ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Ομαδοποίηση δεδομένων μοριακής βιολογίας από τεχνικές RNA-sequencing

Παπαλευθέρη Ελένη

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος

Δρ. Αριστείδης Βραχάτης

Λαμία, 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Ομαδοποίηση δεδομένων μοριακής βιολογίας από τεχνικές RNA-sequencing

Ελένη Παπαλευθέρη

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων

Δρ. Αριστείδης Βραχάτης

Λαμία, 2021

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί* ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*
2. *Δέχομαι ότι η αυτολεξεί* ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*
3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*
4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: …04…/..06…/2021……

Η Δηλούσα.

ΠΑΠΑΛΕΥΘΕΡΗ ΕΛΕΝΗ

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Ομαδοποίηση δεδομένων μοριακής βιολογίας από τεχνικές RNA-sequencing

Ελένη Παπαλευθέρη

Τριμελής Επιτροπή:

Δρ. Αριστείδης Βραχάτης , Επιβλέπων Καθηγητής

Δρ. Αθανάσιος Κακαρούντας, Αναπληρωτής Καθηγητής

Δρ. Σωτήριος Τασουλής , Επίκουρος Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου και επιβλέποντα κ. Αριστείδη Βραχάτη για την καθοδήγησή του και την εμπιστοσύνη που μου έδειξε για την εκπόνηση της παρούσας πτυχιακής εργασίας. Επίσης, ευχαριστώ όλους τους καθηγητές μου για την απόκτηση των πολύτιμων γνώσεων κοντά τους όλα αυτά τα χρόνια.

Σε προσωπικό επίπεδο θα ήθελα να ευχαριστήσω την οικογένειά μου, αλλά και τους φίλους μου που με στήριξαν στο ταξίδι αυτό. Αφιερώνω αυτή την εργασία στον σύζυγό μου.

Περίληψη

Οι απαιτήσεις των επιστημών στην ανακάλυψη εργαλείων ανάλυσης των σύγχρονων υπέρογκων δεδομένων διαρκώς μεγαλώνουν και παράλληλα αναγκαία είναι και η αξιολόγηση αυτών των λογισμικών μηχανικής μάθησης. Οι τεχνικές RNA sequencing, αποτελούν τη βάση για τη δημιουργία των νεοφερμένων αλλά και χρηστικών αυτών λογισμικών. Με την πιστοποίηση της ορθής ομαδοποίησης των δεδομένων αυτών ,μέσω των κατάλληλων μετρικών αξιολόγησης, επιτυγχάνεται τόσο η αναβάθμιση των παλιών όσο και η ανακάλυψη νέων scRNA εργαλείων. Το γεγονός αυτό επιτυγχάνεται με την προγραμματιστική μελέτη της παρούσας πτυχιακής εργασίας. Υλοποιήθηκε ένα σύγχρονο πακέτο λογισμικών ,σε python κώδικες υψηλού επιπέδου προγραμματισμού, παρέχοντας οπτικοποιημένα συμπεράσματα αξιολόγησης των παραπάνω εργαλείων. Έτσι, δίνεται η δυνατότητα βελτιστοποίησης των αδύναμων σημείων των εργαλείων αυτών αλλά και ο χώρος για νέες δημιουργίες προς υλοποίηση στο μέλλον.

Λέξεις κλειδιά: *υπέρογκα δεδομένα, RNA-sequencing, μηχανική μάθηση , ομαδοποίηση, μετρικές αξιολόγησης, scRNA εργαλεία, προγραμματισμός python*

Abstract

The demands of science in the discovery of tools for the analysis of modern big data are constantly growing and at the same time it is necessary to evaluate these machine learning softwares. RNA sequencing techniques are the basis for the creation of these new and useful softwares. By certifying the correct clustering of this data, through the appropriate evaluation metrics, both the upgrade of the old and the discovery of the new scRNA tools is achieved. This is achieved by the programming study of the present dissertation. A modern software package was implemented in high-level programming python codes, providing visualized evaluation conclusions of the above tools. Thus, it is possible to optimize the weak points of these tools but also give the space for new creations to be implemented in the future.

Keywords: *big data, RNA-sequencing, machine learning, clustering, evaluation metrics, scRNA tools, python programming*

Περιεχόμενα

1 ΕΙΣΑΓΩΓΗ 11-13

2 ΔΕΔΟΜΕΝΑ ΜΟΡΙΑΚΗΣ ΒΙΟΛΟΓΙΑΣ ΚΑΙ ΤΕΧΝΙΚΕΣ RNA-SEQUENCING 14

2.1 ΟΡΙΣΜΟΣ RNA SEQUENCING 14
2.2 ΔΙΑΔΙΚΑΣΙΑ RNA SEQUENCING ΣΕ ΜΟΡΙΑΚΑ ΔΕΔΟΜΕΝΑ 14-18

3 ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ 19

3.1 ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ ΚΑΙ PYTHON 19
3.2 ΟΡΙΣΜΟΣ ΚΑΙ ΜΕΘΟΔΟΛΟΓΙΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ 19-22

4 ΟΜΑΔΟΠΟΙΗΣΗ ΔΕΔΟΜΕΝΩΝ 23-25

5 ΕΚΤΙΜΗΣΗ ΑΠΟΔΟΣΗΣ ΑΛΓΟΡΙΘΜΩΝ ΟΜΑΔΟΠΟΙΗΣΗΣ ΜΕ ΜΕΤΡΙΚΕΣ ΑΞΙΟΛΟΓΗΣΗΣ 26

5.1 ADJUSTED RAND SCORE 28-38
5.2 MUTUAL INFORMATION BASED SCORES 39-43
5.3 HOMOGENEITY, COMPLETENESS AND V-MEASURE 44-49
5.4 CALINSKI-HARABASZ INDEX 50-51

6 ΕΡΓΑΛΕΙΑ ΟΜΑΔΟΠΟΙΗΣΗΣ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΑΛΛΗΛΟΥΧΙΑΣ RNA ΕΝΟΣ ΚΥΤΤΑΡΟΥ 51

6.1 PARCS (PHENOTYPING BY ACCELERATED REFINED COMMUNITY-PARTITIONING) 52-54
6.2 TREE-SNE (TREE OF T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING) 54-56
6.3 SNN-CLIQ (SHARED NEAREST NEIGHBOR CLIQUE) 56-58
6.4 GRAPHSCC 58-60
6.5 SCZIDESK 60-62
6.6 GINICLUST3 62-64
6.7 HYPERCLUSTER 64-65

