2158                        last;
2159                    }
2160                    $seqc++;
2161                }
2162            }
2163
2164            foreach my $position (@low_conf)
2165            {
2166                for my $seqtmp ( 0 .. $seqc - 1 )
2167                {
2168                    undef $sequence[$seqtmp][$position];
2169                }
2170            }
2171
2172            open $input_fh, '>', $input or die $!;
2173            for my $seqtmp ( 0 .. $seqc - 1 )
2174            {
2175                print {$input_fh} $sequence[$seqtmp][0];
2176                for ( 1 .. $resc )
2177                {
2178                    if ( defined $sequence[$seqtmp][$_] )
2179                    {
2180                        print {$input_fh} $sequence[$seqtmp][$_];
2181                    }
2182                }
2183                say {$input_fh} q{};
2184            }
2185            close $input_fh;
2186            my $clu_file = $_[0] . '.clu';
2187            system "sreformat clustal $input > $clu_file";
2188        }
2189        elsif ( $counter2 == 0 )
2190        {
2191            $email_data .= <<"ENDHTML";
2192                    <br><center>All this alignment's columns have high confidence values.
2193                    <br>Masking is not needed.</center>
```

```
ENDHTML
    }
    else
    {
        $email_data .= <<"ENDHTML";
                <br><center>Most of this alignment's columns have poor confidence values.
                <br>Masking was <b>NOT</b> performed.</center>
ENDHTML
        system "cp $_[0] $input";
    }
    return 0;
}

#####################################################################
#Removing the TRICHOTOMY word, setting negative values to zero.#
#####################################################################
sub tree_manipulation1
{
    my $yacounter = 0;
    my ( $tree_fh, @tree );
    open $tree_fh, '<', $_[0] or die $!;
    my $line;
    {
        local $/ = "\n";
        while ( $line = <$tree_fh> )
        {
            $line =~ s/TRICHOTOMY//;
            $line =~ s/-\d[.]\d*/0/;
 ##############################################################################
 #Fix the NJ negative branch length artifact, by setting those numbers to zero.#
 ##############################################################################
            $tree[$yacounter] = $line;
            $yacounter++;
        }
    }
    close $tree_fh or die $!;
    open $tree_fh, '>', $_[0] or die $!;
    foreach my $yacounter (@tree)
    {
        print {$tree_fh} $yacounter;
    }
    close $tree_fh or die $!;
    return 0;
}
#############################################################################
#Dividing the bootstrap values by 10, to reach a maximum value of 100.#
#############################################################################
sub tree_manipulation2
{
    my $yacounter = 0;
    my ( $bvdiv, $tree_fh );
    my @tree = ();
    open $tree_fh, '<', $_[0] or die $!;
    my $line;
    {
```

```
2249            local $/ = "\n";
2250            while ( $line = <$tree_fh> )
2251            {
2252                if ( $line =~ /^(\d+):/ )
2253                {
2254                    $bvdiv = $1 / 10.0;
2255                    $line  = $` . $bvdiv . q{:} . $';
2256                }
2257                $tree[$yacounter] = $line;
2258                $yacounter++;
2259            }
2260        }
2261        close $tree_fh or die $!;
2262        open $tree_fh, '>', $_[0] or die $!;
2263        foreach my $yacounter (@tree)
2264        {
2265            print {$tree_fh} $yacounter;
2266        }
2267        close $tree_fh or die $!;
2268        return 0;
2269    }
2270
2271    ############################################################
2272    #Parse the tree to exract distances and bootstrap values.#
2273    ############################################################
2274    sub parser
2275    {
2276        my $cancounter    = 0;
2277        my $ginomenon     = 1;
2278        my $linocounter   = 0;
2279        my $node_distance = 0;
2280        my $plc           = 0;
2281        my $plc2          = 0;
2282        my $plcn          = 0;
2283        my $plcn2         = 0;
2284        my $sscounter     = 0;
2285        my $unicounter    = 0;
2286
2287        my (
2288            $continue,             $degree_of_confidence, $last_yes,
2289            $search_id,            $tree_for_parsingfh,   $tree_node_distancesfh,
2290            $tree_tip_distancesfh, @compare_seq,          @parsed_lines,
2291            @parsed_linesnew,      @parsing_lines,        @star_seq,
2292            @uni_ids,              %distanceshash,        %distanceshash2
2293        );
2294
2295        open $tree_for_parsingfh, '<', $_[0] or die $!;
2296        my $line;
2297        {
2298            local $/ = "\n\n";
2299            ########################
2300            #Get all the sequences.#
2301            ########################
2302            while ( $line = <$tree_for_parsingfh> )
2303            {
```

```
2304            ##########################
2305            #Find the input sequence.#
2306            ##########################
2307            if ( $line =~ /\s"(\w\_\_\_\w+\_\_\_)"/ )
2308            {
2309                $starting_point = $1;
2310                $uni_ids[$unicounter] = $1;
2311                $unicounter++;
2312            }
2313            ##############################
2314            #Find all the other sequences.#
2315            ##############################
2316            if ( $line =~ /"(\w{6,})"/ )
2317            {
2318                if ( $1 !~ /\D\d{1,4}\_/ )
2319                {
2320                    $uni_ids[$unicounter] = $1;
2321                    $unicounter++;
2322                }
2323            }
2324            if ( $' =~ /"(\w{6,})"/ && $1 !~ /\D\d{1,4}\_/ )
2325            {
2326                $uni_ids[$unicounter] = $1;
2327                $unicounter++;
2328            }
2329            $parsing_lines[$linocounter] = $line;
2330            $linocounter++;
2331        }
2332    }
2333
2334    ################################
2335    #Parsing distance between nodes.#
2336    ################################
2337    open $tree_node_distancesfh, '<', $_[1]
2338      or die $!;
2339    my $neoline;
2340    {
2341        local $/ = "\n";
2342        while ( $neoline = <$tree_node_distancesfh> )
2343        {
2344            if ( $neoline !~ /\$/ && $neoline =~ /\w/ )
2345            {
2346                do
2347                {
2348                    ####################################
2349                    #This line contains sequence names.#
2350                    ####################################
2351                    if ( $neoline !~ /[.]/ )
2352                    {
2353                        if (   $neoline =~ /\s?(\w+)/
2354                            || $neoline =~ /(Root)/ )
2355                        {
2356                            $parsed_lines[$plc][$plc2] = $1;
2357                            $plc2++;
2358                            $continue = $';
```

