ARISTOTLE UNIVERSITY OF THESSALONIKI

POLYTECHNIC SCHOOL

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

INTELLIGENT SYSTEMS & SOFTWARE ENGINEERING LABGROUP

DIPLOMA THESIS

Design and Implementation of a Methodology for the Evaluation of Somatic Mutation Identification Algorithms

Elaboration:

Mitsigkolas Anastasios

tasos1109@gmail.com

Student Number: 8666

Supervisors:

Professor **Pericles A. Mitkas**

Dr. **Fotis E. Psomopoulos**

Thessaloniki, June 2020

Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνεται ο κύκλος της φοιτητικής μου πορείας στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης. Δράττομαι έτσι της ευκαιρίας να ευχαριστήσω ορισμένους ανθρώπους που με στήριξαν όλα αυτά τα χρόνια.

Θα ήθελα να ευχαριστήσω τον καθηγητή Περικλή Α. Μήτκα που με δέχτηκε στο εργαστήριο Επεξεργασίας Πληροφορίας και μου εμπιστεύθηκε την εκπόνηση της παρούσας εργασίας.

Ακόμη, θα ήθελα να ευχαριστήσω τον μετά-διδάκτορα κ. Φώτη Ε. Ψωμόπουλο για την συνεργασία μας και τις αμέτρητες γνώσεις με τις οποίες με εφοδίασε. Η διαρκής καθοδήγηση που μου παρείχαν αποτέλεσε τον σημαντικότερο παράγοντα για την περάτωση της εν λόγω εργασίας.

Επιπρόσθετα, θα ήθελα να ευχαριστήσω όλους όσους ενεπλάκησαν από το Ινστιτούτο Εφαρμοσμένων Βιοεπιστημών (INAB|CERTH) στην παροχή δεδομένων καθώς και για τη συμβολή τους στη διαμόρφωση της παρούσας διπλωματικής.

Τέλος, σε προσωπικό επίπεδο θα ήθελα να ευχαριστήσω την οικογένεια μου καθώς και τους φίλους μου που με στήριξαν κατά την διάρκεια όλων αυτών των χρόνων.

Διπλωματική Εργασία

Τίτλος

«Σχεδιασμός και Ανάπτυξη Μεθοδολογίας για την Αξιολόγηση Αλγορίθμων Ανίχνευσης Σωματικών Μεταλλάξεων»

Περίληψη

Με την ολοένα αυξανόμενη ανάπτυξη της επιστήμης των υπολογιστών, τα προγράμματα και οι αλγόριθμοι που αναπτύσσονται γίνονται όλο και περισσότερο εξειδικευμένα και αποδοτικά. Ταυτόχρονα με την ανάπτυξή τους, όμως, η πολυπλοκότητα τους αυξάνεται δραματικά. Για τον βιοπληροφορικό ή/και τον βιολόγο η αυξημένη πολυπλοκότητα στη χρήση τέτοιων προγραμμάτων μπορεί να οδηγήσει σε άσκοπη σπατάλη πόρων και χρόνου.

Πιο συγκεκριμένα η επιλογή παραμέτρων και το λεγόμενο optimization των προγραμμάτων αυτών μπορεί να γίνει τόσο περίπλοκο και να οδηγήσει σε μέτρια ή και κακή λύση ενός βιολογικού προβλήματος.

Κρίνεται λοιπόν αναγκαία η δημιουργία ενός συστηματικού και αυτοματοποιημένου τρόπου βελτιστοποίησης των παραμέτρων εισόδου σε αυτά τα εργαλεία με σκοπό την διευκόλυνση των ερευνητών και την εύρεση της βέλτιστης δυνατής λύσης του προς εξέταση προβλήματος.

Ταυτόχρονα είναι αναγκαία η δημιουργία εργαλείων προσομοίωσης πραγματικών δεδομένων, αφού μόνο με γνωστά δεδομένα που προσεγγίζουν επαρκώς καλά την πραγματικότητα μπορούν να επιτευχθούν οι στόχοι της βελτιστοποίησης παραμέτρων.

Συνοπτικά στην παρούσα διπλωματική επιχειρείται και παρουσιάζεται η ανάπτυξη μεθόδων αυτοματοποίησης και βελτιστοποίησης της εκτέλεσης εργαλείων εύρεσης σωματικών μεταλλάξεων. Πιο συγκεκριμένα:

- Επιχειρείται η δημιουργία εργαλείου παραγωγής τεχνητών δεδομένων amplicons, με εισαγωγή σε αυτά γνωστών μεταλλάξεων προς εύρεση.
- Γίνεται απόπειρα αυτοματοποίησης της επαναληπτικής εκτέλεσης του GATK pipeline.
- Επιχειρείται η βαθμολόγηση των εκτελέσεων ως προς την αποτελεσματικότητά τους, μέσω της σύγκρισης των αποτελεσμάτων με τα αρχικά γνωστά δεδομένα.
- Τέλος εμφανίζονται εμφανισιακά ελκυστικά γραφήματα για την παρουσίαση των αποτελεσμάτων.

Diploma Thesis

Title

«Design and Implementation of a Methodology for the Evaluation of Somatic Mutation Identification Algorithms»

Abstract

In view of the rapid development of the Computer Science, these days, the programs and the algorithms which are invented become more and more sophisticated and efficient. With their spiraling development, their complexity increases dramatically, as well. The increased complexity may cause serious waste of resources and time, as these programs are used by the bioinformaticians or biologists.

More specifically, the choice of the input parameters and the optimization of these programs might become so complex that could lead to the finding of a non-optimum solution of the biological problem.

As a result, there is a necessity of developing a systematic and automated method that can optimize the input parameters of these tools, with the purpose of facilitating the researchers and finding a better possible solution.

The creation of simulation tools that can simulate real data is also necessary, because only if we use artificially created data with known mutations, can we optimize these tools.

The present diploma thesis focuses on developing automated methods and algorithms that can optimize the input parameters of variant discovery tools as we mentioned above. Specifically:

- We attempt to develop a tool which aims at producing artificially sequencing data with known mutations in reasonable time.
- We attempt to automate the GATK by making use of bash scripts.
- We attempt to evaluate the corresponding executions of the GATK pipeline in terms of effectiveness, through the comparison of the pipeline's results and the known mutexed amplicons.
- Finally, we try to visualize the results by making use of graphical infographics.

Short Description of the Problem

As it was mentioned above, the purpose of the bioinformatics is the use of computational and mathematical methods in order to extract useful information from the enormous amount of digital data, which is produced by the sampling, isolation and processing of the genetic material. This challenge demands the development of software and systematic methods to preprocess, analyze and, finally, extract information from the raw genetic material. These tools should be able to recognize patterns, explore trends and use prediction models in order to deliver useful results about the current experiment. Due to the exponential rise of the complexity of the biological problems these tools tend to be more and more complex, as well, not only in their development, but also in their usage. In conclusion, there is a necessity of the creation of efficiency programs and the selection of optimized input parameters. We call this problem the “Parameters Optimization Problem”.

Definition of Mutations

In biology, a mutation is an alteration in the nucleotide sequence of the genome of an organism, virus, or extrachromosomal DNA. Viral genomes can be of either DNA or RNA. Mutations result from errors during DNA replication, mitosis, and meiosis or other types of damage to DNA (such as pyrimidine dimers that may be caused by exposure to radiation or carcinogens), which then may undergo error-prone repair (especially microhomology-mediated end joining) or cause an error during other forms of repair or else may cause an error during replication (translesion synthesis). Mutations may also result from insertion or deletion of segments of DNA due to mobile genetic elements.

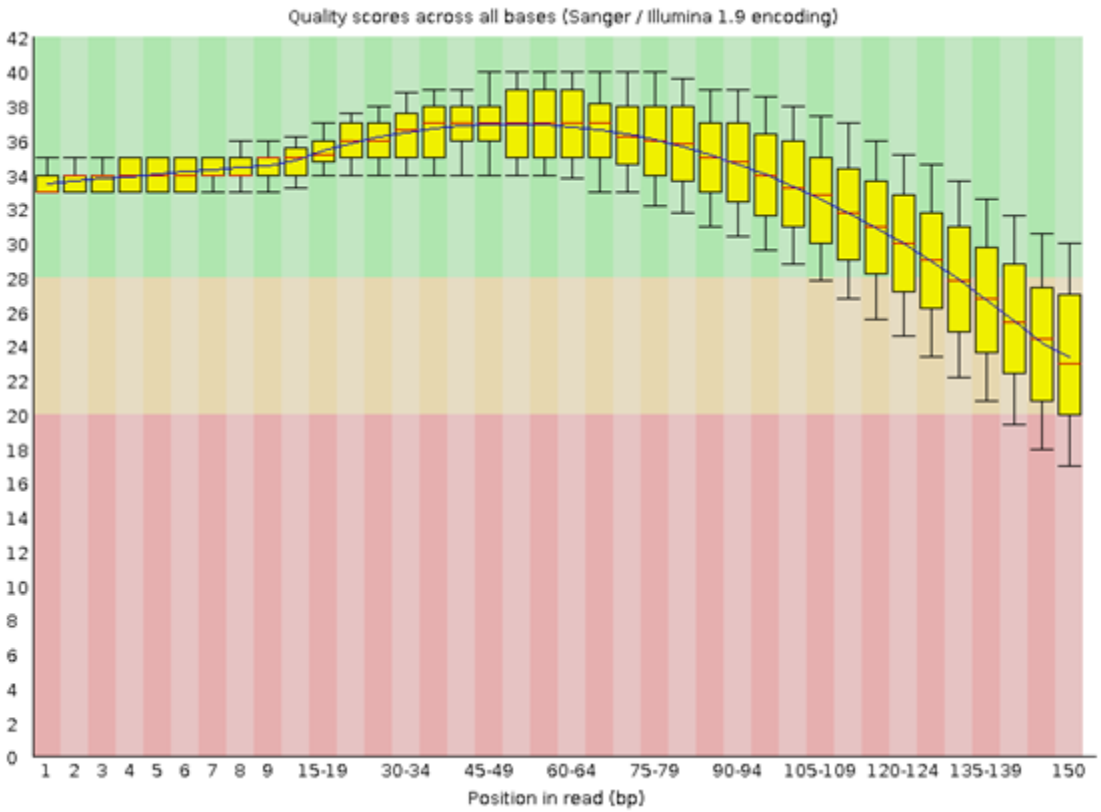
Searching of Somatic Mutations

Hence, the searching of somatic mutations is especially important. For this reason, quite efficient tools for searching somatic mutations have already been developed. These tools are so developed that complex probabilistic models are used in order to find somatic mutations in aligned amplicons of DNA pieces. Due to the fact that the models used by these tools are so complex, it is necessary that these tools be optimized by the bioinformatician in each problem. In this diploma thesis we represent an attempt to automate a parametric analysis with the purpose of selecting optimized input parameters.

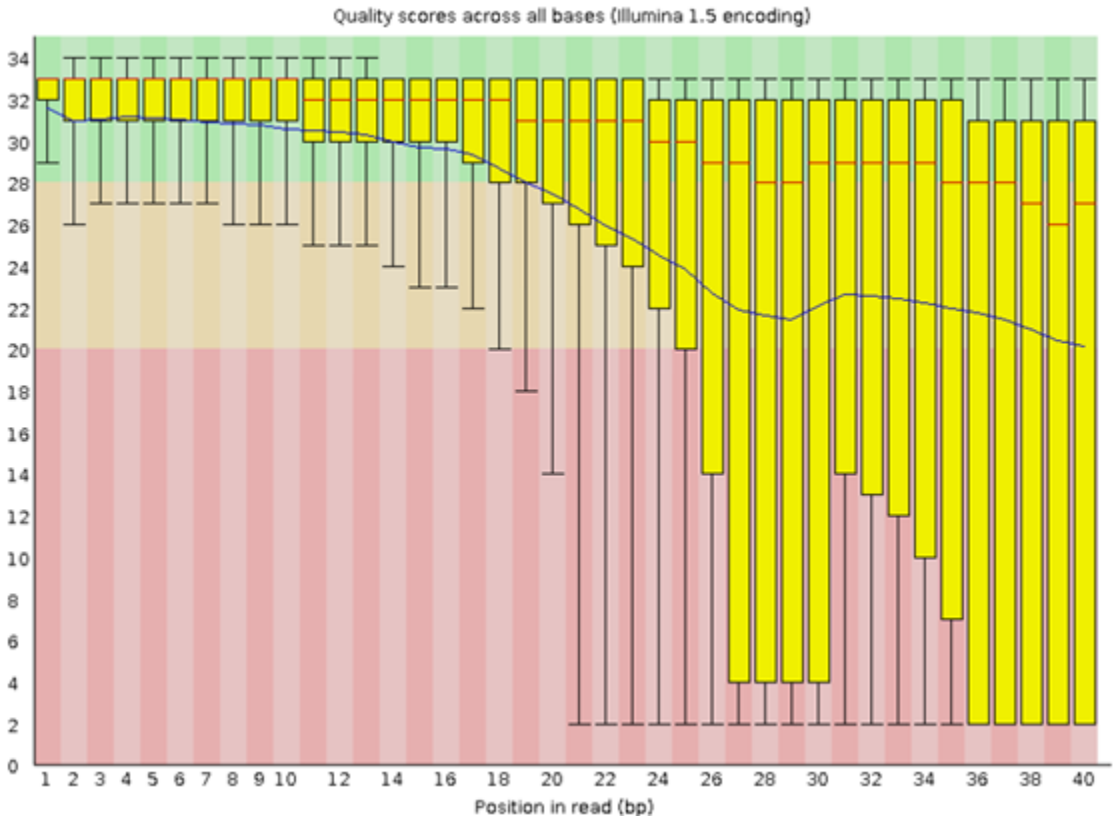
Objectives of the Present Diploma Thesis

- The first goal is the creation of simulated next generation sequencing data following a template with known mutations provided by biologists or other specialists. For that reason, we have developed an artificial data simulator written in R by using parallel programming techniques for the reduction of the execution time. The artificially created sequencing data

consists of 2 reads per “scan”, one from left to right (r1) and one from right to left (r2) and is characterized by pairs of bases and the corresponding probabilities. Therefore, the first step is the sampling of an amplicon in 2-way reads as mentioned, as many times as Mean Coverage and VAF indicate. We note that the 2-way reads must have at least 1 base overlap between them. The second step is to produce qualities which follow realistic distribution for each base that has been generated. The last step is to combine the bases and the corresponding qualities into a single fastq file for each of the 2-way reads. Given a set of known means and standard deviations of qualities per position of bases, here is the result:



As we can see from the above boxPlot the means and the sds of the qualities of our artificial data set follow a distribution of our choice. In comparison, we also provide a real Illumina NGS data boxPlot bellow:



To conclude, with this simulator we are able to construct almost any data set we want to, as long as the means and standard deviations of qualities per position in read are known to us.

- The second goal is an attempt to automate a GATK pipeline through a bash script, in order to have the opportunity to run this pipeline multiple times with different input parameters. More specifically, in this project we try to optimize one GATK's input parameter called "max-read-per-alignment-start". For more details about this parameter see GATK's manual.
- The third goal is to create an algorithm in R, so as to compare the mutations reported by GATK with the known template's mutations. In order to accomplish this, we are looking for changes in the mutation type (insertion, deletion, change), differences between reported coordinates and coordinates of the template's mutations aligned in the reference genome and lastly differences in the bases for each mutation. Only if a reported mutation passes all previous tests, is it classified as a true positive mutation.

In conclusion, firstly, we generate a realistic NGS data set with known mutations, secondly, we run a Variant Discovery pipeline as many times as needed with different input parameters and lastly, we evaluate the success of the corresponding execution through the number of true positive findings. In addition, we studied the way that the scores of the artificial

dataset affect the number of true positive mutations which have been found by the GATK.

Πίνακας περιεχομένων

Ευχαριστίες ... 3
Περίληψη ... 5
Abstract ... 6
Short Description of the Problem ... 7
Definition of Mutations ... 7
Searching of Somatic Mutations ... 7
Objectives of the Present Diploma Thesis ... 7
Πίνακας περιεχομένων ... 11
Πίνακας Εικόνων και Σχημάτων ... 14
Λίστα Πινάκων ... 14
1 Εισαγωγή ... 16
1.1 Εισαγωγή στη Βιοπληροφορική ... 16
1.1.1 Ορισμός και Συνοπτική Επεξήγηση ... 16
1.1.2 Σύντομη Ιστορική Αναδρομή ... 16
1.2 Συνοπτική Περιγραφή του Προβλήματος ... 18
1.2.1 Απλοϊκός Ορισμός – Σωματικές μεταλλάξεις (Somatic mutations) ... 18
1.2.2 Σημαντικότητα Μελέτης Σωματικών Μεταλλάξεων ... 18
1.2.3 Αναζήτηση Σωματικών Μεταλλάξεων ... 19
1.3 Στόχοι της Παρούσας Διπλωματικής ... 19
2 Θεωρητικό Υπόβαθρο ... 21
2.1 Μοριακή Βιολογία ... 21
2.1.1 DNA ... 21
2.1.2 Amplicon ... 21
2.1.3 Amplicon Sequencing ... 22
2.2 Computer Science ... 23
2.2.1 Parallel Computing ... 24
2.3 Χρήσιμα Μαθηματικά Θεωρήματα και Στατιστικές ... 26
2.3.1 Μέση Τιμή (Average) ... 26
2.3.2 Διακύμανση (Variance) ... 26
2.3.3 Τυπική Απόκλιση (Standard Deviation, SD) ... 26
2.3.4 Δημιουργία Τυχαίων Αριθμών ... 27
3 Περιγραφή Αλγορίθμου Προσομοίωσης ... 29
3.1 Εισαγωγή ... 29
3.2 Το πρόβλημα ... 29

3.3 Επεξήγηση Προτύπου Δεδομένων (Template) 29
3.4 Απαιτήσεις Τεχνητού Συνόλου Δεδομένων 31
3.4.1 FASTQ format 31
3.4.2 Απαιτήσεις Τεχνητού Συνόλου Δεδομένων 32
3.5 Απαιτήσεις Προγράμματος Προσομοιωτή 32
3.6 Περιγραφή Αλγορίθμου 33
3.6.1 Προ-επεξεργασία Δεδομένων 33
3.6.2 Δειγματοληψία Δεδομένων 34
3.6.2.1 Παραγωγή Δεδομένων 34
3.6.2.2 Παραγωγή ποιοτήτων 37
3.7 Time Improvements 41
3.8 Testing 44
4 Υλοποίηση Αλγοριθμικής Διαδικασίας 48
4.1 Ροή Εργασίας (pipeline) 48
4.1.1 Εισαγωγή 48
4.1.2 Script 49
4.1.3 Εκδόσεις Tools 51
4.1.4 Σχόλια και Παρατηρήσεις 51
4.1.5 Αυτοματοποίηση Εκτέλεσης 51
4.2 Επεξεργασία Αποτελεσμάτων 52
4.2.1 VcfComp.R 52
4.2.1.1 Συναρτήσεις 52
4.2.1.2 Κύριος Κώδικας 55
4.2.2 summary_of_iterations.R 59
4.2.2.1 Ανάλυση Κώδικα 59
4.2.3 Σύνοψη 60
4.3 Επεξεργασία Αρχείων SAM 60
4.3.1 Περιγραφή Αλγορίθμου sam_cmp.R 61
5 Αποτελέσματα 68
5.1 Εισαγωγή 68
5.2 Δεδομένα 68
5.3 Πειράματα 68
5.3.1 Constant quality 68
5.3.1.1 Constant Quality Q = 12 (ASCII 45) 68
5.3.1.2 Constant Quality Q = 20 (ASCII 53) 71
5.3.1.3 Constant Quality Q = 30 (ASCII 63) 74

5.3.2 Realistic Dataset 78
6 Συμπεράσματα 85
6.1 Γενικά Συμπεράσματα 85
6.2 Μελλοντικές Επεκτάσεις 86
7 Βιβλιογραφία 88
Παράρτημα Α 90

Πίνακας Εικόνων και Σχημάτων

Εικόνα 2-1 22
Εικόνα 2-2 23
Εικόνα 2-3 Γραφική παράσταση δημιουργίας τυχαίας τιμής x_1, από μια τυχαία τιμή της ομοιόμορφης στο διάστημα $[0,1]$ 28
Εικόνα 3-1 Per base qualities for real data - FastQC 38
Εικόνα 3-2: Κανονική κατανομή – Συνάρτηση πυκνότητας πιθανότητας 39
Εικόνα 3-3: Με μαύρο χρώμα απεικονίζεται η αθροιστική συνάρτηση πιθανότητας ενώ με κόκκινο η συνάρτηση μάζας πιθανότητας. Τα τυχαία ομοιόμορφα κατανεμημένα δείγματα κυμαίνονται από το 0-1 στον άξονα y (με μεγαλύτερη πιθανότα να βρίσκονται ανάμεσα στις πράσινες γραμμές) ενώ το τυχαίο αλλά κανονικά κατανεμημένο δείγμα κυμαίνεται από 33-75 μέσω της μαύρης καμπύλης (με μεγαλύτερη πιθανότα να βρίσκονται ανάμεσα στις μπλέ γραμμές). 40
Εικόνα 3-4 From left to right produced reads dataframe. 43
Εικόνα 3-5 Per Base Qualities for artificial data - FastQC 46
Εικόνα 3-6 Per Base Qualities for real data - FastQC 47
Εικόνα 4-1 Παράδειγμα τμήματος του γονιδιώματος αναφοράς στο οποίο έχει γίνει στοίχιση ενός read με 3 χαρακτήρες 'N' στην αρχή του και D κενό (gap) 7 θέσεων. 66
Εικόνα 4-2 Παράδειγμα τμήματος του γονιδιώματος αναφοράς στο οποίο έχει γίνει στοίχιση ενός read με 3 χαρακτήρες 'N' στην αρχή του και I insertion 6 θέσεων. 67
Εικόνα 5-1Template Overview 68
Εικόνα 5-2 FastQC - Per base sequence quality – Q = 12 experiment 69
Εικόνα 5-3 Frequencies of qualities – Q = 12 experiment MRPAS=100 70
Εικόνα 5-4 Frequencies of mutexed positions – Q = 12 experiment MRPAS=100 70
Εικόνα 5-5 Per base quality for used reads – Q = 12 experiment MRPAS=100 71
Εικόνα 5-6 FastQC - Per base sequence quality – Q = 20 experiment 72
Εικόνα 5-7 Frequencies of qualities – Q = 20 experiment MRPAS=100 73
Εικόνα 5-8 Frequencies of mutexed positions – Q = 20 experiment MRPAS=100 74
Εικόνα 5-9 Per base quality for used reads – Q = 20 experiment MRPAS=100 74
Εικόνα 5-10 FastQC - Per base sequence quality – Q = 30 experiment 75
Εικόνα 5-11 Frequencies of qualities – Q = 30 experiment MRPAS=50 77
Εικόνα 5-12 Frequencies of mutexed positions – Q = 30 experiment MRPAS=50 77
Εικόνα 5-13 Per base quality for used reads – Q = 30 experiment MRPAS=50 78
Εικόνα 5-14 1.2 FastQC - Per base sequence quality – realistic experiment 79
Εικόνα 5-15 Frequencies of qualities – realistic experiment MRPAS=50 80
Εικόνα 5-16 Frequencies of mutexed positions – realistic experiment MRPAS=50 81
Εικόνα 5-17 Per base quality for used reads – realistic experiment MRPAS=50 81
Εικόνα 5-18 Frequencies of qualities – realistic experiment MRPAS=100 83
Εικόνα 5-19 Frequencies of mutexed positions – realistic experiment MRPAS=100 83
Εικόνα 5-20 Per base quality for used reads – realistic experiment MRPAS=100 84

Λίστα Πινάκων

Πίνακας 3-1 ASCII_BASE=33 Illumina quality scores 32
Πίνακας 4-1 Επεξήγησης Indels Gatk vs Template format 54
Πίνακας 4-2 61

Πίνακας 5-1 FastQC Details – r1 Q = 12 experiment 69
Πίνακας 5-2 FastQC Details – r2 Q = 12 experiment 69
Πίνακας 5-3 Q = 12 Experiment MRPAS=100 True Positives 70
Πίνακας 5-4 FastQC Details – r1 Q = 20 experiment 71
Πίνακας 5-5 FastQC Details – r2 Q = 20 experiment 71
Πίνακας 5-6 Q = 20 Experiment MRPAS=100 True Positives 72
Πίνακας 5-7 FastQC Details – r1 Q = 30 experiment 74
Πίνακας 5-8 FastQC Details – r2 Q = 30 experiment 75
Πίνακας 5-9 Q = 30 Experiment MRPAS=50 True Positives 75
Πίνακας 5-10 FastQC Details – r1 realistic experiment 78
Πίνακας 5-11 FastQC Details – r2 realistic experiment 78
Πίνακας 5-12 Realistic Experiment MRPAS=50 True Positives 79
Πίνακας 5-13 Realistic Experiment MRPAS=100 True Positives 82
Πίνακας 8-1 90

1 Εισαγωγή

1.1 Εισαγωγή στη Βιοπληροφορική

1.1.1 Ορισμός και Συνοπτική Επεξήγηση

Βιοπληροφορική καλείται ένας διεπιστημονικός κλάδος, και παρόλο που δεν υπάρχει ένας κοινώς αποδεκτός ορισμός, θα μπορούσαμε να κάνουμε μια προσπάθεια ορισμού της ως ο επιστημονικός χώρος όπου η συνύπαρξη της βιολογίας με την πληροφορική, τη στατιστική και τα μαθηματικά εξερευνά νέους τρόπους για την επιστημονική προσέγγιση βιολογικών προβλημάτων, καθώς και την κατά δύναμίν καλύτερη κατανόηση των βασικών αρχών της βιολογίας. Συνεπώς, πρόκειται για ένα γνωστικό χώρο με συγκεκριμένο όσο και ευρύ πεδίο εφαρμογών και αλληλεπίδρασης με τη σύγχρονη δομική, μοριακή πληθυσμιακή και περιβαλλοντική βιολογία. Ο επιστημονικός τομέας της βιοπληροφορικής σήμερα κατατάσσεται, σε παγκόσμιο επίπεδο, ως ένας από τους πλέον αναπτυσσόμενους, αφού έχει ήδη επιδείξει σημαντικά επιτεύγματα και έχει καταλάβει ιδιαίτερο τμήμα της προσοχής του επιστημονικού και επενδυτικού κόσμου. Στο σημείο αυτό αξίζει να αναφερθεί ότι, καθώς η βιοπληροφορική είναι διεπιστημονικός κλάδος, υπάρχουν πολλές και αντικρουόμενες απόψεις σχετικά με τον ορισμό της και με το επιστημονικό της περιεχόμενο. Η απλούστερη προσέγγιση υποστηρίζει ότι η βιοπληροφορική είναι η εφαρμογή κάποιων τεχνολογιών (μαθηματικών, αλγορίθμων, υπολογιστών κ.ο.κ) σε βιολογικά προβλήματα. Η πιο σύνθετη προσέγγιση υποστηρίζει ότι η βιοπληροφορική όχι μόνο εφαρμόζει τεχνολογίες, μεθόδους και άλλες θετικές επιστήμες σε βιολογικά προβλήματα, αλλά προσπαθεί να εδραιώσει και γενικότερους νόμους λειτουργίας αυτών των βιολογικών συστημάτων.

1.1.2 Σύντομη Ιστορική Αναδρομή

Προσπερνώντας τις προσπάθειες μαθηματικοποίησης της γενετικής (που γέννησε τη γενετική πληθυσμών και τη σύγχρονη εξελικτική θεωρία), από την εποχή των Hardy, Weinberg και Fisher, Wright κλπ, ανιχνεύουμε τις απαρχές της υπολογιστικής βιολογίας, στις αρχές της ίδιας της σύγχρονης μοριακής βιολογίας από τη δεκαετία του 1950 και 1960. Αναφέρω συνοπτικά μερικά από τις μεγαλύτερες ανακαλύψεις του χώρου:

- Τα πειράματα του Chargaff έφεραν στο φώς ότι το ποσοστό Αδενίνης είναι το ίδιο με το ποσοστό της Θυμίνης και το ποσοστό Γουανίνης ίσο με αυτό της Κυτοσίνης σε κάθε μόριο DNA. Αυτές ήταν οι πρώτες ενδείξεις για κάποια μορφή ψηφιακής πληροφορίας στις βιολογικές αλληλουχίες.
- Τα παραπάνω αποτελέσματα, στη συνέχεια, χρησιμοποιήθηκαν από τους Watson και Crick για να μπορέσουν να προσδιορίσουν την τρισδιάστατη δομή του DNA.
- Τη δεκαετία του 60 έγιναν επίσης και οι πρωτοποριακές μελέτες των Jacob και Monod στη γονιδιακή ρύθμιση.
- Σχετικά με τις πρωτεΐνες μετά τον προσδιορισμό των πρώτων τρισδιάστατων δομών (ινσουλίνη και μυογλοβίνη), ακολούθησαν τη δεκαετία του 1960 η ταυτοποίηση και η μελέτη μιας σειράς παρόμοιων δομών (λυσοζύμη, παπαΐνη, ριβονουκλεάση κ.ο.κ.), προκαλώντας έκρηξη στη μελέτη της δόμησης και λειτουργίας των πρωτεϊνών, σε ατομικό επίπεδο.

Επίσης δύο από τις σημαντικότερες βιολογικές ανακαλύψεις ήταν η εύρεση της πρωτοταγούς δομής των πρωτεϊνών το 1951, και του RNA το 1967. Όπως βλέπουμε πολλά

από τα προβλήματα που απασχολούν την επιστήμη της βιοπληροφορικής σήμερα προέρχονται από την διεύρυνση του ορίζοντα της βιολογίας τη δεκαετία του 1960.

Τη δεκαετία του 1980 το πεδίο της υπολογιστικής βιολογίας πήρε πλέον μια πιο ξεκάθαρη μορφή. Η μαζική εμφάνιση δημοσιεύσεων σε υψηλού κύρους επιστημονικά περιοδικά με άμεση διατύπωση βιολογικών προβλημάτων αλλά και παρουσίαση σημαντικών επιτευγμάτων, παράλληλα με την πλέον αυξανόμενα ραγδαία χρήση υπολογιστικών συστημάτων οδήγησαν τη δεκαετία του 80 στη ραγδαία αύξηση του όγκου των δεδομένων και στη δημιουργία μεγαλύτερων και πιο οργανωμένων βάσεων βιολογικών δεδομένων. Το 1986 γεννήθηκαν οι δύο πιο γνωστές μέχρι σήμερα βάσεις δεδομένων νουκλεοτιδικών αλληλουχιών (GenBank και EMBL Data Library). Στη συνέχεια όλο και περισσότερες βάσεις δεδομένων έκαναν την εμφάνισή, ενώ ταυτόχρονα προτάθηκε η δημιουργία δικτύων που θα διευκόλυναν την υπολογιστική έρευνα στη βιολογία (EMBnet και BIONET).

Η δεκαετία του 1990 ήταν η δεκαετία κατά την οποία η έρευνα στην υπολογιστική βιολογία εκτινάχθηκε. Φυσικά, για άλλη μια φορά δεν πρέπει να αμελήσουμε να αναφέρουμε ότι η δεκαετία αυτή σημαδεύτηκε επίσης από την ανάπτυξη του διαδικτύου και του παγκοσμίου ιστού, αλλά και από την εξάπλωση των προσωπικών Η/Υ. Πρέπει να σημειώσουμε επίσης, ότι η ευρεία χρήση του όρου «βιοπληροφορική» συντελέστηκε μέσα στη δεκαετία του 1990. Η δεκαετία αυτή σημαδεύτηκε από μια σειρά μεγάλων ανακαλύψεων. Στον τομέα της στοίχισης αλληλουχιών, η πιο σημαντική ίσως δημοσίευση όλων των εποχών στο χώρο, αφορά το BLAST (Basic Local Alignment Search Tool), από επιστήμονες του NCBI το 1990. Το BLAST «πάτησε» πάνω στις ανακαλύψεις για τη στατιστική κατανομή του σκορ (score) της τοπικής στοίχισης (το γνωστό θεώρημα Karlin-Altschul) και πραγματικά ήταν μια επαναστατική συμβολή στον τρόπο που θα διεξάγεται από κει και πέρα η αναζήτηση ομοιότητας σε βάσεις δεδομένων και η στοίχιση, καθώς ήταν πιο γρήγορο από κάθε άλλο αλγόριθμο επιτρέποντας ταχείες αναζητήσεις, αλλά έδινε και για πρώτη φορά μια εκτίμηση για τη στατιστική σημαντικότητα των στοιχίσεων. Στην πρώτη του έκδοση δεν παρήγαγε στοιχίσεις με κενά, αλλά στη δεύτερη, παρείχε και αυτή τη δυνατότητα, ενώ περιλάμβανε και άλλες παραλλαγές όπως το PSI-BLAST.

Η σύγχρονη εποχή μετά το 2000 δεικτοδότησε την ώριμη φάση της υπολογιστικής βιολογίας. Η διδασκαλία της βιοπληροφορικής πλέον γίνεται συστηματική και πανεπιστήμια, ιδρύονται και αναπτύσσονται επιστημονικές και επαγγελματικές οργανώσεις και κυκλοφορούν όλο και περισσότερα επιστημονικά περιοδικά προσανατολισμένα σε αυτόν τον κλάδο. Ο μεγάλος ρυθμός προσδιορισμού των γονιδιωμάτων, οδήγησε στην αλματώδη ανάπτυξη της γονιδιωματικής στις διάφορες μορφές της. Παράλληλα, εμφανίστηκαν οι τεχνικές αλληλούχισης νέας γενιάς (Next Generation Sequencing), οι οποίες έδωσαν νέα πνοή σε μελέτες γονιδιακής έκφρασης (RNAseq), έκαναν εύκολο τον εντοπισμό πολυμορφικών θέσεων και αναμένεται να επηρεάσουν και την προσωποποιημένη ιατρική. Με όλα αυτά τα δεδομένα δημιουργήθηκε μια μεγάλη ανάγκη για περισσότερους αλγόριθμους πρόγνωσης έτσι ώστε να τεθεί ‘σε τάξη’ ο τεράστιος αυτός όγκος δεδομένων, αλλά και μια μεγάλη ανάγκη για δημιουργία εξειδικευμένων βάσεων δεδομένων και οντολογιών που να τις περιγράφουν. Την εποχή αυτή, είδαμε την έκρηξη των διαδικτυακών εφαρμογών (web-servers), αλλά και του ελεύθερου λογισμικού βιοπληροφορικής. Επιπλέον, τα προγράμματα μετα-γονδιωματικής (meta-genomics) έθεσαν νέα αλγοριθμικά προβλήματα στην εύρεση γονιδίων και την ομαδοποίηση δεδομένων [1].

Σήμερα η βιοπληροφορική έχει γίνει αναπόσπαστο κομμάτι πολλών πεδίων της βιολογίας. Ο προγραμματισμός είναι μέρος της μεθοδολογίας, και αναφέρεται σε ειδικές

διοχετεύσεις (pipelines) ανάλυσης που χρησιμοποιούνται συχνά, ειδικά στο πεδίο της γενομικής. Συχνές χρήσεις της βιοπληροφορικής περιλαμβάνουν την αναγνώριση υποψήφιων γονιδίων και μονονουκλεοτιδικών πολυμορφισμών (SNPs). Συχνά αυτή η αναγνώριση γίνεται με σκοπό την κατανόηση της γενετικής βάσης μιας ασθένειας, μοναδικών προσαρμογών, επιθυμητών ιδιοτήτων (ειδικά στα γεωργικά είδη) ή τις διαφορές μεταξύ πληθυσμών.

1.2 Συνοπτική Περιγραφή του Προβλήματος

Όπως αναφέρθηκε εκτενώς παραπάνω, σκοπός της της βιοπληροφορικής είναι, εκτός των άλλων, και η χρησιμοποίηση υπολογιστικών – μαθηματικών μεθόδων για την εξαγωγή συμπερασμάτων, από τον τεράστιο όγκο δεδομένων που προκύπτουν, μέσω της χρησιμοποίησης σύγχρονων εργαστηριακών μεθόδων για την δειγματοληψία, απομόνωση και επεξεργασία δειγμάτων γενετικού υλικού. Αυτή η πρόκληση απαιτεί την ανάπτυξη υπολογιστικών εργαλείων (software), και μεθόδων για την προ επεξεργασία, την ανάλυση και την εξαγωγή χρήσιμων, τελικά, συμπερασμάτων από τη γενετική πληροφορία. Τα εργαλεία αυτά πρέπει να είναι ικανά να εξερευνούν και να ανακαλύπτουν τάσεις, να αναγνωρίζουν πρότυπα, να δημιουργούν μοντέλα πρόβλεψης και τελικά να εκδίδουν προβλέψεις για τις εκβάσεις των πειραμάτων. Εξαιτίας της αυξανόμενης πολυπλοκότητας των προβλημάτων τα προαναφερθέντα εργαλεία ή τουλάχιστον μερικά από αυτά, τείνουν καθημερινά να γίνονται όλο και πιο περίπλοκα, τόσο στη συγγραφή τους όσο και στη χρησιμοποίηση και λειτουργία τους από τον χρήστη (βιοπληροφορικό/βιολόγο). Κρίνεται λοιπόν απαραίτητο να τονίσουμε σε αυτό το σημείο ότι, υπάρχει μεγάλη αναγκαιότητα όχι μόνο για την δημιουργία αποδοτικών εργαλείων αλλά και για την επιλογή των βέλτιστων παραμέτρων στο εκάστοτε πρόβλημα. Από εδώ και στο εξής θα αναφερόμαστε σε αυτό το πρόβλημα ως «Πρόβλημα βελτιστοποίησης παραμέτρων».

Η εύρεση, ταυτοποίηση και συστηματική καταγραφή σωματικών μεταλλάξεων κρίνεται εξαιρετικά σημαντική. Παρόλο που σαφείς και εμπεριστατωμένοι ορισμοί θα δοθούν στο κατάλληλο κεφάλαιο, θα αφιερώσουμε μερικές γραμμές παραθέτοντας έναν απλοϊκό ορισμό των «Σωματικών Μεταλλάξεων», ενώ στη συνέχεια θα αναφέρουμε μερικούς από τους λόγους για τους οποίους η εύρεση αυτών των μεταλλάξεων είναι καίριας σημασίας.

1.2.1 Απλοϊκός Ορισμός – Σωματικές μεταλλάξεις (Somatic mutations)

Στη βιολογία, με τον όρο **μετάλλαξη, μεταλλαγή (mutation)** [2], χαρακτηρίζεται οποιαδήποτε αλλαγή στην αλληλουχία των νουκλεοτιδίων, στο γονιδίωμα ενός οργανισμού, ιού ή εξωχρωμοσωμικού DNA. Ιογενή γονιδιώματα μπορούν να είναι είτε DNA είτε RNA. Στους πολυκύτταρους ευκαρυωτικούς οργανισμούς, αν η μεταβολή προσβάλλει κύτταρα γαμετών, χαρακτηρίζεται γενετική μεταλλαγή και μπορεί να κληρονομηθεί. Αντίθετα, αν προσβληθούν σωματικά κύτταρα (μη φυλετικά), η μεταλλαγή αυτή ονομάζεται **σωματική μεταλλαγή** η οποία δεν κληρονομείται. Οι μεταλλάξεις που έχουν ως αποτέλεσμα αλλαγές γονιδίων (γονιδιακές μεταλλάξεις), είναι αλλαγές στην αλληλουχία βάσεων μόνο σε ένα γονίδιο παράγοντας ένα διαφορετικό αλληλόμορφο.