7 ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ 66

7.1 ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΟΜΑΔΟΠΟΙΗΣΗΣ ΣΕ SINGLE-CELL ΔΕΔΟΜΕΝΑ 66-68
7.2 ΟΠΤΙΚΟΠΟΙΗΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΣΕ BAR-PLOTS & TABLES 68-73

8 ΣΥΜΠΕΡΑΣΜΑΤΑ 74

8.1 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ 74-75
8.2 ΕΠΟΜΕΝΑ ΒΗΜΑΤΑ 75-76

9 ΠΑΡΑΡΤΗΜΑ 77

9.1 ΕΓΧΕΙΡΙΔΙΟ 77
9.1.1 ΤΕΚΜΗΡΙΩΣΗ ΤΩΝ PYTHON SCRIPTS 77-85
9.1.2 Ο ΣΤΟΧΟΣ ΚΑΙ Η ΣΧΕΣΗ ΜΕΤΑΞΥ ΤΩΝ PYTHON SCRIPTS 85-87
9.2 ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΣΕ PYTHON SCRIPTS 88-100
9.3 ΠΡΟΣΑΡΜΟΓΕΣ ΣΤΑ SCRIPTS ΤΩΝ TOOLS 101-132
9.4 ΕΡΓΑΛΕΙΟ ΥΠΟΛΟΓΙΣΜΟΥ ΜΕΣΩΝ ΟΡΩΝ CSV ΑΡΧΕΙΩΝ 133-134

10 ΑΝΑΦΟΡΕΣ - ΒΙΒΛΙΟΓΡΑΦΙΑ 135-138

Ευρετήριο εικόνων-πινάκων

1 ΕΙΚΟΝΑ 3.1 : ΑΛΓΟΡΙΘΜΟΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ – ΠΑΡΑΔΕΙΓΜΑ ΒΑΣΙΣΜΕΝΟ ΣΕ ΔΕΝΔΡΑ ΑΠΟΦΑΣΕΩΝ 20

2 ΕΙΚΟΝΑ 3.2 : ΚΥΡΙΕΣ ΜΕΘΟΔΟΛΟΓΙΕΣ ΤΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ 21

3 ΕΙΚΟΝΑ 4.1 : ΚΥΡΙΑ ΒΗΜΑΤΑ ΑΛΓΟΡΙΘΜΟΥ ΟΜΑΔΟΠΟΙΗΣΗΣ 25

4 ΕΙΚΟΝΑ 5.1 : ΠΙΝΑΚΑΣ «CONFUSION MATRIX» ΜΕ ΜΕΤΡΙΚΕΣ ΤΑΞΙΝΟΜΗΣΗΣ 27

5 ΠΙΝΑΚΑΣ 1 : (TABLE 1) ΣΗΜΕΙΩΣΗ ΓΙΑ ΤΗ ΣΥΓΚΡΙΣΗ ΔΥΟ ΔΙΑΜΕΡΙΣΜΑΤΩΝ (Ο ΠΙΝΑΚΑΣ «CONTINGENCY TABLE» ΓΙΑ ΤΗ ΣΥΓΚΡΙΣΗ ΔΙΑΜΕΡΙΣΜΑΤΩΝ U ΚΑΙ V)..... 32

6 ΠΙΝΑΚΑΣ 2: (TABLE 2) ΤΥΠΟΣ ΓΙΑ ΤΟΝ ΑΡΙΘΜΟ ΖΕΥΓΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ (ΧΩΡΙΣ ΔΙΑΤΑΞΗ) ΤΩΝ ΤΕΣΣΑΡΩΝ ΤΥΠΩΝ 35