```
2359                            if ( $continue =~ /\w/ )
2360                            {
2361                                $neoline = $continue;
2362                            }
2363                            else
2364                            {
2365                                $plc++;
2366                            }
2367                        }
2368                    }
2369                    elsif (
2370                        ############################
2371                        #This line contains values.#
2372                        ############################
2373                        $neoline =~ /(\d+[.]\d+)/
2374                      )
2375                    {
2376                        $distanceshash{ $parsed_lines[ $plc - 1 ][$plc2] } = $1;
2377                        $plc2++;
2378                        $continue = $';
2379                        if ( $continue =~ /\w/ )
2380                        {
2381                            $neoline = $continue;
2382                        }
2383                    }
2384                } while ( $continue =~ /\w/ );
2385                $plc2 = 0;
2386            }
2387        }
2388    }
2389    close $tree_node_distancesfh or die $!;
2390    ########################################################
2391    #Parsing distance from every tip to its closest node.#
2392    ########################################################
2393    open $tree_tip_distancesfh, '<', $_[2]
2394      or die $!;
2395    my $neoline2;
2396    {
2397        local $/ = "\n";
2398        while ( $neoline2 = <$tree_tip_distancesfh> )
2399        {
2400            if ( $neoline2 !~ /\$/ && $neoline2 =~ /\w/ )
2401            {
2402                do
2403                {
2404                    ####################################
2405                    #This line contains sequence names.#
2406                    ####################################
2407                    if ( $neoline2 !~ /[.]/ )
2408                    {
2409                        if ( $neoline2 =~ /\_?\_?\_?(\w{6,})\_?\_?\_?\s/ )
2410                        {
2411                            $parsed_linesnew[$plcn][$plcn2] = $1;
2412                            $plcn2++;
2413                            $continue = $';
```

```
                            if ( $continue =~ /\w/ )
                            {
                                $neoline2 = $continue;
                            }
                            else
                            {
                                $plcn++;
                            }
                        }
                    }
                    ############################
                    #This line contains values.#
                    ############################
                    elsif ( $neoline2 =~ /(\d+[.]\d+)/ )
                    {
                        $distanceshash2{ $parsed_linesnew[ $plcn - 1 ][$plcn2] }
                          = $1;
                        $plcn2++;
                        $continue = $';
                        if ( $continue =~ /\w/ )
                        {
                            $neoline2 = $continue;
                        }
                    }
                } while ( $continue =~ /\w/ );
                $plcn2 = 0;
            }
        }
    }
    close $tree_tip_distancesfh or die $!;

    my $p_l_s = @parsing_lines;
    foreach my $line (@parsing_lines)
    {
        if ( $line =~ /$starting_point/ )
        {
            ####################################
            #Get the node of the input sequence.#
            ####################################
            if ( $line =~ /parts\$(\w)(\w+)/ )
            {
                $star_seq[$sscounter] = $1 . $2;
                $search_id = $1 . $2;
                $sscounter++;
            }
            for ( 0 .. $p_l_s - 1 )
            {
                if ( $parsing_lines[$_] =~ /$search_id"/ )
                {
                    ####################################
                    #Get the node leading to the node...#
                    ####################################
                    if ( $parsing_lines[$_] =~ /parts\$(\w)(\w+)/ )
                    {
                        $star_seq[$sscounter] = $1 . $2;
```

```
2469                        $search_id = $1 . $2;
2470                        $sscounter++;
2471                        $_--;
2472                    }
2473                }
2474            }
2475        }
2476    }
2477    $last_yes = 0;
2478    foreach my $uni (@uni_ids)
2479    {
2480        $sscounter = 0;
2481        foreach my $line (@parsing_lines)
2482        {
2483            if ( $line =~ /$uni/ )
2484            {
2485                ######################################################
2486                #Get the closest node for  all the other sequences.#
2487                ######################################################
2488                if ( $line =~ /parts\$(\w)(\w+)/ )
2489                {
2490                    $compare_seq[$sscounter] = $1 . $2;
2491                    $search_id = $1 . $2;
2492                    $sscounter++;
2493                }
2494                for ( 0 .. $p_l_s - 1 )
2495                {
2496                    if ( $parsing_lines[$_] =~ /$search_id"/ )
2497                    {
2498                        ####################################
2499                        #Get the node leading to the node...#
2500                        ####################################
2501                        if ( $parsing_lines[$_] =~ /parts\$(\w)(\w+)/ )
2502                        {
2503                            $compare_seq[$sscounter] = $1 . $2;
2504                            $search_id = $1 . $2;
2505                            $sscounter++;
2506                            $_--;
2507                        }
2508                    }
2509                }
2510            }
2511        }
2512        foreach my $node (@compare_seq)
2513        {
2514            if ( $node =~ /(\d+)\_?/ )
2515            {
2516                ##################################
2517                #Add the distance for this node.#
2518                ##################################
2519                $node_distance += $distanceshash{"$node"};
2520                $ginomenon *= $1 / 1000.0;
2521            }
2522            foreach my $star_node (@star_seq)
2523            {
```