1.2.2 Σημαντικότητα Μελέτης Σωματικών Μεταλλάξεων

Ο προσδιορισμός σωματικών μεταλλάξεων κρίνεται εξαιρετικά σημαντικός αφού αφορά μεταλλάξεις που συμβαίνουν κατά τη διάρκεια της ζωής του ανθρώπου (επίκτητες) και είτε επηρεάζουν τη ζωή του άμεσα ή έμμεσα ή δεν την επηρεάζουν καθόλου.

Παραδείγματα σωματικών μεταλλάξεων, περιλαμβάνουν αλλαγές στο γεννητικό υλικό σε ένα σωματικό κύτταρο, μεταβάλλοντας το σε καρκινικό.

1.2.3 Αναζήτηση Σωματικών Μεταλλάξεων

Όπως αναφέραμε παραπάνω η εύρεση σωματικών μεταλλάξεων είναι ιδιαίτερα σημαντική. Για αυτόν τον λόγο έχουν ήδη αναπτυχθεί, αρκετά αποτελεσματικά, εργαλεία (software) αναζήτησης μεταλλάξεων παντός τύπου και φυσικά σωματικών. Αυτά τα εργαλεία έχουν αναπτυχθεί σε βαθμό, που πλέον αρκετά περίπλοκα πιθανοτικά μοντέλα χρησιμοποιούνται προκειμένου να γίνει εύρεση σωματικών μεταλλάξεων σε στοιχισμένα 'κομμάτια' γεννητικού υλικού στο γονιδίωμα αναφοράς πχ. του ανθρώπου. Εξαιτίας της αυξημένης πολυπλοκότητας των μοντέλων αυτών κρίνεται πλέον απαραίτητη η επιλογή πληθώρα παραμέτρων (ρυθμίσεων) του εκάστοτε εργαλείου, από τον βιοπληροφορικό, είτε μέσω εμπειρικών μεθόδων, είτε μέσω κάποιας πειραματικής παραμετρικής ανάλυσης. Στην παρούσα διπλωματική θα παρουσιάσουμε μια αρκετά εμπεριστατωμένη προσπάθεια συστηματοποίησης του προβλήματος βελτιστοποίησης παραμέτρων.

1.3 Στόχοι της Παρούσας Διπλωματικής

Όπως αναφέρθηκε παραπάνω βασικός στόχος της παρούσας διπλωματικής είναι η συστηματοποίηση του προβλήματος βελτιστοποίησης παραμέτρων σε εργαλεία εύρεσης σωματικών μεταλλάξεων, δηλαδή, τη δημιουργία ενός συνόλου εργαλείων (αλγορίθμων) αυτοματοποίησης της επαναληπτικής εκτέλεσης εργαλείων με διαφορετικές παραμέτρους εκτέλεσης, για το ίδιο σύνολο δεδομένων εισόδου και τέλος την σύγκριση των αποτελεσμάτων των διαφορετικών εκτελέσεων, την αξιολόγηση τους και την επιλογή των βέλτιστων παραμέτρων. Θα μπορούσε κάποιος να πει ότι πρόκειται για μια παραμετρική ανάλυση ενός αλγορίθμου και επιλογή παραμέτρων με βάση την ελαχιστοποίηση κάποιας μετρικής.

Από τα παραπάνω προκύπτει ότι, δυστυχώς, δεν μπορούμε να βασιστούμε ακριβώς σε αληθινά δεδομένα για διάφορους λόγους που θα αναφερθούν στη συνέχεια, εκ των οποίων ο βασικότερος είναι ότι για να γίνει τελικά η αξιολόγηση της αποτελεσματικότητας ενός εργαλείου είναι, προφανώς, αναγκαίο να γνωρίζουμε εκ των προτέρων τις μεταλλάξεις αυτές. Με άλλα λόγια αντιμετωπίζουμε στην παραμετρική μας ανάλυση, το πρόβλημα, με γνωστή είσοδο.

Συνεπώς, πρώτος στόχος της παρούσας εργασίας είναι η δημιουργία ενός προσομοιωτή, ο οποίος θα παράγει ένα τεχνητό σύνολο δεδομένων, ακολουθώντας ένα πρότυπο το οποίο φυσικά θα συμπεριλαμβάνει τις γνωστές σε εμάς μεταλλάξεις. Το κύριο μέλημα που τέθηκε από την αρχή είναι ο προσομοιωτής να παράγει δεδομένα τα οποία να αντιπροσωπεύουν, όσο το δυνατόν καλύτερα πραγματικά δεδομένα, με την επιλογή των παραμέτρων να αφήνεται στον βιολόγο.

Για τον σκοπό αυτόν προκύπτει η ανάγκη, δεύτερος στόχος, της δημιουργίας ενός γραφικού περιβάλλοντος στο οποίο ο βιολόγος θα έχει τη δυνατότητα να επιλέξει τις παραμέτρους που επιθυμεί, να εμφανίσει στην οθόνη γραφήματα των συνολικών παραμέτρων που επέλεξε και πολλά άλλα.

Αφού δημιουργηθούν τα τεχνητά δεδομένα, ως τρίτος στόχος ορίστηκε η αυτοματοποίηση της ροής του συνόλου των προγραμμάτων προ επεξεργασίας των δεδομένων καθώς και του εργαλείου που επιλέχθηκε για την εύρεση των σωματικών μεταλλάξεων. Με την ολοκλήρωση της επαναληπτικής εκτέλεσης του εργαλείου εύρεσης

μεταλλάξεων, προκύπτουν τα αρχεία εξόδου (vcf) files στα οποία είναι καταγεγραμμένες οι μεταλλάξεις που βρέθηκαν επιτυχώς, καθώς και πολλές άλλες παράμετροι.

Ως τέταρτος στόχος, λοιπόν, τέθηκε η δημιουργία ενός R script το οποίο θα αναλαμβάνει να γνωστοποιήσει αν οι μεταλλάξεις που βρέθηκαν για κάθε επαναληπτική εκτέλεση είναι True Positive, True Negative, False Positive, False Negative, σε ποια κατηγορία ανήκουν κτλ., με βάση πάντα το αρχικό πρωτότυπο (template) από το οποίο δημιουργήθηκε εξ' αρχής το τεχνητό σύνολο δεδομένων (γνωστή είσοδος). Το script αυτό έχει τελικά σκοπό να γνωστοποιήσει και να καταγράψει όλα τα παραπάνω χαρακτηριστικά σε έναν ενιαίο πίνακα, ο οποίος είναι τελικά η έξοδος του προγράμματος και εγγράφεται σε αρχείο. Υπενθυμίζω ότι για κάθε διαφορετική εκτέλεση του εργαλείου εύρεσης μεταλλάξεων, με διαφορετικές παραμέτρους, τελικά προκύπτει από το script του τέταρτου στόχου και ένας διαφορετικός πίνακας μεταλλάξεων. Συνεχίζοντας όλοι αυτοί οι πίνακες πρέπει να φορτωθούν και να αναλυθούν εκ νέου από ένα ξεχωριστό πρόγραμμα του οποίου η συγγραφή συμπεριλαμβάνεται για λόγους απλότητας στον τέταρτο στόχο. Η ανάλυση αυτών των πινάκων περιλαμβάνει, πρακτικά, τη σύγκρισή τους και την εξαγωγή του αποτελέσματος: από ποιες παραμέτρους εκτέλεσης προέκυψε ο «καλύτερος» πίνακας. Με τον όρο καλύτερος εννοούμε την αποτελεσματικότερη εύρεση μεταλλάξεων.

Κάπου εδώ ολοκληρώνεται η διαδικασία αξιολόγησης και βελτιστοποίησης παραμέτρων που θέσαμε ως στόχο από την αρχή.

Αξίζει όμως να αναφερθεί ότι κατά τη διάρκεια του πειραματισμού προέκυψε και η ανάγκη να μελετήσουμε πως τα εργαλεία εύρεσης μεταλλάξεων επηρεάζονται από την ποιότητα των τεχνητών δεδομένων. Στο σημείο αυτό ίσως αναφερθούμε σε όρους οι οποίοι δεν γίνουν πλήρως κατανοητοί. Θα τους ξεκαθαρίσουμε, όμως, στο κατάλληλο κεφάλαιο. Το τεχνητό σύνολο δεδομένων δεν περιέχει τίποτα άλλο πέρα από αλληλουχίες (κομμάτια) DNA. Για κάθε βάση που παράγεται σύμφωνα με το πρότυπο, παράγεται και μία ποιότητα η οποία αντιπροσωπεύει το ποσοστό σφάλματος που είχε η συγκεκριμένη βάση κατά την ανάγνωση της. Η ποιότητα αυτή ποικίλει και ενδέχεται για παράδειγμα να είναι χαμηλή λόγο θορύβου, λανθασμένου διαβάσματος λόγω ευθυγράμμισης κτλ. Ο προσομοιωτής που κατασκευάσαμε στο πρώτο βήμα, παράγει ζευγάρια βάσεις-ποιότητες οι οποίες όμως ακολουθούν μια συγκεκριμένη κατανομή που επιλέγεται από τον βιολόγο. Το αν τελικά η ποιότητα των βάσεων μια μετάλλαξης επηρεάζει την ικανότητα εύρεσης τους από το εξειδικευμένο εργαλείο, είναι αντικείμενο μελέτης του πέμπτου σταδίου – στόχου αυτής της διπλωματικής.

2 Θεωρητικό Υπόβαθρο

Στη συνέχεια του παρόντος εγγράφου πολύ συχνά θα αναφερόμαστε σε όρους μοριακής βιολογίας, επιστήμης υπολογιστών και ανάλυσης και επεξεργασίας δεδομένων, οι οποίοι, προκυμμένου να γίνονται κατανοητοί είναι απαραίτητο να αφιερώσουμε ένα ολόκληρο κεφάλαιο παραθέτοντας κυρίως τους ορισμούς τους, καθώς και, όπου κρίνεται απαραίτητο, επεξηγηματικές λεπτομέρειες.

2.1 Μοριακή Βιολογία

2.1.1 DNA

Η ανακάλυψη της δομής του DNA πραγματοποιήθηκε το 1953 από τους Τζέιμς Γουάτσον (James D. Watson) και Φράνσις Κρικ (Francis Crick). Από πολλούς η ανακάλυψη της διπλής έλικας του DNA θεωρείται ως η μεγαλύτερη βιολογική ανακάλυψη του 20ού αιώνα. Για τη συνεισφορά τους στη μελέτη της δομής του DNA, οι Γουάτσον και Κρικ μοιράστηκαν το 1962 το Βραβείο Νόμπελ με τον Μόρις Γουίλκινς και την Ρόζαλιν Φράνκλιν, οι οποίοι εργάστηκαν προς την ίδια κατεύθυνση.

Ορισμός:

'Το δε(σ)οξυριβο(ζο)νουκλεϊ(νι)κό οξύ (Deoxyribonucleic Acid, DNA) [3] ***είναι νουκλεϊκό οξύ που περιέχει τις γενετικές πληροφορίες που καθορίζουν τη βιολογική ανάπτυξη όλων των κυτταρικών μορφών ζωής και των περισσοτέρων ιών.'***

Το DNA συνήθως έχει τη μορφή διπλής έλικας, αλλά όταν δεν πολλαπλασιάζεται πακετάρεται ως υπερρέλικα (με επιπλέον ελίκωση σε ανώτερη τάξη μεγέθους με την βοήθεια ιστονών και άλλων μορίων).

Πρόκειται για μεγαλομοριακή ένωση που συγκροτείται από αζωτούχες-πρωτεϊνικές βάσεις, φωσφορικές ρίζες και ένα σάκχαρο με πέντε άτομα άνθρακα (πεντόζη), την δε(σ)οξυριβόζη. Στα ευκαρυωτικά κύτταρα ανιχνεύεται κυρίως μέσα στον πυρήνα του κυττάρου αλλά και σε μερικά άλλα οργανίδια, όπως τα μιτοχόνδρια και τα πλαστίδια, επιτρέποντάς τους να αναπαράγονται αυτόνομα (ημιαυτόνομα οργανίδια).

Η διαμόρφωση των μεγάλων μορίων του DNA στο χώρο έχει τη μορφή δύο επιμηκών αλύσεων, οι οποίες συστρέφονται ελικοειδώς μεταξύ τους. Οι αζωτούχες βάσεις στο DNA είναι τέσσερις:

- Κυτοσίνη C
- Γουανίνη G
- Θυμίνη T
- Αδενίνη A

Οι αζωτούχες βάσεις, ανάλογα με την σειρά αλληλουχίας τους σε τριάδες, κωδικοποιούν το μήνυμα για τη σύνθεση των αμινοξέων του κυττάρου στα ριβοσώματα. Εκεί τα αμινοξέα συνδυάζονται, με τη σειρά κατά την οποία μεταφέρθηκαν στο ριβόσωμα και συντίθενται έτσι οι διαφορετικές πρωτεΐνες.

2.1.2 Amplicon

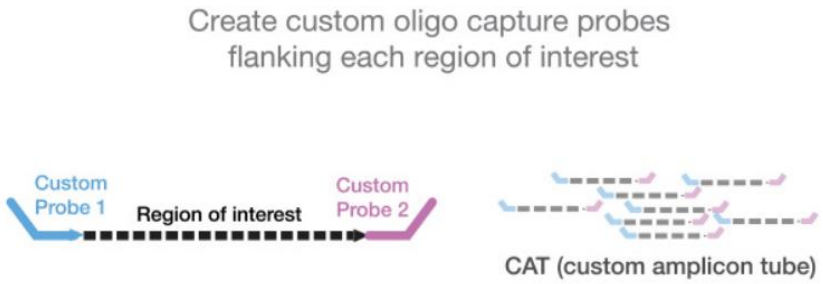
Ορισμός:

'Στη μοριακή βιολογία, Amplicon [4] ***ονομάζεται ένα κομμάτι DNA ή RNA το οποίο είναι η πηγή και/ή το προϊόν διαδικασιών διπλασιασμού ή πολλαπλασιασμού.'***

Μπορεί να διαμορφωθεί τεχνητά χρησιμοποιώντας ποικίλες μεθόδους συμπεριλαμβανομένου αλυσιδωτή αντίδραση πολυμεράσης (PCR) ή αλυσιδωτή αντίδραση λιγάσης (LCR). Με λίγα λόγια οι παραπάνω μέθοδοι χρησιμοποιούνται στη μοριακή βιολογία για την ταχεία δημιουργία εκατομμυρίων ή και δισεκατομμυρίων αντιγράφων ενός συγκεκριμένου κομματιού DNA ή RNA. Ορισμοί για τις παραπάνω μεθόδους δεν θα δοθούν διότι δεν είναι αναγκαίο να εμβαθύνουμε τόσο και ξεφεύγουν από τους στόχους αυτού του εγγράφου.

2.1.3 Amplicon Sequencing

Amplicon Sequencing [5] είναι μία αρκετά στοχευμένη προσέγγιση που επιτρέπει στους ερευνητές να αναλύουν τη γενετική ποικιλομορφία σε συγκεκριμένες γονιδιωματικές περιοχές. Το εξαιρετικά βαθύ sequencing των προϊόντων που προκύπτουν από την PCR (amplicons) επιτρέπει την αποδοτική ταυτοποίηση και χαρακτηρισμό παραλλαγών. Αυτή η μέθοδος χρησιμοποιεί ολιγονουκλεοτίδια (μικρά κομμάτια DNA) σχεδιασμένα για στόχευση και σύλληψη περιοχών ενδιαφέροντος.



Εικόνα 2-1

Η μέθοδος αυτή είναι εξαιρετικά χρήσιμη για την εξερεύνηση **σπάνιων** σωματικών μεταλλάξεων σε περίπλοκα δείγματα όπως για παράδειγμα όγκους αναμεμιγμένους με DNA βλαστοκυττάρων. Μια άλλη κοινή εφαρμογή είναι η αλληλούχιση του βακτηριακού γονιδίου 16S rRNA σε πολλαπλά είδη, μια ευρέως χρησιμοποιούμενη μέθοδος φυλογενότητας και ταξινομίας (phylogeny and taxonomy studies), ιδιαίτερα σε διαφορετικά δείγματα μεταγονιδιωματικών (metagenomics).

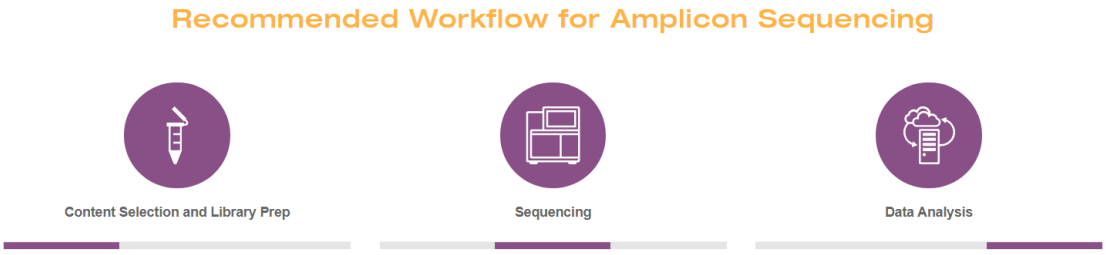
Πλεονεκτήματα της μεθόδου

- Επιτρέπει στους ερευνητές να ανακαλύπτουν αποτελεσματικά, να επικυρώνουν και να ελέγχουν γενετικές παραλλαγές χρησιμοποιώντας μια ιδιαίτερα στοχευμένη προσέγγιση.
- Υποστηρίζει την σύμπλεξη εκατοντάδων έως χιλιάδων Amplicons ανά αντίδραση προκυμμένου να επιτευχθεί υψηλή κάλυψη (Coverage).
- Παρέχει αλληλούχιση υψηλής κάλυψης ακόμη και σε περιοχές δύσκολες να δειγματοληπτηθούν (να γίνουν sequencing), όπως περιοχές πλούσιες σε GC ζεύγη.
- Επιτρέπει την ευελιξία για ένα μεγάλο εύρος πειραματικών σχεδίων.
- Μειώνει το κόστος της αλληλούχισης και τον χρόνο ολοκλήρωσης σε σύγκριση με ευρύτερες προσεγγίσεις όπως η αλληλουχία ολόκληρου του γονιδιώματος.

Η παραπάνω μέθοδος αλληλούχιση επιτρέπει στους ερευνητές την αλληλούχιση περιοχών που κυμαίνονται από μερικά έως εκατοντάδες γονίδια σε μία μόνο εκτέλεση. Επιπλέον ένα βασικό χαρακτηριστικό της μεθόδου είναι ότι έχει την ικανότητα να επιταχύνει την έρευνα αξιολογώντας ταυτόχρονα πολλαπλά γονίδια. Τα δείγματα - βιβλιοθήκες μπορούν να προετοιμαστούν σε μόλις 5-7,5 ώρες και να αλληλουχηθούν σε 17-32 ώρες.

Τέλος αναφέρουμε συνοπτικά ότι τα τρία κύρια βήματα της διαδικασίας είναι:

1. Επιλογή των δειγμάτων και προετοιμασία
2. Αλληλούχιση
3. Ανάλυση δεδομένων



Εικόνα 2-2

2.2 Computer Science

Η αλληλούχιση γονιδίων (genome sequencing) [6] αποτελεί στις μέρες μας ένα αναπόσπαστο και πολυχρησιμοποιούμενο εργαλείο στους τομείς έρευνας της βιολογίας και της βιοϊατρικής. Τα τελευταία χρόνια η ικανότητα παραγωγής δεδομένων αυτών των υπερσύγχρονων μεθόδων έχει αυξηθεί δραματικά, και δεν θα ήταν υπερβολή να πούμε ότι αυξάνεται με εκθετικούς ρυθμούς ακολουθώντας το νόμο του Moore, ενώ παράλληλα το κόστος μειώνεται ραγδαία. Για παράδειγμα η τελευταία τεχνολογία της Illumina, HiSeq 2500, έχει την ικανότητα να παράγει 600 εκατομμύρια σειρές 300bp σε μήκος (2 x 150bp, 180GB όγκος δεδομένων) σε 40 ώρες. Η ταχύτατη αυτή αφομοίωση της πληροφορίας του γενετικού κώδικα αντιπροσωπεύει μία πλούσια πηγή για την εκτεταμένη επέκταση της γνώσης μας σχετικά με τα βιολογικά ζητήματα, αλλά ταυτόχρονα αποτελεί σοβαρή πρόκληση για τους αναλυτές δεδομένων, αφού απαιτούνται νέοι αλγόριθμοι και προγράμματα ικανά να ανταπεξέλθουν στις τεράστιες απαιτήσεις επεξεργαστικής ισχύς που χρειάζονται για την εξαγωγή χρήσιμων αποτελεσμάτων από αυτά τα δεδομένα.

Από τα παραπάνω γίνεται εμφανής η αναγκαιότητα για νέες μεθόδους, εργαλεία και αποδοτικούς τρόπους εκμετάλλευσης της ‘σημερινής’ υπολογιστικής ισχύς που είναι διαθέσιμη. Από ένα γιγαντιαίων διαστάσεων cluster μέχρι έναν προσωπικό υπολογιστή μετρίων επιδόσεων. Η χρησιμοποίηση βελτιστοποιημένων και καλά σχεδιασμένων αλγορίθμων θα πρέπει να είναι ο πρωταρχικός στόχος ενός προγραμματιστή ο οποίος επιχειρεί να δημιουργήσει ένα νέο εργαλείο ή βιβλιοθήκη, αφού αποτελεί το κλειδί για την επίτευξη μέγιστης αποδοτικότητας. Παρόλα αυτά η ραγδαία αύξηση δεδομένων κάνει την εκτέλεση, ακόμη και των πιο βελτιστοποιημένων αλγορίθμων πχ αυτών με χρονική πολυπλοκότητα γραμμική ή τετραγωνική, να μοιάζει απελπιστικά αργή. Στη συνέχεια θα αναφέρουμε επιγραμματικά και κατόπιν θα αναλύσουμε λεπτομερώς κάποιους από τους λόγους, πέρα των τεράστιων συνόλων δεδομένων, για τους οποίους και οι αποδοτικότεροι αλγόριθμοι αποτυγχάνουν να εκτελεστούν σε σύντομο χρονικό διάστημα.

1. Κακή συγγραφή κώδικα.

Τα υπολογιστικά συστήματα στις μέρες μας καθίστανται εξαιρετικά πολύπλοκα. Η χρησιμοποίηση ακόμη και χαμηλού επιπέδου γλώσσες όπως C μπορεί να βοηθήσει στη συγγραφή αποδοτικών αλγορίθμων αλλά απαιτεί πολύ μεγαλύτερη προσπάθεια και γνώσεις από τους μηχανικούς – προγραμματιστές. Μερικά παραδείγματα κακής συγγραφής κώδικα είναι η χρησιμοποίηση πολλών εμφωλευμένων επαναληπτικών δομών, αυξομειούμενου μεγέθους διανύσματα ή πίνακες, περιττές εκχωρήσεις τιμών σε μεταβλητές και πολλά άλλα. Λαμβάνοντας υπόψιν τις διαφορετικές αρχιτεκτονικές συστημάτων, την αυτόματη βελτιστοποίηση των Compiler και πολλές άλλες παραμέτρους καταλαβαίνουμε ότι η απόλυτη εκμετάλλευση της υπολογιστικής ισχύς ενός συστήματος είναι σχεδόν αδύνατη.

2. Χρησιμοποίηση μη αποδοτικών δομών δεδομένων

   Κάθε γλώσσα προγραμματισμού περιλαμβάνει ένα μεγάλο σύνολο δομών δεδομένων. Κάποιες από αυτές διαχειρίζονται από τον πυρήνα πολύ αποδοτικότερα από άλλες για διάφορους λόγους, όπως για παράδειγμα το πού αποθηκεύονται: cash, ram, registers, SSD/HDD. Η χρησιμοποίηση λοιπόν κατάλληλων δομών δεδομένων, λαμβάνοντας πάντα μέριμνα για το trade of ταχύτητας εκτέλεσης και κατάληψης χώρου, κρίνεται απαραίτητη.

3. Σειριακή εκτέλεση παραλληλοποιήσιμων αλγορίθμων.

   Πολλές φορές ο αλγόριθμος που αποφασίζουμε να χρησιμοποιήσουμε είναι από τη ‘φύση’ του παραλληλοποιήσιμος. Αυτό σημαίνει ότι μπορεί με τη χρήση κατάλληλων τεχνικών ή/και εργαλείων να παραλληλοποιηθεί και πρακτικά να εκτελείται παράλληλα/ταυτόχρονα σε πολλούς πυρήνες ενός πολυπύρηνου συστήματος. Στο σημείο αυτό αξίζει να αναφέρουμε ότι και πάλι υπάρχει ένα μεγάλο trade of μεταξύ του χρόνου εκτέλεσης, των απαιτήσεων σε μνήμη και του χρόνου-γνώσεις συγγραφής του προγράμματος, αφού το να παραλληλοποιηθεί αποδοτικά ένας αλγόριθμος είναι μία δύσκολη και χρονοβόρα εργασία και ταυτόχρονα οι απαιτήσεις σε μνήμη αυξάνονται ραγδαία.

Οι λόγοι που αναφέρθηκαν παραπάνω είναι λίγοι από τους πολλούς. Παρόλα αυτά διαλέχθηκαν και αναφέρθηκαν αφού αποτέλεσαν τα κύρια ζητήματα της παρούσας διπλωματικής. Πιο συγκεκριμένα μεγάλο μέρος της της διπλωματικής αυτής ήταν η παραλληλοποίηση του προσομοιωτή του πρώτου σταδίου, με σκοπό τη παραγωγή τεχνητών δεδομένων σε λογικό χρόνο (πχ. μερικές ώρες). Για αυτόν τον λόγο θα συνεχίσουμε στο κεφάλαιο αυτό παραθέτοντας μερικές θεωρητικές γνώσεις σχετικά με τον παράλληλο προγραμματισμό (Parallel Computing).

2.2.1 Parallel Computing

Ορισμός:

‘Στην πληροφορική, παράλληλος προγραμματισμός [7] ***λέγεται η ανάπτυξη εφαρμογών οι οποίες εκμεταλλεύονται την ύπαρξη πολλαπλών επεξεργαστικών μονάδων σε έναν πολυεπεξεργαστή ή πολυυπολογιστή για να επιτύχουν αύξηση των υπολογιστικών επιδόσεων και μείωση του απαιτούμενου χρόνου εκτέλεσης της εφαρμογής.’***

Με βάση τον παραπάνω ορισμό συμπεραίνουμε ότι ο παράλληλος προγραμματισμός μπορεί να ιδωθεί ως ειδική περίπτωση ταυτόχρονου προγραμματισμού, όπου η εκτέλεση γίνεται πραγματικά παράλληλα και όχι ψευδοπαράλληλα.

Τα παράλληλα συστήματα διακρίνονται σε πολυεπεξεργαστές κοινής μνήμης, όπου πολλαπλοί επεξεργαστές επικοινωνούν με μία κοινή μνήμη ενιαίου χώρου διευθύνσεων, και σε πολυυπολογιστές κατανεμημένης μνήμης, όπου πολλαπλά πακέτα επεξεργαστή-ιδιωτικής μνήμης, με τον δικό του χώρο διευθύνσεων το καθένα, διασυνδέονται και επικοινωνούν μεταξύ τους και στις δύο περιπτώσεις η επικοινωνία γίνεται μέσω ενός «δικτύου διασύνδεσης». Τα μοντέλα παράλληλου προγραμματισμού για πολυεπεξεργαστές και πολυυπολογιστές είναι το μοντέλο κοινού χώρου διευθύνσεων (π.χ. πολλαπλές διεργασίες ή νήματα, OpenMP) και το μοντέλο μεταβίβασης μηνυμάτων (π.χ. PVM, MPI), αντιστοίχως. Στο πρώτο οι επεξεργαστικές μονάδες ανταλλάσσουν πληροφορίες προσπελαύνοντας κοινόχρηστες μεταβλητές στην κοινή μνήμη, ενώ στο δεύτερο ανταλλάσσοντας μηνύματα. Κάθε προγραμματιστικό μοντέλο μπορεί να εφαρμοστεί και σε σύστημα μιας αρχιτεκτονικής που δεν είναι η φυσική του (π.χ. πολυνηματικό πρόγραμμα σε πολυυπολογιστή ή πρόγραμμα MPI σε πολυεπεξεργαστή) αλλά συνήθως με χαμηλότερες επιδόσεις.

Συνήθως ένας αλγόριθμος παραλληλοποιείται με διάσπασή του σε πολλαπλά τμήματα τα οποία ανατίθενται σε ξεχωριστά νήματα ή διεργασίες και έτσι εκτελούνται παράλληλα σε διαφορετικές επεξεργαστικές μονάδες. Ωστόσο δεν είναι βέβαιο ότι ένας αλγόριθμος, υλοποιημένος σε κάποιο πρόγραμμα, μπορεί να παραλληλοποιηθεί πάντα. Για παράδειγμα μία υπό - ρουτίνα η οποία υπολογίζει την ακολουθία Φιμπονάτσι μέσω ενός επαναληπτικού βρόχου δεν μπορεί να διασπαστεί σε πολλαπλά τμήματα με διαμοίραση των επαναλήψεων του βρόχου σε μικρότερους υπό - βρόχους, αφού ο υπολογισμός ο οποίος γίνεται σε κάθε επανάληψη εξαρτάται από αυτούς των δύο προηγούμενων επαναλήψεων. Επομένως τελικά όλες οι επαναλήψεις πρέπει να εκτελεστούν σειριακά στο εσωτερικό μίας μόνο διεργασίας. Τέτοιου τύπου εξαρτήσεις είναι το σημαντικότερο εμπόδιο για την παραλληλοποίηση.

Η παραλληλοποίηση ενός σειριακού αλγορίθμου, προκειμένου να εκμεταλλευτούμε την αύξηση των υπολογιστικών επιδόσεων που προσφέρει ο παραλληλισμός, διακρίνεται σε τέσσερα βήματα που εκτελούνται διαδοχικά:

- Η διάσπαση του ολικού προβλήματος σε επιμέρους υπό – προβλήματα[1]. Εξαρτάται από το πρόβλημα και εναπόκειται στον προγραμματιστή.
- Η ανάθεση εργασιών σε οντότητες εκτέλεσης (διακριτές διεργασίες, νήματα ή ίνες). Μπορεί να είναι είτε στατική, όπου κάθε οντότητα αναλαμβάνει ένα προκαθορισμένο σύνολο εργασιών προς εκτέλεση, ή δυναμική, όπου οι εργασίες ανατίθενται μία-μία κατά τον χρόνο εκτέλεσης· μόλις μία οντότητα ολοκληρώσει την τρέχουσα εργασία της ζητά να της δοθεί η επόμενη (Workers). Η δυναμική λύση είναι προτιμότερη σε περίπτωση που οι επεξεργαστές έχουν διαφορετικό φόρτο (π.χ. αν κάποιοι εκτελούν και άλλα προγράμματα) αλλά επισύρει κάποια χρονική επιβάρυνση[2].

[1] Ο ΕΠΙΣΤΗΜΟΝΑΣ ΛΕΣΛΙ ΛΑΜΠΟΡΤ, Ο ΟΠΟΙΟΣ ΠΡΩΤΟΣ ΠΡΟΣΔΙΟΡΙΣΕ ΤΗΝ ΑΠΑΙΤΗΣΗ Η ΠΑΡΑΛΛΗΛΗ ΕΚΤΕΛΕΣΗ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΝΑ ΔΙΝΕΙ ΠΑΝΤΑ ΤΑ ΙΔΙΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕ ΤΗΝ ΑΝΤΙΣΤΟΙΧΗ ΣΕΙΡΙΑΚΗ ΤΟΥ ΕΚΤΕΛΕΣΗ, ΠΡΟΚΕΙΜΕΝΟΥ ΑΥΤΟ ΝΑ ΘΕΩΡΕΙΤΑΙ ΟΡΘΟ. (ΒΙΚΙΠΑΙΔΕΙΑ - Παράλληλος προγραμματισμός, 2020)

[2] Ο ΝΟΜΟΣ ΤΟΥ ΆΜΝΤΑΛ ΧΡΗΣΙΜΟΠΟΙΕΙΤΑΙ ΓΙΑ ΝΑ ΔΙΑΠΙΣΤΩΘΕΙ Η ΑΝΑΜΕΝΟΜΕΝΗ ΜΕΙΩΣΗ ΤΟΥ ΧΡΟΝΟΥ ΕΚΤΕΛΕΣΗΣ ΑΠΟ ΤΗΝ ΠΑΡΑΛΛΗΛΟΠΟΙΗΣΗ ΕΝΟΣ ΣΕΙΡΙΑΚΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ. ΦΕΡΕΙ ΤΟ ΟΝΟΜΑ ΤΟΥ ΤΖΙΝ ΆΜΝΤΑΛ ΠΟΥ ΤΟΝ ΠΡΩΤΟΔΙΑΤΥΠΩΣΕ ΚΑΙ ΔΗΛΩΝΕΙ ΟΤΙ Η ΕΠΙΤΑΧΥΝΣΗ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΤΟ ΟΠΟΙΟ ΑΞΙΟΠΟΙΕΙ ΠΟΛΛΑΠΛΟΥΣ ΕΠΕΞΕΡΓΑΣΤΕΣ ΠΕΡΙΟΡΙΖΕΤΑΙ ΑΠΟ ΤΟΝ ΧΡΟΝΟ ΕΚΤΕΛΕΣΗΣ ΠΟΥ ΑΠΑΙΤΕΙ ΤΟ ΣΕΙΡΙΑΚΟ ΜΕΡΟΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ. Ο ΝΟΜΟΣ ΤΟΥ ΆΜΝΤΑΛ ΣΥΝΗΘΩΣ ΟΡΙΖΕΤΑΙ ΜΕ ΤΗ ΒΟΗΘΕΙΑ ΤΟΥ ΠΑΡΑΚΑΤΩ ΜΑΘΗΜΑΤΙΚΟΥ ΤΥΠΟΥ:

$$\frac{1}{(1-P)+\frac{P}{N}}$$

ΣΤΟΝ ΤΥΠΟ ΑΥΤΟΝ Η ΜΕΤΑΒΛΗΤΗ Ρ ΔΗΛΩΝΕΙ ΤΟ ΠΟΣΟΣΤΟ ΤΩΝ ΣΥΝΟΛΙΚΩΝ ΥΠΟΛΟΓΙΣΜΩΝ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΟΙ ΟΠΟΙΟΙ ΜΠΟΡΟΥΝ ΝΑ ΠΑΡΑΛΛΗΛΟΠΟΙΗΘΟΥΝ ΚΑΙ Η ΜΕΤΑΒΛΗΤΗ Ν ΤΟ ΠΛΗΘΟΣ ΤΩΝ ΔΙΑΘΕΣΙΜΩΝ ΕΠΕΞΕΡΓΑΣΤΩΝ. ΓΙΑ

- Την ενορχήστρωση των οντοτήτων. Εδώ γίνεται ο καθορισμός του τρόπου συντονισμού και επικοινωνίας μεταξύ των οντοτήτων εκτέλεσης, π.χ. φράγματα εκτέλεσης, αποστολή ή λήψη μηνυμάτων κλπ.
- Την αντιστοίχιση οντοτήτων σε επεξεργαστές. Μπορεί να είναι στατική, δηλαδή προκαθορισμένη από τον προγραμματιστή, ή δυναμική, δηλαδή ρυθμιζόμενη από το σύστημα κατά τον χρόνο εκτέλεσης [7].

2.3 Χρήσιμα Μαθηματικά Θεωρήματα και Στατιστικές

2.3.1 Μέση Τιμή (Average)

Μέσος όρος ή αλλιώς **δειγματική μέση τιμή**[8] ενός συνόλου **ν** παρατηρήσεων αποτελεί το σπουδαιότερο και χρησιμότερο μέτρο της Στατιστικής και είναι ένα μέτρο θέσης, δηλαδή δείχνει σχετικά τις θέσεις των αριθμών στους οποίους αναφέρεται.

Γενικά, ορίζεται ως το άθροισμα των παρατηρήσεων δια του πλήθους αυτών. Είναι δηλαδή η μαθηματική πράξη ανεύρεσης της «μέσης απόστασης» ανάμεσα σε δύο ή περισσότερους αριθμούς. Η μέση τιμή συμβολίζεται με $\bar{x}$. Η μέση τιμή υπολογίζεται ως εξής:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

2.3.2 Διακύμανση (Variance)

Διακύμανση [9] είναι η αναμενόμενη τιμή της τετραγωνικής απόκλισης της τυχαίας μεταβλητής από τη μέση τιμή, και άτυπα μετρά **πόσο μακριά** ένα σύνολο (τυχαίων) αριθμών απλώνεται από τη μέση τιμή του. Η διακύμανση υπολογίζεται ως εξής:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

2.3.3 Τυπική Απόκλιση (Standard Deviation, SD)

Στη στατιστική, η **τυπική απόκλιση** [10] (**SD**, εκπροσωπούμενη επίσης από το ελληνικό γράμμα σίγμα **σ** ή **s**) είναι ένα μέτρο που χρησιμοποιείται για να υπολογιστεί το ποσό της μεταβολής ή της διασποράς ενός συνόλου τιμών δεδομένων.

Μια χαμηλή τυπική απόκλιση υποδηλώνει ότι τα σημεία των δεδομένων τείνουν να είναι κοντά στο μέσο όρο (που ονομάζεται επίσης η αναμενόμενη τιμή) του συνόλου, ενώ μία υψηλή τυπική απόκλιση υποδεικνύει ότι τα στοιχεία απλώνονται πάνω από ένα ευρύτερο φάσμα των τιμών. Η τυπική απόκλιση μιας τυχαίας μεταβλητής, ενός στατιστικού πληθυσμού, ενός συνόλου δεδομένων, ή της κατανομής πιθανότητας είναι η τετραγωνική ρίζα της διακύμανσης της. Είναι αλγεβρικά απλούστερη, αν και στην πράξη λιγότερο ισχυρή από τη μέση απόλυτη απόκλιση. Η τυπική απόκλιση υπολογίζεται ως εξής:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

ΠΑΡΑΔΕΙΓΜΑ, ΑΝ ΤΟ P ΕΙΝΑΙ 90% (0,9), ΤΟΤΕ ΤΟ (1-P) ΕΙΝΑΙ 10% ΚΑΙ ΤΟ ΟΛΙΚΟ ΠΡΟΓΡΑΜΜΑ (ΣΥΜΦΩΝΑ ΜΕ ΤΟΝ ΤΥΠΟ ΤΟΥ ΆΜΝΤΑΛ) ΜΠΟΡΕΙ ΝΑ ΕΠΙΤΑΧΥΝΘΕΙ ΤΟ ΠΟΛΥ 10 ΦΟΡΕΣ, ΟΣΟΥΣ ΕΠΕΞΕΡΓΑΣΤΕΣ ΚΑΙ ΑΝ ΧΡΗΣΙΜΟΠΟΙΗΣΟΥΜΕ. ΓΙ' ΑΥΤΟΝ ΤΟΝ ΛΟΓΟ Η ΠΑΡΑΛΛΗΛΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΝΑΙ ΧΡΗΣΙΜΗ ΜΟΝΟ ΓΙΑ ΠΕΡΙΟΡΙΣΜΕΝΟ ΠΛΗΘΟΣ ΕΠΕΞΕΡΓΑΣΤΩΝ (ΜΙΚΡΟ Ν) Η ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΜΕ ΠΟΛΥ ΜΕΓΑΛΗ ΤΙΜΗ P (Π.Χ. Η ΠΡΟΣΘΕΣΗ ΔΥΟ Ν-ΔΙΑΣΤΑΤΩΝ ΔΙΑΝΥΣΜΑΤΩΝ ΕΧΕΙ P 100%, ΑΦΟΥ ΠΡΑΚΤΙΚΩΣ ΔΕΝ ΥΠΑΡΧΕΙ ΑΠΟΚΛΕΙΣΤΙΚΑ ΣΕΙΡΙΑΚΟ ΤΜΗΜΑ ΣΤΟΝ ΑΛΓΟΡΙΘΜΟ). (ΒΙΚΙΠΑΙΔΕΙΑ - Παράλληλος προγραμματισμός, 2020)

2.3.4 Δημιουργία Τυχαίων Αριθμών

Εδώ αναφέρουμε μια πολύ σημαντική εφαρμογή [11] της ομοιόμορφης κατανομής στην δημιουργία ενός δείγματος τιμών $x_1, x_2, x_3, \dots, x_n$ από μία τυχαία μεταβλητή X οποιασδήποτε θεωρητικής κατανομής.