7 ΠΙΝΑΚΑΣ 3: (TABLE 3) ΑΠΛΟΠΟΙΗΜΕΝΟΣ ΠΙΝΑΚΑΣ « CONTINGENCY TABLE» 2 × 2 ΓΙΑ ΣΥΓΚΡΙΣΗ ΔΙΑΜΕΡΙΣΜΑΤΩΝ U ΚΑΙ V 37

8 ΕΙΚΟΝΑ 7.1: ΔΙΑΓΡΑΜΜΑΤΑ ΓΙΑ ΤΟ 1ο DATASET GSE103334 68

9 ΕΙΚΟΝΑ 7.2: ΔΙΑΓΡΑΜΜΑΤΑ ΓΙΑ ΤΟ 2ο DATASET GSE52583 69

10 ΕΙΚΟΝΑ 7.3: ΔΙΑΓΡΑΜΜΑΤΑ ΓΙΑ ΤΟ 3ο DATASET GSE59739 70

11 ΕΙΚΟΝΑ 7.4: ΔΙΑΓΡΑΜΜΑΤΑ ΓΙΑ ΤΟ 4ο DATASET GSE86469B 71

12 ΕΙΚΟΝΑ 7.5: ΠΙΝΑΚΑΣ ΓΙΑ ΣΥΓΚΡΙΣΗ ΤΩΝ TOOLS ΜΕ ΒΑΣΗ ΤΗ ΜΕΤΡΙΚΗ ADJUSTED MUTUAL INFO SCORE 72

13 ΕΙΚΟΝΑ 7.6: ΠΙΝΑΚΑΣ ΓΙΑ ΣΥΓΚΡΙΣΗ ΤΩΝ TOOLS ΜΕ ΒΑΣΗ ΤΗ ΜΕΤΡΙΚΗ ADJUSTED RAND SCORE 72

14 ΕΙΚΟΝΑ 7.7: ΠΙΝΑΚΑΣ ΓΙΑ ΣΥΓΚΡΙΣΗ ΤΩΝ TOOLS ΜΕ ΒΑΣΗ ΤΗ ΜΕΤΡΙΚΗ CALINSKI HARABASZ SCORE 72

15 ΕΙΚΟΝΑ 7.8: ΠΙΝΑΚΑΣ ΓΙΑ ΣΥΓΚΡΙΣΗ ΤΩΝ TOOLS ΜΕ ΒΑΣΗ ΤΗ ΜΕΤΡΙΚΗ COMPLETENESS SCORE 73

16 ΕΙΚΟΝΑ 7.9: ΠΙΝΑΚΑΣ ΓΙΑ ΣΥΓΚΡΙΣΗ ΤΩΝ TOOLS ΜΕ ΒΑΣΗ ΤΗ ΜΕΤΡΙΚΗ HOMOGENEITY SCORE 73

17 ΕΙΚΟΝΑ 7.10: ΠΙΝΑΚΑΣ ΓΙΑ ΣΥΓΚΡΙΣΗ ΤΩΝ TOOLS ΜΕ ΒΑΣΗ ΤΗ ΜΕΤΡΙΚΗ V-MEASURE SCORE 73

18 ΕΙΚΟΝΑ 9.1 : ΜΟΡΦΟΛΟΓΙΑ ΠΙΝΑΚΑ ΣΥΓΚΡΙΣΕΩΝ 83

19 ΕΙΚΟΝΑ 9.2 : ΔΙΑΓΡΑΜΜΑ ΜΕΤΡΙΚΩΝ HYPERCLUSTER 86

20 ΕΙΚΟΝΑ 9.3 : OUT.TXT ΑΡΧΕΙΟ 87

1. Εισαγωγή

Η διεπιστημονικότητα χαρακτηρίζει τον τομέα της Βιοπληροφορική ,καθώς η Βιολογία με την Πληροφορική, την Στατιστική αλλά και τα Μαθηματικά συνθέτουν μία πρωτοπόρα επιστήμη η οποία εξερευνά διαρκώς νέες διαδικασίες για την επίλυση ποικίλων βιολογικών προβλημάτων και την πλήρη κατανόηση των βασικών αρχών της Βιολογίας όμως και της ίδιας της ζωής. Πρόκειται για ένα ευρύ γνωστικό πεδίο το οποίο εφαρμόζεται ιδανικά στη σύγχρονη μοριακή ,δομική, πληθυσμιακή αλλά και περιβαλλοντική βιολογία. Ο διαρκώς αναπτυσσόμενος κλάδος της Βιοπληροφορικής έχει επιδείξει σημαντικά επιτεύγματα και κατέχει κεντρική θέση στις σύγχρονες εξελίξεις των επιστημών της ζωής. Μπορούμε πια να ανακτήσουμε, να συγκρίνουμε και να παρατηρήσουμε σημαντικά γεγονότα, εξετάζοντας λεπτομερώς το σύνολο των νουκλεϊκών οξέων, δηλαδή του DNA και του RNA , το οποίο οδηγεί στην εξερεύνηση πολλών γονιδίων και τελικά των πρωτεϊνών που παράγονται από εκείνα ,αλλά και των μεταβολικών μονοπατιών τα οποία συνθέτουν έναν οργανισμό (Baxevanis & Ouellette, 2016) . Μια κύρια κατεύθυνσή της βιοπληροφορικής αφορά την άντληση πληροφοριών από τις αλληλουχίες αυτές, με σκοπό την απόκτηση γνώσεων που θα συμβάλουν στην επίλυση πολλών προβλημάτων. Τα προγράμματα αποκρυπτογράφησης του γονιδιώματος του ανθρώπου, και όχι μόνο, συμβάλλουν στο γεγονός αυτό. Στις σύγχρονες ερευνητικές διαδικασίες της επιστήμης της βιολογίας ,λοιπόν, απαραίτητη είναι η εκμετάλλευση των υπέρογκων δεδομένων που καθημερινά ,λόγω της ραγδαίας εξέλιξης του ανθρώπου και των επιστημών, συσσωρεύονται και τελικά η εξαγωγή αποτελεσμάτων από εκείνα δυσκολεύει όλο και περισσότερο. Οι νέες ανακαλύψεις και η διαρκής πρόοδος στη δημιουργία μεθόδων ανίχνευσης επαναλαμβανόμενων προτύπων, παράγουν μοντέλα πρόβλεψης και λήψης αναγκαίων αποφάσεων. Η εξόρυξη δεδομένων και η ομαδοποίησή τους ,χρησιμοποιούνται για την ανάλυση αυτού του μεγάλου όγκου βιολογικών και μη δεδομένων τα οποία ονομάζονται «big data» (Baxevanis & Ouellette, 2016 ; Phil , 2013) .

Στην παρούσα πτυχιακή εργασία εφαρμόστηκαν πολλαπλά σύγχρονα εργαλεία ομαδοποίησης της βάση δεδομένων scRNA-tools, η οποία αποτελεί έναν κατάλογο πακέτων λογισμικού για την ανάλυση των

δεδομένων της αλληλουχίας RNA ενός κυττάρου.[4] Στη συνέχεια, δημιουργήθηκε ένα νέο πακέτο λογισμικού αποτελούμενο από python scripts τα οποία συμβάλουν στην αξιολόγηση των παραπάνω εργαλείων αλλά και την αυτοματοποιημένη οπτικοποίηση της απόδοσής τους τόσο σε μορφή «bar plots» όσο και «πινάκων συγκρίσεων» των εργαλείων ανά δύο μεταξύ τους. Έτσι, δίνετε μια ολοκληρωμένη εικόνα της χρηστικότητας του πακέτου που δημιουργήθηκε, βοηθώντας μελλοντικά στην βελτιστοποίηση των υπαρχόντων αλλά και τη δημιουργία νέων εργαλείων ομαδοποίησης και γενικότερα ανάλυσης μεγάλου όγκου δεδομένων.