```
2524                ######################################################################
2525                #Find the common node for the input sequence and every other one.#
2526                ######################################################################
2527                  if ( $node eq $star_node )
2528                  {
2529                      foreach my $star_node (@star_seq)
2530                      {
2531                          ######################################
2532                          #Add distances until the common node.#
2533                          ######################################
2534                          if (   $star_node ne $node
2535                              && $star_node =~ /(\d+)\_?/ )
2536                          {
2537                              $node_distance += $distanceshash{"$star_node"};
2538                              $ginomenon *= $1 / 1000.0;
2539                          }
2540                          ###################################
2541                          #Remove the common node's distance.#
2542                          ###################################
2543                          if ( $star_node eq $node )
2544                          {
2545                              if ( defined $distanceshash{"$star_node"} )
2546                              {
2547                                  $node_distance -= $distanceshash{"$star_node"};
2548                                  last;
2549                              }
2550                              else
2551                              {
2552                                  $node_distance = $distanceshash{"$star_node"};
2553                                  last;
2554                              }
2555                          }
2556                      }
2557                      ################################################
2558                      #For every sequence, except for the input one...#
2559                      ################################################
2560                      if ( $uni !~ /$starting_point/ )
2561                      {
2562                          ######################################
2563                          #Add the tip-to-closest-node distances.#
2564                          ######################################
2565                          $node_distance +=
2566                            $distanceshash2{"$uni"} +
2567                            $distanceshash2{"$starting_point"};
2568                         ###########################################################
2569                         #If distance is 0, then the sequences are overlapping!?!#
2570                         ###########################################################
2571                          if ( $node_distance == 0 )
2572                          {
2573                              $degree_of_confidence = $ginomenon / 0.1;
2574                              $candidate[$cancounter] =
2575                                $degree_of_confidence . q{ } . $uni;
2576                              $cancounter++;
2577                          }
2578                          ################################
```

```
2579                        #Calculate the confidence value.#
2580                        ################################
2581                        else
2582                        {
2583                            $degree_of_confidence = $ginomenon / $node_distance;
2584                            $candidate[$cancounter] =
2585                              $degree_of_confidence . q{ } . $uni;
2586                            $cancounter++;
2587                        }
2588                    }
2589                    $last_yes = 1;
2590                    last;
2591                }
2592                if ( $last_yes == 1 )
2593                {
2594                    last;
2595                }
2596            }
2597            if ( $last_yes == 1 )
2598            {
2599                last;
2600            }
2601        }
2602        if ( $last_yes == 0 )
2603        {
2604            $node_distance = 0;
2605            $ginomenon     = 1;
2606            ######################################################################
2607            #If the input sequence had only the root common with a sequence...#
2608            ######################################################################
2609            foreach my $node (@compare_seq)
2610            {
2611                if ( $node =~ /(\d+)\_?/ )
2612                {
2613                    ######################################################
2614                    #Add the distance from the sequence's node to root.#
2615                    ######################################################
2616                    $node_distance += $distanceshash{"$node"};
2617                    $ginomenon *= $1 / 1000.0;
2618                }
2619            }
2620            foreach my $star_node (@star_seq)
2621            {
2622                if ( $star_node =~ /(\d+)\_?/ )
2623                {
2624                    ############################################################
2625                    #Add the distance from the input sequence's node to root.#
2626                    ############################################################
2627                    $node_distance += $distanceshash{"$star_node"};
2628                    $ginomenon *= $1 / 1000.0;
2629                }
2630            }
2631            ########################################
2632            #Add the tip-to-closest-node distances.#
2633            ########################################
```

```
2634                $node_distance +=
2635                  $distanceshash2{"$uni"} + $distanceshash2{"$starting_point"};
2636                $degree_of_confidence = $ginomenon / $node_distance;
2637                $candidate[$cancounter] = $degree_of_confidence . q{ } . $uni;
2638                $cancounter++;
2639            }
2640            $node_distance = 0;
2641            $ginomenon     = 1;
2642            @compare_seq   = ();
2643            $last_yes      = 0;
2644        }
2645        @candidate = map join( q{ }, @{$_} ),
2646          sort { $a->[0] <=> $b->[0] } map { [split] } @candidate;
2647        ################################################################################
2648        #Sort the sequences in descending order, according to the confidence value.#
2649        ################################################################################
2650        @candidate = reverse @candidate;
2651        $cand_sans[0] = $one;
2652        my $neocounter = 1;
2653        foreach my $can (@candidate)
2654        {
2655            if ( $can =~ /\s(\w+)/ )
2656            {
2657                $cand_sans[$neocounter] = $1;
2658                $neocounter++;
2659            }
2660        }
2661        return @candidate, @cand_sans, $starting_point;
2662    }
2663
2664    ##############################################################
2665    #Calculate the Z-values and the probabilities for each hit.#
2666    ##############################################################
2667    sub compute_probability
2668    {
2669        my @numbers;
2670        my $numcounter = 0;
2671        my $re         = 0;
2672        ############################
2673        #Get the Confidence values.#
2674        ############################
2675        foreach my $can (@candidate)
2676        {
2677            if (   $can !~ /$starting_point/
2678                && $can =~ /(\d?\d?\d?.?\d+e?-?\d*) \w+/ )
2679            {
2680                $numbers[$numcounter] = $1;
2681                $numcounter++;
2682            }
2683        }
2684        ############################################
2685        #Calculate the mean and standard deviation.#
2686        ############################################
2687        my $mean               = mean(@numbers);
2688        my $standard_deviation = stddev(@numbers);
```