Είναι γνωστές αρκετές μέθοδοι, οι οποίες δημιουργούν τυχαίους αριθμούς από μία ομοιόμορφη κατανομή (ομοιόμορφα κατανεμημένους) στο διάστημα $[0, 1]$. Κάθε ηλεκτρονικός υπολογιστής, π.χ. έχει πρόγραμμα το οποίο δίνει τυχαία δείγματα από μία ομοιόμορφη κατανομή στο $[0, 1]$.

Ας υποθέσουμε ότι μια συνεχής τυχαία μεταβλητή X έχει συνάρτηση πυκνότητας πιθανότητας $f_x(x)$ και αθροιστική κατανομή πιθανότητας $F_x(x)$. Ακόμη, θεωρούμε και μία ομοιόμορφα κατανεμημένη μεταβλητή *U* στο διάστημα $(0, 1)$.

Επιθυμούμε να βρούμε μία συνάρτηση G η οποία για κάθε τυχαίο αριθμό u της ομοιόμορφης να δίνει ένα τυχαίο αριθμό x της $f_x(x)$, δηλαδή, $x = G(u)$, έτσι ώστε

$$P(X \leq x) = P(U \leq u)$$

Αλλά είναι γνωστό ότι

$$P(X \leq x) = F_x(x)$$

$$P(U \leq u) = F_u(u) = \frac{u - 0}{1 - 0} = u$$

Δηλαδή θέλουμε να ισχύει

$$F_x(x) = F_u(u) = u$$

$$F_x(x) = u$$

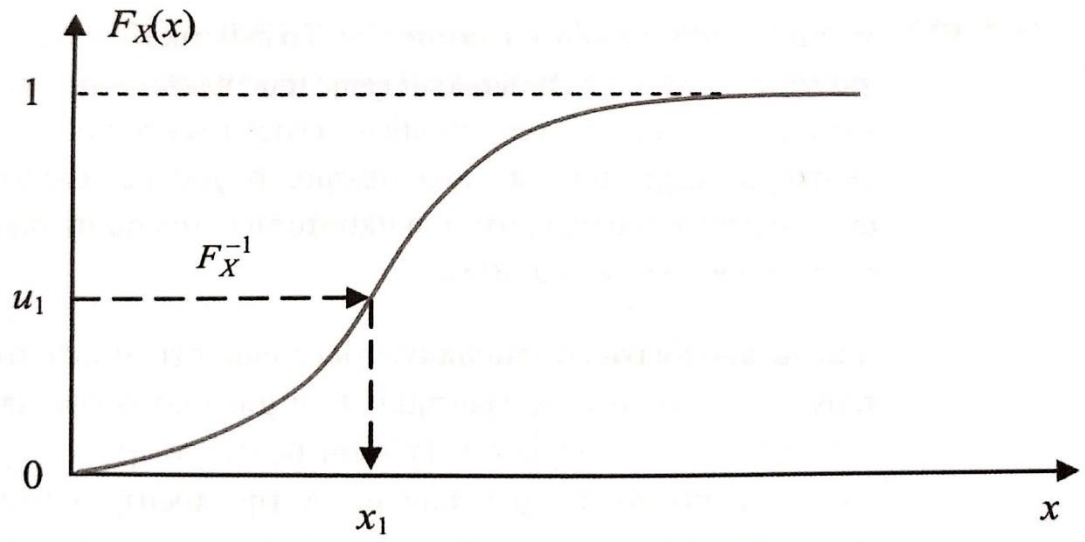
$$x = F_x^{-1}(u)$$

Με άλλα λόγια, η συνάρτηση που επιθυμώ να προσδιορίσω είναι η, $G = F_x^{-1}$, δηλαδή η αντίστροφη της αθροιστικής πιθανότητας της X.

Έτσι, για να δημιουργήσω ένα δείγμα τιμών $x_1, x_2, x_3, \dots, x_n$ από μία τυχαία μεταβλητή με αθροιστική κατανομή πιθανότητας $F_x(x)$, ακολουθώ τα εξής βήματα:

- Βρίσκω την αναλυτική μορφή της $F_x(x)$,
- Ορίζω την αντίστροφη $G = F_x^{-1}$,
- Επιλέγω τυχαία μία τιμή u από την ομοιόμορφη $[0, 1]$,
- Θέτω $x = F_x^{-1}(u)$.

Ο μηχανισμός δημιουργίας τυχαίου δείγματος που μόλις αναπτύξαμε, περιγράφεται γραφικά και στην Εικόνα 2-3. Επιλέγεται τυχαία μία τιμή u_1 από την ομοιόμορφη $[0, 1]$, στη συνέχεια με την χρήση της F_x^{-1} μεταφέρεται στον άξονα x, δίνοντας μία τυχαία τιμή x_1 για την X.



Εικόνα 2-3 Γραφική παράσταση δημιουργίας τυχαίας τιμής x_1, από μια τυχαία τιμή της ομοιόμορφης στο διάστημα $[0,1]$

Παράδειγμα:

Σε μία τυχαία μεταβλητή X, η οποία παριστάνει τον χρόνο καλής λειτουργίας ενός συστήματος η αθροιστική κατανομή πιθανότητας είναι $F_x(x) = 1 - e^{-\lambda x}$. Για να δημιουργήσουμε ένα τυχαίο δείγμα το οποίο θα παριστάνει τους χρόνους λειτουργίας n τέτοιων συστημάτων ακολουθούμε τα εξής βήματα:

- Με τον υπολογιστή μας επιτυγχάνουμε από την ομοιόμορφη $[0, 1]$ τυχαίο δείγμα τιμών $u_1, u_2, \dots, u_n$.
- Θέτουμε για κάθε τιμή u_κ, $u_\kappa = 1 - e^{-\lambda x_\kappa}$,
- Μετασχηματίζουμε ως προς x_κ, $x_\kappa = -\frac{1}{\lambda}\ln(1 - u_\kappa)$.

Οι προκύπτουσες τιμές $x_1, x_2, x_3, \dots, x_n$ είναι τυχαίο δείγμα τιμών από την μεταβλητή X, η οποία ακολουθεί την κατανομή $F_x(x) = 1 - e^{-\lambda x}$.

3 Περιγραφή Αλγορίθμου Προσομοίωσης

3.1 Εισαγωγή

Όπως έγινε κατανοητό από τα εισαγωγικά κεφάλαια, η όλη διαδικασία και μελέτη σωματικών μεταλλάξεων ξεκινάει από ένα τεχνητό σετ δεδομένων. Σε αυτό το κεφάλαιο θα αναλύσουμε βήμα προς βήμα τα προγράμματα που δημιουργήθηκαν και θα ενώσουμε όλα τα κομμάτια προκειμένου να γίνει πλήρως κατανοητός ο τρόπος δημιουργίας του τεχνητού συνόλου δεδομένων. Από εδώ και στο εξής κάθε τμήμα κώδικα που αναλύεται μπορεί να βρεθεί στο GitHub repository https://github.com/tasos51/amitsigk-thesis. Τα βήματα που θα περιγραφούν στις επόμενες ενότητες είναι τα εξής:

- Περιγραφή του προβλήματος.
- Επεξήγηση του προτύπου δεδομένων.
- Συλλογή απαιτήσεων του τεχνητού συνόλου δεδομένων.
- Συλλογή απαιτήσεων του προγράμματος.
- Πλήρης περιγραφή του αλγορίθμου.
- Time improvements.
- Testing των υπο-ρουτίνων.
- Επαλήθευση αποτελεσμάτων.

3.2 Το πρόβλημα

Το πρόβλημα που μελετάται και αναλύεται στο παρόν κεφάλαιο είναι η δημιουργία ενός αρκετά μεγάλου συνόλου reads ακολουθώντας ένα πρότυπο. Το πρότυπο αυτό θα δειγματοληπτηθεί μερικά εκατομμύρια φορές, θα εμπλουτιστεί με δεδομένα ποιότητας, τα οποία θα υποδεικνύουν με ρεαλιστικό τρόπο πόσο «καλά» έγινε η αλληλούχιση των amplicons στην υποτίθεται πραγματική εκτέλεση του πειράματος και τέλος τα δεδομένα θα εγγραφούν σε αρχεία σε fastq format. Επιπλέον όλη η διαδικασία απαιτείται να γίνεται σε πεπερασμένο χρόνο μερικών ωρών με ταυτόχρονα καλή διαχείριση της μνήμης RAM ενός μέσου προσωπικού υπολογιστή.

3.3 Επεξήγηση Προτύπου Δεδομένων (Template)

Το πρόγραμμα του προσομοιωτή απαιτεί ως είσοδο ένα πρότυπο δεδομένων (template) file, το οποίο είναι πρακτικά ένα αρχείο πίνακα, όπου οι κάθε στήλη του χωρίζεται από την επόμενη με τον χαρακτήρα tab (tab delimited file). Το αρχείο αυτό περιλαμβάνει τους τύπους όλων των amplicons που θέλουμε να δημιουργήσουμε και να εξάγουμε στο τέλος εκτέλεσης του προγράμματος, καθώς και λεπτομέρειες και παραμέτρους που χρειάζονται για τη δειγματοληψία των συμβολοσειρών βάσεων DNA. Στη συνέχεια παραθέτουμε ένα κομμάτι του template file σε πίνακα:

AmpliconID	Type	Sequence	MeanCoverage	VAF	REF	ALT
CSF3R.exon.17.line.1.chr1.36931697.36932509_tile_1	Ref	TTTACAATACTGAAGTTATAGGAAACAAGCA CAAAAGGCCATTGGGTGGGGCTGGATGGAGC ATGATCTGGTCCTTAAAGTATGCAGATCGCCT GGGAGGCCCAGCCTATGGAGATTGGGAGGAG AGGGAGATGCTGGTGACTGGAGATGGTGAGA GCCTGGGCTGGGGTAGTTTTTAGTCATGGGCT TATGGACCCTCCCCTCTTCTCCAGCTAG	11000			
CSF3R.exon.17.line.1.chr1.36931697.36932509_tile_1	Mut1	TTTACAATACTGAAGTTATAGGAAACAAGCA CAAAAGGCCATTGGGTGGGGCTGGATGGAGC ATGATCTGGTCCTTAAAGTATGCAGATCGCCT GCGAGGCCCAGCCTATGGAGATTGGGAGGAG AGGGAGATGCTGGTGACTGGAGATGGTGAGA GCCTGGGCTGGGGTAGTTTTTAGTCATGGGCT TATGGACCCTCCCCTCTTCTCCAGCTAG		0.08	G	C

CSF3R.exon.17.line.1.chr1.36931697.36932509_tile_1	Mut2	TTTACAATACTGAAGTTATAGGAAACAAGCA CAAAAGGCCATTGGGTGGGGCTGGATGGAGC TTGATCTGGTCCTTAAAGTATGCAGATCGCCT GGGAGGCCCAGCCTATGGAGATTGGGAGGAG AGGGAGATGCTGGTGACTGGAGATGGTGAGA GCCTGGGCTGGGGTAGTTTTTAGTCATGGGCT TATGGACCCTCCCCTCTTCTCCAGCTAG		0.3	A	T
MPL.exon.10.line.5.chr1.43814933.43815030_tile_1	Ref	GGCGGTGGACGGAGATCTGGGGTCACAGAGC GAACCAAGAATGCCTGTTTACAGGCCTTCGG CTCCACCTGGTCCACCGCCAGTCTCCTGCCTG GCGGGGGCGGTACCTGTAGTGTGCAGGAAAC TGCCACCTCAGCAGCAGCAGGCCCAGGACGG CGCTGAGGCCCAGCACTAGATGCAGAGCGGT CACCAAGGAGATCCAGGCTAGGAG	3700			
MPL.exon.10.line.5.chr1.43814933.43815030_tile_1	Mut1	GGCGGTGGACGGAGATCTGGGGTCACAGAGC GAACCAAGAATGCCTGTTTACAGGCCTTCGG CTCCACCTGGTCCACCGCCAGTGTCTCCTGCC TGGCGGGGGCGGTACCTGTAGTGTGCAGGAA ACTGCCACCTCAGCAGCAGCAGGCCCAGGAC GGCGCTGAGGCCCAGCACTAGATGCAGAGCG GTCACCAAGGAGATCCAGGCTAGGAG		0.54	-	GT
PTEN.exon.7.line.9.chr10.89717609.89717776_tile_1	Ref	TTTAACCATGCAGATCCTCAGTTTGTGGTCTG CCAGCTAAAGGTGAAGATATATTCCTCCAATT CAGGACCCACACGACGGGAAGACAAGTTCAT GTACTTTGAGTTCCCTCAGCCGTTACCTGTGT GTGGTGATATCAAAGTAGAGTTCTTCCACAA ACAGAACAAGATGCTAAAAAA	1400			
PTEN.exon.7.line.9.chr10.89717609.89717776_tile_1	Mut1	TTTAACCATGCAGATCCTCAGTTTGTGGTCTG CCAGCTAAAGGTGAAGATATATTCCTCCAATT CAGGACCCACACGACGGGAAGACAAGTTCAT GTACTTTGAGTTCCCTCAGCCGTTACCTTGTG TGGTGATATCAAAGTAGAGTTCTTCCACAAAC AGAACAAGATGCTAAAAAA		0.1	G	-

Οι στήλες του πίνακα είναι οι εξής:

- AmpliconID: Πρόκειται για μία συμβολοσειρά η οποία λειτουργεί ως μοναδικό χαρακτηριστικό του τύπου του amplicon του αντιστοιχεί στην κάθε γραμμή. Το AmpliconID πρέπει να δίνεται στο εξής format: xxxxxxxxxxxx_tile_tileNumber
- Type: Η στήλη "Type" υποδεικνύει το είδος του amplicon της κάθε γραμμής. Πιο συγκεκριμένα μπορεί να περιλαμβάνει τις συμβολοσειρές 'Ref', 'Mut1', Mut2, …, 'Mut_x'.
    - Στην περίπτωση 'Ref' (reference) σημαίνει ότι στη συγκεκριμένη γραμμή δεν έχουν εισαχθεί μεταλλάξεις.
    - Αντίθετα στις περιπτώσεις 'Mut_x' έχουν εισαχθεί τεχνητές μεταλλάξεις και ο αριθμός x συμβολίζει την x–ιοστή μετάλλαξη μετά το τελευταίο 'Ref'.
- Sequence: Πρόκειται για τη συμβολοσειρά βάσεων προς δειγματοληψία. Περιλαμβάνει συμβολοσειρά αναφοράς ή συμβολοσειρά με μετάλλαξη ανάλογα με τις περιπτώσεις 'Ref' ή 'Mut_x' αντίστοιχα.
- MeanCoverage: Πρόκειται για τον συνολικό αριθμό των φορών που θα δειγματοληπτηθεί το εκάστοτε amplicon στο 'Ref'. Παραμένει κενό για τα 'Mut_x'.
- VAF: (Allele frequency) Πρόκειται για το ποσοστό επί τις εκατό που εμφανίζεται το εκάστοτε αλληλόμορφο ως προς το 'Ref' στο οποίο ανήκει. Ο αριθμός των φορών που θα δειγματοληπηθεί το 'Mut_x' προκύπτει από τον πολλαπλασιασμό $\mathrm{VAF} * \mathrm{MeanCoverage}$ του 'Ref' στο οποίο ανήκει.
- REF: Παραμένει κενό στα πεδία των 'Ref', ενώ περιέχει το αλληλόμορφο αναφοράς στα πεδία των 'Mut_x'.
- ALT: Παραμένει κενό στα πεδία των 'Ref', ενώ περιέχει το εναλλακτικό αλληλόμορφο (με εισηγμένη μετάλλαξη) στα πεδία των 'Mut_x'.

  Υποσημείωση: Ένα ορφανό πεδίο τύπου 'Mut_x', ενώ απουσιάζει το αντίστοιχο πεδίο του 'Ref' δεν μπορεί να υπάρχει. Πρακτικά τα πεδία REF, ALT στα 'Mut_x'

υποδεικνύουν μια γνωστή εισαγμένη μετάλλαξη στη συμβολοσειρά (sequence) με την αλλαγή να έγινε σε κάποια θέση από την βάση του REF στη/στις βάσεις του ALT. Οι περιπτώσεις που διακρίνουμε είναι τρείς:

1. Εισαγωγή βάσης (insertion)

1	TTTACAATACTGAAGTTATAGGAAACAAGCA
2	TTTACAA***T***TACTGAAGTTATAGGAAACAAGCA

2. Διαγραφή βάσης (deletion)

1	TTTACAATACTGAAGTTATAGGAAACAAGCA
2	TTTAC_ATACTGAAGTTATAGGAAACAAGCA

3. Αλλαγή βάσης (change)

1	TTTACAATACT***G***AAGTTATAGGAAACAAGCA
2	TTTACAATACT***C***AAGTTATAGGAAACAAGCA

3.4 Απαιτήσεις Τεχνητού Συνόλου Δεδομένων

Στο σημείο αυτό θα αναλύσουμε λεπτομερώς τη μορφή του συνόλου δεδομένων που θέλουμε να έχει ως έξοδο το πρόγραμμά μας. Κατόπιν θα καταγράψουμε-συλλέξουμε τις απαιτήσεις στις οποίες θέλουμε να συμμορφώνεται το σύνολο αυτό δεδομένων.

3.4.1 FASTQ format

FASTQ format είναι μία μορφή κειμένου για την αποθήκευση βιολογικών σειρών βάσεων και τα αντίστοιχα δεδομένα ποιότητας (quality scores). Τα γράμματα των συμβολοσειρών αλλά και αυτά των ποιοτήτων κωδικοποιούνται σε μορφή ASCII code για συντομία.

Ένα αρχείο fastq το οποίο περιέχει μόνο μία εγγραφή (μοναδική συμβολοσειρά) μοιάζει όπως παρακάτω (στο αρχείο δεν συμπεριλαμβάνεται η αρίθμηση γραμμών):

```
1 @SEQ_ID
2 GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
3 +
4 !''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

- Η γραμμή 1 ξεκινά με τον χαρακτήρα ‘@’ και ακολουθείται από ένα μοναδικό χαρακτηριστικό όνομα της βιολογικής σειράς πχ. AmpliconID και μία προαιρετική περιγραφή.
- Η γραμμή 2 είναι η ακατέργαστη βιολογική συμβολοσειρά βάσεων.
- Η γραμμή 3 ξεκινά με τον χαρακτήρα ‘+’ και προαιρετικά ακολουθείται από το ίδιο χαρακτηριστικό της γραμμής 1 και οποιαδήποτε περιγραφή.
- Η γραμμή 4 κωδικοποιεί τις τιμές ποιότητας για την συμβολοσειρά της γραμμής 2, και πρέπει να έχει το ίδιο ακριβώς μήκος γραμμάτων με τη συμβολοσειρά της γραμμής 2. [12]

Το σκόρ ποιότητας μιας βάσης γνωστό και ως Phred ή Q score είναι ένας ακέραιος αριθμός που αντιπροσωπεύει την εκτιμώμενη πιθανότητα σφάλματος, ότι για παράδειγμα η βάσει είναι λάθος.

Έστω P είναι η πιθανότητα σφάλματος, τότε:

$$P = 10^{-Q/10} \Leftrightarrow Q = -10log_{10}(P)$$

Τα Q scores [13] στην πλειονότητα των περιπτώσεων, όπως και στο πρόγραμμα του προσομοιωτή, κωδικοποιούνται ως χαρακτήρες ASCII με βάση το 33 όπως παρακάτω:

Πίνακας 3-1 ASCII_BASE=33 Illumina quality scores

ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 $	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (	18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

3.4.2 Απαιτήσεις Τεχνητού Συνόλου Δεδομένων

Στη συνέχεια της παραγράφου θα αναφερόμαστε στο τεχνητό σύνολο δεδομένων προς παραγωγή ως **ΤΣΔ**.

Απαιτήσεις:

- Το **ΤΣΔ** θα πρέπει να προκύπτει από τη δειγματοληψία του προτύπου δεδομένων (template) δειγματοληπτώντας όλες τις συμβολοσειρές βάσεων.
- Το **ΤΣΔ** θα πρέπει να αποθηκευτεί σε fastq format όπως ακριβώς περιγράφεται στην ενότητα 3.4.1.
- Το **ΤΣΔ** θα πρέπει να έχει πεδίο '+' (γραμμή 2) πανομοιότυπο με το πεδίο '@' (γραμμή 1).
- Τα πεδία '@', '+' του **ΤΣΔ** θα πρέπει να αναγράφουν το μοναδικό αναγνωριστικό του amplicon , το AmpliconID, ακολουθούμενο από έναν μοναδικό αύξον αριθμό που υποδεικνύει το coverage που βρίσκεται το συγκεκριμένο amplicon. Για παράδειγμα έστω ότι πρόκειται για το amplicon με AmpliconID: XXXX το οποίο προέκυψε από την 153η δειγματοληψία στη σειρά τότε τα πεδία '@', '+' θα είναι '@XXXX_153' και '+XXXX_153' αντίστοιχα.
- Το **ΤΣΔ** θα πρέπει να έχει πεδίο ποιοτήτων (γραμμή 4), με τις ποιότητες κωδικοποιημένες σε ***ASCII_BASE=33*** Illumina όπως ακριβώς καταγράφεται στον παραπάνω πίνακα.
- Οι ποιότητες (Q score) του **ΤΣΔ** θα πρέπει να ακολουθούν συγκεκριμένη κατανομή κατά μήκος της συμβολοσειράς με γνωστά στατιστικά που θα επιλέγει ο χρήστης για κάθε θέση.
- Το **ΤΣΔ** θα πρέπει να εγγραφεί σε αρχείο με κατάληξη '.fastq'.

3.5 Απαιτήσεις Προγράμματος Προσομοιωτή

Απαιτήσεις:

- Το πρόγραμμα του προσομοιωτή θα πρέπει να δέχεται ως είσοδο την απόλυτη διαδρομή στο δίσκο του πρότυπου αρχείου δεδομένων (template).
- Το πρότυπο αρχείο δεδομένων θα πρέπει να βρίσκεται αυστηρά στο format που επεξηγείται στη ενότητα 3.3.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να δέχεται ως είσοδο τον αριθμό των threads που θέλει ο χρήστης να χρησιμοποιήσει προκειμένου να επιταχύνει τη διαδικασία παραγωγής των reads. (Reads ονομάζονται τα διαφορετικά δείγματα ενός amplicon)
- Το πρόγραμμα του προσομοιωτή θα πρέπει θα πρέπει να δέχεται ως είσοδο την επιλογή του χρήστη (yes/no) για την συμπίεση των δεδομένων εξόδου.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να δέχεται ως είσοδο το μήκος των reads σε αριθμό βάσεων που θέλουμε να δημιουργηθούν.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να δέχεται ως είσοδο την απόλυτη διαδρομή στο δίσκο του αρχείου καταγραφής της μέσης τιμής της κατανομής της ποιότητας ανά θέση βάσης.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να δέχεται ως είσοδο την απόλυτη διαδρομή στο δίσκο του αρχείου καταγραφής της τυπικής απόκλισης της κατανομής της ποιότητας ανά θέση βάσης.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να δέχεται ως είσοδο την απόλυτη διαδρομή στο δίσκο του φακέλου που επιθυμεί ο χρήστης να αποθηκεύσει τα αρχεία αποτελεσμάτων.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να μπορεί να δειγματοληπτεί το πρότυπο αρχείο δεδομένων.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να μπορεί να εμπλουτίζει τα δεδομένα δειγματοληψίας με δεδομένα ποιότητας, τα οποία να ακολουθούν συγκεκριμένη κατανομή ποιότητας, με γνωστή μέση τιμή και τυπική απόκλιση ανά θέση.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να μπορεί να συμπιέσει τα δεδομένα που προκύπτουν από τις δύο παραπάνω απαιτήσεις.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να μπορεί να αποθηκεύσει τα συμπιεσμένα ή μη δεδομένα που δημιούργησε.
- Το πρόγραμμα του προσομοιωτή θα πρέπει να μπορεί εκτελέσει το έργο του σε λογικό χρόνο, ανάλογο του μεγέθους των δεδομένων που θα του ζητηθεί να επεξεργαστεί και να αποθηκεύσει.

Υποσημείωση: Η τελευταία απαίτηση, μάλλον, είναι η λιγότερο σαφώς καθορισμένη. Αυτό συμβαίνει διότι η πολυπλοκότητα του προγράμματος είναι αρκετά μεγάλη και δεν μπορεί να προσδιοριστεί επακριβώς, με αποτέλεσμα, να μην έχουμε την δυνατότητα να καθορίσουμε βελτίωση της. Παρακάτω θα γίνει εμφανής η ανάγκη για βελτιστοποίηση του χρόνου εκτέλεσης και θα περιγραφεί ο τρόπος με τον οποίο επιτεύχθηκε αυτό.

3.6 Περιγραφή Αλγορίθμου

Υποθέτοντας ότι το αρχείο δεδομένων έχει εισαχθεί κανονικά ο αλγόριθμός μας ξεκινά με μια στοιχειώδη προ-επεξεργασία των δεδομένων. Τα βήματα του αλγορίθμου κατηγοριοποιούνται σε υπό-ενότητες παρακάτω.

3.6.1 Προ-επεξεργασία Δεδομένων

Σκοπός της προ-επεξεργασίας δεδομένων είναι να υπολογιστεί ο αριθμός των φορών που θα γίνει δειγματοληψία των mutexed amplicons (πεδία ‘Mut_x’). Ο υπολογισμός αυτός

γίνεται από τον πολλαπλασιασμό της τιμής VAF του εκάστοτε mutexed amplicon με την τιμή του MeanCoverage του πατρικού amplicon τύπου 'Ref'. Αφού υπολογιστεί ο αριθμός αυτός καταχωρείται στη αντίστοιχη κενή θέση του πίνακα στη στήλη MeanCoverage. Θεωρούμε $Ref_{i\dots m}$ τα m διαφορετικά 'Ref' πεδία και $Mut_{i,j\dots n}$ τα n διαφορετικά 'Mut' πεδία που ανήκουν στο i-οστό 'Ref'. Τότε η πράξη γίνεται:

$$Mut_{i,j}.MeanCoverage = Mut_{i,j}.VAF * Ref_i.MeanCoverage$$

Αφού υπολογιστούν οι αριθμοί MeanCoverage για όλα τα παιδιά $Mut_{i,j\dots n}$, $i = c$ ενός $Ref_{i=c}$ οι τιμές τους προστίθενται και αφαιρούνται από το MeanCoverage του $Ref_{i=c}$ όπως παρακάτω:

$$Ref_i.MeanCoverage = Ref_i.MeanCoverage - \sum_{j=1}^{n} Mut_{i,j}$$

Οι παραπάνω υπολογισμοί επιτυγχάνονται μέσω μίας επαναληπτικής διαδικασίας η οποία προσπελαύνει τα τις γραμμές του πίνακα του πρότυπου αρχείου δεδομένων. Μόλις βρεθεί πατρικό amplicon τύπου 'Ref' εκτελούνται οι υπολογισμοί που περιεγράφηκαν παραπάνω για τα παιδιά πεδία 'Mut' μέχρις ότου βρεθεί καινούργιο πατρικό amplicon τύπου 'Ref'.

Το αποτέλεσμα της προ-επεξεργασίας είναι η δημιουργία ενός πίνακα με τα amplicons, πανομοιότυπου με τον πίνακα του πρότυπου αρχείου, με τη μόνη διαφορά ότι πλέον στα πεδία τύπου 'Mut' η στήλη MeanCoverage δεν είναι πλέον κενή, αλλά, περιέχει τον αριθμό των φορών που πρέπει να δειγματοληπτηθούν.

3.6.2 Δειγματοληψία Δεδομένων

Στο σημείο αυτό θα περιγράψουμε πως ο αλγόριθμος του προσομοιωτή, δοθείσης μιας ακολουθίας βάσεων, amplicon, παράγει n αριθμό κομματιών (reads) διαβάζοντας το amplicon από τα αριστερά προς τα δεξιά (read1) και n αριθμό reads διαβάζοντας το amplicon από τα δεξιά προς τα αριστερά (read2).

Μπορούμε να χωρίσουμε το πρόβλημα της παραγωγής reads δε δύο ανεξάρτητα μεταξύ τους υπο-προβλήματα. Το πρώτο είναι η δημιουργία reads από τις συμβολοσειρές βάσεων του πρότυπου αρχείου δεδομένων, με τυχαίο αλλά μεθοδικό τρόπο. Το δεύτερο είναι η παραγωγή συμβολοσειρών ποιοτήτων με τρόπο τέτοιο ώστε ναι μεν οι ποιότητες να είναι τυχαίες αλλά να ακολουθούν **ανά θέση** Gaussian-Κανονική κατανομή με γνωστή μέση τιμή και τυπική απόκλιση. Θα ονομάσουμε το πρώτο πρόβλημα **παραγωγή δεδομένων** ενώ το δεύτερο **παραγωγή ποιοτήτων** και θα τα αναλύσουμε ξεχωριστά.

3.6.2.1 Παραγωγή Δεδομένων

Η παραγωγή των δεδομένων πρέπει να γίνεται με τέτοιον τρόπο ώστε να προσεγγίζει όσο το δυνατόν καλύτερα την δειγματοληψία των δεδομένων σε ένα Sequencing Machine. Υποθέτουμε λοιπόν ότι έχουμε μία συμβολοσειρά βάσεων μήκους 20 χαρακτήρων όπως παρακάτω:

Amplicon | TTTACAATACTGAAGTTATA

Σκοπός μας είναι να ‘κόψουμε’ δύο κομμάτια (read1, read2) από το παραπάνω amplicon μήκους m χαρακτήρων με επικάλυψη τουλάχιστον ενός χαρακτήρα μεταξύ των δύο reads, ξεκινώντας από μία τυχαία θέση εκκίνησης στο amplicon. Επιπλέον βασική προϋπόθεση είναι ότι αν το μήκος m είναι αρκετά μεγάλο ώστε να ξεφύγουμε εκτός από το amplicon είτε προς τα αριστερά είτε προς τα δεξιά είτε και προς τις δύο κατευθύνσεις, το αποτέλεσμα που θα προκύψει θα συμπληρωθεί με τον κατάλληλο αριθμό χαρακτήρων ‘N’ προς όποια κατεύθυνση είναι αναγκαίο για το κάθε read. Ο αλγόριθμος που χρησιμοποιήθηκε είναι ο εξής:

Έστω η παραπάνω συμβολοσειρά, ορίζουμε τους δείκτες αρχής και τέλους όπως παρακάτω:

```
1 TTTACAATACTGAAGTTATA
2 1                   n
```

Δηλαδή $string_{idxs} \in [1, n]$, επιλέγουμε μία τυχαία θέση εκκίνησης

$random_{start} = rand(1, n)$ και ένα τυχαίο μήκος παραθύρου

$random_{offset} = rand(0, m - 1)$

όπου m είναι το μήκος των reads που θέλουμε να εξάγουμε, και η συνάρτηση rand(a,b) θεωρούμε ότι επιστρέφει έναν τυχαίο αριθμό που ακολουθεί κανονική κατανομή μεταξύ των αριθμών a και b. Τέλος από εδώ και στο εξής θα θεωρούμε τις μεταβλητές L.r1, H.r1, L.r2, H.r2 ως τον ανώτατο και τον κατώτατο δείκτη του read στο amplicon.

From left to right (R1):

$$H.r1 = random_{start} + random_{offset}$$

$$L.r1 = H.r1 - m + 1$$

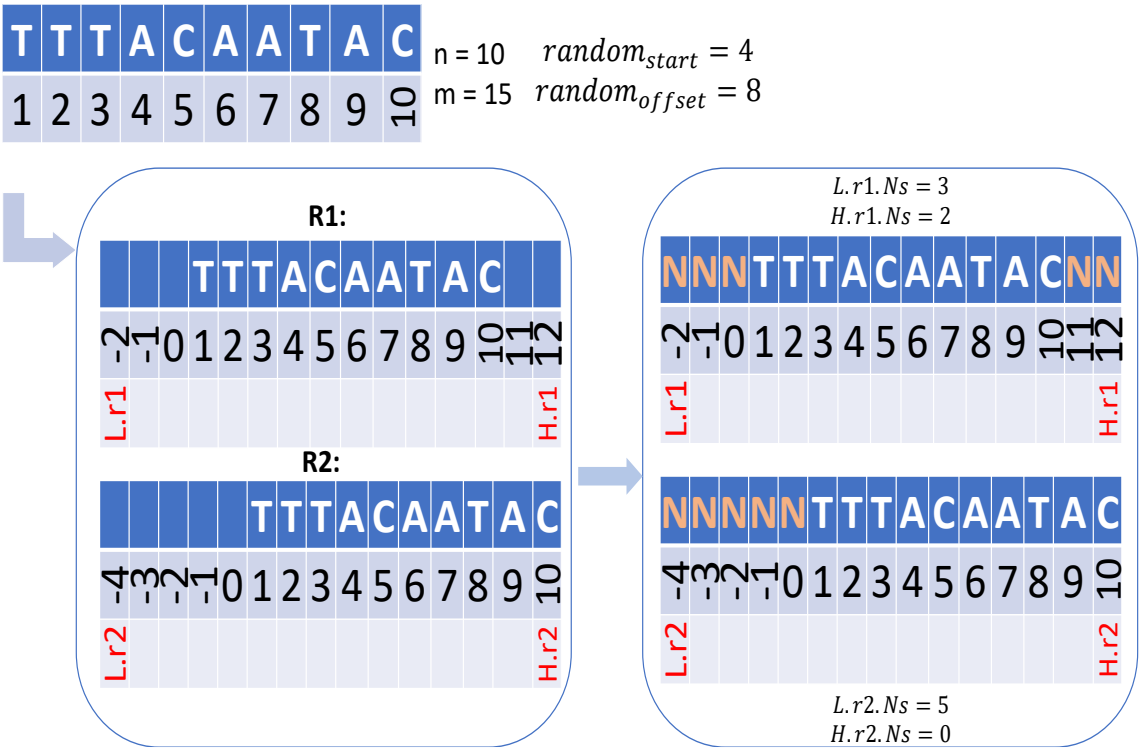
From right to left (R2):

$$L.r2 = random_{start} - random_{offset}$$

$$H.r2 = L.r2 + m - 1$$

Two sides Ns calculation:

$$L.Ns = \{$$
$$if\ L.rx \leq 0$$
$$abs(L.rx) + 1$$
$$else$$
$$0$$
$$\}$$
$$H.Ns = \{$$
$$if\ H.rx > n$$
$$H.rx - n$$
$$else$$
$$0$$
$$\}$$



Στο πρώτο τμήμα του ψευδοκώδικα line1-3, υπολογίζουμε του δείκτες του πρώτου read1, αυτού που στην πραγματικότητα θα διαβάζονταν από το scanner από τα αριστερά προς τα δεξιά.

- Line2: Υπολογίζουμε τον ανώτερο (πιο δεξί) δείκτη στο amplicon προσθέτοντας στην τυχαία θέση εκκίνησης το τυχαίο offset, το πόσο δηλαδή θέλουμε να απομακρυνθούμε προς τα δεξιά από την θέση εκκίνησης.
- Line3: Υπολογίζουμε τον κατώτερο (πιο αριστερό) δείκτη στο amplicon αφαιρώντας από το δεξί το μήκος του επιθυμητού read και προσθέτοντας τον αριθμό 1.

Στο δεύτερο τμήμα του ψευδοκώδικα line4-6, υπολογίζουμε του δείκτες του δεύτερου read2, αυτού που στην πραγματικότητα θα διαβάζονταν από το scanner από τα δεξιά προς τα αριστερά. Παρόλο που στην πραγματικότητα το read2 θα διαβάζονταν αντεστραμμένο, το scanner αυτοματοποιημένα θα γυρνούσε τη συμβολοσειρά προς την ορθή κατεύθυνση. Συνεπώς αυτό που αρκεί να κάνουμε στα πλαίσια της προσομοίωσης είναι να υπολογίσουμε έναν αριστερό και έναν δεξί δείκτη πάνω στο amplicon με τρόπο τέτοιο ώστε να έχει επικάλυψη τουλάχιστον ενός χαρακτήρα με το read1 και να αντιγράψουμε τους χαρακτήρες που δεικτοδοτούνται από τους δείκτες.

- Line5: Υπολογίζουμε τον κατώτερο (πιο αριστερό) δείκτη στο amplicon, αφαιρώντας από την τυχαία θέση εκκίνηση το τυχαίο offset, το πόσο δηλαδή θέλουμε να απομακρυνθούμε προς τα αριστερά από την θέση εκκίνησης.
- Line6: Υπολογίζουμε τον ανώτερο (πιο δεξί) δείκτη στο amplicon, προσθέτοντας από τον αριστερό το μήκος του επιθυμητού read (m) και αφαιρώντας τον αριθμό 1.

Στο τρίτο και τελευταίο τμήμα του ψευδοκώδικα line7-19, υπολογίζουμε τον αριθμό των χαρακτήρων N που πρέπει να προστεθούν στην αρχή και στο τέλος της συμβολοσειράς-κομμάτι που έχουμε εξάγει από το amplicon μέσω της παραπάνω διαδικασίας. Τονίζουμε ότι έχοντας γνωστούς τους δείκτες $L.r_x$ και $H.r_x$ για x = 1,2 δηλαδή για οποιαδήποτε από τα δύο

reads, μπορούμε πολύ εύκολα να υπολογίσουμε τον αριθμό των χαρακτήρων 'N' που πρέπει να προστεθούν από τα αριστερά $(L.Ns)$ και από τα δεξιά $(H.Ns)$.

- Line 8-13: Ο αριθμός των L.Ns που πρέπει να προστεθούν στα αριστερά του read ισούται με την απόλυτη τιμή του δείκτη $L.r_x$ αν ο δείκτης $L.r_x < 0$, ενώ είναι 0 σε κάθε άλλη περίπτωση.
- Line 14-19: Ο αριθμός των H.Ns που πρέπει να προστεθούν στα δεξιά του read ισούται με τη διαφορά $H.r_x - n$ όπου n το μήκος του amplicon αν ο δείκτης $H.r_x > 0$, ενώ είναι 0 σε κάθε άλλη περίπτωση.

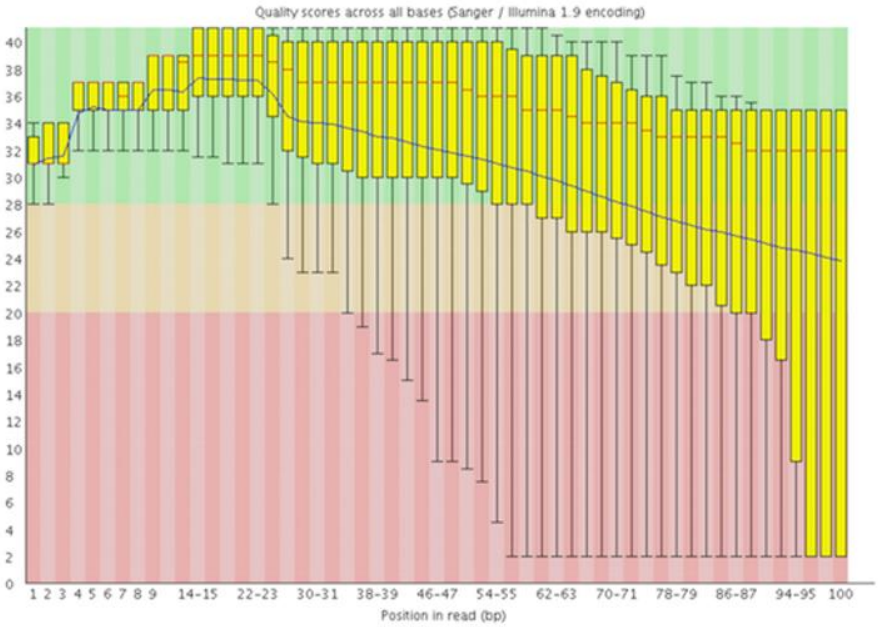
Υποσημειώσεις:

1. Ο αλγόριθμος λειτουργεί για οποιοδήποτε μήκος m, n. Συνεπώς διακρίνουμε 3 περιπτώσεις:
    a. Αν $m > n$ σημαίνει ότι το ζητούμενο κομμάτι θα είναι μεγαλύτερο από το μήκος του amplicon και, προφανώς, θα συμπληρωθεί ή στην αρχή ή και στο τέλος ή και στα δύο με χαρακτήρες 'N', ανάλογα με τη θέση εκκίνησης.
    b. Αν $m < n$ σημαίνει ότι το ζητούμενο κομμάτι θα είναι μικρότερο σε μήκος από το amplicon και, προφανώς, θα συμπληρωθεί μόνο στην αρχή ή μόνο στο τέλος ή σε κανένα από τα δύο με χαρακτήρες 'N', ανάλογα με τη θέση εκκίνησης.
    c. Αν $m = n$ σημαίνει ότι το ζητούμενο κομμάτι θα είναι ίσο με το μήκος τους amplicon και, προφανώς, θα συμπληρωθεί μόνο στην αρχή ή μόνο στο τέλος ή σε κανένα από τα δύο με χαρακτήρες 'N', ανάλογα με τη θέση εκκίνησης.
2. Ο αλγόριθμος θα παράγει ζευγάρια reads τα οποία θα έχουν επικάλυψη **τουλάχιστον** μίας βάσης μεταξύ τους ανάλογα, με το τυχαίο offset.
3. Ο αλγόριθμος θα παράγει ζευγάρια reads τα οποία θα περιλαμβάνουν **τουλάχιστον** μία βάση του amplicon, ανάλογα με την τυχαία θέση εκκίνησης. Σε αυτή την περίπτωση όλες οι υπόλοιπες θέσεις θα συμπληρωθούν με χαρακτήρες 'N'.
4. Ο αλγόριθμος θα εκτελεστεί επαναληπτικά μαζί με το κομμάτι της παραγωγής ποιοτήτων που θα αναλύσουμε παρακάτω MeanCoverage φορές για κάθε 'Ref' και 'Mut' amplicon που υπάρχει στο πρότυπο αρχείο δεδομένων.

3.6.2.2 Παραγωγή ποιοτήτων

Η παραγωγή των ποιοτήτων πρέπει να γίνει με τέτοιον τρόπο ώστε από τη μία να ικανοποιούνται οι απαιτήσεις συστήματος και από την άλλη να είναι αρκετά τυχαίες και ρεαλιστικές. Για να προχωρήσουμε παρακάτω, λοιπόν, καλό θα ήταν να εξηγήσουμε λεπτομερώς τι εννοούμε όταν αναφερόμαστε σε τυχαίες και ρεαλιστικές ποιότητες. Ο καλύτερος τρόπος για να γίνει αυτό είναι να μελετήσουμε ένα σύνολο πραγματικών δεδομένων.

Παρακάτω βλέπουμε ένα box plot το οποίο δημιουργήθηκε από το FastQC για πραγματικά δεδομένα.



Εικόνα 3-1 Per base qualities for real data - FastQC

Παρατηρούμε ότι αποτυπώνονται στατιστικά που προέκυψαν από την ανάλυση των ποιοτήτων όλων των reads ανά θέση βάσης. Συμπεραίνουμε τα εξής:

1. Από τις μεσαίες (κόκκινες) γραμμές των boxes καταλαβαίνουμε ότι οι μέση τιμή της ποιότητας ανά βάση είναι διαφορετική.
2. Από το μέγεθος των boxes καταλαβαίνουμε ότι η τυπική απόκλιση των ποιοτήτων ανά θέση βάσης διαφέρει αρκετά. Πιο συγκεκριμένα παρατηρούμε ότι στις αρχικές βάσεις οι ποιότητες είναι λιγότερο ‘απλωμένες’ γύρω από τη μέση τιμή τους ενώ όσο πλησιάζουμε στην ακραία αριστερή βάση (100) οι ποιότητες ‘απλώνονται’ όλο και περισσότερο γύρω από τη μέση τιμή.

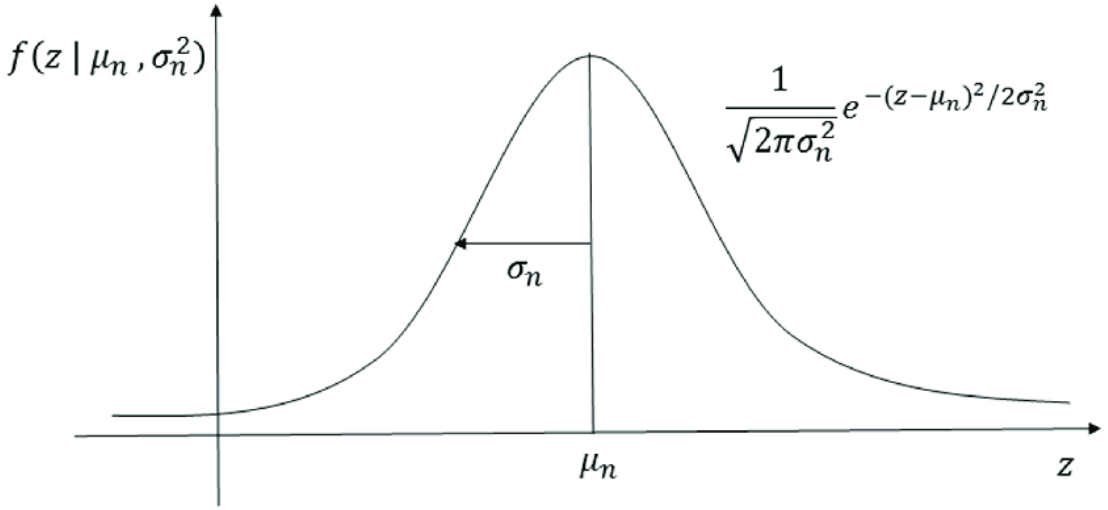
Τα παραπάνω συμπεράσματα είναι πολύ χρήσιμα τόσο για την κατανόηση των πραγματικών δεδομένων όσο και για την κατάστρωση των απαιτήσεων του τεχνητού συνόλου δεδομένων. Οδηγούμαστε λοιπόν στην διαπίστωση ότι το τεχνητό σετ δεδομένων που θέλουμε να δημιουργήσουμε πρέπει να είναι εμπλουτισμένο με ποιότητες οι οποίες προκειμένου να προσεγγίζουν ένα πραγματικό πείραμα θα πρέπει να παράγονται εντελώς τυχαία κάθε φορά, αλλά να έχουν συγκεκριμένη μέση τιμή ποιότητας και τυπική απόκλιση ανά βάση ανά read. Συνεπώς θεωρούμε αναγκαία τη συστηματική καταγραφή των μέσων τιμών και τυπικών αποκλίσεων ανά βάση, σε δύο διαφορετικά αρχεία τα οποία θα δέχεται ως είσοδο το πρόγραμμά του προσομοιωτή, όπως ακριβώς αναφέρεται στις Απαιτήσεις Προγράμματος Προσομοιωτή στην ενότητα 3.5. Για παράδειγμα αν θέλουμε παραγωγή reads μήκους 150 βάσεων έκαστο, χρειαζόμαστε 150 μέσες τιμές ποιότητας και 150 διαφορετικές αποκλίσεις.

Στο σημείο αυτό αξίζει να τονισθεί ότι όταν λέμε συστηματική καταγραφή ανά θέση εννοούμε, m διαφορετικούς μέσους όρους ποιότητας και m διαφορετικές τυπικές αποκλίσεις, όπου m το μήκος των reads που επιθυμούμε. Και για την περίπτωση των μέσων όρων και για την περίπτωση των τυπικών αποκλίσεων, ο τρόπος με τον οποίο θα παραχθούν είναι εντελώς ελεύθερος αρκεί όλες οι τιμές να ανήκουν σε ένα συγκεκριμένο κλειστό σύνολο, αυτό το οποίο χαρακτηρίζει τις ανώτερες και κατώτερες τιμές της ποιότητας, όπως φαίνεται στον Πίνακας 3-1. Για παράδειγμα θα μπορούσε να γίνει χειροκίνητα σε ένα excel

spreadsheet τοποθετώντας τις τιμές μία προς μία με το πληκτρολόγιο ή ακολουθώντας έναν μαθηματικό νόμο-συνάρτηση, ο οποίος αυτοματοποιημένα θα μας έφτιαχνε ένα διάνυσμα για τους μέσους όρους και ένα για τις τυπικές αποκλίσεις, με τις τιμές που επιθυμούμε.

Στη συνέχεια, για λόγους απλότητας θα προσεγγίσουμε το πρόβλημα για μία μόνο ποιότητα και μετά θα το γενικεύσουμε για το σύνολο των ποιοτήτων ανά read.

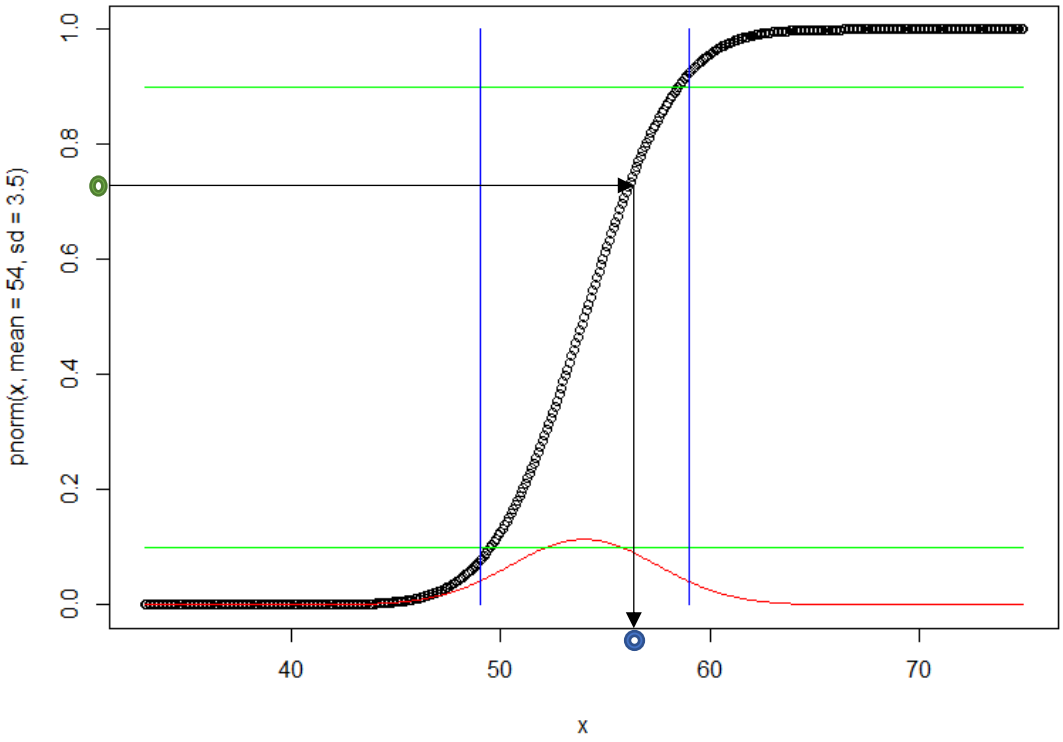
Η κατανομή που επιλέγεται να χρησιμοποιηθεί έχοντας γνωστές, για μία θέση την μέση τιμή και την τυπική απόκλιση προκειμένου να φράξουμε την τυχαία ποιότητα που θα παραχθεί είναι η Gaussian κανονική κατανομή. Έχοντας λοιπόν γνωστή μέση τιμή μ_n και τυπική απόκλιση σ_n προκύπτει ότι η τυχαία τιμή της ποιότητας θα είναι περιορισμένη στην καμπάνα γύρω από τη μέση τιμή και ανοιγμένη όσο η τυπική απόκλιση, όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 3-2: Κανονική κατανομή – Συνάρτηση πυκνότητας πιθανότητας

Το πρόβλημα που προκύπτει στο σημείο αυτό είναι το πως θα φτιάξουμε έναν αλγόριθμό, ο οποίος θα παράγει τυχαίες τιμές ποιότητας οι οποίες θα ακολουθούν όμως συγκεκριμένη κατανομή με γνωστή μέση τιμή και τυπική απόκλιση, αφού είναι γνωστό ότι ο υπολογιστής διαθέτει προγράμματα – συναρτήσεις που παράγουν τυχαία δείγματα ακολουθώντας πάντα ομοιόμορφη κατανομή.

Χρησιμοποιώντας την εφαρμογή που αναλύεται στην ενότητα 2.3.4, μπορούμε εύκολα να δημιουργήσουμε τυχαία δείγματα που ακολουθούν **οποιαδήποτε** κατανομή, συμπεριλαμβανομένου στην περίπτωσή μας και της κανονικής, από δείγματα μίας ομοιόμορφης κατανομής, που φυσικά ο υπολογιστής είναι ικανός. Απαραίτητη προϋπόθεση για αυτή τη διαδικασία είναι να γνωρίζουμε την αθροιστική συνάρτηση της κατανομής που επιθυμούμε. Στην προκείμενη περίπτωση η R διαθέτει στη core library συνάρτηση της αθροιστικής συνάρτησης της κανονικής κατανομής $\mathrm{pnorm}()$ και συνεπώς μπορούμε εύκολα να δημιουργήσουμε τα τυχαία αλλά κανονικά κατανεμημένα δείγματά μας.



Εικόνα 3-3: Με μαύρο χρώμα απεικονίζεται η αθροιστική συνάρτηση πιθανότητας ενώ με κόκκινο η συνάρτηση μάζας πιθανότητας. Τα τυχαία ομοιόμορφα κατανεμημένα δείγματα κυμαίνονται από το 0-1 στον άξονα y (με μεγαλύτερη πιθανότα να βρίσκονται ανάμεσα στις πράσινες γραμμές) ενώ το τυχαίο αλλά κανονικά κατανεμημένο δείγμα κυμαίνεται από 33-75 μέσω της μαύρης καμπύλης (με μεγαλύτερη πιθανότα να βρίσκονται ανάμεσα στις μπλέ γραμμές).

Έχοντας γνωστά τα όρια των τιμών των ποιοτήτων σε ακέραιους αριθμούς ASCII code L_q, H_q, ο υπολογισμός περιγράφεται από τον παρακάτω ψευδοκώδικα:

```
1  L_q = 33, H_q = 75, mean = ct, sd = ct
2  x_norm = seq(L_q, H_q, by = 0.1)
3  y_norm = pnorm(x_norm, mean, sd)
4  result = uniToOthers(x_norm, y_norm, runif(1))
5
6  uniToOthers <- function(x, y, y_rnd) {
7    idx = which.min(abs(y - y_rnd))
8    x_rnd = x[idx]
9    return(x_rnd)
10 }
```

Με έντονη γραφή σημειώνονται οι μεταβλητές διανύσματα.

Line1: Γίνεται η αρχικοποίηση των μεταβλητών της κατώτερης ποιότητας $L_q = 33$, της ανώτερης ποιότητας $H_q = 75$, της μέσης τιμής της κανονικής κατανομής $mean = ct$ ίση με μία σταθερά και της τυπικής απόκλισης της κανονικής κατανομής $sd = ct$ ίση επίσης με μία σταθερά.

Line2: Γίνεται η αρχικοποίηση του διανύσματος των τιμών του άξονα x για την παραγωγή της αθροιστικής συνάρτησης της κανονικής κατανομής.

Line3: Γίνεται η αρχικοποίηση του διανύσματος των τιμών της αθροιστικής συνάρτησης της κανονικής κατανομής με μέση τιμή $mean = ct$ και τυπική απόκλιση $sd = ct$ καλώντας τη συνάρτηση της R $pnorm()$.

Line6: Δημιουργία της συνάρτησης $uniToOthers(\boldsymbol{x}, \boldsymbol{y}, y\_rnd)$ συντομογραφία της φράσης uniform to others υποδηλώνοντας το ότι παίρνει ως ορίσματα μια οποιαδήποτε αθροιστική συνάρτηση κατανομής πιθανότητας σε μορφή δύο διανυσμάτων και ένα τυχαίο δείγμα y_{rnd} από μία ομοιόμορφη κατανομή και το 'μετατρέπει' σε τυχαίο δείγμα της δοθείσης κατανομής, κάνοντας χρήση της εφαρμογής στην ενότητα 2.3.4.

Line7: Καλώντας τη συνάρτηση $which.\min(abs(\boldsymbol{y} - y_{rnd}))$ με όρισμα την απόλυτη τιμή της διαφοράς του διανύσματος $\boldsymbol{y}$ με το τυχαίο ομοιόμορφα κατανεμημένο δείγμα y_rnd, επιστρέφεται ο δείκτης της τιμής του διανύσματος $\boldsymbol{y}$ για την οποία ελαχιστοποιείται η απόλυτη τιμή της διαφοράς αυτής.

Line8: Γνωρίζοντας τον δείκτη της τιμή για την οποία ελαχιστοποιείται η απόλυτη διαφορά, idx μπορούμε να γυρίσουμε πίσω στο διάνυσμα x και να εξάγουμε την τιμή του, $x_{rnd} = x[idx]$, το οποίο είναι το τυχαίο δείγμα που ακολουθεί την κατανομή πιθανότητας της επιλογής μας. Σε αυτήν την περίπτωση την κανονική.

Line9: Τέλος επιστρέφουμε το τυχαίο **κανονικά** κατανεμημένο δείγμα x_{rnd}. Αυτή ακριβώς είναι η εφαρμογή του θεωρήματος στην ενότητα 2.3.4 , όπως περιγράφεται και στην Εικόνα 3-3.

Line4: Στη γραμμή αυτή γίνεται η κλίση της συνάρτησης $uniToOthers(\boldsymbol{x}, \boldsymbol{y}, y\_rnd)$ με το τυχαίο ομοιόμορφα κατανεμημένο δείγμα να έχει παραχθεί από τη συνάρτηση της R $runif(1)$, με όρισμα 1 για την παραγωγή ενός δείγματος. Το αποτέλεσμα της συνάρτησης $uniToOthers()$ εκχωρείται στη μεταβλητή $result$.

Υποσημειώσεις:

1. Όπως αναφέραμε παραπάνω, ο αλγόριθμος αυτός παράγει μία τυχαία ποιότητα κανονικά κατανεμημένη ή με άλλα λόγια, τυχαία αλλά φραγμένη από μία κανονική συνάρτηση πυκνότητας πιθανότητας, με γνωστή μέση τιμή και τυπική απόκλιση. Συνεπώς θα πρέπει να εκτελεστεί τόσες φορές όσος ο αριθμός των βάσεων στο read, για κάθε διαφορετικό read. Για παράδειγμα εάν χρειαζόμαστε 1 million reads δύο κατευθύνσεων, μήκους 150 βάσεων το καθένα έχουμε $2 \cdot 1.000.000 * 150 = 300.000.000$ εκτελέσεις. Ήδη έχει αρχίσει να γίνεται εμφανής η τεράστια πολυπλοκότητα του αλγορίθμου καθώς και η αναγκαιότητα για βελτιστοποιήσεις.
2. Ενώ είναι εύκολο να υπολογίσουμε και την χρονική αλλά και την χωρική πολυπλοκότητα του αλγορίθμου είναι ίσως ανώφελο, αφού χρησιμοποιούμε αρκετές έτοιμες συναρτήσεις της R για τις οποίες δεν γνωρίζουμε την πολυπλοκότητα.
3. Τα βήματα 1 – 2 μπορούν να εκτελεστούν μόνο μία φορά και απλά να χρησιμοποιείται το περιεχόμενο των μεταβλητών για τους παρακάτω υπολογισμούς σε μία επανάληψη για κάθε βάση.
4. Τέλος καλό θα ήταν να αναφερθεί ότι ο παραπάνω αλγόριθμος υλοποιήθηκε στην R χωρίς να δοθεί ιδιαίτερη σημασία στη χρησιμοποίηση αποδοτικών δομών δεδομένων της γλώσσας, τεχνικές αποδοτικού τρόπου γραφής R και εξοικονόμηση μνήμης προκειμένου ο κώδικας να είναι εύκολα αναγνώσιμος.

3.7 Time Improvements

Από τις πρώτες προσπάθειες εκτέλεσης άρχισε να γίνεται εμφανής η ανάγκη για βελτιστοποιήσεις στον τομέα του χρόνου εκτέλεσης του προγράμματος.

Οι στόχοι που τέθηκαν εξ' αρχής ήταν η ολοκλήρωση της παραγωγής μερικών εκατομμυρίων reads σε χρόνο όχι περισσότερο από μερικές ώρες.

Για αυτό τον λόγο η κύρια μέθοδος επιτάχυνσης της εκτέλεσης του προγράμματος που επιλέχθηκε είναι η παραλληλοποίηση του αλγορίθμου παραγωγής των δεδομένων και των αντίστοιχων ποιοτήτων τους, προκειμένου να γίνει πλήρης εκμετάλλευση των πολυπύρηνων προσωπικών υπολογιστών.

Το πρώτο βήμα λοιπόν, όπως αναφέρθηκε στις απαιτήσεις προγράμματος είναι ο χρήστης να μπορεί να επιλέξει τον αριθμό των πυρήνων που θα χρησιμοποιηθούν, δίνοντας τον ως όρισμα κατά την εκτέλεση.

Το δεύτερο βήμα είναι η εισαγωγή της βιβλιοθήκης "$doParallel$" και η δημιουργία των threads με την εντολή:

$$registerDoParallel(cores)$$

Με όρισμα $cores$ τον αριθμό των πυρήνων που έδωσε ο χρήστης.

Στη συνέχεια έχουμε τη δυνατότητα να χρησιμοποιήσουμε πληθώρα μεθόδων προκειμένου να παραλληλοποιήσουμε τα τμήματα του αλγορίθμου που μας ενδιαφέρουν. Στο πρόγραμμα του προσομοιωτή χρησιμοποιήθηκε η συνάρτηση $foreach()$ για την παράλληλη εκτέλεση επαναληπτικής διαδικασίας, όπως θα εξηγήσουμε στη συνέχεια αναλυτικά.

Μετά την προ-επεξεργασία των δεδομένων εισόδου (πρότυπο δεδομένων), καταλήγουμε με μία δομή δεδομένων τύπου πίνακα της R, dataframe, με τα δεδομένα φορτωμένα σε μορφή πίνακα και την στήλη MeanCoverage πλήρως συμπληρωμένη. Υπενθυμίζουμε ότι πλέον ο αριθμός MeanCoverage είναι ο αριθμός των φορών που θα δειγματοληπτηθεί το κάθε read. Στη συνέχεια πρόκειται να εκτελεστεί ο αλγόριθμος της παραγωγής δεδομένων και των αντίστοιχων ποιοτήτων τους σειριακά όπως εξηγήθηκε σε προηγούμενες ενότητες. Οι προσπάθειες παραλληλοποίησης που έγιναν είναι τρείς. Παρακάτω θα παρομοιάσουμε τα διαθέσιμα threads, δηλαδή αυτά που δεν χρησιμοποιούνται σε μία στιγμή, ως εργάτες (workers), για την ευκολότερη κατανόηση των προσπαθειών παραλληλοποίησης.

1 <u>Διάσπαση του προβλήματος σε επαναλήψεις ανά read:</u>
Η πρώτη προσπάθεια παραλληλοποίησης αφορά την διάσπαση του προβλήματος σε διεργασίες όπου ο κάθε εργάτης (worker) αναλαμβάνει την δειγματοληψία ενός amplicon, μία φορά από τα αριστερά προς τα δεξιά και μία από τα δεξιά προς τα αριστερά και την παραγωγή των αντίστοιχων ποιοτήτων τους. Αφορά δηλαδή την παραγωγή ενός και μόνο ζευγαριού reads. Η κάθε αυτή διεργασία ανατίθεται σε οποιοδήποτε thread είναι διαθέσιμο και έχει τελειώσει την εργασία του εκείνη τη στιγμή. Πρόκειται δηλαδή για μία επαναληπτική διαδικασία τόσων φορών όσο και το άθροισμα των τιμών των MeanCoverage όλων των amplicons του προτύπου, που η βιβλιοθήκη "$doParallel$" αναλαμβάνει να εκτελέσει παράλληλα μέσω της συνάρτησης $foreach()$. Πιο συγκεκριμένα κάθε επανάληψη αναλαμβάνεται από ένα ξεχωριστό thread. Μετά το πέρας κάθε διεργασίας – thread η συνάρτηση συνδυάζει τα αποτελέσματα σε μία λίστα και το ελευθερωμένο thread αναλαμβάνει την επόμενη δειγματοληψία. Η επιστρεφόμενη λίστα του κάθε thread περιλαμβάνει

2 dataframes, ένα για κάθε κατευθυνόμενο read. Μετά το πέρας κάθε thread η επιστρεφόμενη λίστα συνδυάζεται σε μία ενιαία και το κάθε dataframe μοιάζει όπως παρακάτω:

	AmpliconID	Sequence	Length	Q	Idx	Type
1	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 1...	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...	150	BCACBDACDCCDECFDDDCDGFDEFGFBHDGCHEDEH...	1	Ref
2	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 2...	NNTTTACAATACTGAAGTTATAGGAAACAAGCACAAAAG...	150	BBCCCCBDBCCBEDDECFDDFGDFFFEEDGEFDIEEFCH...	1	Ref
3	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 3...	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...	150	CCDEEDDEDDCDCDDCDDEADBHEDFBCCGDHFDFEH...	1	Ref
4	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 4...	ATGGAGCATGATCTGGTCCTTAAAGTATGCAGATCGCC...	150	DCBDDBBBDCCEDEDFDBEDEDBFFFDFDEEFFDHFDH...	1	Ref
5	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 5...	AGAGGGAGATGCTGGTGACTGGAGATGGTGAGAGCC...	150	BBECDCDECDDEDBFECECFEDDGFCEGFCCFFED@E...	1	Ref
6	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 6...	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...	150	BDBFBECCDCEDCCCEFFDCEDFEHFIHDGFHECHGFGF...	1	Ref
7	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 7...	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...	150	BBCCCCCCCDDFCDEECFCFCHCEDBDHGCFFGEEHF...	1	Ref
8	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 8...	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...	150	CBCBDBCBBDCDDDCDFDFFFDDCFECFFDCHFEFDFEF...	1	Ref
9	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 9...	GAGATGGTGAGAGCCTGGGCTGGGGTAGTTTTTAGT...	150	CCDBDBDCCCCEBCCGDCEDDGDFFDGFEFFGHGGGF...	1	Ref
10	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 1...	GGGGTAGTTTTTAGTCATGGGCTTATGGACCCTCCCC...	150	CDCCBC@CEFDCCDCBCDECDEEFDCFHFCGECHGHF...	1	Ref
11	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 1...	NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN...	150	BCCBCDDDBDDCFBBEFDDFFDEGHEFFFGEGEDDCEE...	1	Ref
12	CSF3R.exon.17.line.1.chr1.36931697.36932509_ 1...	AAGCACAAAAGGCCATTGGGTGGGGCTGGATGGAGC...	150	ACCEEDDCDDCCFCEEEEGDCAGFFFFEDFDBEHJEGFEJ...	1	Ref

Εικόνα 3-4 From left to right produced reads dataframe.

a. Η στήλη AmpliconId περιέχει τα ονόματα των amplicons όπως ακριβώς είναι καταγεγραμμένα στο πεδίο '@' του προτύπου αρχείου, ακολουθούμενα από ένα αύξον αριθμό της δειγματοληψίας.
b. Η στήλη Sequence περιέχει τις συμβολοσειρές βάσεων.
c. Η στήλη Length περιέχει το μήκος κάθε συμβολοσειράς για λόγους επαλήθευσης.
d. Η στήλη Q περιέχει τη συμβολοσειρά των ποιοτήτων για κάθε read κωδικοποιημένη σε ASCII κώδικα.
e. Η στήλη Idx περιέχει τον αριθμό της σειράς του amplicon στο πρότυπο αρχείο για λόγους επαλήθευσης. Αν για παράδειγμα είναι 1 το συγκεκριμένο read αφορά το amplicon στη γραμμή 1 του προτύπου και ούτω καθεξής.
f. Η στήλη Type περιέχει τον τύπο του amplicon που αφορά το συγκεκριμένο read. Για παράδειγμα 'Ref', 'Mut_x'.

Σύντομα έγινε εμφανές πως η μέθοδος αυτή δεν προσέφερε και πολλά από άποψη αποδοτικότητας και συνεπώς εγκαταλείφθηκε. Στο συμπέρασμα αυτό καταλήξαμε από το γεγονός ότι αφενός η εργασία που ανατέθηκε σε κάθε worker – thread ήταν μικρή, δηλαδή πολύ σύντομα οι workers τελείωναν τη δουλειά τους, αφετέρου ο συνδυασμός έστω και μικρών σε όγκο δεδομένων στο τέλος του κάθε thread είναι αρκετά χρονοβόρος και υπολογιστικά απαιτητικός. Γενικά είναι περισσότερο αποδοτική η ανάθεση βαριών εργασιών σε κάθε worker και ο συνδυασμός μεγάλου όγκου πληροφορίας στο τέλος, αφού η διαδικασία της ανάθεσης μιας εργασίας αλλά και ο συνδυασμός των δεδομένων - αποτελεσμάτων καταναλώνουν αρκετούς υπολογιστικούς πόρους. Παρόλα αυτά με την ανάθεση βαριών εργασιών πιθανών να αυξηθούν ραγδαία και οι απατήσεις του προγράμματος σε μνήμη, οπότε πρέπει ο σχεδιαστής του αλγορίθμου να είναι αρκετά προσεκτικός. Σε περίπτωση μη διαθεσιμότητας αρκετής μνήμης μπορούν να χρησιμοποιηθούν και άλλες τεχνικές όπως για παράδειγμα εγγραφή ενδιάμεσων αποτελεσμάτων σε αρχεία.

2 <u>Διάσπαση του προβλήματος σε επαναλήψεις ανά amplicon:</u>
Η δεύτερη προσπάθεια παραλληλοποίησης αφορά την διάσπαση του προβλήματος σε διεργασίες όπου ο κάθε εργάτης (worker) αναλαμβάνει την δειγματοληψία ενός amplicon MeanCoverage φορές από τα αριστερά προς τα δεξιά και MeanCoverage φορές από τα δεξιά προς τα αριστερά και την παραγωγή των αντίστοιχων ποιοτήτων τους. Αφορά δηλαδή την παραγωγή τόσων ζευγαριών reads όσων αναγράφονται στη

στήλη MeanCoverage. Δηλαδή κάθε φορά που ένα worker είναι διαθέσιμος αναλαμβάνει να δειγματοληπτήσει ένα ολόκληρο amplicon. Στην περίπτωση αυτή η επαναληπτική διαδικασία προσπαελαύνει τις γραμμές του προτύπου αρχείου και τις αναθέτει ως διεργασίας στα ελεύθερα threads. Μετά το πέρας μίας διεργασίας τα αποτελέσματα αποθηκεύονται σε λίστα με δύο dataframe και συνδυάζονται σε μία ενιαία, όπως ακριβώς στην προηγούμενη περίπτωση, με την μόνη διαφορά ότι τώρα έχουμε μεγαλύτερο όγκο δεδομένων. Συμπερασματικά αναφέρουμε ότι αυτή η μέθοδος μείωσε αρκετά τον χρόνο εκτέλεσης και θεωρήθηκε πολύ αποτελεσματική. Πετύχαμε παραγωγή 2 εκατομμυρίων reads προς κάθε κατεύθυνση σε περίπου 4,5 ώρες.

3 Διάσπαση του προβλήματος σε επαναλήψεις ανά amplicon με ενδιάμεση εγγραφή δεδομένων σε αρχεία:
Κατά τη διάρκεια εκτέλεσης πειραμάτων, για ερευνητικούς σκοπούς θεωρήθηκε χρήσιμο να μεγαλώσουμε το μέγεθος των δεδομένων κατά μία τάξη μεγέθους. Αυτό επιτεύχθηκε με τον πολλαπλασιασμό της στήλης MeanCoverge του προτύπου αρχείου με το 10. Δυστυχώς η μέθοδος 2 που μέχρι τώρα ήταν η πιο αποτελεσματική απέτυχε στην παραγωγή ενός τόσο μεγάλου συνόλου δεδομένων λόγω περιορισμού μνήμης. Πιο συγκεκριμένα κάθε φορά που ένας worker τελείωνε τη δουλεία του, συνδύαζε τα δεδομένα του με τα ήδη παραχθέντα, ενώ η ενιαία λίστα διατηρούνταν στη μνήμη, μέχρι που αυτή εξαντλήθηκε. Ο τρόπος με τον οποίο αντιμετωπίστηκε αυτό το πρόβλημα είναι ο εξής: Επιλέγονται αρχικά τόσα amplicons προς επεξεργασία όσος και ο αριθμός των threads που επέλεξε ο χρήστης και ανατίθεται το καθένα σε διαφορετικό thread προς επεξεργασία. Μόλις ολοκληρωθεί η επεξεργασία όλων αυτών των amplicons που επιλέχθηκαν τα αποτελέσματα συνδυάζονται σε κοινή λίστα και κατόπιν εγγράφονται σε αρχείο. Στη συνέχεια αναλαμβάνονται τα επόμενα amplicons (threads σε αριθμό) και ούτω καθεξής. Συμπερασματικά αυτή η μέθοδος κάνει εξαιρετικά καλή διαχείριση μνήμης με κόστος την όχι και τόσο καλή εκμετάλλευση ενός πολυπύρηνου συστήματος. Αυτό συμβαίνει διότι από τα x threads που εκτελούνται, κάποια μπορεί να τελειώσουν σημαντικά νωρίτερα από κάποια άλλα, αν ο αριθμός MeanCoverage σε αυτά είναι αρκετά μικρότερος, και συνεπώς θα πρέπει να περιμένουν και τα υπόλοιπα να τελειώσουν χωρίς να αναλαμβάνουν καμία άλλη διεργασία, μέχρι τα αποτελέσματα να εγγραφούν στο αρχείο.

Τελικά διατηρήθηκαν δύο versions του προσομοιωτή, μία με την παραλληλοποίηση της δεύτερης μεθόδου για σχετικά μικρά σύνολα δεδομένων και μία με την παραλληλοποίηση της τρίτης μεθόδου για αρκετά μεγαλύτερα σύνολα. Μία πιθανή νέα μέθοδος παραλληλοποίησης θα αναφερθεί στο κεφάλαιο των επεκτάσεων.

3.8 Testing

Για την επαλήθευση της ακεραιότητας του αλγορίθμου δημιουργήθηκε πρόγραμμα επαλήθευσης των αποτελεσμάτων. Η διαδικασία επαλήθευσης των αποτελεσμάτων χωρίζεται σε δύο τμήματα. Το πρώτο αφορά την παραγωγή των δεδομένων, ενώ το δεύτερο αφορά την παραγωγή των ποιοτήτων.

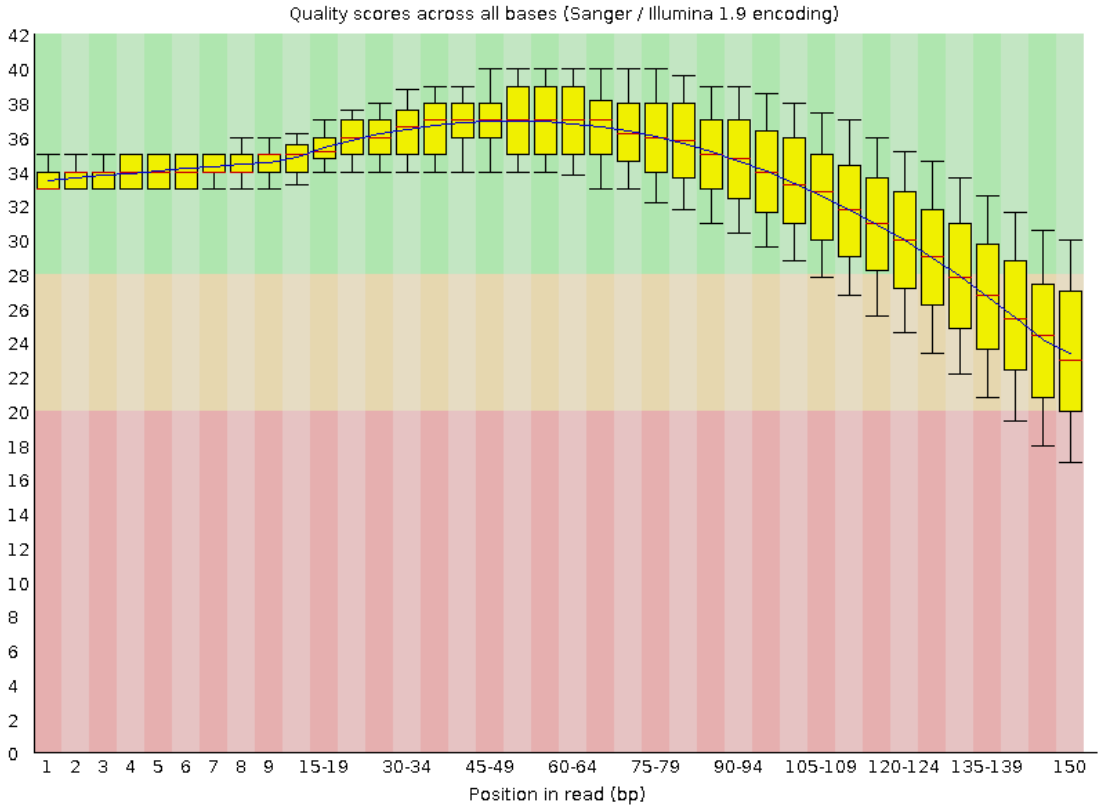
Για την επαλήθευση των παραχθέντων δεδομένων δημιουργήθηκε R script το οποίο εκτελεί τα παρακάτω βήματα:

1 Φορτώνονται τα τεχνητά reads που παρήχθησαν από τον προσομοιωτή σε μορφή λίστας. Η λίστα αυτή περιέχει δύο dataframes, ένα για τα reads αριστερής και ένα για τα reads δεξιάς κατεύθυνσης.
2 Σε όλα τα reads αντικαθίστανται όλοι οι χαρακτήρες 'N' με χαρακτήρες κενού.
3 Μετράτε και καταχωρείται σε διάνυσμα $Length$ το νέο μήκος όλων των reads ξεχωριστά. Σημειώνουμε ότι στο νέο μήκος δεν προσμετρώνται οι χαρακτήρες κενού του δεύτερου βήματος.
4 Σε επαναληπτική διαδικασία για κάθε read ελέγχεται αν η μέγιστη κοινή υπο-ακολουθία μεταξύ του read χωρίς τους χαρακτήρες 'N' και του read με τους χαρακτήρες 'N'. Εάν η μέγιστή κοινή υπο-ακολουθία ταυτίζεται με το read χωρίς τους χαρακτήρες 'N' τότε καταχωρείται σε διάνυσμα η τιμή TRUE, διαφορετικά η τιμή FALSE. Αναφέρουμε ότι, για την εύρεση της μέγιστης κοινής υπο-ακολουθίας χρησιμοποιήθηκε η συνάρτηση LCS όπως αναφέρεται στο documentation [14].
5 Στη συνέχεια ελέγχεται αν καταχωρήθηκε τιμή FALSE στο παραπάνω διάνυσμα. Αν όχι η επαλήθευση κατέληξε επιτυχώς. Αν ναι σημαίνει ότι δημιουργήθηκαν reads χωρίς καμία βάση και συνεπώς θα πρέπει να ελέγξουμε ξανά τον αλγόριθμο παραγωγής δεδομένων.

Για την επαλήθευση των παραχθέντων ποιοτήτων χρησιμοποιήθηκε το εργαλείο FastQC στην παρακάτω έκδοση:

$$FastQC\ v0.11.9$$

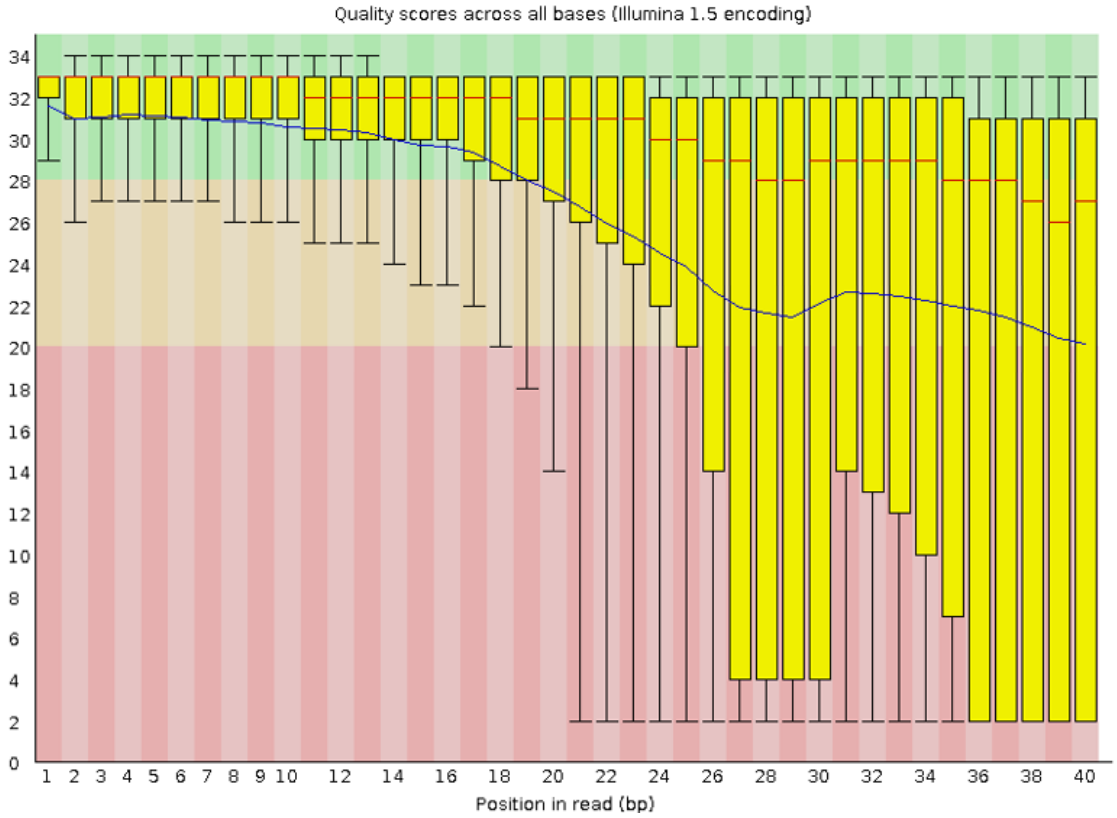
Στόχος αυτού του βήματος είναι να επιβεβαιώσουμε γραφικά αν οι παραχθείσες ποιότητες ακολουθούν τη συγκεκριμένη κατανομή που επέλεξε ο χρήστης, σε μορφή συστηματικής καταγραφής των μέσων τιμών και των τυπικών αποκλίσεων ανά θέση. Αφού εκτελέστηκε το FastQC η αναφορά έδωσε το εξής γράφημα:



Εικόνα 3-5 Per Base Qualities for artificial data - FastQC

Πρόκειται για ένα boxplot γράφημα στο οποίο βλέπουμε, μεταξύ άλλων, τη μέση τιμή ανά θέση (κόκκινη γραμμή) και την τυπική απόκλιση επίσης ανά θέση, των ποιοτήτων.

Συγκριτικά παραθέτουμε και ένα γράφημα πραγματικών δεδομένων που παράχθηκε από το FastQC.



Εικόνα 3-6 Per Base Qualities for real data - FastQC

Το αποτέλεσμα είναι πολύ καλό δεδομένου ότι ακόμη και αν τα γραφήματα διαφέρουν αρκετά, το πρόγραμμά μας έχει τη δυνατότητα να παράγει οποιαδήποτε κατανομή ποιοτήτων η οποία εξαρτάται αποκλειστικά από την επιλογή των μέσων τιμών και τυπικών αποκλίσεων από τον χρήστη.

4 Υλοποίηση Αλγοριθμικής Διαδικασίας

4.1 Ροή Εργασίας (pipeline)

4.1.1 Εισαγωγή

Έχοντας δημιουργήσει το τεχνητό σύνολο δεδομένων, ήρθε η στιγμή να τρέξουμε μία σειρά προγραμμάτων προκειμένου να αναλύσουμε τα δεδομένα που προκύπτουν από τα τεχνητά reads και στη συνέχεια να αποπειραθούμε να ‘ψάξουμε‘ με συγκεκριμένα εργαλεία για σωματικές μεταλλάξεις. Υπενθυμίζουμε ότι έχουν εισαχθεί γνωστές μεταλλάξεις και οι ποιότητες των reads ακολουθούν συγκεκριμένη και αρκετά ρεαλιστική κατανομή. Ο πρώτος στόχος αυτού του κεφαλαίου είναι να παραθέσουμε τις εντολές εκτέλεσης όλων των εργαλείων που χρησιμοποιήθηκαν (pipeline) και να τις αναλύσουμε ξεχωριστά. Ο δεύτερος στόχος είναι η λεπτομερής καταγραφή, σε μορφή πίνακα, των εργαλείων που χρησιμοποιήθηκαν και των εκδόσεών τους. Ο τρίτος και τελευταίος στόχος είναι ένας υποτυπώδης σχολιασμός του χρόνου εκτέλεσης του pipeline και των αποτελεσμάτων, καθώς και μία μικρή περιγραφή του τρόπου που χρησιμοποιήθηκε για να γίνει αυτοματοποιημένη η εκτέλεση του.

4.1.2 Script