Ειδικότερα, ακολουθεί μία παρουσίαση των περιεχομένων των κεφαλαίων της παρούσας εργασίας, με σκοπό την κατανόηση της μεθοδολογίας που ακολουθήθηκε για την διεκπεραίωσή της. Πιο αναλυτικά,

- Στο **1ο κεφάλαιο:**
  το οποίο αποτελεί την παρούσα ενότητα παρουσιάζεται η ανάγκη των σύγχρονων επιστημών στην εύρεση εργαλείων ανάλυσης δεδομένων αλλά και αξιολόγησής τους.
- Στο **2ο κεφάλαιο:**
  ορίζεται η έννοια του RNA sequencing, καθώς αποτελεί μια τεχνική ανάλυσης των βιολογικών δεδομένων πάνω στα στάδια της οποίας βασίζεται η δημιουργία των σύγχρονων εργαλείων-λογισμικών.
- Στο **3ο κεφάλαιο:**
  ορίζεται η μηχανική μάθηση η οποία αποτελεί την απαρχή της ανακάλυψης των πιο εξελιγμένων και χρηστικών τρόπων αξιοποίησης των υπέρογκων βιολογικών και μη δεδομένων.
- Στο **4ο κεφάλαιο:**
  δίνεται έμφαση στην επεξήγηση της έννοιας της ομαδοποίησης των δεδομένων σε συστάδες.
- Στο **5ο κεφάλαιο:**
  αναλύεται ο τρόπος αξιολόγησης των αποτελεσμάτων της συσταδοποίησης με τις πιο δημοφιλής μετρικές, οι οποίες ορίζονται σε βάθος.
- Στο **6ο κεφάλαιο:**
  παρουσιάζονται τα scRNA εργαλεία που εφαρμόστηκαν αλλά και το πρωταρχικό όραμα της κατασκευής τους.
- Στο **7ο κεφάλαιο:**

παραθέτω την υλοποίηση και την αξιολόγηση των ομαδοποιήσεων που διενεργήθηκε καθώς και ανάλογο σχολιασμό των οπτικοποιημένων αποτελεσμάτων.

- Στο **8ο κεφάλαιο:**
  περιέχονται τα συμπεράσματα που προκύπτουν απ' την παρούσα έρευνα όπως και επόμενα βήματα προς υλοποίηση στο μέλλον.
- Στο **9ο κεφάλαιο:**
  Παρατίθενται ένα επεξηγηματικό εγχειρίδιο, οι κώδικες του πρακτικού μέρους της παρούσας πτυχιακής εργασίας αλλά και οι κώδικες των scRNA tools,καθώς ορισμένοι τροποποιήθηκαν ελαφρώς για τους σκοπούς της εργασίας.
- Και τέλος στο **10ο κεφάλαιο:**
  περιλαμβάνονται οι αναφορές και οι βιβλιογραφικές παραπομπές σε επιστημονικά άρθρα (papers) και βιβλία που χρησιμοποιήθηκαν.

2. Δεδομένα μοριακής βιολογίας και τεχνικές RNA-Sequencing

2.1 Ορισμός RNA sequencing

Το RNA sequencing (RNA-seq) είναι μια sequence-based τεχνολογία η οποία με χρήση μεθόδων Next-Generation Sequencing (NGS) δείχνει τόσο την παρουσία όσο και την ποσότητα του RNA που υπάρχει σε ένα βιολογικό δείγμα, μια δεδομένη στιγμή. Παράλληλα, αναλύει το συνεχώς μεταβαλλόμενο κυτταρικό μεταγράφωμα (cellular transcriptome) (Costa-Silva et al., 2017). Το Next-Generation Sequencing (NGS) ή αλλιώς γνωστό και ως High-Throughput Sequencing, έχει εξαιρετικά υψηλή απόδοση, μέγιστη επεκτασιμότητα και μεγάλη ταχύτητα, επιτρέποντας στους ερευνητές να εκτελούν πολλές εφαρμογές και να αναλύουν βιολογικά συστήματα σε υψηλό επίπεδο (Koboldt et al., 2013). Τελευταία, η αλληλούχηση επόμενης γενιάς (NGS) με τις τεχνικές αλληλούχησης RNA (RNA-seq), έχει παρουσιάσει σημαντική πρόοδο και κάνει διαθέσιμα όλο και περισσότερα δεδομένα μεταγραφωμάτων και όχι μόνο.

2.2 Διαδικασία RNA-sequencing σε μοριακά δεδομένα

Το read mapping, ή αλλιώς alignment, αποτελεί το πρώτο βήμα πραγματοποίησης της διαδικασίας του RNA-sequencing . Εκατομμύρια short reads, δηλαδή μικρές αλληλουχίες αζωτούχων βάσεων, αντιστοιχίζονται σε ένα αρχικό γονιδίωμα αναφοράς. Ωστόσο, αυτές οι μικρές αλληλουχίες αντιστοιχούν σε πολλά κομμάτια, τόσο σε σωστά όσο και σε λανθασμένα. Άρα, αποθηκεύονται όλες οι τοποθεσίες της αντιστοίχισης. Ο τελικός στόχος είναι να βρεθεί η τοποθεσία στην οποία η αλληλουχία ταιριάζει καλύτερα. Στα περισσότερα υπάρχοντα alignment προγράμματα γίνεται ένα πρώτο πέρασμα ώστε να βρεθεί μια μειωμένη λίστα με τις πιθανές τοποθεσίες και στην συνέχεια ακολουθεί μια πλήρης αξιολόγηση των υποψήφιων τοποθεσιών από περίπλοκους αλγορίθμους που υλοποιούν local alignment.

Στη συνέχεια, έχοντας αποκτήσει τις γονιδιακές τοποθεσίες για όσο το δυνατόν περισσότερες αλληλουχίες, γίνεται η προσπάθεια να

κατηγοριοποιηθούν τα reads σε κάποιες βιολογικά ουσιώδεις μονάδες όπως για παράδειγμα στα εξώνια (exons), στις μεταγραφές (transcripts) ή στα γονίδια (genes).

Τα εξώνια (exons) είναι νουκλεοτιδικές αλληλουχίες που συνδέονται ομοιοπολικά μεταξύ τους και έτσι δημιουργούν το ώριμο messenger RNA (mRNA). Τα transcripts είναι προϊόντα μονόκλωνων ριβονουκλεϊκών οξέων (RNA) που συντίθενται κατά την μετάφραση του DNA και σχηματίζουν άλλα προϊόντα ώριμου RNA όπως το αγγελιοφόρο mRNA (messenger RNA), μεταφορικό tRNA (transport RNA) ή ριβοσωμικό rRNA (ribosomal RNA). Τα γονίδια (genes) είναι αλληλουχίες βάσεων του DNA στις οποίες περιέχονται πληροφορίες που καθορίζουν την σύνθεση μορίων RNA ή πολυπεπτιδίων, δηλαδή που οδηγούν στην έκφραση του DNA.