```
2689      foreach my $can (@candidate)
2690      {
2691          if (   $can !~ /$starting_point/
2692              && $can =~ /(\d?\d?\d?.?\d+e?-?\d*) (\w+)/ )
2693          {
2694              ######################
2695              #Calculate Z-values.#
2696              ######################
2697              my $z;
2698              if ( $standard_deviation ne '0' )
2699              {
2700                  $z = ( $1 - $mean ) / $standard_deviation;
2701              }
2702              else
2703              {
2704                  $z = ( $1 - $mean ) / 0.1;
2705              }
2706              ############################################################
2707              #For positive Z-values, calculate the level of confidence.#
2708              ############################################################
2709              if ( $z > 0.674 )
2710              {
2711                  my $erf  = erfc( $z / sqrt 2 );
2712                  my $prob = ( 1 - $erf ) * 100;
2713                  $can = $z . q{ } . $prob . q{ } . $2;
2714                  $realign[$re] = $2;
2715                  $re++;
2716              }
2717              else
2718              {
2719                  $can = $z . q{ } . $2;
2720              }
2721          }
2722      }
2723      return @candidate, @realign;
2724  }
2725
2726  ###############################################################################
2727  #Find the GOs shared between the input sequence and every other sequence.#
2728  ###############################################################################
2729  sub GOs_in_common
2730  {
2731      my @input_gos;
2732      my $input_counter = 0;
2733      if ( $one ne 'QUERY' )
2734      {
2735          foreach my $seq (@cand_sans)
2736          {
2737              chomp $seq;
2738              if ( $seq eq $one )
2739              {
2740                  my $dbfetch = get("http://www.uniprot.org/uniprot/$one.txt");
2741                  if ( !( defined $dbfetch ) )
2742                  {
2743                      for ( 0 .. 3 )
```

```
2744                    {
2745                        $dbfetch =
2746                          get("http://www.uniprot.org/uniprot/$one.txt");
2747                        if ( defined $dbfetch )
2748                        {
2749                            last;
2750                        }
2751                    }
2752                }
2753                do
2754                {
2755                    ###############################################################
2756                    #Parse the input sequence's UniProt flat file for GO terms.#
2757                    ###############################################################
2758                    if ( $dbfetch =~ /GO; GO:(\d+);/ )
2759                    {
2760                        $input_gos[$input_counter] = $1;
2761                        $input_counter++;
2762                        $dbfetch = $';
2763                    }
2764                } while ( $dbfetch =~ /GO; GO:(\d+);/ );
2765                $one_gos = @input_gos;
2766                if ( $one_gos == 0 )
2767                {
2768                    last;
2769                }
2770            }
2771            else
2772            {
2773                my $common_prop  = q{};
2774                my $ncommon_prop = q{};
2775                my @go_list      = ();
2776                my $ncpi         = 0;
2777                my $eq_counter   = 0;
2778                my $res_counter  = 0;
2779                my $dbfetch = get("http://www.uniprot.org/uniprot/$seq.txt");
2780                if ( !( defined $dbfetch ) )
2781                {
2782
2783                    for ( 0 .. 3 )
2784                    {
2785                        $dbfetch =
2786                          get("http://www.uniprot.org/uniprot/$seq.txt");
2787                        if ( defined $dbfetch )
2788                        {
2789                            last;
2790                        }
2791                    }
2792                }
2793                do
2794                {
2795            ###########################################################################
2796            #Parse every other sequence's UniProt flat file for GO terms in common.#
2797            ###########################################################################
2798                    if ( defined $dbfetch
```

```
2799                        && $dbfetch =~ /GO; GO:(\d+);\s(.+);/ )
2800                    {
2801                        $go_list[$ncpi] = $2;
2802                        $ncpi++;
2803                        foreach my $go (@input_gos)
2804                        {
2805                            if ( $go == $1 )
2806                            {
2807                                $eq_counter++;
2808                                $common_prop .= $2 . "\n";
2809                            }
2810                        }
2811                        $res_counter++;
2812                        $dbfetch = $';
2813                    }
2814                } while ( $dbfetch =~ /GO; GO:(\d+);\s(.+);/ );
2815                foreach my $term (@go_list)
2816                {
2817                    $term =~ s/[?]/\[?]/;
2818                    if ( $common_prop !~ /$term/ )
2819                    {
2820                        $ncommon_prop .= $term . "\n";
2821                    }
2822                }
2823                my $neq_counter = $res_counter - $eq_counter;
2824                ########################################
2825                #If no GO terms are found, put in "NA"#
2826                ########################################
2827                if ( $eq_counter == 0 && $res_counter == 0 )
2828                {
2829                    $eq_counter  = 'NA';
2830                    $neq_counter = 'NA';
2831                }
2832                $common{$seq} = $eq_counter . q{ } . $neq_counter;
2833                chomp $common_prop;
2834                chomp $ncommon_prop;
2835                $textcommon{$seq}  = $common_prop;
2836                $textncommon{$seq} = $ncommon_prop;
2837            }
2838        }
2839        return $one_gos, %common, %textcommon, %textncommon;
2840    }
2841    else
2842    {
2843        foreach my $seq (@cand_sans)
2844        {
2845            chomp $seq;
2846            if ( $seq ne $one )
2847            {
2848                my $ontologies  = q{};
2849                my @go_list     = ();
2850                my $ncpi        = 0;
2851                my $res_counter = 0;
2852                my $dbfetch = get("http://www.uniprot.org/uniprot/$seq.txt");
2853                if ( !( defined $dbfetch ) )
```