```
1  fastqc /$r1Name /$r2Name -o /results
2  bwa index Homo_sapiens.GRCh37.dna.primary_assembly.fasta
3  bwa mem -M
   -t 4 Homo_sapiens.GRCh37.dna.primary_assembly.fasta
   /$r1Name /$r2Name > /SampleName.sam
4  picard CreateSequenceDictionary R
   =/Homo_sapiens.GRCh37.dna.primary_assembly.fasta O
   =/Homo_sapiens.GRCh37.dna.primary_assembly.dict
5  samtools faidx
   /Homo_sapiens.GRCh37.dna.primary_assembly.fasta
6  samtools view -@ 16 -Sb /SampleName.sam
                  > /SampleName.bam
7  samtools fixmate -@ 16
                  -O bam /SampleName.bam
                  /SampleName.fixed.bam
8  samtools sort -@ 16
                  -o /SampleName.fixed.sorted.bam
                  /SampleName.fixed.bam
9  samtools view -@ 16 -h -F 0x904
                  -b /SampleName.fixed.sorted.bam
                  > /SampleName.fixed.sorted.uniq.bam
10 picard AddOrReplaceReadGroups I
                  =/SampleName.fixed.sorted.uniq.bam O
                  =/SampleName.fixed.sorted.uniq.rg.bam RGID
                  =1 RGLB=lib1 RGPL=illumina RGPU
                  =unit1 RGSM=SampleName
11 samtools index /SampleName.fixed.sorted.uniq.rg.bam
12 gatk Mutect2 --max-reads-per-alignment-start 50
   --reference Homo_sapiens.GRCh37.dna.primary_assembly.fasta
   --input /SampleName.fixed.sorted.uniq.rg.bam
   --tumor-sample SampleName
   --output /SampleName_GATK_4_1_0_0_variants_MRPAS$i.vcf.gz
   > /SampleName.Mutect2_4_1_0_0_MRPAS$i.out 2>&1
13 gunzip /SampleName_GATK_4_1_0_0_variants_MRPAS$i.vcf.gz
```

Επεξήγηση εντολών:

1 **‘fastqc’** Δέχεται είσοδο τα αρχεία r1.name.fastq, r2.name.fastq των reads των δύο κατευθύνσεων και παράγει ως έξοδο μία στατιστική ανάλυση για το κάθε ένα. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [15].

2 **‘bwa index’** Δέχεται είσοδο το γονιδίωμα αναφορά σε μορφή fasta και δημιουργεί δείκτες για το γονιδίωμα αναφοράς. Πρακτικά δημιουργεί μία βάση δεδομένων για να καταστεί πιο εύκολή η μετέπειτα επεξεργασία. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [16].

3 **‘bwa mem’** Δέχεται είσοδο το γονιδίωμα αναφορά σε μορφή fasta καθώς και τα αρχεία r1.name.fastq, r2.name.fastq των reads των δύο κατευθύνσεων, αναλαμβάνει το local alignment των reads στο γονιδίωμα αναφοράς και παράγει έξοδο αρχείο ‘SampleName.sam’. Τα αρχεία .sam είναι αρχεία αποθήκευσης ‘στοιχισμένων’ reads και το πρότυπο που χρησιμοποιείται περιγράφεται λεπτομερώς εδώ [17]. Τέλος η

παράμετρος -t ακολουθείται από τον αριθμό ων πυρήνων που θέλουμε να χρησιμοποιηθούν. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [16].

4 **'picard CreateSequenceDictionary'** Δέχεται είσοδο το γονιδίωμα αναφορά σε μορφή fasta μετά την παράμετρο R= , και παράγει ως έξοδο ένα αρχείο με όνομα το όνομα του αρχείου εισόδου και κατάληξη .dict. Αυτό είναι ένα είδος ευρετηρίου που διευκολύνει την εύρεση τμημάτων στο γονιδίωμα αναφοράς και είναι απαραίτητο για την υπόλοιπη διαδικασία. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [18].

5 **'samtools faidx'** Δέχεται είσοδο το γονιδίωμα αναφορά σε μορφή fasta και παράγει ως έξοδο επιπλέον δείκτες σε αρχείο .fai, του γονιδιώματος αναφοράς απαραίτητους για την υπόλοιπη διαδικασία. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [19].

6 **'samtools view'** Δέχεται είσοδο το αρχείο εξόδου του βήματος 3, 'SampleName.sam' και το μετατρέπει στη δυαδική του μορφή 'SampleName.bam', χρησιμοποιώντας 16 πυρήνες μετά την εντολή '-@'. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [19].

7 **'samtools fixmate'** Δέχεται είσοδο το δυαδικό αρχείο εξόδου του βήματος 6, 'SampleName.bam' και παράγει ως έξοδο το αρχείο με όνομα 'SampleName.fixed.bam' τροποποιώντας ελαφρώς τις συντεταγμένες των reads και γεμίζοντας κάποιες κενές μεταβλητές, χρησιμοποιώντας 16 πυρήνες μετά την εντολή '-@'. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [19].

8 **'samtools sort'** Δέχεται είσοδο το δυαδικό αρχείο εξόδου του βήματος 7, 'SampleName.fixed.bam' και παράγει ως έξοδο το αρχείο με όνομα 'SampleName.fixed.sorted.bam', με τη διαφορά ότι πλέον όλα τα reads είναι ταξινομημένα στο αρχείο ανάλογα με την πιο αριστερή συντεταγμένη του κάθε read ως προς το γονιδίωμα αναφοράς, χρησιμοποιώντας 16 πυρήνες μετά την εντολή '-@'. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [19].

9 **'samtools view'** Δέχεται είσοδο το δυαδικό αρχείο εξόδου του βήματος 8, 'SampleName.fixed.sorted.bam' και παράγει ως έξοδο το αρχείο με όνομα 'SampleName.fixed.sorted.uniq.bam', το οποίο αφαιρεί ότι reads δεν έχουν αντιστοιχισθεί (unmapped) ή έχουν και δευτερεύουσες αντιστοιχίσεις (flag -F Ox904). Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [19].

10 **'picard AddOrReplaceReadGroups'** Δέχεται είσοδο το δυαδικό αρχείο εξόδου του βήματος 9, 'SampleName.fixed.sorted.uniq.bam' και παράγει ως έξοδο το αρχείο με όνομα 'SampleName.fixed.sorted.uniq.rg.bam', στο οποίο προστίθεται σε κάθε εγγραφή το όνομα του αντίστοιχου δείγματος. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [18].

11 **'samtools index'** Δέχεται είσοδο το δυαδικό αρχείο εξόδου του βήματος 10 'SampleName.fixed.sorted.uniq.rg.bam', και παράγει ως έξοδο επίσης δείκτες για το συγκεκριμένο αρχείο, απαραίτητους για την μετέπειτα επεξεργασία. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [19].

12 **'gatk Mutec2'** Πρόκειται για το εργαλείο εύρεσης μεταλλάξεων. Η εντολή αυτή δέχεται ως όρισμα εισόδου το --reference γονιδίωμα αναφοράς, το --input το δυαδικό αρχείο του βήματος 10 που πρακτικά είναι όλα τα τεχνητά reads μετά την προ-επεξεργασία και παράγει ως έξοδο δύο αρχεία. Το πρώτο --output είναι αρχείο με κατάληξη .vcf.gz (Variant Call Format) καθώς πρόκειται για ένα συμπιεσμένο format αποθήκευσης των μεταλλάξεων που έχουν βρεθεί και πληθώρα άλλων παραμέτρων. Το δεύτερο, με κατάληξη .out είναι το αρχείο εξόδου του προγράμματος στο οποίο τυπώνονται κυρίως διαγνωστικά χρήσης, όπως χρόνος εκτέλεσης πιθανά σφάλματα κτλ. Επιπλέον η εντολή αυτή δέχεται και πληθώρα παραμέτρων. Αυτή που μας ενδιαφέρει περισσότερο είναι το --max-reads-per-

alignment-start όπου στην προκείμενη περίπτωση τέθηκε ίση με 50. Πρόκειται για τον μέγιστο αριθμό reads ανά πιο αριστερή θέση στοίχισης που θέλουμε το σύστημα να συνυπολογίσει για την εύρεση μεταλλάξεων σε εκείνη την περιοχή. Για περισσότερες λεπτομέρειες ανατρέξτε εδώ [20].

13 **'gunzip'** Αυτή η εντολή αναλαμβάνει την αποσυμπίεση του αρχείου εξόδου μεταλλάξεων του προηγούμενου βήματος. Η έξοδος του είναι ένα ασυμπίεστο αρχείο με κατάληξη .vcf.

4.1.3 Εκδόσεις Tools

Tool	**Version**
samtools using htslib	1.7 1.7
picard	2.21.9
The Genome Analysis Toolkit (GATK)	4.1.6.0
HTSJDK	2.21.2
FastQC	0.11.9
BWA	bwakit-0.7.15_x64-linux

4.1.4 Σχόλια και Παρατηρήσεις

Αναφορικά ο χρόνος εκτέλεσης του script δεν ήταν πάνω από 5-6 ώρες συμπεριλαμβανομένου του indexing (εντολή 2). Σημειώνουμε ότι το indexing που προαναφέρθηκε είναι αναγκαίο να γίνει μόνο την πρώτη φορά της εκτέλεσης του pipeline. Αν για κάποιον λόγο θέλουμε να επαναλάβουμε την εκτέλεση του pipeline χρησιμοποιώντας το **ίδιο** γονιδίωμα αναφοράς, δεν χρειάζεται να ξανά εκτελέσουμε την γραμμή 2, με την προϋπόθεση να έχουμε κρατήσει τα απαραίτητα αρχεία εκείνου του βήματος, μειώνοντας έτσι δραματικά τον χρόνο εκτέλεσης στις 2 ώρες περίπου για έναν μέσο προσωπικό υπολογιστή με 4 πυρήνες και 8gb RAM. Τέλος αναφέρουμε ότι μετά το πέρας του παραπάνω script έχουμε πλέον στη διάθεση μας σε αρχείο, εύκολα χρησιμοποιήσιμο, όλες τις σωματικές μεταλλάξεις που βρέθηκαν από το GATK.

4.1.5 Αυτοματοποίηση Εκτέλεσης

Λαμβάνοντας υπόψιν την απλότητα του παραπάνω script αλλά και την ανάγκη για αυτοματοποιημένη εκτέλεση – στη συνέχεια θα χρειαστεί να τρέξουμε όλο το pipeline επαναλαμβανόμενα για διαφορετικές παραμέτρους εισόδου του GATK – αποφασίστηκε να χρησιμοποιήσουμε την απλή μέθοδο bash scripting. Συνεπώς όλες οι παραπάνω εντολές τοποθετήθηκαν σε ένα αρχείο script.sh. Στη συνέχεια, στις πρώτες γραμμές του αρχικοποιήθηκαν μεταβλητές με τα ονόματα των αρχείων των reads, την διαδρομή των αρχείων του γονιδιώματος αναφοράς, τη διαδρομή των αρχείων του προγράμματος BWA κτλ. Στη συνέχεια δόθηκε άδεια εκτέλεσης στο αρχείο script.sh και το script ήταν πλέον έτοιμο να εκτελεστεί σειριακά και αυτοματοποιημένα. Στη συνέχεια της διπλωματικής θα αναλύσουμε την προσπάθεια που έγινε σχετικά με την επαναληπτική εκτέλεση του script αλλάζοντας κάθε φορά τις παραμέτρους εισόδου του GATK. Για τον σκοπό αυτό χρησιμοποιήθηκε επαναληπτική διαδικασία τύπου for loop στο εσωτερικό του script με την μεταβλητή μετρητή i να παίρνει τις απαραίτητες τιμές και να τις εκχωρεί στα κατάλληλα ορίσματα εισόδου του GATK. Μια πιθανή επέκταση της αυτοματοποίησης της ροής του script αυτού θα αναφερθεί στο κατάλληλο κεφάλαιο των επεκτάσεων και βελτιώσεων.

4.2 Επεξεργασία Αποτελεσμάτων

Έχοντας δημιουργήσει ένα .vcf αρχείο για κάθε εκτέλεση του GATK με διαφορετική παράμετρο εισόδου --max-reads-per-alignment-start που επιλέξαμε από μία λίστα, κρίνεται πλέον απαραίτητη η δημιουργία ενός R script, το οποίο θα φορτώνει αυτά τα διαφορετικά αρχεία .vcf, θα βαθμολογεί τα αποτελέσματα ως προς τις γνωστές μεταλλάξεις που έχουν εισαχθεί και θα παράγει μία σύνοψη στατιστικών για κάθε διαφορετική εκτέλεση. Με αυτόν τον τρόπο θα μπορούμε όχι μόνο να δούμε στατιστικά για το αποτέλεσμα της κάθε εύρεσης αλλά και να κάνουμε μια σύγκριση μεταξύ τους ώστε να συμπεράνουμε ποια τιμή της παραμέτρου --max-reads-per-alignment-start δούλεψε καλύτερα και βοήθησε στο να βρεθούν περισσότερες μεταλλάξεις. Στη συνέχεια του παρόντος κεφαλαίου θα αναλύσουμε δύο R script που δημιουργήθηκαν για αυτόν τον σκοπό. Το πρώτο 'VcfComp.R' διαβάζει αναδρομικά από πλήθος φακέλων τα αρχεία .vcf τα αναλύει και παράγει ως έξοδο μία σύνοψη για κάθε ένα από αυτά. Το δεύτερο 'summary_of_iterations.R' διαβάζει αναδρομικά τις συνόψεις που παρήχθησαν από το πρώτο script και παράγει μία εντελώς καινούργια σύνοψη από όλες τις εκτελέσεις μαζί. Συνεπώς αυτή η ενότητα διαρθρώνεται σε δύο μεγάλα υπο-κεφάλαια στα οποία θα αναλυθούν ξεχωριστά τα δύο script, τα προβλήματα που προέκυψαν και οι λύσεις που δόθηκαν. Μελλοντικές επεκτάσεις και νέα πράγματα που θα μπορούσαν να γίνου προς βελτίωση αυτής της διαδικασίας θα αναφερθούν στο κεφάλαιο των μελλοντικών επεκτάσεων.

4.2.1 VcfComp.R

Σε αυτό το κεφάλαιο θα παραθέσουμε και κατόπιν θα εξηγήσουμε τις συναρτήσεις οι οποίες χρησιμοποιήθηκαν. Στη συνέχεια θα εξηγήσουμε τον κύριο κώδικα σε βήματα.

4.2.1.1 Συναρτήσεις

Στον παρακάτω πίνακα απεικονίζεται αριθμημένο το τμήμα δήλωσης του τύπου της κάθε συνάρτησης. Τα ορίσματα που αντιστοιχούν σε διανύσματα ή πίνακες σημειώνονται με έντονη γραφή:

1 $findPosIndexes <- function(vcf\_pos, \mathbf{startPos}, \mathbf{endPos})$
2 $isInsideBound <- function(\mathbf{indexes})$
3 $findIndelsType <- function(ref, alt)$
4 $changeIndelsFormat <- function(ref, alt, type)$
5 $findRelations <- function(\mathbf{templateRef}, \mathbf{templateAlt}, \mathbf{vfcRef}, \mathbf{vfcAlt}, \mathbf{posIndexes})$
6 $startDif <- function(string1, string2)$
7 $perPosOfSpeChrom <- function(data)$

1 Ορίσματα εισόδου:
   a. Μία συντεταγμένη ή θέση (position) που αντιστοιχεί σε μία μετάλλαξη (vcf_pos) που βρέθηκε και έχει αναφερθεί στο vcf αρχείο.
   b. Διάνυσμα όλων των πιο αριστερών (startPos) θέσεων των γνωστών μεταλλάξεων του προτύπου αρχείου ως προς το γονιδίωμα αναφοράς.
   c. Διάνυσμα όλων των πιο δεξιών (endPos) θέσεων των γνωστών μεταλλάξεων του προτύπου αρχείου ως προς το γονιδίωμα αναφοράς.

   Έξοδος:

a. Η συνάρτηση αυτή επιστρέφει διάνυσμα δεικτών στο οποίο έχουν καταγραφεί οι δείκτες γραμμών των διανυσμάτων startPos, endPos για τις οποίες η τιμή vcf_pos βρίσκεται ενδιάμεσα στις προηγούμενες δύο, δηλαδή:

$$\boldsymbol{startPos} <= vcf\_pos \,\&\, vcf\_pos <= \boldsymbol{endPos}$$

Σημείωση:

Η συνάρτηση σχεδιάστηκε ώστε να καλείται μέσω της οικογένειας συναρτήσεων apply() της R. Συνεπώς θα κληθεί για πληθώρα τιμών vcf_pos και συνεπώς θα επιστραφεί λίστα δεικτών. Κάθε γραμμή της λίστας θα αντιπροσωπεύει μία τιμή vcf_pos και κάθε στήλη έναν δείκτη των τιμών startPos, endPos στις οποίες θα βρίσκεται ανάμεσα.

2 Ορίσματα εισόδου:

a. Μονοδιάστατο διάνυσμα δεικτών (κάθε κελί του διανύσματος περιέχει και έναν δείκτη).

Έξοδος:

a. Η συνάρτηση αυτή επιστρέφει λογική τιμή TRUE/FALSE. Η τιμή αυτή προκύπτει από την καταμέτρηση των δεικτών διανύσματος εισόδου. Αν το διάνυσμα εισόδου περιέχει έναν ή περισσότερους δείκτες επιστρέφεται η τιμή TRUE. Διαφορετικά αν το διάνυσμα εισόδου δεν περιέχει κανέναν δείκτη, δηλαδή είναι μηδενικού μήκους επιστρέφεται η τιμή FALSE.

Σημείωση:

Η συνάρτηση σχεδιάστηκε ώστε να καλείται μέσω της οικογένειας συναρτήσεων apply() της R. Συνεπώς θα κληθεί για πληθώρα μονοδιάστατων διανυσμάτων δεικτών και συνεπώς θα επιστραφεί διάνυσμα πολλαπλών TRUE/FALSE.

3 Ορίσματα εισόδου:

a. String ref που υποδεικνύει τι υπήρχε στη θέση που βρέθηκε η μετάλλαξη.
b. String alt που υποδεικνύει τι βρέθηκε στη θέση της μετάλλαξης.

Έξοδος:

a. String που παίρνει τιμές μεταξύ των change, deletion, insertion. Η έξοδος αυτή προκύπτει από συγκρίσεις μήκους βάσεων μεταξύ των δύο ορισμάτων και πρακτικά πρόκειται για το είδος της μετάλλαξης.

Σημείωση:

Η συνάρτηση σχεδιάστηκε ώστε να καλείται μέσω της οικογένειας συναρτήσεων apply() της R. Συνεπώς θα κληθεί για δύο μονοδιάστατα διανύσματα. Το ένα θα περιέχει τιμές ref και το άλλο θα περιέχει τις αντίστοιχες τιμές alt. Συνεπώς θα επιστραφεί διάνυσμα πολλαπλών τιμών change/deletion/insertion, μία τιμή για κάθε ζευγάρι ref, alt.

4 Ορίσματα εισόδου:

a. String ref που υποδεικνύει τι υπήρχε στη θέση που βρέθηκε η μετάλλαξη.
b. String alt που υποδεικνύει τι βρέθηκε στη θέση της μετάλλαξης.

c. String type μεταξύ των τιμών change/deletion/insertion όπως περιεγράφηκε στην παραπάνω συνάρτηση.

Έξοδος:

a. String ref που υποδεικνύει τι υπήρχε στη θέση που βρέθηκε η μετάλλαξη.
b. String alt που υποδεικνύει τι βρέθηκε στη θέση της μετάλλαξης.
Με τη διαφορά σε σχέση με τα ορίσματα εισόδου ότι θα πλέον θα βρίσκονται σε διαφορετικό format. Για παράδειγμα:

Πίνακας 4-1 Επεξήγησης Indels Gatk vs Template format

Type	**GATK Format**		**Template Format**	
	Ref:	Alt:	Ref:	Alt:
Change	A	T	A	T
Deletion	TG	T	G	_
Insertion	G	GTTA	_	TTA

Η διαφοροποίηση έγκειται στο γεγονός ότι το GATK κρατάει πάντα την προηγούμενη βάση από τη μετάλλαξη που βρέθηκε. Στην περίπτωση του insertion προσθέτει τις βάσεις που εισήχθησαν στα δεξιά της προ υπάρχουσας βάσης. Στην περίπτωση του deletion αφαιρεί τις βάσεις που διαγράφηκαν από τα δεξιά της προ υπάρχουσας βάσης. Σε αντιπαράθεση έρχεται το πρότυπο στο οποίο χρησιμοποιούμε συμβολισμό με κάτω παύλες για να χαρακτηρίσουμε την διαγραφή κάποιων βάσεων ή την προσθήκη άλλων σε κενό.

Σημείωση:

Η συνάρτηση σχεδιάστηκε ώστε να καλείται μέσω της οικογένειας συναρτήσεων apply() της R. Συνεπώς θα κληθεί για δύο μονοδιάστατα διανύσματα. Το ένα θα περιέχει τιμές ref ενώ το άλλο τις αντίστοιχες τιμές alt σε GATK format. Θα επιστραφεί data.frame δύο στηλών, για τις τιμές ref, alt σε template.

5 Ορίσματα εισόδου:

a. Μονοδιάστατο διάνυσμα των τιμών ref του template.
b. Μονοδιάστατο διάνυσμα των τιμών alt του template.
c. Μονοδιάστατο διάνυσμα των τιμών ref του αρχείου vcf.
d. Μονοδιάστατο διάνυσμα των τιμών alt του αρχείου vcf.
e. Λίστα δεικτών. Οι γραμμές της οποίας αντιστοιχούν στις μεταλλάξεις που βρέθηκαν από το GATK και οι στήλες στους δείκτες των alt amplicons του template σύμφωνα με τη σχέση της συνάρτησης 1. Για παράδειγμα η πρώτη γραμμή αντιστοιχεί στην GATK μετάλλαξη 1 και οι δείκτες που ακολουθούν στις επόμενες στήλες υποδηλώνουν μεταξύ ποιων amplicons τις συντεταγμένες (εκκίνησης και τέλους) η μετάλλαξη αυτή βρίσκεται ανάμεσα.

Έξοδος:

Η συνάρτηση αυτή επιστρέφει ένα dataframe δύο στηλών, η πρώτη εκ των οποίων είναι ο αριθμός γραμμής της μετάλλαξης του template ενώ η δεύτερη είναι ο αριθμός γραμμής της μετάλλαξης του vcf αρχείου που αναφέρθηκε

από το GATK. Μεταξύ αυτών των δύο έγινε ταύτιση. Δηλαδή μία μετάλλαξη που βρέθηκε από το GATK ταυτίστηκε με μία γνωστή από το template και ως προς τα όρια θέσεων και ως προς την μετάλλαξη, βάση προς βάση. Μπορεί να πει κανείς ότι αυτός είναι ένας πίνακας σχέσης (φιλίας) μεταξύ των γνωστών μεταλλάξεων του template και των μεταλλάξεων που αναφέρθηκαν από το GATK. Στο σημείο αυτό προκύπτει ένα πρόβλημα. Οι μεταλλάξεις που ταυτίστηκαν μπορεί να βρέθηκαν επακριβώς (βάση προς βάση) σε σχέση με αυτές του template αλλά σε διαφορετική θέση ως προς το γονιδίωμα αναφοράς. Παρόλα αυτά σίγουρα θα βρέθηκαν μεταξύ των ορίων θέσεων $[startPos, endPos]$ που υποδεικνύει το template.

6 Ορίσματα εισόδου:
   a. String 1.
   b. String 2.

   Έξοδος:

   a. Επιστρέφεται ο δείκτης – χαρακτήρα - των strings για τον οποίο εντοπίζεται η πρώτη διαφορά (σε χαρακτήρα) μεταξύ τους. Ο επιστρεφόμενος δείκτης μπορεί να πάρει τιμές μεταξύ του 1 και του μήκους του μικρότερου string.

4.2.1.2 Κύριος Κώδικας

Στο σημείο αυτό εξηγούμε τον κύριο κώδικα που αναλαμβάνει την κλήση των συναρτήσεων τη διαχείριση των δομών και συνεπώς την επεξεργασία των δεδομένων.

1 Βήμα: Στο βήμα αυτό εκχωρούνται τα ορίσματα εισόδου (vcf file path, template file path, results path, coordinates file path) σε μεταβλητές και στη συνέχεια γίνονται οι απαραίτητοι έλεγχοι ακεραιότητας για τα ορίσματα αυτά.
2 Βήμα: Στο βήμα αυτό γίνεται ο έλεγχος για το αν ο αριθμός των γραμμών του αρχείου vcf είναι μηδενικός. Αν είναι μηδενικός το πρόγραμμα διακόπτεται αφού σημαίνει ότι δεν βρέθηκε καμία μετάλλαξη. Σε διαφορετική περίπτωση η εκτέλεση του προγράμματος συνεχίζεται κανονικά.

Υποσημείωση: Πριν προχωρήσουμε στο παρακάτω βήμα πρέπει να αναφέρουμε ότι κατά τη διάρκεια συγγραφής του κώδικα βρέθηκε πιο αξιόπιστη πηγή των συντεταγμένων αρχής και τέλους των ref amplicons (αρχείο coordinates), από την εξαγωγή τους μέσω του ampliconID. Το αρχείο αυτό διαβάζεται επίσης στο πρώτο βήμα.

3 Βήμα: Σε αυτό το βήμα εκτελείται ένα sanity check μεταξύ των συμβολοσειρών βάσεων του αρχείου coordinates και των ref amplicons του template προκειμένου να βεβαιωθούμε αν το αρχείο coordinates είναι πλήρες και αφορά όλα τα refs του template. Ταυτόχρονα δημιουργείται και ένας πίνακας φιλίας μεταξύ των αριθμών των γραμμών του coordinates και των αριθμών των γραμμών του template, διότι μπορεί να είναι σε διαφορετική σειρά μεταξύ τους.
4 Βήμα: Γνωρίζοντας τη συντεταγμένη εκκίνησης όλων των refs του template, γνωρίζουμε και τη θέση εκκίνησης όλων των αλληλόμορφων. Συνεπώς αν προσθέσουμε σε αυτή τη θέση τον δείκτη της πρώτης διαφορετικής βάσης μεταξύ ενός ref και του αντίστοιχου αλληλόμορφού του, προκύπτει η συντεταγμένη της μετάλλαξης. Το βήμα αυτό λοιπόν, αναλαμβάνει τον υπολογισμό των συντεταγμένων των μεταλλάξεων για όλα τα alt amplicons. Ο δείκτης που πρέπει να προστεθεί στη

θέση εκκίνησης υπολογίζεται από την κλήση της συνάρτησης startDif(), με ορίσματα ένα ref και το αντίστοιχο αλληλόμορφό του alt.