Η πιο απλή προσέγγιση ,της διασικασίας του RNA sequencing, σ' αυτό το βήμα απαριθμεί τα «reads» που επικαλύπτουν τα εξώνια σε ένα γονίδιο. Ωστόσο, χρησιμοποιούνται και εναλλακτικοί τρόποι σύνοψης που περιέχουν «reads» σε όλο το μήκος του γονιδίου ενσωματώνοντας έτσι και αλληλουχίες ιντρονίων (introns). Οι αλληλουχίες των ιντρονίων αφαιρούνται μέσω RNA splicing κατά την ωρίμανση του RNA, πράγμα που σημαίνει ότι δεν εκφράζονται στο τελικό mRNA.

Το επόμενο πολύ βασικό βήμα του RNA-seq , είναι η κανονικοποίηση (normalization) κατά την οποία τα «count reads» που έχουν συνοψιστεί παραπάνω μετασχηματίζονται επιτρέποντας έτσι σημαντικές συγκρίσεις μεταξύ δειγμάτων. Η κανονικοποίηση είναι σημαντική αφού το ποσοστό του mRNA το οποίο αντιστοιχίζεται σ' ένα συγκεκριμένο γονίδιο μπορεί να διαφέρει μεταξύ βιολογικών καταστάσεων. Η πραγματική έκφραση ενός γονιδίου λαμβάνεται ως το ποσό του mRNA ανά κύτταρο που παράγει το γονίδιο αυτό. (Evans et al. , 2018)

Μέσω των κανονικοποιημένων πλέον δεδομένων και με τη χρήση είτε στατιστικών μεθόδων είτε μεθόδων μηχανικής μάθησης, αναγνωρίζονται τα differentially expressed γονίδια. Τέτοιες μέθοδοι ομαδοποιούνται σε δυο κατηγορίες, τις παραμετρικές και τις μη-παραμετρικές. Στην επιστήμη της Βιοπληροφορικής δεν υπάρχει ακριβής συμφωνία για το ποιά μέθοδος υπερτερεί δίνοντας την απαραίτητη εγκυρότητα των αποτελεσμάτων.

Οι παραμετρικές μέθοδοι προσπαθούν να συλλάβουν όλη την πληροφορία που παρέχουν τα δεδομένα με τη βοήθεια συγκεκριμένων παραμέτρων. Προβλέπονται οι τιμές άγνωστων δεδομένων παρατηρώντας κάποιο υιοθετημένο μοντέλο και τις παραμέτρους του. Όταν χρησιμοποιούνται παραμετρικές μέθοδοι γίνεται η υπόθεση πως, μετά την κανονικοποίηση, η τιμή έκφρασης ενός γονιδίου αντιστοιχίζεται σε μία συγκεκριμένη κατανομή (π.χ. κατανομή Poisson, Negative Binomial).

Αντιθέτως, η χρήση μη-παραμετρικών μεθόδων μπορεί να συλλαμβάνει περισσότερη πληροφορία για την κατανομή των δεδομένων ,καθώς δε χρησιμοποιούνται γνωστές κατανομές με συγκεκριμένες παραμέτρους. Αυτό είναι εφικτό καθώς τέτοιες μέθοδοι υποθέτουν πως η κατανομή των δεδομένων δεν μπορεί να εκφραστεί από κάποιο σαφώς ορισμένο αριθμό παραμέτρων κι έτσι το ποσό της πληροφορίας που εξάγεται από τα δεδομένα αυξάνεται μέσω πολυπλοκότερων κατανομών.

Αποτέλεσμα του RNA sequencing είναι η Βιολογική διορατικότητα (Biological Insight). Δηλαδή, η δημιουργία λιστών με γονίδια «Differentially Expressed» δεν είναι πάντα ο απώτερος σκοπός, καθώς παρατηρώντας τις αλλαγές στην έκφραση ορισμένων ομάδων γονιδίων ο ερευνητής μπορεί να οδηγηθεί σε μεγαλύτερη βιολογική γνώση πάνω στο σύστημα που εξετάζει.

Γενικότερα, μέχρι και πριν 20 χρόνια οι μικροσυστοιχίες του DNA είχαν αναδυθεί ως μια ισχυρή τεχνική μέτρησης των μεταγράφων (transcripts) mRNA (γονιδιακή έκφραση). Έχουν χρησιμοποιηθεί περισσότερο από κάθε άλλη τεχνική ώστε, να αξιολογήσουν τις διαφορές στην αφθονία mRNA σε διαφορετικά βιολογικά δείγματα. Όμως στην πορεία των ετών, αναδείχθηκε η πιο ισχυρή τεχνική του RNA-seq (Heilkenbrinker et al., 2013; McGettigan, 2013). Η διαδικασία της μέτρησης της γονιδιακής έκφρασης και στις δύο τεχνικές αποτελείται από τα ακόλουθα στάδια (Edwards ,2004) δημιουργώντας υψηλής απόδοσης δεδομένα γονιδιακής έκφρασης:

- **Στάδιο Σχεδιασμού:**

  Επιλέγονται τα βιολογικά δείγματα για τη σύγκριση της γονιδιακής έκφρασης.

- **Στάδιο Προετοιμασίας RNA και παγίδων συμπληρωματικού DNA :**

Στην περίπτωση της τεχνικής των μικροσυστοιχιών, απομονώνεται, μετατρέπεται και μαρκάρεται το RNA με φθορίζουσες χρωστικές ενώ στην περίπτωση της τεχνικής RNA-seq, μετατρέπεται το RNA σε cDNA και αποθηκεύεται σε μια βιβλιοθήκη.

- **Στάδιο Απόκτησης Δεδομένων:**

Ανακτώνται τα δεδομένα, δηλαδή τα υβριδοποιημένα δείγματα για τις μικροσυστοιχίες, τα οποία περιέχουν σε στερεά μορφή συμπληρωματικού (complementary) DNA ή ολιγονουκλεοτιδίων που αντιστοιχούν σε γνωστά γονίδια. Αντίστοιχα για τη περίπτωση του RNA-seq, εκτελείται αλληλούχιση επόμενης γενιάς.