```
2854                {
2855                    for ( 0 .. 3 )
2856                    {
2857                        $dbfetch =
2858                          get("http://www.uniprot.org/uniprot/$seq.txt");
2859                        if ( defined $dbfetch )
2860                        {
2861                            last;
2862                        }
2863                    }
2864                }
2865                do
2866                {
2867            ##########################################################################
2868            #Parse every other sequence's UniProt flat file for GO terms in common.#
2869            ##########################################################################
2870                    if ( defined $dbfetch
2871                        && $dbfetch =~ /GO; GO:(\d+);\s(.+);/ )
2872                    {
2873                        $go_list[$ncpi] = $2;
2874                        $ncpi++;
2875                        $res_counter++;
2876                        $dbfetch = $';
2877                    }
2878                  } while ( defined $dbfetch
2879                    && $dbfetch =~ /GO; GO:(\d+);\s(.+);/ );
2880                foreach my $term (@go_list)
2881                {
2882                    $term =~ s/[?]/\[?]/;
2883                    $ontologies .= $term . "\n";
2884                }
2885                ######################################
2886                #If no GO terms are found, put in "NA"#
2887                ######################################
2888                if ( $res_counter == 0 )
2889                {
2890                    $res_counter = 'NA';
2891                }
2892                $texts{$seq} = $res_counter;
2893                chomp $ontologies;
2894                $texts2{$seq} = $ontologies;
2895            }
2896        }
2897        return %texts, %texts2;
2898    }
2899 }
2900
2901 ############################
2902 #Email sending subroutine.#
2903 ############################
2904 sub send_email
2905 {
2906     my $msg = MIME::Lite->new(
2907         Subject => "Pinda Job Result: $_[0]",
2908         From    => 'Pinda@orion.mbg.duth.gr',
```

```
2909            To      => $_[1],
2910            Type    => 'text/html',
2911            Data    => $email_data
2912        );
2913        $msg->send();
2914        return 0;
2915    }
```

4.4 Pinda.R

Το Pinda.R εξάγει τα δεδομένα αποστάσεων και τιμών bootstrap από το παραγόμενο δενδρόγραμμα, ενώ επίσης απεικονίζει το δένδρο.

```
1   #!/usr/bin/Rscript
2
3   ########################################
4   #Author: Dimitrios - Georgios Kontopoulos#
5   ########################################
6
7   ######################################
8   #Script in the R programming language#
9   # for drawing and parsing NJ-trees,  #
10  #       bootstrapped or not          #
11  ######################################
12
13  ##########################################################################
14  #This program is free software: you can redistribute it and/or modify#
15  #it under the terms of the GNU General Public License as              #
16  #published by the Free Software Foundation, either version 3 of the   #
17  #License, or (at your option) any later version.                      #
18  #                                                                     #
19  #For more information, see http://www.gnu.org/licenses/.              #
20  ##########################################################################
21
22  args <- commandArgs(TRUE)
23  #r <- getOption('repos')
24  #r['CRAN'] <- 'http://cran.cc.uoc.gr/'
25  #options(repos=r)
26  #install.packages(c('ape','ade4') lib='../R')
27  if (args[1] == "-parser") {
28      library(ade4, lib.loc = "../R/")  #load ADE4 package
29      tree <- newick2phylog(scan(args[2], what = ""), add.tools = FALSE)
30      treestructure <- tree[4]
31      print(treestructure)
32  } else if (args[1] == "-lengths_1") {
33      library(ade4, lib.loc = "../R/")
34      tree <- newick2phylog(scan(args[2], what = ""), add.tools = FALSE)
35      treestructure <- tree[3]
36      print(treestructure)
37  } else if (args[1] == "-lengths_2") {
38      library(ade4, lib.loc = "../R/")
39      tree <- newick2phylog(scan(args[2], what = ""), add.tools = FALSE)
40      treestructure <- tree[2]
```