5 Βήμα: Στο βήμα αυτό διαχωρίζονται από τον πίνακα template τα ref amplicons με τα alt amplicons. Πλέον στον πίνακα template υπάρχουν καταχωρημένα μόνο τα alt amplicons, ενώ στον πίνακα templateRefs υπάρχουν καταχωρημένα μόνο τα ref amplicons.

6 Βήμα: Σε αυτό το βήμα καλείται η συνάρτηση findPosIndexes() μέσω της οικογένειας συναρτήσεων apply(). Το σχήμα συναρτήσεων καλείται για όλες τις συντεταγμένες των μεταλλάξεων που αναφέρθηκαν από το GATK, και τους πίνακες των θέσεων εκκίνησης και τέλους όλων των alt amplicon του template. Το αποτέλεσμα είναι ο πίνακας δεικτών όπως αναφέρεται στη συνάρτηση 1.

7 Βήμα: Έχοντας υπολογίσει τον πίνακα δεικτών από το παραπάνω βήμα καλούμε τη συνάρτηση isInsideBound() μέσω της οικογένειας συναρτήσεων apply(). Το αποτέλεσμα είναι ένα πίνακας λογικών τιμών για όλες τις συντεταγμένες των μεταλλάξεων που αναφέρθηκαν από το GATK, οι οποίες υποδεικνύουν πότε μία μετάλλαξη βρίσκεται εντός ορίων αρχής και τέλους έστω σε ένα amplicon του template.

8 Βήμα: Σε αυτό το βήμα καλείται η συνάρτηση findIndelsType() μέσω της οικογένειας συναρτήσεων apply(). Το αποτέλεσμα είναι διάνυσμα τιμών change/insertion/deletion, μήκους όσος ο αριθμός των μεταλλάξεων που αναφέρθηκαν από το GATK.

9 Βήμα: Σε αυτό το βήμα καλείται η συνάρτηση changeIndelsFormat() μέσω της οικογένειας συναρτήσεων apply(), για κάθε μετάλλαξη που αναφέρθηκε από το GATK. Το αποτέλεσμα είναι η αλλαγή του format των στηλών ref, alt σε format συμβατό με αυτό του template.

10 Βήμα: Σε αυτό το βήμα καλείται η συνάρτηση findRelations() με σκοπό το ταίριασμα των μεταλλάξεων που βρέθηκαν από το GATK και τις γνωστές του template. Το αποτέλεσμα είναι ένα πίνακας φιλίας δύο στηλών. Η πρώτη έχει το templateID ενώ η δεύτερη έχει το vcfID. Για κάθε γραμμή σημαίνει ότι υπάρχει και μία αντιστοιχία.

Υποσημείωση: Έχοντας βρει τον πίνακα φιλίας μεταξύ των μεταλλάξεων του GATK και αυτών του template, είναι εύκολο πλέον να κάνουμε μία καταγραφή σε νέους πίνακες τις μεταλλάξεις που μας ενδιαφέρουν. Για λόγους απλότητας απαιτείται η κατασκευή ενός ενιαίου πίνακα που θα έχει καταγεγραμμένες όλες τις μεταλλάξεις που χαρακτηρίζονται ως True Positives, True Negatives, False Positives, False Negatives. **There was a mutex is a positive class**.

- **True Positive (TP)** => Χαρακτηρίζεται μία μετάλλαξη η οποία αναφέρθηκε από το GATK και ταυτόχρονα υπήρχε στις γνωστές εισαχθείσες. Με άλλα λόγια έγινε πλήρης ταύτιση.
- **False Positive (FP)** => Χαρακτηρίζεται μία μετάλλαξη η οποία αναφέρθηκε από το GATK αλλά στην πραγματικότητα δεν υπήρχε στις γνωστές εισαχθείσες. Σε αυτήν την περίπτωση διακρίνουμε τις εξής υποπεριπτώσεις:
    - Η μετάλλαξη βρέθηκε τελείως εκτός ορίων οποιουδήποτε amplicon του template.
    - Η μετάλλαξη βρέθηκε εντός ορίων κάποιου amplicon του template αλλά είναι λανθασμένη.

- **True Negative (TN)** => Χαρακτηρίζεται μία μετάλλαξη η οποία δεν βρέθηκε από το GATK και δεν υπήρχε πουθενά στο template. Οι μεταλλάξεις αυτές δεν καταγράφονται για ευνόητους λόγους.
- **False Negative (FN)** => Χαρακτηρίζεται μία μετάλλαξη η οποία δεν αναφέρθηκε από το GATK αλλά υπήρχε στις γνωστές εισαχθείσες του template.

Τα δεδομένα που πρέπει να καταγράψουμε (στήλες) είναι τα παρακάτω:

.Template => τιμή που είναι καταγεγραμμένη στο template
.Data => τιμή που είναι καταγεγραμμένη στο vcf fille

1. CHROM => Χρωμόσωμα
2. POS.Template => Θέση μετάλλαξης
3. REF.Template => Reference
4. ALT.Template => Alternative αλληλόμορφο
5. AF.Template => Allele frequency
6. DP.Template => Depth
7. IsInsideBoundaries.Data => Λογική τιμή, αν βρίσκεται εντός ορίων ενός amplicon
8. Positive => True/False
9. POS.Data => Θέση μετάλλαξης
10. IndelType.Data => change/insertion/deletion
11. REF.data => Reference
12. ALT.Data => Alternative αλληλόμορφο
13. AF.Data => Allele frequency
14. AD.Data => Allele depth (splited in x1, x2)
15. AD.Template => Allele depth, Υπολογίζεται από : DP.Template * AF.Template
16. DP.Data => Info Depth
17. MBQ.Data => Mean Base Quality
18. MMQ.Data => Mean Mapping Quality
19. DiffDP => DP.Template - $DP.Data_{(info)}$
20. DiffAD => AD.Template - $AD.Data_{(x2)}$
21. DiffAF => AF.Template - AF.Data
22. PerDiffInfo.DP => (AD.Template - $DP.Data_{(info)}$) / AD.Template
23. PerDiffFormat.DP => (AD.Template - $DP.Data_{(format)}$) / AD.Template
24. PerDiffAD => (AD.Template – ($AD.Data_{(x1)}$ + $AD.Data_{(x2)}$)) / AD.Template
25. AbsoluteMatchingTemplateId => Για TP το id του τροποποιημένου template που ανήκει.
26. AbsoluteMatchingOfficialTemplateId => Για TP το id του αρχικού template που ανήκει.
27. Sequence => Συμβολοσειρά βάσεων. Υπάρχει μόνο στα δεδομένα template.
28. StartPos.Template => Συντεταγμένη εκκίνησης ενός amplicon
29. Negative => True/False

Υποσημείωση για τις τιμές AD, DP_{format}, DP_{info}:

- Οι τιμές AD περιλαμβάνουν αφιλτράριστα reads, αλλά δεν συμπεριλαμβάνουν uninformative reads.
- Οι τιμές DP συμπεριλαμβάνουν uninformative reads.
    - Οι τιμές DP_{format} είναι για φιλτραρισμένα reads.
    - Οι τιμές DP_{info} είναι για αφιλτράριστα reads.

Γενικά οι παραπάνω όροι του φιλτραρίσματος και των uninformative reads είναι κάπως περίπλοκοι. Για περισσότερες λεπτομερείς μπορείτε να ανατρέξετε εδώ [21].

11 Βήμα: Σε αυτό το βήμα κατασκευάζουμε δύο πίνακες με στήλες αυτές που αναφέρθηκαν στην υποσημείωση του βήματος 10. Ο πρώτος αφορά τις μεταλλάξεις

οι οποίες, σύμφωνα με τον πίνακα φιλίας, έχουν κάποια σχέση με αυτές του template και ο δεύτερος όλες οι μεταλλάξεις του template που δεν έχουν καμία σχέση, δηλαδή όλα τα **FN**. Στη συνέχεια μετονομάζουμε τις στήλες σε κατάλληλα ονόματα και κάνουμε merge τους δύο πίνακες. Αν ένας από τους δύο πίνακες είναι κενός, δηλαδή αν όλες οι μεταλλάξεις **έχουν** κάποια σχέση μεταξύ τους ή αν όλες οι μεταλλάξεις **δεν έχουν** κάποια σχέση μεταξύ τους το merging θα εκτελεστεί κανονικά με τον έναν εκ των δύο πίνακα κενό. Ο πίνακας μετά το merging ονομάζεται **final_res**.

12 Βήμα: Σε αυτό το βήμα εγγράφουμε τον πίνακα final_res του παραπάνω βήματος σε αρχείο με όνομα που ακολουθεί το εξής format: results.table_ημερομηνία παραγωγής_MRPASxxx όπου το MRPAS προκύπτει από τα αρχικά της παραμέτρου του GATK --max-read-per-alignment-start και ο αριθμός xxx είναι η τιμή αυτής της μεταβλητής της συγκεκριμένης εκτέλεσης.

13 Βήμα: Σε αυτό το βήμα κατασκευάζουμε συνοπτική λίστα με πλήθος στατιστικών (summary). Υπολογίζονται τα εξής:

- Template’s number of mutexes
- Template’s overall depth
- GATK’s number of mutexes
- GATK’s overall INFO DP
- GATK’s overall FORMAT DP
- GATK’s overall AD
- Uninformative reads (INFO.DP - AD)
- TruePositives
- FalsePositives
- FalseNegatives
- Data that is inside boundaries
- Data that isn't inside boundaries
- GATK % of insertions
- GATK % of deletions
- GATK % of changes
- % TruePositives
- % FalsePositives
- % FalseNegatives
- % of positives of specific CHROM
- % of Positives that is inside boundaries
- Mean of absolutes values of difference of AF
- SD of absolutes values of difference of AF
- Mean of absolutes values of percentage DP - INFO.DP
- SD of absolutes values of percentage DP - INFO.DP
- Mean of absolutes values of percentage DP - FORMAT.DP
- SD of absolutes values of percentage DP - FORMAT.DP
- Mean of absolutes values of percentage DP – AD
- SD of absolutes values of percentage DP – AD

14 Βήμα: Εγγραφή της παραπάνω λίστας σε αρχείο.

15 Βλέπε παράρτημα στο τέλος του εγγράφου.

Συνοψίζοντας το παραπάνω πρόγραμμα δέχεται ως είσοδο τα γνωστά δεδομένα του template και την πρόβλεψη του GATK σε vcf format και εξάγει έναν ενιαίο πίνακα

αποτελεσμάτων με τις πληροφορίες – στήλες που μας ενδιαφέρουν και ένα summary στατιστικών.

4.2.2 summary_of_iterations.R

Έχοντας κατασκευάσει πίνακες με τις πληροφορίες που μας ενδιαφέρουν για κάθε εκτέλεση του GATK με διαφορετικές παραμέτρους εισόδου ή με διαφορετική τιμή μίας παραμέτρου, έχει φτάσει πλέον η στιγμή να παραθέσουμε σε έναν ενιαίο πίνακα ίδιες στήλες διαφορετικών εκτελέσεων τη μία δίπλα στην άλλη, ώστε να μπορέσουμε να τις συγκρίνουμε μεταξύ τους και να εξάγουμε αποτελέσματα. Είναι πολύ πιθανό να κριθεί στη συνέχεια χρήσιμη η δημιουργία γραφημάτων κοινών στηλών για όλες τις διαφορετικές εκτελέσεις του εργαλείου εύρεσης μεταλλάξεων πχ. GATK. Στη συνέχεια αυτής της παραγράφου θα αναλύσουμε λεπτομερώς τα βήματα ενός νέου script που δημιουργήθηκε για τον παραπάνω σκοπό.

4.2.2.1 Ανάλυση Κώδικα

1 Βήμα: Στο βήμα αυτό εκχωρούνται τα ορίσματα εισόδου, σε μεταβλητές και γίνονται οι απαραίτητοι έλεγχοι εγκυρότητας. Μεταξύ άλλων το όρισμα που μας ενδιαφέρει ιδιαίτερα είναι το --columns. Πρόκειται για ένα comma delimited string στο οποίο καταγράφονται οι στήλες του πίνακα του προηγούμενου script, που μας ενδιαφέρει να κρατήσουμε τη μία δίπλα στην άλλη. Με άλλα λόγια μπορούμε να επιλέξουμε ποιες στήλες θέλουμε το script να αντιπαραθέσει τη μία δίπλα στην άλλη σε ομάδες για τις διαφορετικές εκτελέσεις του GATK όπως παρακάτω.

Έστω 5 εκτελέσεις για διαφορετικές τιμές --max-read-per-alignment-start που ανήκουν στο διακριτό σύνολο $[50, 100, 1000, 1500, 2000]$.

Έστω επίσης ότι επιλέγουμε τις στήλες που υποδεικνύονται μέσω του string εισόδου --columns = AD.Template,AF.Template,POS.Data,REF.data. Τότε το αποτέλεσμα θέλουμε να είναι ένας πίνακας με την παρακάτω μορφή.

AD. Template	AF. Template	MRPAS50. POS.Data	MRPAS50. REF.data	MRPAS100. POS.Data	MRPAS100. REF.data	MRPAS1000. POS.Data	MRPAS1000. REF.data	MRPAS1500. POS.Data	MRPAS1500. REF.data	MRPAS2000. POS.Data	MRPAS2000. REF.data

Γίνεται λοιπόν εμφανές ότι είναι σημαντικό να μην υπάρχουν επαναλήψεις κοινών στηλών *.template και αυτές να τοποθετηθούν στην αρχή με σκοπό αναφοράς. Στη συνέχεια θέλουμε να τοποθετήσουμε όλες τις υπόλοιπες στήλες *.Data με ομαδοποίηση κατά την τιμή της παραμέτρου --max-read-per-alignment-start. Συνεπώς αφού διασπάσουμε το string μεταξύ των χαρακτήρων ‘,’ ξεχωρίζουμε τις στήλες σε αυτές που περιέχουν τη λέξη κλειδί ‘template’ και σε αυτές που περιέχουν τη λέξη ‘data’.

2 Βήμα: Σε αυτό το βήμα διαβάζουμε αναδρομικά όλα τα αρχεία που περιέχονται στον φάκελο από τον οποίο εκτελείται το script και καταχωρούμε σε λίστα αυτά των οποίων το όνομά τους περιέχει το sub-string ‘MRPAS’ όπως στο format: **‘results.table_ημερομηνία παραγωγής_MRPASxxx ’**. Έπειτα από το πλήρες όνομα του αρχείου κόβεται το κομμάτι ‘MRPASxxx’ και δημιουργείται μεταβλητή με το όνομα αυτό στην οποία καταχωρούνται τα δεδομένα του αρχείου με το ίδιο όνομα. Μετά την ολοκλήρωση αυτού του βήματος θα έχουμε δημιουργήσει τόσες

μεταβλητές όσα και τα αρχεία με τα αποτελέσματα του προηγούμενου script και με περιεχόμενο τον πίνακα αυτών.

3 Βήμα: Σε αυτό το βήμα αντιγράφουμε τις στήλες που αναφέρονται στο string εισόδου, από τους πίνακες με ονόματα MRPASxxx σε μία τελική δομή πίνακα ενώ προηγουμένως έχουμε κρατήσει ένα αντίγραφο των στηλών template στην ίδια δομή.

4 Βήμα: Σε αυτό το βήμα κάνουμε μία στρογγυλοποίηση και παίρνουμε την απόλυτη τιμή όλων των στοιχείων του τελικού πίνακα.
Υποσημείωση: Η στρογγυλοποίηση γίνεται στο τρίτο δεκαδικό ψηφίο με σκοπό την προβολή του τελικού πίνακα σε υπολογιστικό φύλλο excel. Το πρόβλημα που πρόκυπτε χωρίς στρογγυλοποίηση ήταν η υπερχείλιση τιμών σε κελιά του φύλου excel λόγω μεγάλου πλήθους δεκαδικών ψηφίων.

5 Βήμα: Τέλος εγγράφουμε τον τελικό πίνακα του παραπάνω βήματος σε αρχείο με όνομα 'overall_summary.table'.

Συνοψίζοντας το παραπάνω πρόγραμμα δέχεται ως είσοδο τα αρχεία – πίνακες που προκύπτουν από το VcfComp.R script. Υπενθυμίζουμε ότι αυτοί οι πίνακες περιέχουν για όλες τις μεταλλάξεις TP, TN, FP, FN και πληθώρα στηλών δεδομένων. Μετά την εκτέλεση εξάγει έναν ενιαίο πίνακα αποτελεσμάτων με τις πληροφορίες – στήλες που μας ενδιαφέρουν και ένα summary στατιστικών που αφορούν το σύνολο των εκτελέσεων του GATK.

4.2.3 Σύνοψη

Σε αυτό το κεφάλαιο, κατασκευάσαμε δύο R scripts για την επεξεργασία των αποτελεσμάτων και την εξαγωγή χρήσιμων συμπερασμάτων. Στο πρώτο τμήμα κεφ. 4.2.1 εξηγούμε το script VcfComp.R το οποίο διαβάζει ένα αρχείο vcf αποτέλεσμα μίας εκτέλεσης του GATK και το συγκρίνει με τη γνωστή είσοδο template. Χαρακτηρίζει τις μεταλλάξεις ως TP, FP, TN, FN και κατασκευάζει συνοπτικό πίνακα με αυτές και πληθώρα άλλων στοιχείων που τις αφορούν. Τέλος κατασκευάζει και ένα summary στατιστικών που αφορούν την επιτυχία της συγκεκριμένης αυτής εκτέλεσης. Αφού το παραπάνω script εκτελεστεί για όλα τα vcf των διαφορετικών εκτελέσεων είναι χρήσιμο να κάνουμε και μία υποτυπώδη σύγκριση μεταξύ τους. Το στόχο αυτό έρχεται να καλύψει το επόμενο script, summary_of_iterations.R, του κεφαλαίου 4.2.2.

Προκυμμένου να συγκρίνουμε τα αποτελέσματα μεταξύ τους το πρόγραμμα διαβάζει τους πίνακες εξόδου του προηγούμενου βήματος και στη συνέχεια κατασκευάζει έναν νέο ενιαίο πίνακα με στήλες από τους προηγούμενους, τις οποίες επιλέγει ο χρήστης μέσω των ορισμάτων εκτέλεσης. Η επεξεργασία των αποτελεσμάτων περιλαμβάνει ακόμη ένα βήμα το οποίο όμως αποφασίστηκε να αναλυθεί στο επόμενο κεφάλαιο.

4.3 Επεξεργασία Αρχείων SAM

Κατά την επεξεργασία των αποτελεσμάτων προέκυψε η ανάγκη να βρούμε από το σύνολο των εκατομμυρίων reads ποια χρησιμοποιήθηκαν για την εύρεση των μεταλλάξεων, αλλά ταυτόχρονα και την κατανομή των θέσεων και ποιοτήτων των μεταλλάξεων αυτών στα reads. Με άλλα λόγια θέλουμε να βρούμε ποιες ήταν οι θέσεις και οι ποιότητες στις οποίες βρέθηκαν οι μεταλλάξεις ως προς τις συντεταγμένες των reads και τις συχνότητές με τις οποίες εμφανίζονται. Για παράδειγμα έστω ότι έχουμε 4 reads από το ίδιο amplicon μήκους 150 χαρακτήρων. Επιπλέον έστω ότι βρέθηκε μία μετάλλαξη και στα 4 reads. Δηλαδή και τα 4 αυτά reads περιείχαν την ίδια μετάλλαξη. Στο παράδειγμά μας το $position \in [1,150]$, $quality \in [0,42]$ βρέθηκαν όπως παρακάτω.

Πίνακας 4-2

Read Number	Position	Quality
A	75	40
B	75	40
C	60	30
D	55	25

Συνεπώς αυτό που θέλουμε να βρούμε είναι ότι στο σύνολο των χρησιμοποιημένων reads κατά το 50% οι μεταλλάξεις βρέθηκαν στη θέση 75 με ποιότητα 40, στο 25% στη θέση 60 με ποιότητα 30 και στο υπόλοιπο 25% στη θέση 55 με ποιότητα 25. Αυτά τα δεδομένα μπορούν κάλλιστα να απεικονισθούν και με ένα box plot [22].

Επιπλέον θέλουμε να βρούμε και την κατανομή των ποιοτήτων, σε όλες τις θέσεις, όλων των reads που χρησιμοποιήθηκαν για την εύρεση όλων των μεταλλάξεων. Με άλλα λόγια θέλουμε να κάνουμε αυτό που κάνει το FastQC στο πρώτο του διάγραμμα (Per base sequence quality) με τη μόνη διαφορά ότι θέλουμε να συμπεριλάβουμε σε αυτό μόνο τα reads που χρησιμοποιήθηκαν στην εύρεση των μεταλλάξεων.

Για την πραγματοποίηση των παραπάνω στόχων χρειαζόμαστε το αρχείο των στοιχισμένων reads με κατάληξη .sam ή το αντίστοιχο δυαδικό του με κατάληξη .bam και επιπλέον το προϊόν - πίνακα με τις TP, FP, TN, TP μεταλλάξεις, του script που περιγράφεται στο κεφάλαιο 4.2.1.

Στη συνέχεια θα περιγράψουμε τον αλγόριθμο με τον οποίο πετύχαμε τους παραπάνω στόχους, τα αρχεία δεδομένων που χρησιμοποιήθηκαν καθώς και τα διαγράμματα τα οποία εμφανίσαμε.

4.3.1 Περιγραφή Αλγορίθμου sam_cmp.R

Αρχικά θα εξηγήσουμε βήμα προς βήμα το φόρτωμα των δεδομένων, την προ-επεξεργασία αυτών καθώς και τις βιβλιοθήκες που χρησιμοποιήθηκαν. Κατόπιν θα παραθέσουμε τμήμα του κώδικα σε μορφή ψευδοκώδικα και θα εξηγήσουμε την κάθε γραμμή του αναλυτικά. Τέλος σημειώνουμε την αναφορά [17] για το documentation του sam format ώστε ο αναγνώστης να μπορεί να ανατρέξει για περισσότερες λεπτομέρειες και επεξηγήσεις των όρων που θα χρησιμοποιηθούν στη συνέχεια.

Βιβλιοθήκες που χρησιμοποιήθηκαν:

- **Rsamtools:** Χρησιμοποιήθηκε για την εισαγωγή – φόρτωμα των δεδομένων του .bam αρχείου και την βασική επεξεργασία τους μέσω της R.
- **stringr:** Χρησιμοποιήθηκε προκείμενου να γίνει εφικτή η χρησιμοποίηση regular expressions για την επεξεργασία strings.
- **ggplot2:** Χρησιμοποιήθηκε για την εμφάνιση ελκυστικών γραφημάτων.
- **reshape2:** Χρησιμοποιήθηκε για την επεξεργασία της δομής δεδομένων πριν αυτή εκχωρηθεί ως όρισμα σε συνάρτηση της βιβλιοθήκης **ggplot2**.

Περιγραφή βημάτων:

Βήμα 1: Φορτώνουμε τα δεδομένα από το .bam αρχείο. Στο σημείο αυτό αναφέρουμε ότι τα αρχεία αυτά είναι αρκετά μεγάλα και παρά το γεγονός ότι η βιβλιοθήκη **Rsamtools**

αναλαμβάνει να διαβάσει το αρχείο με αρκετά αποδοτικό τρόπο, οι απαιτήσεις σε μνήμη ram είναι αρκετά μεγάλες.

Βήμα 2: Φορτώνουμε τον πίνακα των επιβεβαιωμένων μεταλλάξεων ως απλό tab-delimited file.

Βήμα 3: Χρησιμοποιώντας τη συνάρτηση str_extract_all() της βιβλιοθήκης **stringr** και την regular expression ***|([0-9]+)|([MIDNSHPX=])** σπάμε στα cigar codes όπως το παρακάτω παράδειγμα:

Αρχικό cigar code: **15S35M45D18S**

Τελικό cigar code:

15	S	35	M	45	D	18	S

Τελικά προκύπτει ένας δυσδιάστατος πίνακας με γραμμές όσες και τα cigar codes που υπάρχουν στο αρχείο και στήλες όσες προκύπτουν από τη διάσπαση του μεγαλύτερου cigar code.

Βήμα 4: Εκχωρούμε σε ένα μονοδιάστατο διάνυσμα, lm_pos, τις left most coordinates όλων των reads από το .bam αρχείο. Πρόκειται για τις πιο αριστερές συντεταγμένες των reads ως προς τις συντεταγμένες του γονιδιώματος αναφοράς.

Βήμα 5: Δημιουργούμε ένα νέο μονοδιάστατο διάνυσμα, rm_pos, με τις πιο δεξιές συντεταγμένες όλων των reads, οι οποίες υπολογίζονται, προσθέτοντας στις πιο αριστερές το μήκος του read, αφαιρώντας όλους τους μη χρήσιμους χαρακτήρες 'N' όπως υποδεικνύει το cigar code και τέλος αφαιρώντας το 1. Δηλαδή αφαιρώντας το άθροισμα όλων των αριθμών που προηγούνται του 'S' χαρακτήρα στο cigar code.

$$rm_{pos_i} = lm_{pos_i} + length(read_i) - sum(s_i) - 1$$

Για παράδειγμα έστω read με πιο αριστερή θέση στο γονιδίωμα αναφοράς την συντεταγμένη 1900, μήκος read 150 χαρακτήρες και $sum(s_i) = 33$ όπως στο παραπάνω παράδειγμα, τότε η πιο δεξιά συντεταγμένη θα είναι: $rm_{pos_i} = 1900 + 150 - 33 - 1 = 2016$.

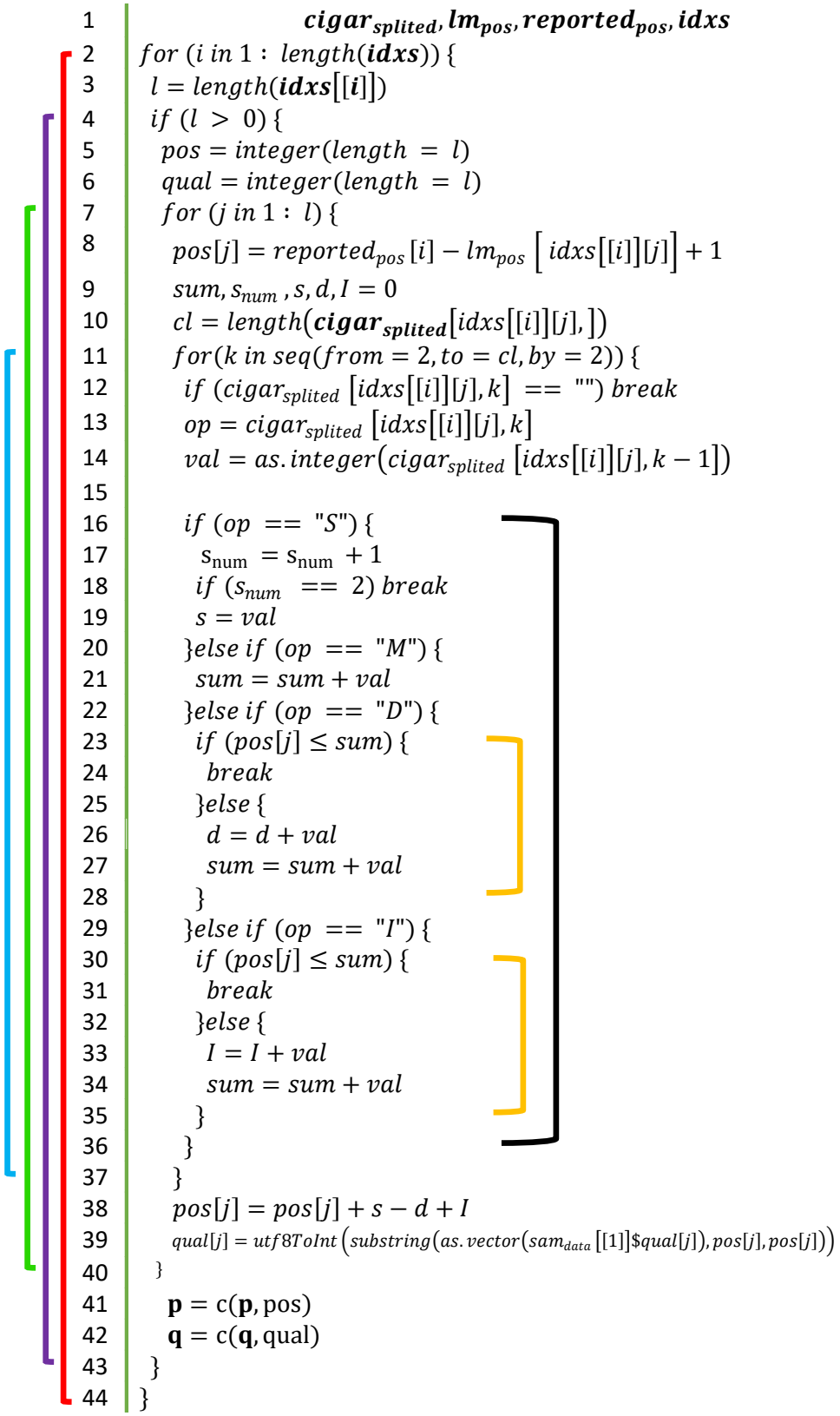
Βήμα 6: Στο σημείο αυτό αναζητούμε ποια από τα reads χρησιμοποιήθηκαν για την εύρεση της κάθε μίας μετάλλαξης. Γνωρίζοντας τις πιο δεξιές και τις πιο αριστερές συντεταγμένες των reads καθώς και τις θέσεις στις οποίες βρέθηκαν οι μεταλλάξεις μπορούμε εύκολα να βρούμε τους δείκτες των reads των οποίων οι συντεταγμένες συμπεριλαμβάνουν μία μετάλλαξη. Για παράδειγμα αν η θέση μίας μετάλλαξης βρίσκεται ενδιάμεσα στις συντεταγμένες ενός read σημαίνει ότι το δεύτερο χρησιμοποιήθηκε για την εύρεση της πρώτης. Συνεπώς κατασκευάζουμε μία λίστα φιλίας της οποίας οι γραμμές αντιπροσωπεύουν τις True Positive μεταλλάξεις, ενώ οι στήλες, έχουν αποθηκευμένους τους δείκτες των reads που χρησιμοποιήθηκαν για την εύρεση της εκάστοτε μετάλλαξης. Με έντονη γραφή σημειώνονται τα μονοδιάστατα διανύσματα. Η σχέση υπολογισμού είναι η εξής:

$$\boldsymbol{idxs_i} = which\left(\boldsymbol{lm_{pos}} \leq mut_{pos_i} \;\&\&\; mut_{pos_i} \leq \boldsymbol{rm_{pos}}\right)$$

Βήμα 7: Γνωρίζοντας τα cigar codes των reads αλλά, τα reads που χρησιμοποιήθηκαν για την εύρεση των μεταλλάξεων αλλά και τις απόλυτες θέσεις στις οποίες βρέθηκαν αυτές οι μεταλλάξεις ως προς το γονιδίωμα αναφοράς, μπορούμε πλέον να υπολογίσουμε τις σχετικές συντεταγμένες – θέσεις που βρέθηκαν αυτές οι μεταλλάξεις ως προς τις συντεταγμένες των reads οι οποίες κυμαίνονται από 0 – μήκος του read. Τα γράμματα του cigar code που θα χρησιμοποιηθούν στον παρακάτω αλγόριθμο περιορίζονται στα S,M,D,I.