- **Στάδιο Ανάλυσης Δεδομένων:**

Στις μικροσυστοιχίες η έκφραση των δεδομένων αναλύεται για να ταυτοποιήσει τα διαφοροποιημένα γονίδια, χρησιμοποιώντας ανάλυση διακυμάνσεων (ANOVA) και διάγραμμα διασποράς (scatter plots) ή ομαδοποίηση γονιδίων/δειγμάτων (clustering). Για RNA-seq τα raw reads χαρτογραφούνται σε ένα αναφορικό μεταγράφωμα (reference transcriptome) (ή γονιδίωμα) και συναρμολογείται˙ σε ορισμένες περιπτώσεις η συναρμολόγηση προηγείται της ευθυγράμμισης. Η πληθικότητα (counts) των reads χρησιμοποιείται για να εξαχθεί το σχετικό επίπεδο έκφρασης των εξονίων ή των μεταγράφων.

- **Στάδιο Βιολογικής Επιβεβαίωσης:**

Πάνω σε αυτά τα ευρήματα εκτελείται το τελευταίο στάδιο της βιολογικής επιβεβαίωσης και για τις δύο περιπτώσεις. Τα αποτελέσματα αποθηκεύονται σε μια βάση δεδομένων ώστε έτσι να διαμοιράζονται και να εκτελούνται περαιτέρω αναλύσεις. Πιο συγκεκριμένα, τα δεδομένα αποθηκεύονται συνήθως στη βάση δεδομένων του NCBI (National Center for Biotechnology Information) και πιο συγκεκριμένα στη βάση δεδομένων SRA (Sequence Read Archive[1]), αλλά και σε άλλες βάσεις δεδομένων όπως αυτή

[1] http://www.ncbi.nlm.nih.gov/sra

της Ensembl [2], που αφορά κυρίως ολόκληρα γονιδιώματα . Τα δεδομένα που αποθηκεύονται στην SRA συνοδεύονται από πληροφορίες που αφορούν τη μελέτη, πληροφορίες του βιολογικού δείγματος, τα πειραματικά δεδομένα, το είδος της πλατφόρμας, την ανάλυση αλλά και το χρόνο υποβολής των δεδομένων (Kodama et al., 2012).

[2] http://www.ensembl.org/index.html

3. Μηχανική μάθηση

3.1 Μηχανική μάθηση και Python

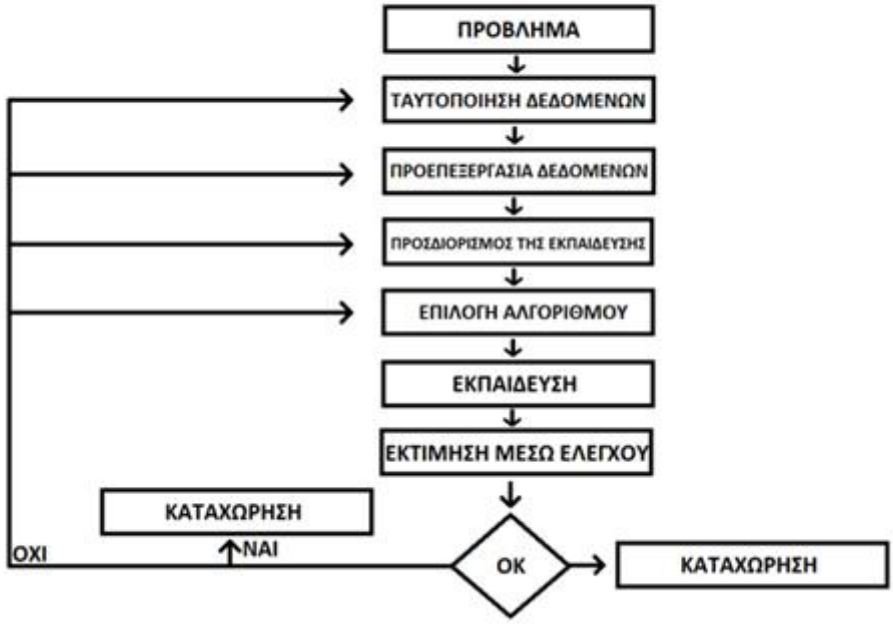
Η αλματώδης ανάπτυξη της τεχνολογίας και οι νέες απαιτήσεις της επιστήμης της Βιολογίας έχουν δημιουργήσει την ανάγκη επεξεργασίας αλλά και ανάλυσης όλο και μεγαλύτερου όγκου δεδομένων και πληροφοριών τα οποία εντάσσονται στην κατηγορία των «big data» . Ο όγκος και ο ρυθμός με τον οποίο παράγονται αυτά τα δεδομένα οδηγεί στην ανάπτυξη νέων αποτελεσματικότερων τεχνικών και μεθόδων επεξεργασίας τους με σκοπό να είναι σε θέση να αξιοποιηθούν από την επιστημονική κοινότητα.

Η επεξεργασία των μεγάλων δεδομένων πραγματοποιείται με διάφορα «εργαλεία». Όμως, με την υψηλού επιπέδου δυναμική γλώσσα προγραμματισμού Python η επεξεργασία αυτή γίνεται αποδοτικότερα και ευκολότερα. Ο προγραμματισμός των σύγχρονων συστημάτων μηχανικής μάθησης γίνεται συνήθως με αυτή τη γλώσσα προγραμματισμού ακόμη και όταν η Python δεν είναι η βασική γλώσσα προγραμματισμού του συστήματος. Ωστόσο, αυτή η γλώσσα χρησιμοποιείται από τους περισσότερους επιστήμονες του κλάδου των δεδομένων λόγω της πληθώρας βιβλιοθηκών, των έτοιμων αλγορίθμων μηχανικής μάθησης αλλά και επεξεργασίας δεδομένων. Γι’ αυτό, στην προγραμματιστική μελέτη της παρούσας εργασίας χρησιμοποιήθηκε η γλώσσα αυτή, παρουσιάζοντας την εφαρμογή της στον τομέα της Βιοπληροφορικής μέσω της μηχανικής μάθησης.