```
41      print(treestructure)
42  } else {
43      library(ape, lib.loc = "../R/")  # load APE package
44      png(filename = args[1], width = 750, height = 550, units = "px",
45          bg = "transparent")  # set output file
46      tree <- read.tree(args[2])  # read tree
47      plot(tree, use.edge.length = TRUE, show.node.label = TRUE,
48          font = 2, no.margin = TRUE, node.pos = 2)  # plot tree
49      add.scale.bar(length = 0.05, col = "red", lcol = "red")  # add scale bar
50      dev.off()  #quit
51  }
```

Κεφάλαιο 5

Επίλογος

Ἡ Ἰθάκη σ᾽ ἔδωσε τ᾽ ὡραῖο ταξίδι.
Χωρὶς αὐτὴν δὲν θά ᾽βγαινες στὸν δρόμο.
Ἄλλα δὲν ἔχει νὰ σὲ δώσει πιά.

Κι ἂν πτωχικὴ τὴν βρεῖς, ἡ Ἰθάκη δὲν σὲ γέλασε.
Ἔτσι σοφὸς ποὺ ἔγινες, μὲ τόση πεῖρα,
ἤδη θὰ τὸ κατάλαβες οἱ Ἰθάκες τὶ σημαίνουν.
Κωνσταντίνος Π. Καβάφης

Σε αυτή τη διπλωματική εργασία παρουσιάστηκε ένα υπολογιστικό πρόγραμμα για τη διευκόλυνση της μελέτης των γονιδιακών διπλασιασμών. Θέλουμε να πιστεύουμε ότι ο στόχος επετεύχθη: το πρόγραμμα είναι αρκετά αξιόπιστο και ταυτόχρονα φιλικό προς το μέσο χρήστη. Θεωρούμε ότι οι αναλύσεις που πραγματοποιούνται υπερκαλύπτουν τις ανάγκες μιας πρώτης εκτίμησης γεγονότων διπλασιασμού, προσθέτοντας ένα ακόμη βέλος στην υπολογιστική "φαρέτρα" ενός "κλασικού" μοριακού βιολόγου. Όσο οι βάσεις δεδομένων θα εμπλουτίζονται, τόσο θα αυξάνει η ακρίβεια και η ευαισθησία του προγράμματος. Επιπλέον, λόγω της ευκολίας στην ενσωμάτωση τρίτων προγραμμάτων, βελτιωμένα ή καινοτόμα λογισμικά βιοπληροφορικής μπορούν σταδιακά να εντάσσονται στο πρωτόκολλο του Pinda. Όλα αυτά που αναφέρθηκαν αναμένεται να καταστήσουν αυτή την εφαρμογή ένα ακόμα πολύτιμο εργαλείο στη μελέτη της μοριακής εξέλιξης.

Βιβλιογραφία

1. Russel,P. J. (2009) **iGenetics: Μια Μεντελική Προσέγγιση**, Ακαδημαϊκές Εκδόσεις Ι. Μπάσδρα και ΣΙΑ Ο.Ε., Αλεξανδρούπολη, Ελλάδα, τόμοι Ι και ΙΙ.

2. Zhang,J. (2003) **Evolution by gene duplication: an update**. Trends in Ecology & Evolution, 18, 292-298.

3. Watson,J. D., Caudy,A. A., Myers,R. M., Witkowski,J. A. (2007) **Ανασυνδυασμένο DNA**, Ακαδημαϊκές Εκδόσεις Ι. Μπάσδρα και ΣΙΑ Ο.Ε., Αλεξανδρούπολη, Ελλάδα.

4. Brown,T. A. (2010) **Γονιδιώματα**, Εκδόσεις Π.Χ. Πασχαλίδης Ε.Π.Ε., Αθήνα, Ελλάδα.

5. Alberts,B., Bray,D., Hopkin,K., Johnson,A., Lewis,J., Raff,M., Roberts,K., Walter,P. (2006) **Βασικές Αρχές Κυτταρικής Βιολογίας**, Εκδόσεις Π.Χ. Πασχαλίδης Ε.Π.Ε., Αθήνα, Ελλάδα, τόμος Ι.

6. Τριανταφυλλίδης,Κ. (2007) **Κλασική και Μοριακή Γενετική**, Εκδοτικός Οίκος Αδελφών Κυριακίδη Α.Ε., Θεσσαλονίκη, Ελλάδα.

7. Barton,N. M., Briggs,D. E. G., Eisen,J. A., Goldstein,D. B., Patel,N. H. (2007) **Evolution**, Cold Spring Harbor Laboratory Press, New York, United States of America.

8. Lewin,B. (2004) **Genes VIII**, Ακαδημαϊκές Εκδόσεις Ι. Μπάσδρα και ΣΙΑ Ο.Ε., Αλεξανδρούπολη, Ελλάδα, τόμος Ι.

9. Κοσσίδα,Σ. (2008) **Βιοπληροφορική: Δυνατότητες και προοπτικές**, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, Ελλάδα.

10. Λουκάς,Μ. Γ. (2000) **Γενετική**, Εκδόσεις Αθ. Σταμούλης, Αθήνα, Ελλάδα, τόμος Ι.

11. Madigan,M. T., Martinko,J. M., Parker,J. (2007) **Brock: Βιολογία των Μικροοργανισμών**, Πανεπιστημιακές Εκδόσεις Κρήτης, Ηράκλειο, Ελλάδα, τόμος Ι.

12. Gu,X. (2011) **Statistical Theory & Methods for Evolutionary Genomics**, Oxford University Press, Chippenham, Great Britain.

13. Wu,M., Chatterji,S., Eisen,J. A. (2012) **Accounting for alignment uncertainty in phylogenomics.** PloS one, 7, e30288.

14. Saitou,N., Nei,M. (1987) **The neighbor-joining method: a new method for reconstructing phylogenetic trees**. Molecular biology and evolution, 4, 406-25.

15. Felsenstein,J. (1985) **Confidence limits on phylogenies: an approach using the bootstrap.** Evolution, 39, 783–791.

16. Hillis,D. M., Bull,J. J. (1993) **An empirical test of bootstrapping as a method for assessing confidence in phylogenetic analysis.** Systematic Biology, 42, 182.

17. Wikipedia (2012) **Gene Ontology.** `http://en.wikipedia.org/wiki/Gene_ontology`.

18. du Plessis,L., Škunca,N., Dessimoz,C. (2011) **The what, where, how and why of gene ontology--a primer for bioinformaticians.** Briefings in bioinformatics, 12, 723-35.

19. Bekman,S., Cholet,E. (2003) **Practical mod_perl**, O'Reilly & Associates, Inc., Sebastopol, California, United States of America.

20. Pierce,C. (2006) **Οδηγός της Perl**, Εκδόσεις Μ. Γκιούρδας, Αθήνα, Ελλάδα.

21. Pollock,J. (2004) **JavaScript: a beginner's guide, Second Edition**, McGraw-Hill/Osborne, Emeryville, California, United States of America.

22. R Development Core Team (2011) **R: A language and environment for**

statistical computing. http://www.R-project.org.

23. Paradis,E., Claude,J., Strimmer,K. (2004) **APE: Analyses of Phylogenetics and Evolution in R language.** Bioinformatics, 20, 289-290.

24. Dray,S., Dufour,A. B. (2007) **The ade4 package: implementing the duality diagram for ecologists.** Journal of Statistical Software, 22.

25. Altschul,S. F., Madden,T. L., Schaffer,A. A., Zhang,J., Zhang,Z., Miller,W., Lipman,D. J. (1997) **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.** Nucleic acids research, 25, 3389-402.

26. Apweiler,R., Bairoch,A., Wu,C. H., Barker,W. C., Boeckmann,B., Ferro,S., Gasteiger,E., Huang,H., Lopez,R., Magrane,M., Martin,M. J., Natale,D. A., O'Donovan,C., Redaschi,N., Yeh,L. - S. L. (2004) **UniProt: the universal protein knowledgebase.** Nucleic acids research, 32, D115–D119.

27. Lawrence Livermore National Laboratory (2008) **SLURM: A Highly Scalable Resource Manager.** https://computing.llnl.gov/linux/slurm/slurm.html.

28. Zhang,Z., Schwartz,S., Wagner,L., Miller,W. (2000) **A greedy algorithm for aligning DNA sequences.** Journal of Computational Biology, 7, 203-14.

29. Sievers,F., Wilm,A., Dineen,D., Gibson,T. J., Karplus,K., Li,W., Lopez,R., McWilliam,H., Remmert,M., Söding,J., Thompson,J. D., Higgins,D. G. (2011) **Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega**. Molecular systems biology, 7, 1-6.

30. Lassmann,T., Frings,O., Sonnhammer,E. L. L. (2009) **Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features.** Nucleic acids research, 37, 858-65.

31. Wu,M., Chatterji,S., Eisen,J. A. (2012) **Accounting for alignment uncertainty in phylogenomics.** PloS one, 7, e30288.

32. Thompson,J. D., Higgins,D. G., Gibson,T. J. (1994) **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence**

weighting, position-specific gap penalties and weight matrix choice. Nucleic acids research, 22, 4673.

33. Free Software Foundation (2011) **GNU Affero General Public License.** `http://www.gnu.org/licenses/agpl.html`.

34. Fujimi,T. J., Nakajyo,T., Nishimura,E., Ogura,E., Tsuchiya,T., Tamiya,T. (2003) **Molecular evolution and diversification of snake toxin genes, revealed by analysis of intron sequences.** Gene, 313, 111-118.

35. Tsujimura,T., Hosoya,T., Kawamura,Sh. (2010) **A single enhancer regulating the differential expression of duplicated red-sensitive opsin genes in zebrafish**. PLoS genetics, 6, e1001245.

Κεφάλαιο 6

Παράρτημα - Βοηθητικά Scripts

6.1 db_temp_check.sh

Bash script για τον εντοπισμό προσωρινών αρχείων βάσεων δεδομένων που παραμένουν μετά από δυο ημέρες από την προγραμματισμένη ανανέωσή τους. Η ύπαρξή τους οφείλεται πιθανόν σε διακοπή ρεύματος ή σε απώλεια σύνδεσης Internet.