a. S: Πρόκειται για τους χαρακτήρες ‘N’ που διαγράφηκαν από την αρχή και το τέλος του read, αν για παράδειγμα cigar code = 15S…35S σημαίνει ότι υπάρχουν 15 ‘N’ στην αρχή του read και 35 ‘N’ στο τέλος του, τα οποία δεν συμπεριλαμβάνονται στον υπολογισμό των συντεταγμένων.
b. M: Πρόκειται για τον αριθμό των βάσεων που ταυτίστηκαν πλήρως με ένα κομμάτι του γονιδιώματος αναφοράς. Αν για παράδειγμα έχουμε cigar code = …15M… σημαίνει ότι 15 βάσεις του read ταυτίστηκαν πλήρως με 15 βάσεις στη σειρά του γονιδιώματος αναφοράς.
c. D: Πρόκειται για το μήκος του κενού που δημιουργείται όταν ένα read σπάει σε δύο ή περισσότερα κομμάτια. Μπορεί δηλαδή το πρώτο μισό ενός read να ταυτιστεί πλήρως με μία περιοχή του γονιδιώματος αναφοράς ενώ το δεύτερο μισό να ταυτιστεί πλήρως με μία άλλη περιοχή παραπέρα, δημιουργώντας ένα κενό μεταξύ τους μήκους D θέσεων.
d. I: Πρόκειται για το μήκος του κενού που δημιουργείται αυτή τη φορά στο ref, όταν ένα read περιέχει τμήμα το οποίο δεν ταυτίζεται με καμία περιοχή στο γονιδίωμα αναφοράς. Μπορεί δηλαδή το πρώτο τμήμα ενός read να ταυτιστεί πλήρως με μία περιοχή του γονιδιώματος, το μεσαίο τμήμα να μην ταυτιστεί με καμία περιοχή του γονιδιώματος, ενώ το τελευταίο τμήμα να ταυτιστεί επίσης πλήρως, δημιουργώντας έτσι ένα νοητό κενό στις συντεταγμένες του γονιδιώματος αναφοράς. Το παραπάνω προκύπτει όταν έχουμε εισαγωγή (insertion) στο read.

Αρχίζει λοιπόν να γίνεται κατανοητό ότι για να υπολογίσουμε με ακρίβεια την θέση μίας μετάλλαξης σε ένα read ως προς τη σχετική συνταγμένη του read θα πρέπει να λάβουμε υπόψη στους υπολογισμούς μας όλα τα παραπάνω γράμματα του cigar code. Εξαιτίας της αυξημένης πολυπλοκότητας της λογικής που χρησιμοποιήθηκε για την επίτευξη της παραπάνω εργασίας, κρίνεται απαραίτητο να παραθέσουμε τον ψευδοκώδικα που χρησιμοποιήθηκε και να τον αναλύσουμε γραμμή προς γραμμή. Επιπλέον αναφέρουμε ότι με έντονη γραφή υποδεικνύονται οι μεταβλητές διανύσματα και πίνακες καθώς και το ότι χρησιμοποιούνται χρωματιστές αγκύλες στον ψυδοκώδικα και αντίστοιχος χρωματικός κώδικας στην ανάλυσή του με σκοπό την καλύτερη και ευκολότερη κατανόηση των τμημάτων του.



Line1: Θεωρούμε γνωστές τις λίστες $\boldsymbol{cigar_{splited}}, \boldsymbol{idxs}$ και τα μονοδιάστατα διανύσματα $\boldsymbol{lm_{pos}}, \boldsymbol{reported_{pos}}$ τα οποία υπολογίστηκαν και έχουν την μορφή όπως περιγράφηκαν στα παραπάνω βήματα.

Line2: Επαναληπτική διαδικασία κατά μήκος των γραμμών της λίστας $\boldsymbol{idxs}$, χρησιμοποιώντας τον δείκτη i. Υπενθυμίζουμε ότι η λίστα αυτή περιέχει τους δείκτες των reads που χρησιμοποιήθηκαν για την εύρεση της κάθε μετάλλαξης, η οποία αναπαρίσταται από τις γραμμές της ίδια της λίστας $\boldsymbol{idxs}$.

Line3: Εκχωρούμε το μήκος της $i-$οστής γραμμής της λίστας $\boldsymbol{idxs}$ στη μεταβλητή l.

Line4: Αν το μήκος της $i-$οστής γραμμής της λίστας $\boldsymbol{idxs}$, μεταβλητή l, είναι θετικό η συγκεκριμένη επανάληψη ως προς i θα συνεχιστεί, διαφορετικά, θα προχωρήσουμε στην επόμενη επανάληψη.

Line5: Αρχικοποιούμε κενό, μονοδιάστατο διάνυσμα, ακεραίων, $\boldsymbol{pos}$, μήκους l.

Line6: Αρχικοποιούμε κενό, μονοδιάστατο διάνυσμα, ακεραίων, $\boldsymbol{qual}$, μήκους l.

Line7: Επαναληπτική διαδικασία κατά μήκος των στηλών της $i-$οστής γραμμής της λίστας $\boldsymbol{idxs}$, χρησιμοποιώντας τον δείκτη j.

Line8: Εκτελείται η πράξη της $j-$στής reported θέσης μετάλλαξης μείον την πιο αριστερή συντεταγμένη του $\boldsymbol{idxs}[[i]][j]-$στού read συν 1. Πρακτικά αφαιρούμε από τη θέση της μετάλλαξης τη συντεταγμένη αρχής ενός read που χρησιμοποιήθηκε και προσθέτουμε 1. Με αυτόν τον τρόπο βρίσκουμε πόσο απέχει η μετάλλαξη από την αρχή του read.

Line9: Δημιουργούμε και μηδενίζουμε τις μεταβλητές: sum, s_{num}, s, d, l.

Line10: Εκχωρούμε στη μεταβλητή cl (αρχικά του ονόματος cigar length) το πλήθος των αριθμών και των γραμμάτων του cigar code του $\boldsymbol{idxs}[[i]][j]-$στού read, μετρώντας το μήκος της $\boldsymbol{idxs}[[i]][j]-$στής γραμμής του πίνακα $\boldsymbol{cigar_{splited}}$.

Line11: Επαναληπτική διαδικασία κατά μήκος των στηλών της $\boldsymbol{idxs}[[i]][j]-$στής γραμμής του πίνακα $\boldsymbol{cigar_{splited}}$, ξεκινώντας από το δεύτερο κελί με βήμα 2, χρησιμοποιώντας τον δείκτη k. Σε αυτό το σημείο θέλουμε να προσπελάσουμε μόνο τα γράμματα του cigar code αυτού του read. Υπενθυμίζουμε ότι τα γράμματα βρίσκονται στα κελιά με ζυγούς δείκτες.

Line12: Αν το $k-$στό γράμμα του $\boldsymbol{idxs}[[i]][j]-$στού cigar code είναι κενός χαρακτήρας, σπάμε την επαναληπτική διαδικασία που προσπελαύνει τα γράμματα του $\boldsymbol{idxs}[[i]][j]-$στού cigar code, διαφορετικά συνεχίζουμε παρακάτω.

Line13: Εκχωρούμε στη μεταβλητή op τον χαρακτήρα του cigar code από την θέση k.

Line14: Εκχωρούμε στη μεταβλητή val την τιμή του cigar code από την θέση $k-1$.

Line16: Αν ο χαρακτήρας $op = "S"$…

Line17: Αυξάνουμε τον μετρητή μέτρησης χαρακτήρων "S", s_{num} κατά 1…

Line18: Αν ο μετρητής s_{num} έχει φτάσει στην τιμή 2 σημαίνει ότι έχουμε φτάσει στο τέλος του read και σταματάμε τους υπολογισμούς σπάζοντας την επανάληψη κατά μήκος του cigar code…

Line19: Διαφορετικά κρατάμε στην μεταβλητή s την τιμή της μεταβλητής val. Αυτός ο αριθμός, πρακτικά, είναι το πλήθος των χαρακτήρων 'N' στην αρχή του εξεταζόμενου read.

Line20: Αν ο χαρακτήρας $op = "M"$…

Line21: Προσθέτουμε στην μεταβλητή sum την μεταβλητή val. Υπενθυμίζουμε ότι ο αριθμός "M" είναι ο αριθμός των βάσεων που έχουν ταυτιστεί απόλυτα. Συνεπώς κρατάμε άθροισμα αυτών των αριθμών για να ξέρουμε πόσο έχουμε «προχωρήσει» κατά μήκος του read.

Line22: Αν ο χαρακτήρας $op = "D"$…

Line23: Αν η απόσταση της μετάλλαξης από την αρχή του read, $pos[j]$, της γραμμής 8, είναι μικρότερη ή ίση της απόστασης που έχουμε «προχωρήσει» στο read…

Line24: σταματάμε τους υπολογισμούς σπάζοντας την επανάληψη κατά μήκος του cigar code, αφού, η μετάλλαξη βρίσκεται πριν το gap Εικόνα 4-1 …

Line26: διαφορετικά προσθέτουμε $d = d + val$ όπου val ο αριθμός των κενών και d τα είδη καταμετρημένα κενά…

Line27: και προσθέτουμε $sum = sum + val$ όπου val ο αριθμός των κενών και sum το πόσο έχουμε «προχωρήσει» στο read.

Line29: Αν ο χαρακτήρας $\mathrm{op} = \text{"I"}$…

Line30: Αν η απόσταση της μετάλλαξης από την αρχή του read, $pos[j]$, της γραμμής 8, είναι μικρότερη ή ίση της απόστασης που έχουμε «προχωρήσει» στο read…

Line31: σταματάμε τους υπολογισμούς σπάζοντας την επανάληψη κατά μήκος του cigar code, αφού, η μετάλλαξη βρίσκεται πριν το Εικόνα 4-2 …

Line33: διαφορετικά προσθέτουμε $I = I + val$ όπου val ο αριθμός των εισηγμένων βάσεων στο read και I ο είδη καταμετρημένος αριθμός των εισηγμένων βάσεων στο read…

Line34: και προσθέτουμε $sum = sum + val$ όπου val ο αριθμός των εισηγμένων βάσεων και sum το πόσο έχουμε «προχωρήσει» στο read.

Line38: Προσθέτουμε στην απόσταση της μετάλλαξης από την αρχή του read, $pos[j]$, την μεταβλητή s, συν την μεταβλητή I και αφαιρούμε την μεταβλητή d. Πρακτικά προσθέτουμε το πλήθος των χαρακτήρων 'N' της αρχής του read, αφαιρούμε το πλήθος των κενών ανάμεσα στην αρχή του read και στη θέση της μετάλλαξης και τέλος προσθέτουμε τον αριθμό των βάσεων που έχουν εισαχθεί και δεν έχουν ταυτιστεί ανάμεσα στην αρχή του read και τη θέση της μετάλλαξης. Στο σημείο αυτό σημειώνουμε ότι οι αριθμοί d, I μπορεί να είναι μηδενικοί αν η μετάλλαξη βρίσκεται πριν τα αντίστοιχα κενά που υποδεικνύουν.

Line39: Βρίσκουμε την ποιότητα που αντιστοιχεί στη θέση που υπολογίστηκε στο παραπάνω βήμα, $pos[j]$, παίρνοντας από το string ποιοτήτων $sam_{data}[[1]]\$qual[j]$ το substring μεταξύ των θέσεων $pos[j]$ $και$ $pos[j]$ δηλαδή τον χαρακτήρα σε εκείνη τη θέση και μετατρέποντάς τον σε ακέραιο.

Line41: Ενώνουμε το διάνυσμα pos της τοπικής επανάληψης, με το διάνυσμα $\boldsymbol{p}$ της εξωτερικής επανάληψης όλων των reads.

Line42: Ενώνουμε το διάνυσμα $qual$ της τοπικής επανάληψης, με το διάνυσμα $\boldsymbol{q}$ της εξωτερικής επανάληψης όλων των reads.

Βήμα 8: Τυπώνουμε ιστόγραμμα του διανύσματος $\boldsymbol{p}$ «Histogram of Positions».

Βήμα 9: Τυπώνουμε ιστόγραμμα του διανύσματος $\boldsymbol{q}$ «Histogram of Qualities».

Βήμα 10: Τυπώνουμε ιστόγραμμα των ποιοτήτων σε όλες τις θέσεις όλων των reads που χρησιμοποιήθηκαν και υποδεικνύονται από τη λίστα $\boldsymbol{idxs}$. Επειδή το μέγεθος των δεδομένων σε αυτό το βήμα ήταν τεράστιο έγινε χρήση της βιβλιοθήκης ggplot, αφού κάνει πολύ καλύτερη διαχείριση μνήμης.

Παράδειγμα D:

Reference Positions	145	146	147	148	149	150	151	152	153	.....	161	162	163	164
Reference Genome		T	T	T	A	C	A	A	T	A	gap	C	T	G	A
Mapped Read	NNN	T	T	T	A	C	A	A	T	A	gap	G	T	G	A
Read Positions	123	4	5	6	7	8	9	10	11	12	.....	13	14	15	16

Εικόνα 4-1 Παράδειγμα τμήματος του γονιδιώματος αναφοράς στο οποίο έχει γίνει στοίχιση ενός read με 3 χαρακτήρες 'N' στην αρχή του και D κενό (gap) 7 θέσεων.

Έστω λοιπόν ότι η μετάλλαξη αναφέρθηκε από το GATK στη θέση 161 με τη λογική ότι $ref = C, alt = G$ και στο .bam file αναφέρεται ότι το read ξεκινά στη θέση 145 καθώς και ότι το cigar code θα είναι: $3S9M7D4M$

Τότε η θέση της μετάλλαξης υπολογίζεται ως εξής:

$$pos[j] = reported_{pos} - lm_{pos} + 1 = 161 - 145 + 1 = 17$$

και επειδή $17 > 9$ θέτουμε $d = 7$ και έχουμε:

$$\widehat{pos[j]} = 17 + 3s - 7d = 13$$

Που όπως φαίνεται και στο παράδειγμα είναι η θέση της μετάλλαξης ως προς τις σχετικές συντεταγμένες του read.

Παράδειγμα I:

Reference Positions	145	146	147	148	149	150	151	152	153	----------							154	155	156	157
Reference Genome		T	T	T	A	C	A	A	T	A	----------						C	T	G	A
Mapped Read	NNN	T	T	T	A	C	A	A	T	A	T	G	C	T	G	C	G	T	G	A
Read Positions	123	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Εικόνα 4-2 Παράδειγμα τμήματος του γονιδιώματος αναφοράς στο οποίο έχει γίνει στοίχιση ενός read με 3 χαρακτήρες 'N' στην αρχή του και I insertion 6 θέσεων.

Έστω λοιπόν ότι η μετάλλαξη αναφέρθηκε από το GATK στη θέση 154 με τη λογική ότι $ref = C, alt = G$ και στο .bam file αναφέρεται ότι το read ξεκινά στη θέση 145 καθώς και ότι το cigar code θα είναι: $3S9M6I4M$

Τότε η θέση της μετάλλαξης υπολογίζεται ως εξής:

$$pos[j] = reported_{pos} - lm_{pos} + 1 = 154 - 145 + 1 = 10$$

και επειδή $10 > 9$ θέτουμε $I = 6$ και έχουμε:

$$\widehat{pos[j]} = 10 + 3s + 6I = 19$$

Που όπως φαίνεται και στο παράδειγμα είναι η θέση της μετάλλαξης ως προς τις σχετικές συντεταγμένες του read.

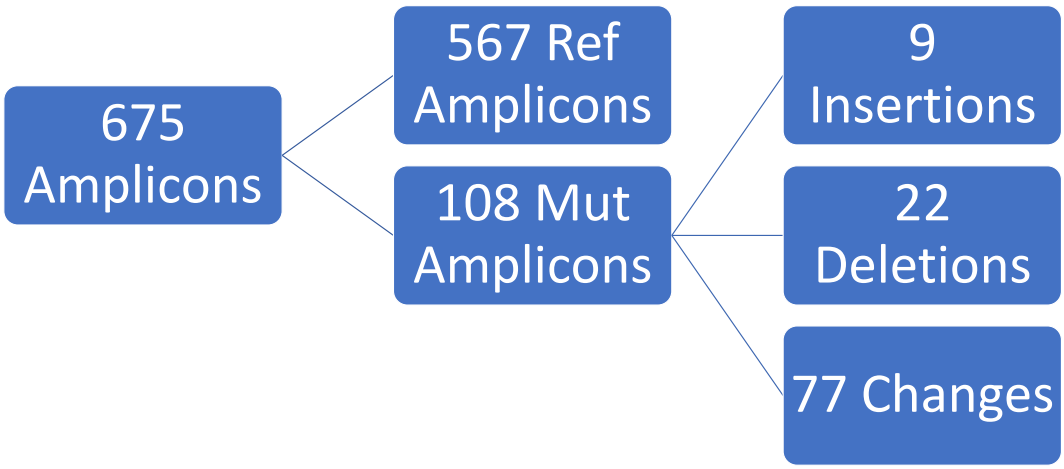
5 Αποτελέσματα

5.1 Εισαγωγή

Στην παράγραφο αυτή θα αναφερθούμε στα πειράματα που έγιναν χρησιμοποιώντας τεχνητά δεδομένα που παράχθηκαν από τον προσομοιωτή του πρώτου βήματος της παρούσας διπλωματικής. Έπειτα θα παρουσιάσουμε πληθώρα αποτελεσμάτων σε μορφή γραφημάτων και πινάκων για όλα τα πειράματα που διεξήχθησαν.

5.2 Δεδομένα

Τα δεδομένα που χρησιμοποιήθηκαν, όπως προαναφέρθηκε, δημιουργήθηκαν από τον προσομοιωτή που παρουσιάσαμε στο πρώτο τμήμα αυτής της διπλωματικής. Για κάθε πείραμα δημιουργήθηκε ξεχωριστό σύνολο δεδομένων, με διαφορετική κατανομή ποιοτήτων ανά θέση. Όλα τα σύνολα όμως στηρίχθηκαν στο πρότυπο δεδομένων που εξηγείται στο κεφάλαιο 3.3 και το οποίο δημιουργήθηκε από ειδικούς. Το πρότυπο δεδομένων που χρησιμοποιήθηκε περιλαμβάνει 675 amplicons από τα οποία, τα 567 αντιστοιχούν σε reference amplicons, ενώ τα 108 σε mutexed amplicons, δηλαδή ref amplicons που έχει εισαχθεί κάποια μετάλλαξη. Από τα 108 mutexed amplicons τα 9 από αυτά αντιστοιχούν σε μεταλλάξεις εισαγωγής, τα 22 από αυτά αντιστοιχούν σε μεταλλάξεις διαγραφής ενώ τα 77 από αυτά σε μεταλλάξεις αλλαγής. Παρακάτω παραθέτουμε συνοπτικό πίνακα – δένδρο με τα βασικά χαρακτηριστικά του προτύπου δεδομένων που περιγράφηκαν παραπάνω.



Εικόνα 5-1Template Overview

5.3 Πειράματα

5.3.1 Constant quality

5.3.1.1 Constant Quality Q = 12 (ASCII 45)

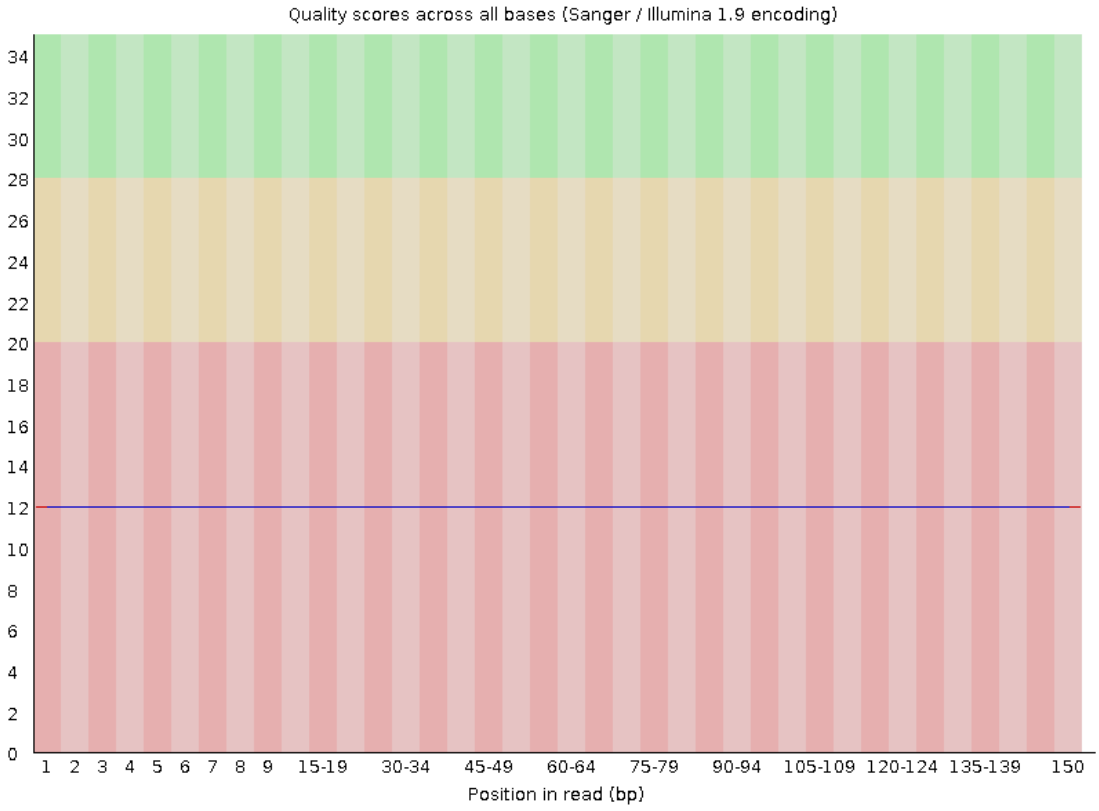
Πείραμα με σύνολο δεδομένων του οποίου οι ποιότητες ακολουθούν σταθερή κατανομή ποιοτήτων ίση με $Q = 12$. Το πείραμα εκτελέστηκε μία φορά για παράμετρο εισόδου max-read-per-alignment-start $= 100$.

Πίνακας 5-1 FastQC Details – r1 Q = 12 experiment

Measure	Value
Filename	r1.compressed.c45_x1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46

Πίνακας 5-2 FastQC Details – r2 Q = 12 experiment

Measure	Value
Filename	r2.compressed.c45_x1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46

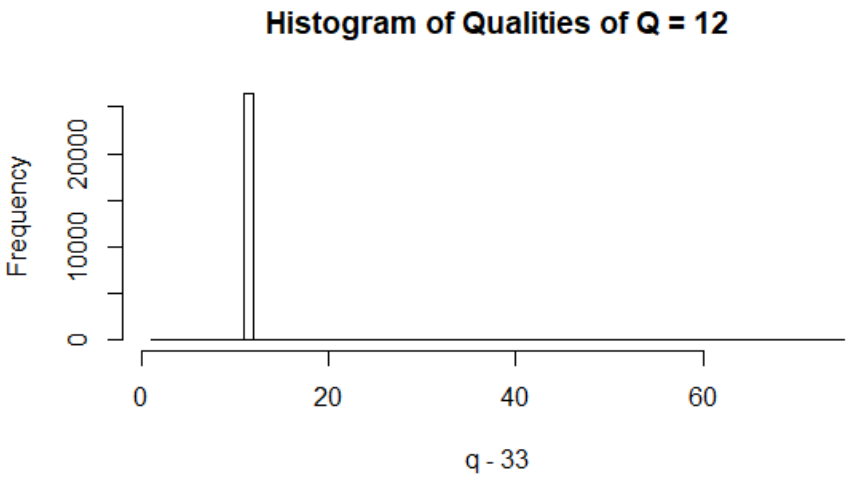


Εικόνα 5-2 FastQC - Per base sequence quality – Q = 12 experiment

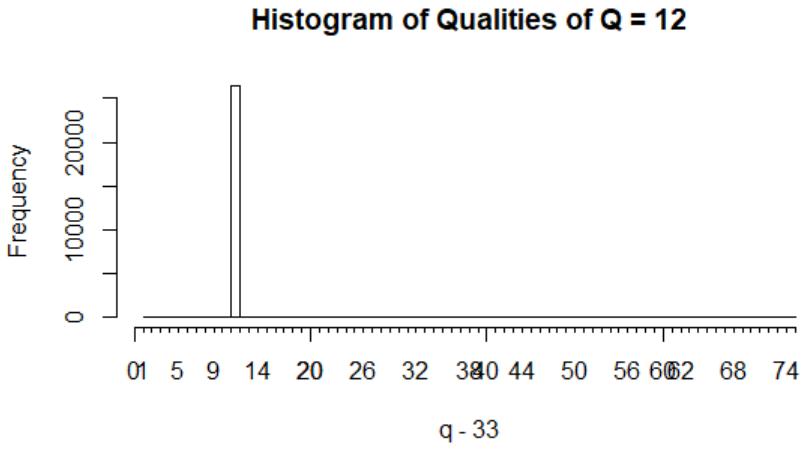
1 max-read-per-alignment-start = 100

Πίνακας 5-3 Q = 12 Experiment MRPAS=100 True Positives

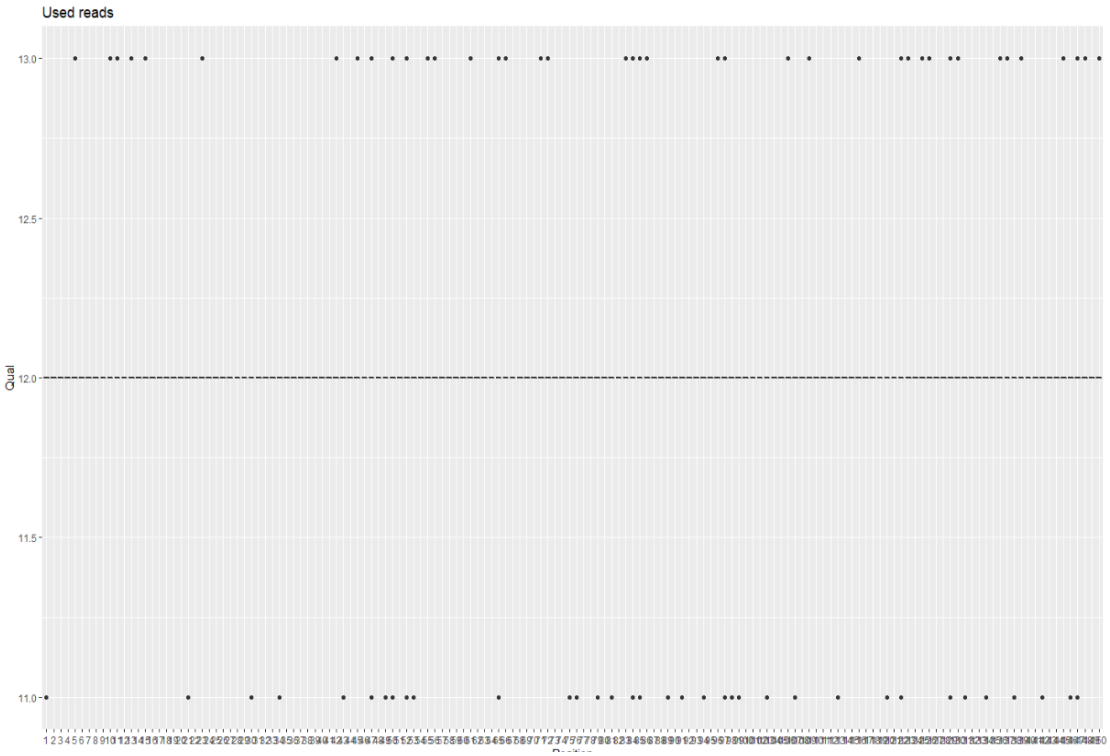
AF.Template	DP.Template	IndelType.Data	AF.Data	DP.Data
0.56	9500	change	0.615	8171
0.06	9500	deletion	0.044	9383
0.21	1700	deletion	0.21	1707



Εικόνα 5-3 Frequencies of qualities – Q = 12 experiment MRPAS=100



Εικόνα 5-4 Frequencies of mutexed positions – Q = 12 experiment MRPAS=100



Εικόνα 5-5 Per base quality for used reads – Q = 12 experiment MRPAS=100

5.3.1.2 Constant Quality Q = 20 (ASCII 53)

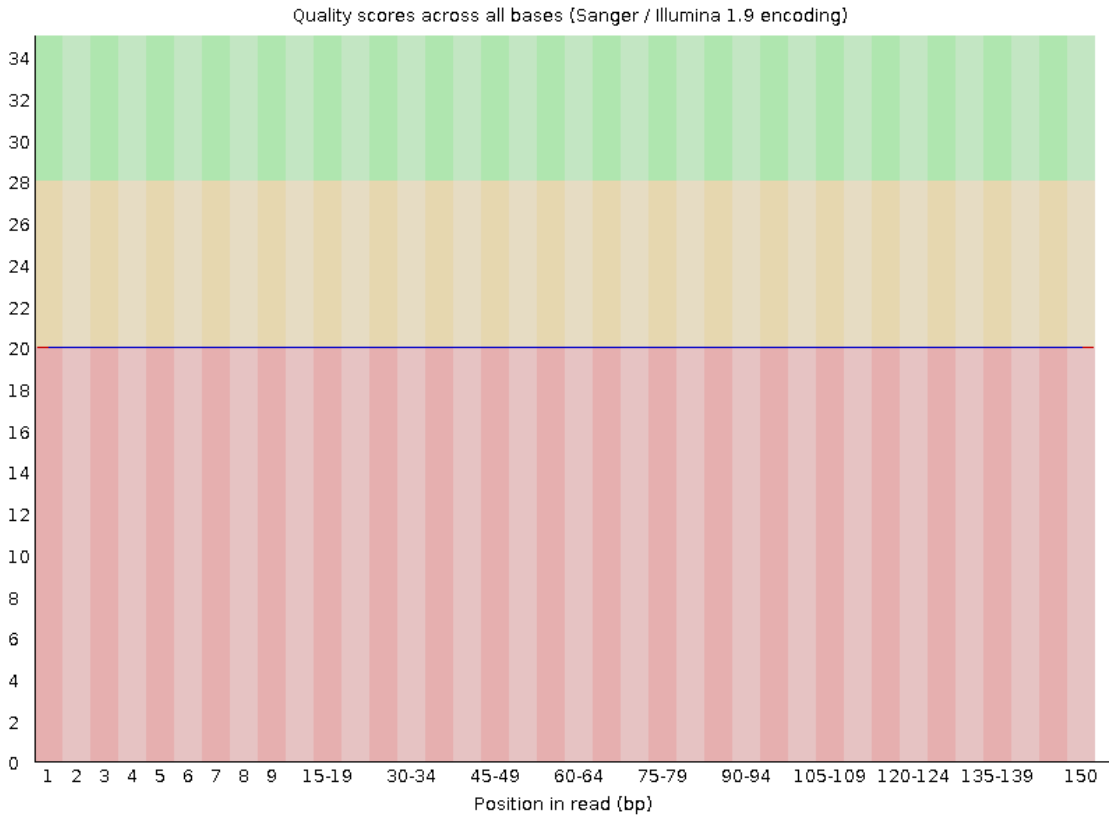
Πείραμα με σύνολο δεδομένων του οποίου οι ποιότητες ακολουθούν σταθερή κατανομή ποιοτήτων ίση με $Q = 20$. Το πείραμα εκτελέστηκε μία φορά για παράμετρο εισόδου max-read-per-alignment-start $= 100$.

Πίνακας 5-4 FastQC Details – r1 Q = 20 experiment

Measure	Value
Filename	r1.compressed.c53_x1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46

Πίνακας 5-5 FastQC Details – r2 Q = 20 experiment

Measure	Value
Filename	r2.compressed.c53_x1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46



Εικόνα 5-6 FastQC - Per base sequence quality – Q = 20 experiment

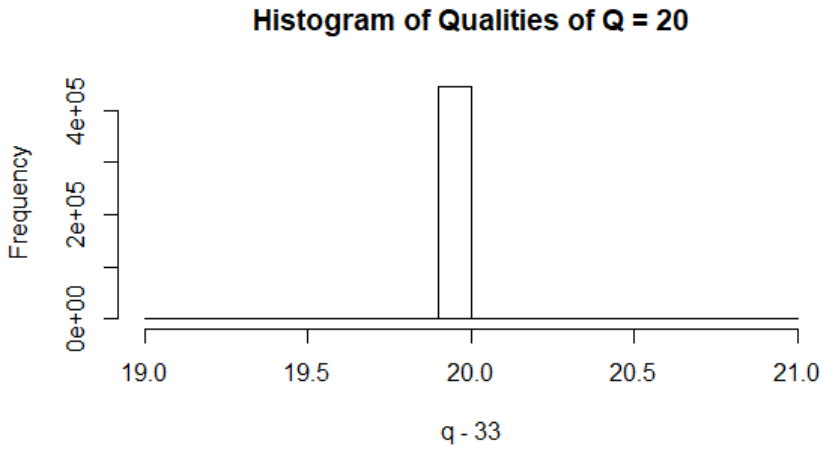
1 max-read-per-alignment-start = 100

Πίνακας 5-6 Q = 20 Experiment MRPAS=100 True Positives

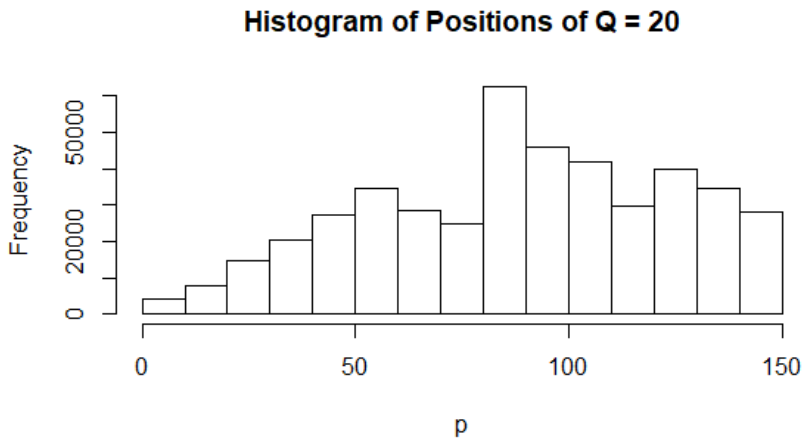
AF. Template	DP. Template	IndelType. Data	AF. Data	DP. Data
0.3	11000	change	0.296	6637
0.08	11000	change	0.078	9646
0.23	2500	insertion	0.233	2274
0.19	3500	change	0.184	4154
0.31	4200	change	0.307	1002
0.1	1400	deletion	0.094	895
0.64	5400	change	0.642	2699
0.24	5400	insertion	0.247	3539
0.07	7900	deletion	0.069	8232
0.32	10200	change	0.314	6290
0.05	9000	change	0.047	5023
0.78	9000	change	0.781	9975
0.21	8000	deletion	0.207	7975
0.07	11400	change	0.071	7490
0.38	11400	deletion	0.348	9915
0.46	5500	change	0.458	4863
0.12	5900	change	0.118	5985
0.09	4300	change	0.093	2503
0.41	4600	deletion	0.412	4043
0.05	1700	change	0.05	1789
0.1	700	change	0.087	777
0.16	5700	change	0.155	6693
0.34	7300	insertion	0.335	5909
0.05	2900	change	0.05	2987
0.34	3100	change	0.347	2831
0.43	8900	change	0.427	8017
0.87	6700	change	0.873	7345
0.41	1800	change	0.418	2025
0.25	1400	insertion	0.255	832
0.13	1700	change	0.122	1749
0.1	3000	deletion	0.097	1610
0.67	3000	change	0.753	1607
0.05	6000	change	0.045	4412
0.2	6000	change	0.2	6476
0.34	2100	change	0.326	1744
0.1	500	change	0.108	545
0.78	1000	change	0.775	1144
0.45	4000	change	0.437	3978
0.34	4000	deletion	0.785	4154

0.05	500	change	0.05	518
0.3	2800	deletion	0.3	2829
0.14	2800	change	0.137	3255
0.39	5800	change	0.382	4667
0.78	4000	change	0.78	2776
0.06	4000	change	0.06	4118
0.27	3000	change	0.272	3286
0.47	2500	change	0.466	2561
0.63	3500	change	0.616	961
0.56	9500	change	0.563	6855
0.06	9500	deletion	0.062	6816
0.19	1000	change	0.19	1171
0.56	900	change	0.547	965
0.08	800	deletion	0.091	518
0.78	800	change	0.77	754
0.32	8000	change	0.321	5849
0.05	8000	change	0.05	8159
0.09	4300	change	0.095	2925

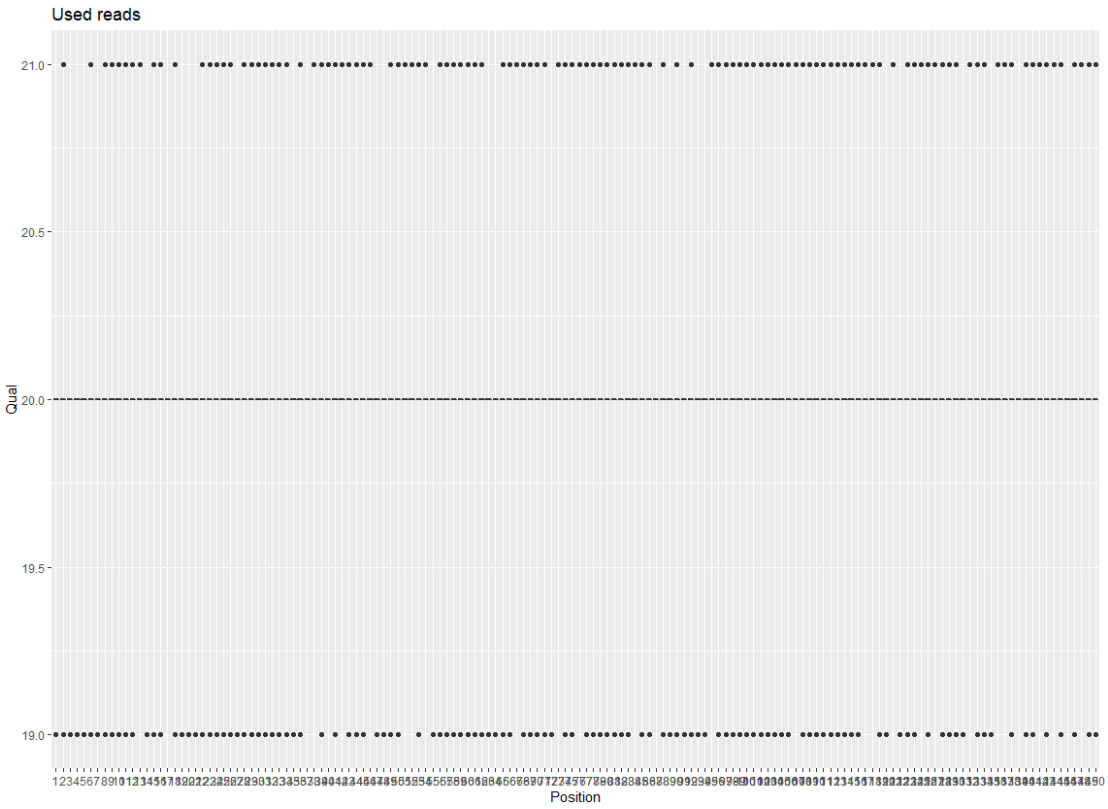
0.23	4300	deletion	0.222	3293
0.38	2800	change	0.373	2969
0.07	2100	change	0.069	2073
0.26	3700	change	0.259	3667
0.05	6030	insertion	0.055	4851
0.42	3200	deletion	0.431	1690
0.25	2300	deletion	0.248	2197
0.47	9000	change	0.48	3839
0.12	9000	change	0.118	2628
0.21	1700	deletion	0.213	1717
0.08	6200	change	0.081	6151
0.35	2510	change	0.346	2423
0.26	4870	change	0.26	5399
0.31	1500	deletion	0.303	1434
0.12	3650	change	0.115	3734
0.19	4600	change	0.193	4769
0.62	7100	change	0.617	7708
0.09	1470	change	0.082	1425



Εικόνα 5-7 Frequencies of qualities – Q = 20 experiment MRPAS=100



Εικόνα 5-8 Frequencies of mutexed positions – Q = 20 experiment MRPAS=100



Εικόνα 5-9 Per base quality for used reads – Q = 20 experiment MRPAS=100

5.3.1.3 Constant Quality Q = 30 (ASCII 63)

Πείραμα με σύνολο δεδομένων του οποίου οι ποιότητες ακολουθούν σταθερή κατανομή ποιοτήτων ίση με $Q = 30$. Το πείραμα εκτελέστηκε μία φορά για παράμετρο εισόδου max-read-per-alignment-start $= 50$.

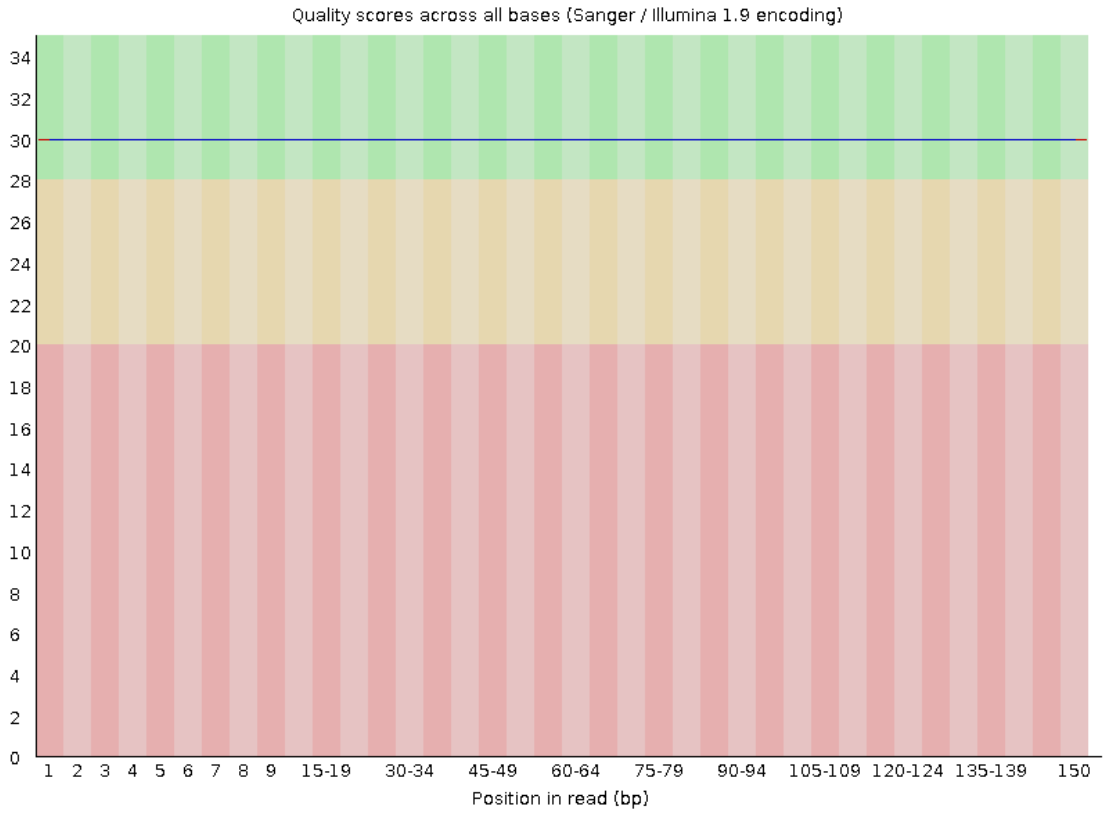
Πίνακας 5-7 FastQC Details – r1 Q = 30 experiment

Measure	Value
Filename	r1.medium.compressed.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9

Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46

Πίνακας 5-8 FastQC Details – r2 Q = 30 experiment

Measure	Value
Filename	r2.medium.compressed.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46



Εικόνα 5-10 FastQC - Per base sequence quality – Q = 30 experiment

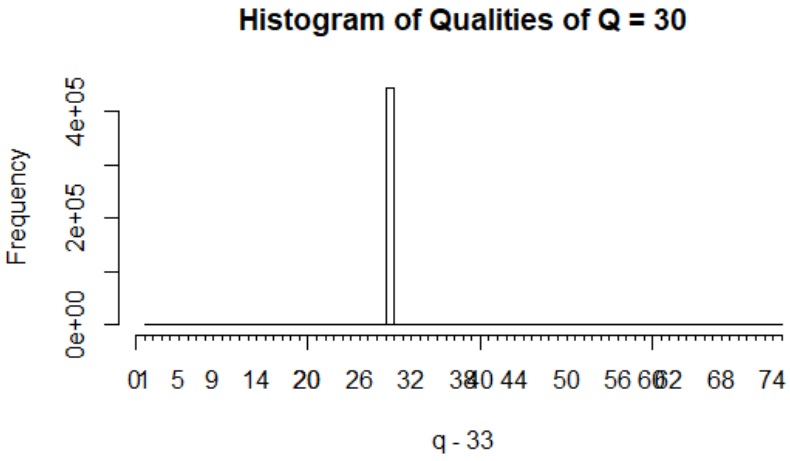
1 max-read-per-alignment-start = 50

Πίνακας 5-9 Q = 30 Experiment MRPAS=50 True Positives

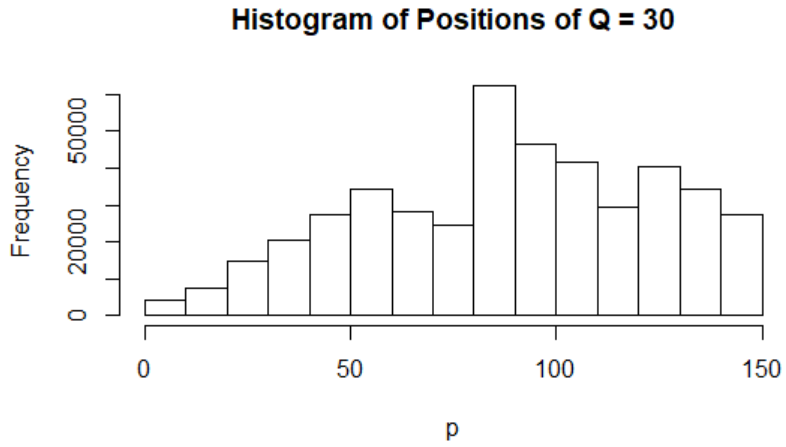
AF. Template	DP. Template	IndelType. Data	AF. Data	DP. Data
0.3	11000	change	0.302	3442
0.08	11000	change	0.079	5080
0.23	2500	insertion	0.236	2292
0.19	3500	change	0.194	4037
0.31	4200	change	0.278	959
0.1	1400	deletion	0.098	1462
0.64	5400	change	0.665	2240
0.24	5400	insertion	0.24	5187
0.07	7900	deletion	0.071	5932

0.32	10200	change	0.32	3293
0.05	9000	change	0.046	2648
0.78	9000	change	0.776	6433
0.21	8000	deletion	0.2	5571
0.07	11400	change	0.071	3786
0.38	11400	deletion	0.316	5704
0.46	5500	change	0.458	4574
0.12	5900	change	0.12	5312
0.09	4300	change	0.094	2460
0.41	4600	deletion	0.388	3800
0.05	1700	change	0.051	1780
0.1	700	change	0.091	759
0.16	5700	change	0.16	6086
0.34	7300	insertion	0.335	4194
0.05	2900	change	0.053	2913
0.34	3100	change	0.34	3209
0.43	8900	change	0.424	4884
0.87	6700	change	0.868	6499
0.41	1800	change	0.415	1995
0.25	1400	insertion	0.247	1417
0.13	1700	change	0.126	1726
0.1	3000	deletion	0.101	3106
0.67	3000	change	0.744	3106
0.05	6000	change	0.051	3509
0.2	6000	change	0.199	5388
0.34	2100	change	0.348	1663
0.1	500	change	0.092	467
0.78	1000	change	0.787	1185
0.45	4000	change	0.448	3463
0.34	4000	deletion	0.785	4003
0.05	500	change	0.052	512
0.3	2800	deletion	0.292	2778
0.14	2800	change	0.142	3135

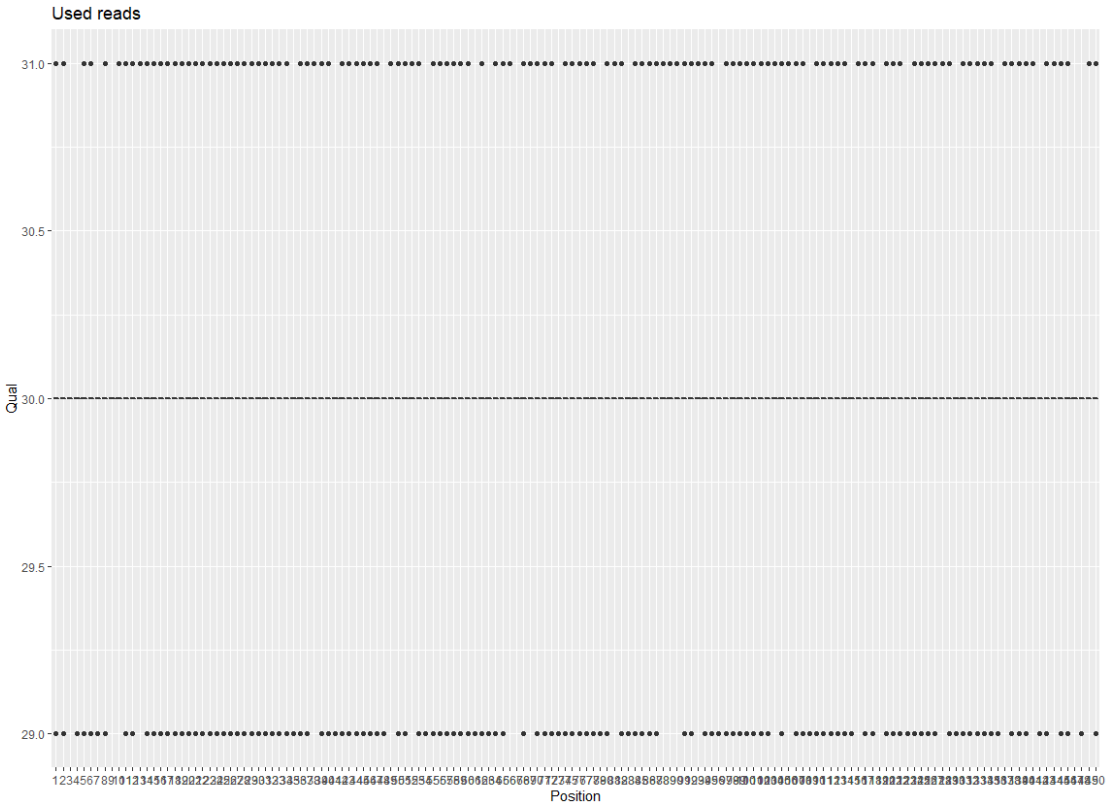
0.39	5800	change	0.387	3789
0.78	4000	change	0.784	2706
0.06	4000	change	0.06	4116
0.27	3000	change	0.27	3275
0.47	2500	change	0.472	2573
0.63	3500	change	0.639	1028
0.56	9500	change	0.557	4443
0.06	9500	deletion	0.067	4423
0.19	1000	change	0.185	1134
0.56	900	change	0.561	926
0.08	800	deletion	0.089	589
0.78	800	change	0.773	796
0.32	8000	change	0.326	3545
0.05	8000	change	0.052	5482
0.09	4300	change	0.085	2858