3.2 Ορισμός και μεθοδολογίες της μηχανικής μάθησης

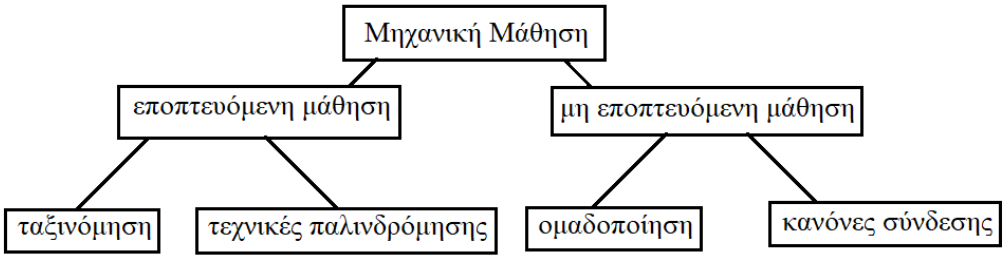
Ο όρος μηχανική μάθηση (Machine Learning) αναφέρεται στον τρόπο με τον οποίο προγράμματα ηλεκτρονικών υπολογιστών, εκπαιδεύονται από τις «εμπειρίες» τους αναγνωρίζοντας σύνθετα πρότυπα μέσα από πολλαπλά δεδομένα εισόδου. Με αυτόν τον τρόπο ένα υπολογιστικό σύστημα μπορεί αυτόνομα να ακολουθήσει «έξυπνες» αποφάσεις. Γι’ αυτό η μηχανική μάθηση αποτελεί την βασική αρχή της τεχνητής νοημοσύνης. (Phil ,2013)

Η διαδικασία της μηχανικής μάθησης σε διάγραμμα ροής δεδομένων:



Εικόνα 3.1 *αλγόριθμος μηχανικής μάθησης – παράδειγμα βασισμένο σε δένδρα αποφάσεων*

Η μηχανική μάθηση περιλαμβάνει μια ιεραρχία μεθοδολογιών, η οποία απεικονίζεται στην εικόνα 3.2 , που αναλύονται παρακάτω.



Εικόνα 3.2 *Κύριες μεθοδολογίες της μηχανικής μάθησης*

Σύμφωνα με τους Han , Kamber και Pei (Han et al., 2012) η μηχανική μάθηση γενικά περιλαμβάνει : την καθοδηγούμενη μάθηση , την μη καθοδηγούμενη μάθηση, την ημι-καθοδηγούμενη μάθηση και την ενεργή μάθηση.

Η καθοδηγούμενη μάθηση (Supervised learning) είναι στην πραγματικότητα όρος ταυτόσημος με την κατηγοριοποίηση (classification) και λειτουργεί με μηχανικά αναγνωρίσιμα πρότυπα τα οποία χρησιμοποιούνται ως παραδείγματα με σκοπό την εκπαίδευση των συστημάτων. Το μαθησιακό μοντέλο χρησιμοποιεί επισημασμένα πρότυπα αναλύοντας τα δεδομένα εκπαίδευσης και μαθαίνει τελικά να ομαδοποιεί τα δεδομένα εισόδου του. Ένας αλγόριθμος καθοδηγούμενης μάθησης παράγει ένα μοντέλο το οποίο στη συνέχεια μπορεί να λειτουργήσει ως πρότυπο. Όταν εκτελεσθεί το βέλτιστο σενάριο του αλγόριθμου, το σύστημα καθορίζει την «ετικέτα»-κλάση της ομάδας και το αποτέλεσμα ταυτίζεται με την πραγματικότητα. Έτσι ο αλγόριθμος καταφέρνει να διαχωρίζει με σωστό τρόπο άγνωστα μέχρι τώρα παραδείγματα για εκείνον. Μερικοί αλγόριθμοι καθοδηγούμενη μάθηση είναι τα νευρωνικά δίκτυα, η γραμμική παλινδρόμηση, τα δέντρα αποφάσεων και άλλοι.

Η μη καθοδηγούμενη μάθηση (Unsupervised learning) είναι ουσιαστικά η διαδικασία της συσταδοποίησης. Στη διαδικασία της μη καθοδηγούμενης μάθησης, τα δεδομένα εισόδου δεν είναι κατηγοριοποιημένα , δηλαδή δε διαθέτουν κάποιο αναγνωριστικό που να τα διαχωρίζει ,οπότε το μοντέλο δεν υποθέτει αλλά δημιουργεί αυτόνομα

δικά του πρότυπα ,ως συστάδες δεδομένων ,χωρίς κάποια σημασιολογική αξία. Έτσι, η αξιολόγησή των αρχικά μη κατηγοριοποιημένων δεδομένων εισόδου δεν πραγματοποιείται με κάποιο συγκεκριμένο τρόπο. Οπότε, τα αποτελέσματα στις δομές που παράγονται δεν είναι ακριβή. Γι' αυτό και ξεχωρίζει η μη καθοδηγούμενη μάθηση από την καθοδηγούμενη η οποία όπως ήδη αναφέρθηκε μέσω ενός βέλτιστου σεναρίου μπορεί να δώσει αποτελέσματα με μεγάλη ακρίβεια. Παραδείγματα αλγορίθμων μη καθοδηγούμενης μάθησης είναι η ομαδοποίηση (clustering), η ανίχνευση ανωμαλιών και άλλοι.

Η ενεργή μάθηση (Active learning) είναι ένα μαθησιακό μοντέλο, όπου οι χρήστες καθορίζουν το αποτέλεσμα της διαδικασίας της μάθησης. Ο εμπειρογνώμονας είναι εκείνος που δίνει ένα πρότυπο για τον αλγόριθμο. Αυτό το πρότυπο μπορεί να προέρχεται από μια διαδικασία καθοδηγούμενης ή μη καθοδηγούμενης μάθησης. Με τον τρόπο αυτό βελτιώνεται η ποιότητα του μαθησιακού μοντέλου.