```
1  #!/bin/bash
2
3  ##########################################################
4  #Bash script to check for remaining db temporary files.#
5  ##########################################################
6
7  ########################################
8  #Author: Dimitrios - Georgios Kontopoulos#
9  ########################################
10
11 ########################################################################
12 #This program is free software: you can redistribute it and/or modify#
13 #it under the terms of the GNU General Public License as             #
14 #published by the Free Software Foundation, either version 3 of the  #
15 #License, or (at your option) any later version.                     #
16 #                                                                    #
17 #For more information, see http://www.gnu.org/licenses/.             #
18 ########################################################################
19
20 ######################################################
21 #Check for remaining database temporary directories.#
22 ######################################################
23
24 set -e
25
26 touch /tmp/whut
27 if [ -d "/usr/local/databases/Swissprot.1" ]
28 then
29    echo "Two days after the last update, the Swissprot.1 directory is still there.
30    Did a power failure take place? Please, take a look." >> /tmp/whut
```

```
31 fi
32
33 if [ -d "/usr/local/databases/UniProt.1" ]
34 then
35    echo "Two days after the last update, the UniProt.1 directory is still there.
36    Did a power failure take place? Please, take a look." >> /tmp/whut
37 fi
38
39 if [ -d "/usr/local/databases/nt.1" ]
40 then
41    echo "Two days after the last update, the nt.1 directory is still there.
42    Did a power failure take place? Please, take a look." >> /tmp/whut
43 fi
44
45 if [ -s "/tmp/whut" ]
46 then
47    mail Pinda -s 'DB temp files are still there!' < /tmp/whut
48 fi
49
50 rm /tmp/whut
51 exit 0
```

6.2 db_update.sh

Bash script για την ενημέρωση των βάσεων δεδομένων.