0.23	4300	deletion	0.225	4398
0.38	2800	change	0.371	2934
0.07	2100	change	0.066	2070
0.26	3700	change	0.255	3561
0.05	6030	insertion	0.051	3896
0.42	3200	deletion	0.417	3195
0.25	2300	deletion	0.253	2180
0.47	9000	change	0.48	2000
0.12	9000	change	0.104	1376
0.21	1700	deletion	0.209	1694
0.08	6200	change	0.079	5296
0.35	2510	change	0.356	2360
0.26	4870	change	0.26	5296
0.31	1500	deletion	0.321	1284
0.12	3650	change	0.117	3647
0.19	4600	change	0.19	4707
0.62	7100	change	0.619	5825
0.09	1470	change	0.099	1411



Εικόνα 5-11 Frequencies of qualities – Q = 30 experiment MRPAS=50



Εικόνα 5-12 Frequencies of mutexed positions – Q = 30 experiment MRPAS=50



Εικόνα 5-13 Per base quality for used reads – Q = 30 experiment MRPAS=50

5.3.2 Realistic Dataset

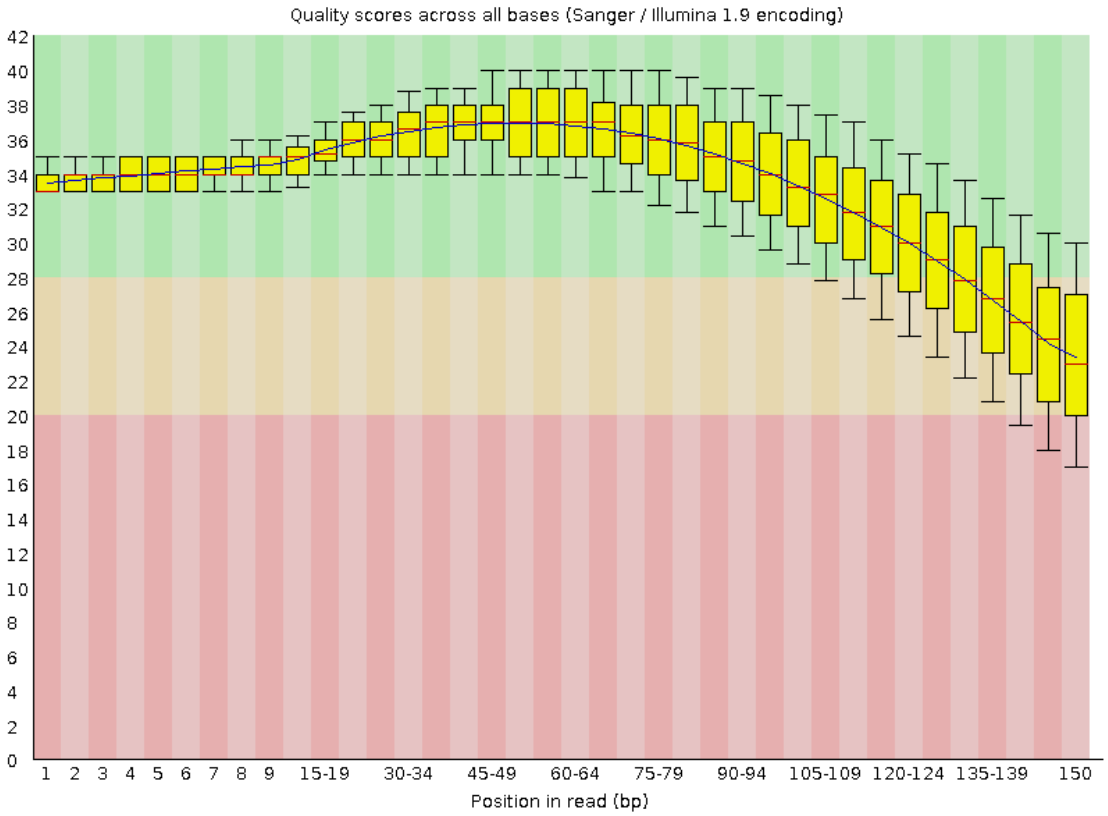
Πείραμα με σύνολο δεδομένων του οποίου οι ποιότητες ακολουθούν μία ρεαλιστική κατανομή ποιοτήτων. Το πείραμα εκτελέστηκε δύο διαφορετικές φορές με διαφορετική παράμετρο εισόδου max-read-per-alignment-start $\in \{50,100\}$.

Πίνακας 5-10 FastQC Details – r1 realistic experiment

Measure	Value
Filename	r1.realistic.compressed.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46

Πίνακας 5-11 FastQC Details – r2 realistic experiment

Measure	Value
Filename	r2.realistic.compressed.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	1996110
Sequences flagged as poor quality	0
Sequence length	150
%GC	46



Εικόνα 5-14 1.2 FastQC - Per base sequence quality – realistic experiment

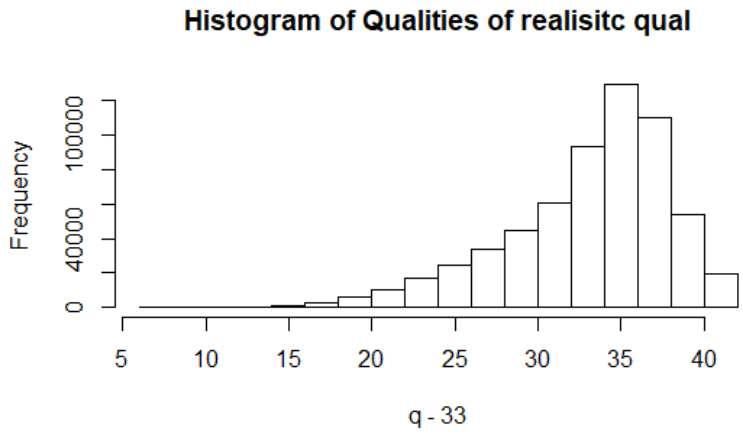
1 max-read-per-alignment-start = 50

Πίνακας 5-12 Realistic Experiment MRPAS=50 True Positives

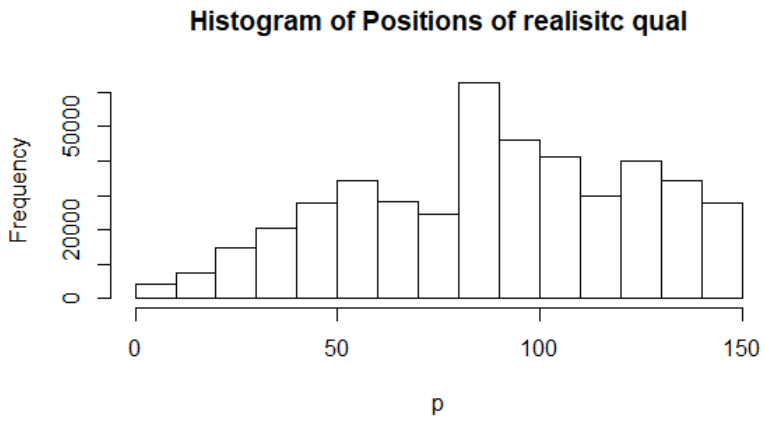
AF. Template	DP. Template	IndelType. Data	AF. Data	DP. Data
0.3	11000	change	0.3	3441
0.08	11000	change	0.081	5085
0.23	2500	insertion	0.238	2318
0.19	3500	change	0.188	4118
0.31	4200	change	0.3	915
0.1	1400	deletion	0.098	887
0.64	5400	change	0.641	2224
0.24	5400	insertion	0.248	3077
0.07	7900	deletion	0.07	5902
0.32	10200	change	0.317	3298
0.05	9000	change	0.053	2649
0.78	9000	change	0.777	6457
0.21	8000	deletion	0.201	5574
0.07	11400	change	0.073	3787
0.38	11400	deletion	0.314	5699
0.46	5500	change	0.457	4568
0.12	5900	change	0.115	5299
0.09	4300	change	0.087	2382
0.41	4600	deletion	0.389	3821
0.05	1700	change	0.054	1756
0.1	700	change	0.099	705
0.16	5700	change	0.158	6051
0.34	7300	insertion	0.337	4192
0.05	2900	change	0.05	2949
0.34	3100	change	0.332	2727
0.43	8900	change	0.428	4882
0.87	6700	change	0.866	6526
0.41	1800	change	0.407	2008
0.25	1400	insertion	0.267	782
0.13	1700	change	0.13	1701
0.1	3000	deletion	0.097	1638
0.67	3000	change	0.74	1636
0.05	6000	change	0.053	3500
0.2	6000	change	0.199	5402
0.34	2100	change	0.346	1729
0.1	500	change	0.103	456
0.78	1000	change	0.772	1173

0.45	4000	change	0.459	3918
0.34	4000	deletion	0.787	4063
0.05	500	change	0.051	512
0.3	2800	deletion	0.298	2754
0.14	2800	change	0.143	3203
0.39	5800	change	0.393	3787
0.78	4000	change	0.778	2698
0.06	4000	change	0.059	4088
0.27	3000	change	0.268	3227
0.47	2500	change	0.47	2540
0.63	3500	change	0.628	990
0.56	9500	change	0.561	3845
0.06	9500	deletion	0.057	3826
0.19	1000	change	0.197	1155
0.56	900	change	0.55	925
0.08	800	deletion	0.096	552
0.78	800	change	0.778	758
0.32	8000	change	0.323	3541
0.05	8000	change	0.048	5476

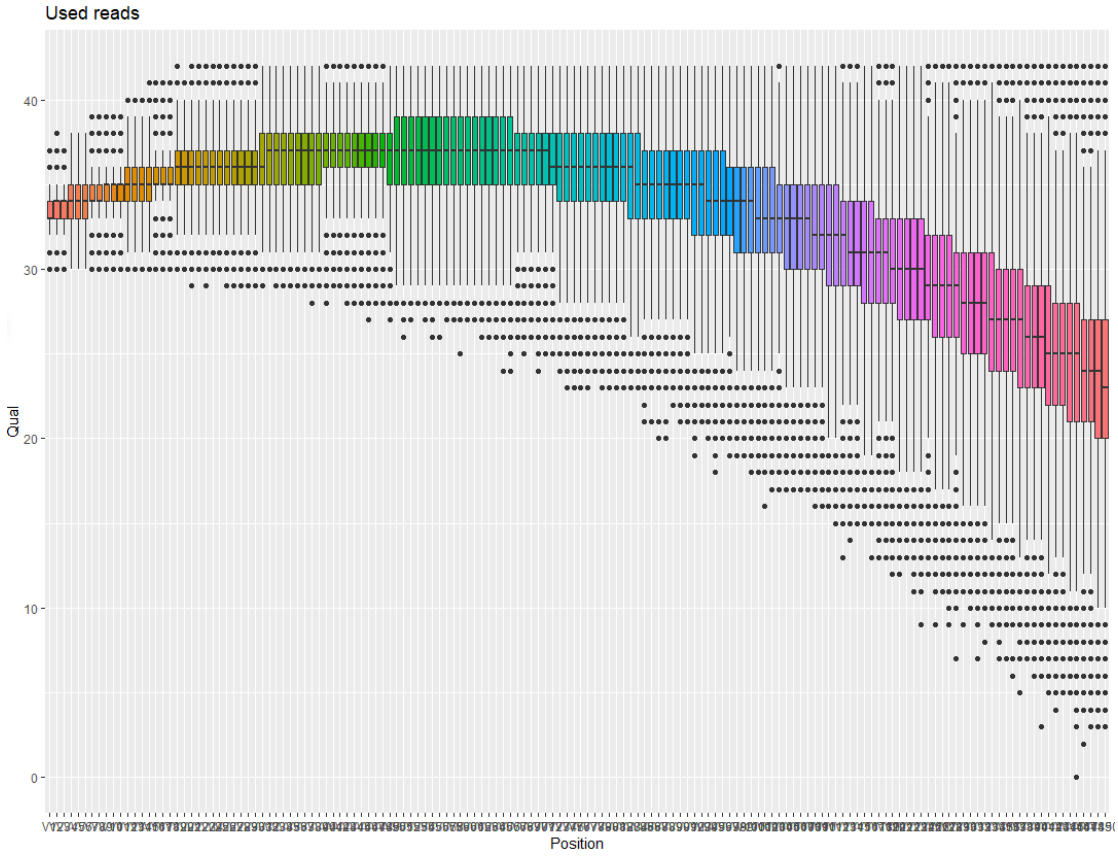
0.09	4300	change	0.089	2862
0.23	4300	deletion	0.231	3204
0.38	2800	change	0.381	2881
0.07	2100	change	0.072	2068
0.26	3700	change	0.269	3480
0.05	6030	insertion	0.052	3868
0.42	3200	deletion	0.41	1660
0.25	2300	deletion	0.256	2175
0.47	9000	change	0.469	1995
0.12	9000	change	0.111	1372
0.21	1700	deletion	0.205	1663
0.08	6200	change	0.08	5365
0.35	2510	change	0.353	2345
0.26	4870	change	0.26	5376
0.31	1500	deletion	0.305	1287
0.12	3650	change	0.115	3660
0.19	4600	change	0.19	4802
0.62	7100	change	0.618	5852
0.09	1470	change	0.093	1388



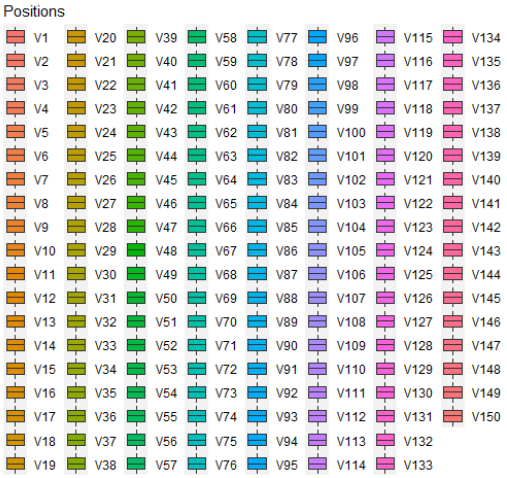
Εικόνα 5-15 Frequencies of qualities – realistic experiment MRPAS=50



Εικόνα 5-16 Frequencies of mutexed positions – realistic experiment MRPAS=50



Εικόνα 5-17 Per base quality for used reads – realistic experiment MRPAS=50



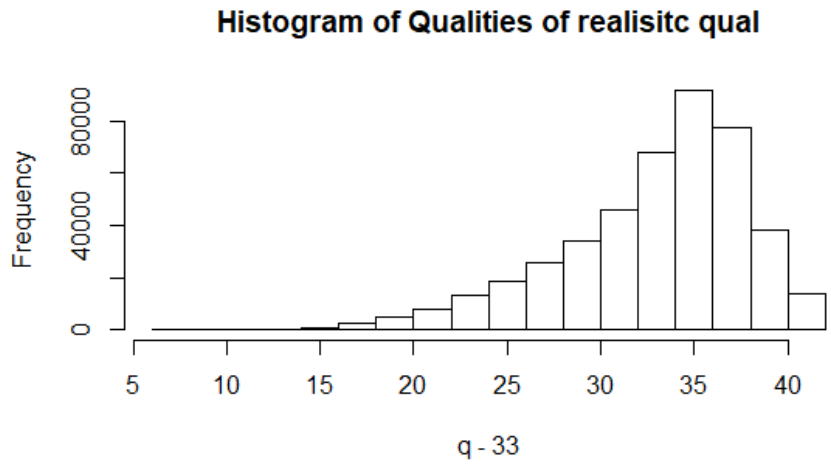
2 max-read-per-alignment-start = 100

Πίνακας 5-13 Realistic Experiment MRPAS=100 True Positives

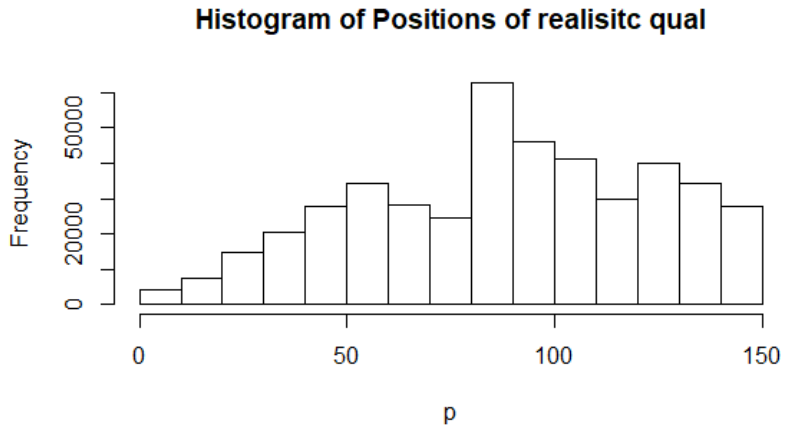
AF. Template	DP. Template	IndelType. Data	AF. Data	DP. Data
0.3	11000	change	0.3	6678
0.08	11000	change	0.083	9609
0.23	2500	insertion	0.24	2353
0.19	3500	change	0.187	4128
0.31	4200	change	0.3	915
0.1	1400	deletion	0.094	908
0.64	5400	change	0.645	2677
0.24	5400	insertion	0.247	3560
0.07	7900	deletion	0.07	8146
0.32	10200	change	0.317	6211
0.05	9000	change	0.051	4943
0.78	9000	change	0.78	9945
0.21	8000	deletion	0.206	7847
0.07	11400	change	0.072	7530
0.38	11400	deletion	0.354	9946
0.46	5500	change	0.455	4881
0.12	5900	change	0.117	5973
0.09	4300	change	0.085	2465
0.41	4600	deletion	0.393	4104
0.05	1700	change	0.053	1774
0.1	700	change	0.099	705
0.16	5700	change	0.159	6440
0.34	7300	insertion	0.338	5911
0.05	2900	change	0.051	2979
0.34	3100	change	0.334	2752
0.43	8900	change	0.43	8006
0.87	6700	change	0.867	7307
0.41	1800	change	0.41	2014
0.25	1400	insertion	0.264	798
0.13	1700	change	0.13	1734
0.1	3000	deletion	0.097	1659
0.67	3000	change	0.742	1655
0.05	6000	change	0.052	4405
0.2	6000	change	0.197	6319
0.34	2100	change	0.347	1771
0.1	500	change	0.1	498
0.78	1000	change	0.775	1183
0.45	4000	change	0.459	3981
0.34	4000	deletion	0.789	4160
0.05	500	change	0.054	538
0.3	2800	deletion	0.297	2784
0.14	2800	change	0.143	3212
0.39	5800	change	0.394	4673
0.78	4000	change	0.778	2781
0.06	4000	change	0.059	4150
0.27	3000	change	0.268	3227
0.47	2500	change	0.472	2561
0.63	3500	change	0.629	993
0.56	9500	change	0.559	6840
0.06	9500	deletion	0.059	6791
0.19	1000	change	0.196	1167
0.56	900	change	0.553	963
0.08	800	deletion	0.094	561
0.78	800	change	0.784	767
0.32	8000	change	0.316	5978
0.05	8000	change	0.051	8157
0.09	4300	change	0.091	2919
0.23	4300	deletion	0.23	3210

0.38	2800	change	0.379	2906
0.07	2100	change	0.074	2109
0.26	3700	change	0.268	3520
0.05	6030	insertion	0.052	4899
0.42	3200	deletion	0.41	1687
0.25	2300	deletion	0.252	2219
0.47	9000	change	0.464	3866
0.12	9000	change	0.12	2649
0.21	1700	deletion	0.209	1699

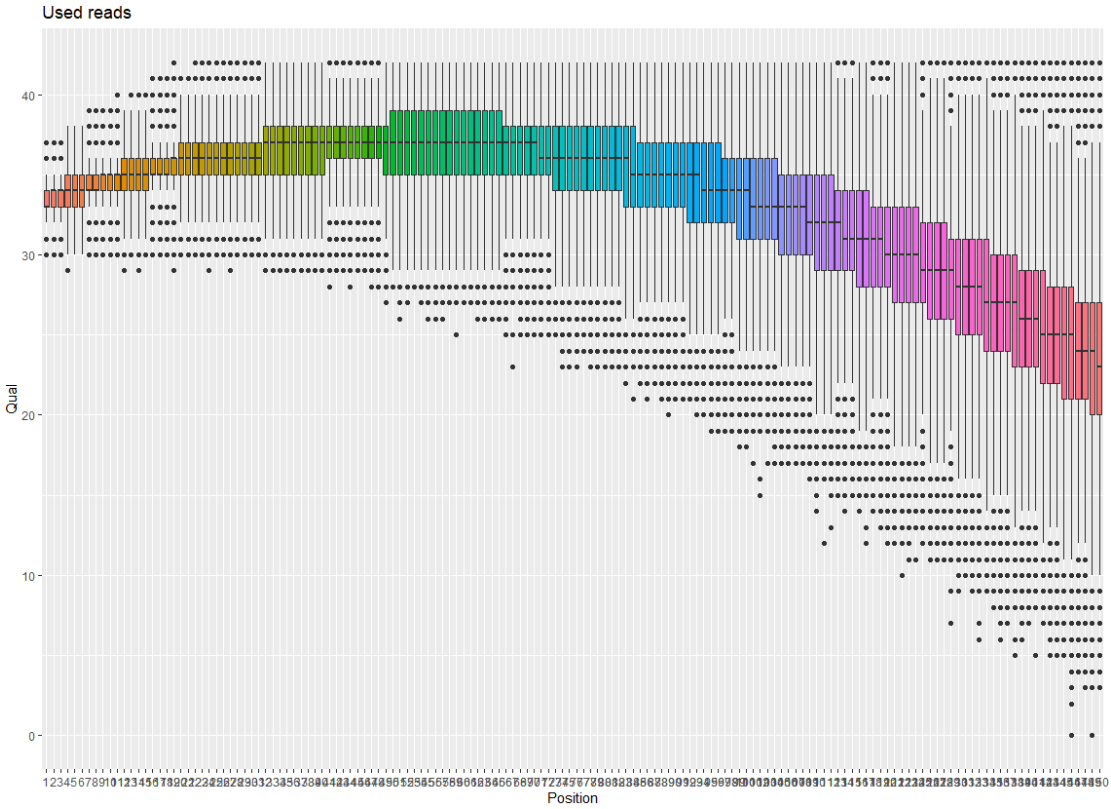
0.08	6200	change	0.08	6074
0.35	2510	change	0.352	2379
0.26	4870	change	0.261	5483
0.31	1500	deletion	0.306	1309
0.12	3650	change	0.115	3696
0.19	4600	change	0.19	4945
0.62	7100	change	0.615	7755
0.09	1470	change	0.092	1413



Εικόνα 5-18 Frequencies of qualities – realistic experiment MRPAS=100



Εικόνα 5-19 Frequencies of mutexed positions – realistic experiment MRPAS=100



Εικόνα 5-20 Per base quality for used reads – realistic experiment MRPAS=100

6 Συμπεράσματα

6.1 Γενικά Συμπεράσματα

Συνοψίζοντας, η εύρεση σωματικών μεταλλάξεων μέσω υπολογιστικών μεθόδων απαιτεί τη χρήση τεχνητού συνόλου δεδομένων. Όσο πιο ρεαλιστικό είναι το τεχνητό σύνολο δεδομένων, τόσο καλύτερη μελέτη των εργαλείων εύρεσης σωματικών μεταλλάξεων μπορεί να γίνει. Συνεπώς, πρώτος και βασικότερος στόχος της παρούσας διπλωματικής είναι η δημιουργία ενός εργαλείου παραγωγής τεχνητών δεδομένων, ακολουθώντας ένα πρότυπο που μας δόθηκε από ειδικούς, με τις παραμέτρους των δεδομένων να επιλέγονται από τον χρήστη και φυσικά να προσεγγίζουν όσο το δυνατόν καλύτερα τα πραγματικά πειράματα amplicons sequencing. Έπειτα τα δεδομένα αναλύθηκαν μέσω ήδη υπαρχόντων εργαλείων και η ροή εκτέλεσης τους αυτοματοποιήθηκε, προκυμμένου να εκτελεστούν επαναληπτικά για διαφορετικά σύνολα δεδομένων και διαφορετικές παραμέτρους εισόδου (ρυθμίσεις). Με το τρόπο αυτόν θα μπορούσαμε να εξάγουμε χρήσιμα συμπεράσματα και πιθανόν να βελτιστοποιήσουμε τα εργαλεία αυτά. Συγκεκριμένα το εργαλείο που χρησιμοποιήθηκε κατά κόρον είναι το GATK Mutec 2. Προκειμένου λοιπόν να επιτευχθεί ο στόχος της βελτιστοποίησης, γίνεται, σε τελευταίο στάδιο, ανάλυση και αξιολόγηση των αποτελεσμάτων των διαφορετικών εκτελέσεων των εργαλείων αυτών σε μία πληθώρα πειραμάτων, τα αποτελέσματα των οποίων παρουσιάστηκαν γραφικά στο προηγούμενο κεφάλαιο. Ο κώδικας των αλγορίθμων που συγγράφηκαν για την επίτευξη των προαναφερθέντων στόχων είναι διαθέσιμος προς οποιονδήποτε σε GitHub repository στο εξής link: https://github.com/tasos51/amitsigk-thesis. Στη συνέχεια αναφέρουμε τα συμπεράσματα που εξαγάγαμε κατά την ανάλυση του προβλήματος και αφορούν κυρίως το GATK και τον τρόπο που διαχειρίζεται τα δεδομένα.

1. Αρχικά, και όπως γίνεται εμφανές από τα διαφορετικά πειράματα του προηγούμενου κεφαλαίου, ο αριθμός των true positive μεταλλάξεων που βρίσκονται από το GATK εξαρτάται άμεσα από την κατανομή που ακολουθούν οι ποιότητες των reads. Για παράδειγμα ένα σύνολο δεδομένων με κακές ποιότητες θα οδηγήσει στη εύρεση πολύ λιγότερων μεταλλάξεων από ότι ένα με πολύ καλές ποιότητες.
2. Επιπλέον το GATK φαίνεται να παρουσιάζει πολύ μεγάλη ευαισθησία στο πλήθος των reads όσον αφορά τις μεταλλάξεις τις οποίες βρίσκει. Δηλαδή mutexed amplicons με χαμηλές συχνότητες (και συνεπώς μικρό πλήθος reads) δυσκολεύουν την εύρεση της μετάλλαξης.
3. Επιπροσθέτων το γεγονός ότι το GATK κάνει αυτόματα downsampling και φιλτράρει τα δεδομένα, φέρει ως αποτέλεσμα το να μην αντιστοιχούν μερικές μετρικές που υπολογίζουμε στις πραγματικές τιμές. Για παράδειγμα ακόμα και αν αυξήσουμε κατά πολλές τάξεις μεγέθους την παράμετρο που υποδεικνύει πόσα reads θα λαμβάνει υπόψη, υπάρχει περίπτωση η έκβαση του αποτελέσματος να είναι σχεδόν ίδια την περίπτωση όπου η παραπάνω παράμετρος είναι χαμηλή, λόγω του ότι το GATK θα κάνει και πάλι downsampling τα δεδομένα.
4. Επίσης το GATK μεταβάλλοντας την παράμετρο που αλλάζει πόσο πολλά reads λαμβάνει υπόψη, έχει μία ιδιαίτερη συμπεριφορά όσον αφορά τον εντοπισμό των μεταλλάξεων. Θα μπορούσε κάποιος να υποθέσει ότι αυξάνοντας την παράμετρο αυτή, όλο και περισσότερα reads θα λαμβάνονται υπόψη και συνεπώς μεγαλύτερη η πιθανότητα να βρεθεί μία μετάλλαξη. Στην πραγματικότητα όμως, είναι πού πιθανό το GATK να μην λαμβάνει τόσο σοβαρά υπόψη μεταλλάξεις με εξαιρετικά

μεγάλη συχνότητα αφού μία τέτοια στην πραγματικότητα δεν μπορεί εύκολα να καταταχθεί ως μετάλλαξη.

Τελικά, γίνεται εύκολα αντιληπτό ότι το GATK εργαλείο προς μελέτη έχει μία αρκετά περίπλοκη συμπεριφορά σε ότι αφορά την εύρεση σωματικών μεταλλάξεων. Συμπερασματικά ο παράγοντας που επηρεάζει περισσότερο είναι οι ποιότητες των reads κάτι που είτε μπορούμε να επιλέξουμε μέσω του προσομοιωτή, είτε μπορούμε να εστιάσουμε σε καλής ποιότητας sequencing data σε μία αληθινή εφαρμογή, εφόσον αυτό είναι δυνατό.

6.2 Μελλοντικές Επεκτάσεις

Τα τελικά προϊόντα της παρούσας διπλωματικής αποτελούν το πρόγραμμα του προσομοιωτή για την παραγωγή του τεχνητού συνόλου δεδομένων, bash script με το pipeline εκτέλεσης του GATK καθώς και πληθώρα αλγορίθμων για την αυτοματοποιημένη επεξεργασία των δεδομένων και την αξιολόγηση των εκτελέσεων. Όπως είναι φυσικό, όλα τα προαναφερθέντα προγράμματα επιδέχονται πληθώρα βελτιώσεων.

Το πρόγραμμα του προσομοιωτή έχει μεγάλα περιθώρια βελτίωσης ως προς τον χρόνο εκτέλεσής, πρώτον με τη χρήση πιο αποδοτικών μεθόδων προγραμματισμού και καλύτερους τρόπους συγγραφής κώδικα γλώσσας R και δεύτερον με την αλλαγή του τρόπου παραλληλοποίησης του αλγορίθμου. Για παράδειγμα προτείνεται μία ελαφρώς τροποποιημένη μέθοδος παραλληλοποίησης της τρίτης μεθόδου που αναλύθηκε στην ενότητα 3.7. Προκειμένου να μην διατηρούνται τεραστίων διαστάσεων λίστες στη μνήμη RAM αλλά ταυτόχρονα να μην υπάρχουν workers που δεν χρησιμοποιούνται προτείνεται εγγραφή των δεδομένων σε αρχείο αμέσως μετά την περάτωση της εργασίας του κάθε worker. Τονίζεται ότι πρώτον: με αυτή τη μέθοδο λόγο της τυχαιότητας της σειράς εκτέλεσης του κάθε thread τα blocks αποτελεσμάτων θα εγγραφούν με τυχαία σειράς το αρχείο, πράγμα που δεν μας απασχολεί καθόλου αφού τα ανακατεμένα αποτελέσματα είναι επιθυμητά ή αδιάφορα. Δεύτερον κάθε φορά που γίνεται εγγραφή στο αρχείο από ένα thread, το αρχείο θα πρέπει να κλειδώνεται ούτως ώστε να αποφευχθεί η ταυτόχρονη εγγραφή από δύο threads ταυτόχρονα και συνεπώς η καταστροφή των δεδομένων.

Όσον αφορά την βελτίωση της αυτοματοποίησης της ροής του pipeline που αναλύεται στο κεφάλαιο 4, προτείνεται η χρησιμοποίηση κάποιας γλώσσας διαχείρισης αυτοματοποιημένων ροών για επιστημονικές εφαρμογές όπως η Cromwell. Με τη χρησιμοποίηση κάποιας παρεμφερούς τεχνολογίας θα επιτευχθεί εύκολη επαναχρησιμοποίηση του pipeline καθώς και δυνατότητα παράλληλης εκτέλεσης ολόκληρου του pipeline σε cluster.

Σε ότι αφορά το πρώτο script της επεξεργασίας των αποτελεσμάτων μία σημαντική βελτίωση που θα μπορούσε να πραγματοποιηθεί είναι η παροχή των τεχνητών δεδομένων μαζί με την ακριβή θέση των μεταλλάξεων στα amplicons, ούτως ώστε να μην χρειάζεται να κάνουμε αναζήτηση των μεταλλάξεων που βρέθηκαν από το GATK σε ποια reads ανήκουν. Άλλωστε αν αντιμετωπίσουμε το πρόβλημα όπως περιεγράφηκε στην αρχή της διπλωματικής, δηλαδή με γνωστή είσοδο, θα πρέπει σε αυτήν να συμπεριλαμβάνονται και οι θέσεις των τεχνητών μεταλλάξεων που εισήχθησαν. Με αυτόν τον τρόπο όχι μόνο θα ελέγχουμε αν βρέθηκαν οι μεταλλάξεις σωστά, αλλά και αν βρέθηκαν στην κατάλληλη θέση ως προς το γονιδίωμα αναφοράς.

Τέλος σε ότι αφορά την εύρεση των μεταλλάξεων, ενδεχομένως να είναι απαραίτητη η σύγκριση των μεταλλάξεων που βρέθηκαν όχι μόνο με τις γνωστές του template αλλά και

με τις συμπληρωματικές του. Ίσως η παραπάνω διαδικασία να αυξήσει δραματικά τις true positives μεταλλάξεις που βρίσκονται αφού μπορούμε να υποθέσουμε ότι πολλές από αυτές έλαβαν χώρα στον δεύτερο κλώνο του DNA. Παρόλα αυτά είναι δύσκολο να εξακριβωθεί αν κάτι τέτοιο ισχύει στην πραγματικότητα αφού μέχρι τώρα μελετάμε artificially data.

Οποιεσδήποτε επεκτάσεις γίνουν και αφορούν τα παραπάνω ζητήματα καλό θα είναι να συνοδεύονται και από το κατάλληλο documentation και ασφαλώς, πλήρη σχολιασμό στις αλλαγές του κώδικα.

7 Βιβλιογραφία

[1] Π. Γ. Μπάγκος, Βιοπληροφορική, Πανεπιστήμιο Θεσσαλίας, 2015.

[2] «Μετάλλαξη - ΒΙΚΙΠΑΙΔΕΙΑ,» 15 Apr 2018. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%9C%CE%B5%CF%84%CE%AC%CE%BB%CE%BB%CE%B1%CE%BE%CE%B7.

[3] «DNA - ΒΙΚΙΠΑΙΔΕΙΑ,» 23 May 2020. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/DNA.

[4] «Amplicon - WIKIPEDIA,» 8 June 2020. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Amplicon.

[5] «Illumina - Introduction to Amplicon Sequencing,» [Ηλεκτρονικό]. Available: https://www.illumina.com/techniques/sequencing/dna-sequencing/targeted-resequencing/amplicon-sequencing.html.

[6] Y. Cai, J. Zheng, Y. Yang, V. Mai, Q. Mao και Y. Sun, «ESPRIT-Forest: Parallel clustering of massive amplicon sequence data in subquadratic time,» 24 April 2017.

[7] «ΒΙΚΙΠΑΙΔΕΙΑ - Παράλληλος προγραμματισμός,» 18 Απριλίου 2020. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%A0%CE%B1%CF%81%CE%AC%CE%BB%CE%BB%CE%B7%CE%BB%CE%BF%CF%82_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CF%8C%CF%82.

[8] «ΒΙΚΙΠΑΙΔΕΙΑ - Μέσος όρος,» 16 Μάρτιος 2020. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%9C%CE%AD%CF%83%CE%BF%CF%82_%CF%8C%CF%81%CE%BF%CF%82.

[9] «ΒΙΚΙΠΑΙΔΕΙΑ - Διακύμανση,» 23 Σεπτεμβρίου 2019. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%BA%CF%8D%CE%BC%CE%B1%CE%BD%CF%83%CE%B7.

[10] «ΒΙΚΙΠΑΙΔΕΙΑ - Τυπική απόκλιση,» 30 Δεκεμβρίου 2019. [Ηλεκτρονικό].

[11] Ζ. Χ. Γιώργος, «Δημιουργία Τυχαίων Αριθμών,» σε *Πιθανότητες και Στατιστική για Μηχανικούς*, σοφια, pp. 279-281.

[12] «WIKIPEDIA - FASTQ format,» 11 Μάιος 2020. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/FASTQ_format.

[13] «Quality (Phred) scores,» [Ηλεκτρονικό]. Available: https://drive5.com/usearch/manual/quality_score.html.

[14] T. Petzoldt, «rdocumentation - Algorithm For The Longest Common Subsequence Problem,» [Ηλεκτρονικό]. Available: https://www.rdocumentation.org/packages/qualV/versions/0.3-3/topics/LCS.

[15] «Babraham Bioinformatics - FastQC,» [Ηλεκτρονικό]. Available: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

[16] «Manual Reference - bwa - Burrows-Wheeler Alignment Tool,» [Ηλεκτρονικό]. Available: http://bio-bwa.sourceforge.net/bwa.shtml.

[17] «Sequence Alignment/Map Format Specification,» 30 Apr 2020. [Ηλεκτρονικό]. Available: https://samtools.github.io/hts-specs/SAMv1.pdf.

[18] «Broadinstitute - Picard - Command Line Overview,» [Ηλεκτρονικό]. Available: http://broadinstitute.github.io/picard/command-line-overview.html.

[19] «Samtools - Manual Page,» 6 December 2019. [Ηλεκτρονικό]. Available: http://www.htslib.org/doc/samtools.html.

[20] «broadinstitute - gatk,» [Ηλεκτρονικό]. Available: https://gatk.broadinstitute.org/hc/en-us.

[21] S. B. Institute, «GATK Forums Broad Institute,» 27 April 2015. [Ηλεκτρονικό]. Available: https://gatkforums.broadinstitute.org/gatk/discussion/5477/dp-and-ad. [Πρόσβαση 25 Jun 2020].

[22] «WIKIPEDIA - Box plot,» 31 May 2020. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Box_plot.

Παράρτημα Α

Προσδιορισμός Συντεταγμένων Μεταλλάξεων

Στο κεφάλαιο 4.2.1.2 γίνεται επεξήγηση των βημάτων του αλγορίθμου σύγκρισης των αποτελεσμάτων του GATK με τις γνωστές μεταλλάξεις του template. Σε ότι αφορά το Βήμα 15: Εξαιτίας του συμπεράσματος ότι στο template που μας δόθηκε δεν καταγράφονται με μεγάλη ακρίβεια οι θέσεις αρχής και τέλους των amplicon αλλά ούτε και οι θέσεις των μεταλλάξεων ως προς το γονιδίωμα αναφοράς, προκύπτει η ανάγκη για τον προσδιορισμό των θέσεων με ακρίβεια έτσι ΄ώστε όχι μόνο να ελέγχουμε αν οι μεταλλάξεις βρίσκονται σωστά αλλά και στη σωστή θέση. Στη συνέχεια όταν μας δόθηκε το αρχείο 'coodrinates.xls' γρήγορα αντιληφθήκαμε ότι οι μεταλλάξεις βρίσκονταν από το GATK σε μία θέση ελαφρώς μετατοπισμένη

$$< 50\ \theta\acute{\epsilon}\sigma\epsilon\iota\varsigma$$

από την αντίστοιχη καταγραφή τους στο αρχείο coordinates. Συνεπώς θέλαμε να δούμε αν αυτή η μετατόπιση σχετίζεται κάπως με το είδος της μετάλλαξης (insertion/deletion/change). Σε αυτό το βήμα λοιπόν, με ένα μικρό κομμάτι κώδικα το οποίο για λόγους απλότητας δεν θα αναλυθεί υπολογίσαμε τις διαφορές θέσεις των TP μεταλλάξεων που βρέθηκαν από το GATK με τις ήδη καταγεγραμμένες θέσεις του αρχείου coordinates, ανά amplicon. Για παράδειγμα αν ένα amplicon έχει ένα insertion και ένα change τότε ο πίνακας που δημιουργούμε είναι ο εξής:

Πίνακας 0-1

AmpliconID	**Position.diff**	**IndelType.Data**
SMC3.p.381.line.346.chr10.112343990.112343992_tile_1	-26;-25	change;insertion

Το συμπέρασμα που προκύπτει από την παραπάνω διαδικασία είναι ότι όλες οι μεταλλάξεις έχουν κάποια σταθερή μετατόπιση στην οποία προστίθεται μία τιμή συ σχετιζόμενη με τις βάσεις που εμπλέκονται στο είδος της μετάλλαξης. Αν και χρειάζεται περισσότερη διερεύνηση για να καταλήξουμε σε ασφαλή συμπεράσματα, ίσως αυτή η μεθοδολογία δεν έχει σημασία πλέον, αφού κανονικά για να αντιμετωπίσουμε το πρόβλημα με γνωστή είσοδο θα πρέπει να μας δίνονται ακριβώς οι θέσεις αρχής και τέλους των amplicons και οι θέσεις των μεταλλάξεων.