```
1  #!/bin/bash
2
3  ################################################
4  #Bash script to keep the databases up to date.#
5  ################################################
6
7  ##########################################
8  #Author: Dimitrios - Georgios Kontopoulos#
9  ##########################################
10
11 ##########################################################################
12 #This program is free software: you can redistribute it and/or modify#
13 #it under the terms of the GNU General Public License as              #
14 #published by the Free Software Foundation, either version 3 of the   #
15 #License, or (at your option) any later version.                      #
16 #                                                                     #
17 #For more information, see http://www.gnu.org/licenses/.             #
18 ##########################################################################
19
20
21 ##########################################################################
22 #Get the Swiss-Prot db, format it and also copy it to the UniProt folder.#
23 ##########################################################################
24 mkdir /usr/local/databases/Swissprot.1/
25 cd /usr/local/databases/Swissprot.1/
26
27 Swiss_status="999"
28 while [ "$Swiss_status" -ne "0" ]; do
```

```
29    rm *.gz
30    nice -n +19 wget ftp://ftp.ebi.ac.uk/pub/databases/uniprot/knowledgebase/uniprot_sprot.fasta.gz
31    nice -n +19 gunzip uniprot_sprot.fasta.gz
32    Swiss_status="$?"
33 done
34
35 mkdir /usr/local/databases/UniProt.1/
36 cp uniprot_sprot.fasta ../UniProt.1/
37 nice -n +19 makeblastdb -in uniprot_sprot.fasta -dbtype prot -parse_seqids
38 rm uniprot_sprot.fasta
39
40 ##############################################################################
41 #Get the TrEMBL db, concatenate it with the Swiss-Prot one and format it.#
42 ##############################################################################
43 cd ../UniProt.1/
44
45 Uni_status="999"
46 while [ "$Uni_status" -ne "0" ]; do
47    rm *.gz
48    nice -n +19 wget ftp://ftp.ebi.ac.uk/pub/databases/uniprot/knowledgebase/uniprot_trembl.fasta.gz
49    nice -n +19 gunzip uniprot_trembl.fasta.gz
50    Uni_status="$?"
51 done
52
53 cat uniprot_sprot.fasta uniprot_trembl.fasta > UniProt.fasta
54 rm -rf uniprot_sprot.fasta uniprot_trembl.fasta
55 nice -n +19 makeblastdb -in UniProt.fasta -dbtype prot -parse_seqids
56 rm UniProt.fasta
57
58 ################################################
59 #Check for running processes (blastp/psiblast).#
60 ################################################
61 while true
62 do
63    if (ps ax | grep -v grep | grep blastp) > /dev/null && ( ps ax | grep -v grep | grep psiblast) > /dev/null;
64    then
65       sleep 60
66    else
67       rm -rf /usr/local/databases/Swissprot/ && mv /usr/local/databases/Swissprot.1/ /usr/local/databases/Swissprot/
68       rm -rf /usr/local/databases/UniProt/ && mv /usr/local/databases/UniProt.1/ /usr/local/databases/UniProt/
69       break
70    fi
71 done
72
73 ####################################
74 #Get the nt database and format it.#
75 ####################################
76 mkdir /usr/local/databases/nt.1/
77 cd /usr/local/databases/nt.1/
78
79 nt_status="999"
80 while [ "$nt_status" -ne "0" ]; do
81    rm *.gz
82    nice -n +19 wget ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nt.gz
83    nice -n +19 gunzip nt.gz
```

```
84      nt_status="$?"
85  done
86
87  mv nt nt.fasta
88  nice -n +19 makeblastdb -in nt.fasta -dbtype nucl -parse_seqids
89  rm nt.fasta
90
91  while true
92  do
93      if (ps ax | grep -v grep | grep blastn) > /dev/null;
94      then
95          sleep 60
96      else
97          rm -rf /usr/local/databases/nt/ && mv /usr/local/databases/nt.1/ /usr/local/databases/nt/
98          break
99      fi
100 done
101 exit 0
```

6.3 remove_temp.sh

Bash script για τη διαγραφή προσωρινών αρχείων που δημιουργήθηκαν τουλάχιστον πριν από 10 ημέρες.

```
1   #!/bin/bash
2
3   ##########################################################
4   #Bash script for temporary files removal after 10 days.#
5   ##########################################################
6
7   ########################################
8   #Author: Dimitrios - Georgios Kontopoulos#
9   ########################################
10
11  ##########################################################################
12  #This program is free software: you can redistribute it and/or modify#
13  #it under the terms of the GNU General Public License as             #
14  #published by the Free Software Foundation, either version 3 of the  #
15  #License, or (at your option) any later version.                     #
16  #                                                                    #
17  #For more information, see http://www.gnu.org/licenses/.             #
18  ##########################################################################
19
20  remove_temp_files ()
21  {
22  ##########################################################################
23  #Find files older than 10 days in the current directory and remove them.#
24  ##########################################################################
25      find . -mtime +11 -name "*" -exec rm {} \;
26  }
27
28  cd /var/www/Pinda/outs/blast/
29  remove_temp_files
30
31  cd /var/www/Pinda/outs/psiblast/
```

```
32  remove_temp_files
33
34  cd /var/www/Pinda/parsing/
35  remove_temp_files
36
37  cd /var/www/Pinda/results/final_alns/multalign/
38  remove_temp_files
39
40  cd /var/www/Pinda/results/final_alns/multalign/conf/
41  remove_temp_files
42
43  cd /var/www/Pinda/results/trees/drawn/
44  remove_temp_files
45
46  cd /var/www/Pinda/results/trees/phs/
47  remove_temp_files
48
49  cd /var/www/Pinda/results/trees/phbs/
50  remove_temp_files
51
52  cd /var/www/Pinda/results/trees/zips/
53  remove_temp_files
54
55  cd /var/www/Pinda/seq/final_seq/
56  remove_temp_files
57
58  cd /var/www/Pinda/tmps/blast/
59  remove_temp_files
60
61  exit 0
```