Παράλληλα, η ημι-καθοδηγούμενη μάθηση (Semi-supervised learning) ορίζεται από τεχνικές μάθησης που κάνουν χρήση επισημασμένων προτύπων και μη επισημασμένων προτύπων. Η ημι-καθοδηγούμενη μάθηση δίνει καλύτερα αποτελέσματα όταν έχει να κάνει με «δυαδικά» προβλήματα, δηλαδή προβλέποντας δύο περιπτώσεις - ομάδες διαχωρίζοντας πλήρως τα όρια των δύο συστάδων μεταξύ τους. Για παράδειγμα, αν η μία κατηγορία αφορά «νίκες» και η άλλη «ήττες» χρησιμοποιούμε τα επισημασμένα πρότυπα ,δηλαδή την εποπτευόμενη μάθηση, για να οριστεί καλύτερα το όριο το οποίο καθορίζει την απόφαση για το που ανήκει κάθε δεδομένο εισόδου. (Han et al., 2012)

4. Ομαδοποίηση δεδομένων (clustering)

Κατά την εξόρυξη δεδομένων αρχικά πραγματοποιείται η εύρεση συσχετίσεων μεταξύ των δεδομένων με χρήση των κανόνων συσχέτισης. Έπειτα, η ταξινόμηση δηλαδή η κατηγοριοποίησή τους σε προκαθορισμένες κλάσεις με τη βοήθεια δέντρων αποφάσεων, νευρωνικά δίκτυα, κατηγοριοποίησης bayesian κλπ. και τέλος η συσταδοποίηση των δεδομένων ,για παράδειγμα με ιεραρχικούς αλγόριθμους. Πιο συγκεκριμένα, η ταξινόμηση (classification) σχετίζεται με την ανάπτυξη ενός μοντέλου που προβλέπει την κλάση των στιγμιότυπων ενός προβλήματος. Το μοντέλο αυτό χτίζεται με βάση ένα σύνολο δεδομένων εκπαίδευσης και η απόδοσή του αξιολογείται με βάση ένα σύνολο δεδομένων ελέγχου. Παράλληλα, η συσταδοποίηση (clustering) ,όπως θα αναφερθεί και παρακάτω ,έχει να κάνει με τον διαχωρισμό των δεδομένων σε ομάδες. Η ομαδοποίηση , καθώς χαρακτηρίζεται από την ικανότητά της να κάνει άμεσα τις απαραίτητες αλλαγές ,ώστε να ανταποκρίνεσαι συνεχώς σε μεταβαλλόμενες ανάγκες, εφαρμόζεται σε όλους σχεδόν τους επιστημονικούς τομείς. Μερικοί κλάδοι που βρίσκει εφαρμογή είναι η Μηχανική Μάθηση, η Βιολογία, η Ιατρική, η Αστρονομία, η Οικονομία, το Μάρκετινγκ και άλλοι πολλοί.

Πιο συγκεκριμένα, η ομαδοποίηση ή συσταδοποίηση (clustering) είναι η διαδικασία που οργανώνει πρότυπα ,όπως παρατηρήσεις, δεδομένα ή διανύσματα, σε ομάδες ή αλλιώς συστάδες (clusters). Τα μέλη κάθε συστάδας ομοιάζουν σύμφωνα με διάφορα κριτήρια .
Η συσταδοποίηση βασίζεται κυρίως στην ομοιογένεια ,όπου στοιχεία εντός της ίδιας συστάδας είναι παρόμοια μεταξύ τους και στον διαχωρισμό ,όπου στοιχεία σε διαφορετικές συστάδες είναι διαφορετικά μεταξύ τους (Jain et al. , 1999). Η ομαδοποίηση αποτελεί μάθηση χωρίς επίβλεψη, η οποία ξεκινά από μια βάση δεδομένων που αποτελείται από δεδομένα εκπαίδευσης και έχει ως σκοπό να διαχωριστούν τα εκπαιδευόμενα στοιχεία σε ομάδες. Κατά καιρούς έχουν αναπτυχθεί διάφοροι αλγόριθμοι ομαδοποίησης για τη συγκέντρωση διαφορετικών τύπων δεδομένων τα οποία παράγονται από διαφορετικές πηγές ενώ για το ίδιο σύνολο δεδομένων, διαφορετικοί αλγόριθμοι θα δώσουν ξεχωριστά αποτελέσματα. Ειδικότερα, δεν υπάρχει ένας μόνο τρόπος να μπορεί να προκύψει η ακριβέστερη συσταδοποίηση για ένα σύνολο δεδομένων εισόδου. Έτσι, έχουμε πολλαπλές υλοποιήσεις αλγόριθμων ομαδοποίησης που επιλέγονται αναλόγως το αντικείμενο μελέτης και τη μορφή των δεδομένων. (Sharma & Asuda. , 2018)

Η διαδικασία της ομαδοποίησης ενός πλήθους δεδομένων πραγματοποιείται όπως ο Buhmann (2002) εισήγαγε ακολουθώντας κάποια βασικά στάδια υλοποίησης με το εξής μοντέλο:

- αναπαράσταση δεδομένων,
- μοντελοποίηση,
- βελτιστοποίηση και
- επικύρωση.

Ακόμη, οι Jain et al. (1999) παρέχουν ένα μοντέλο που επεκτάθηκε από τους Theodoridis & Koutroumbas (2009) και τροποποιήθηκε από τον Skourtaniotis (2016) περιγράφοντας τη διαδικασία της συσταδοποίησης με τα εξής στάδια:

- συλλογή δεδομένων,
- προετοιμασία δεδομένων,
- μέτρο εγγύτητας,
- κριτήριο ομαδοποίησης,
- αλγόριθμος ομαδοποίησης και
- επικύρωση.

Η συλλογή δεδομένων είναι η αρχική φάση της διαδικασίας ομαδοποίησης. Ο ερευνητής συνήθως καταλήγει σε μερικά μη επεξεργασμένα πρωτογενή δεδομένα που είναι ακατάλληλα για άμεση επεξεργασία. Έτσι, απαιτείται η προετοιμασία των δεδομένων που περιλαμβάνει την επιλογή χαρακτηριστικών, τα δεδομένα προ επεξεργάζονται και μετασχηματίζονται. Ο σκοπός αυτών των σημαντικών διαδικασιών είναι η δημιουργία ενός λειτουργικού συνόλου δεδομένων που θα εισαχθούν στον αλγόριθμο ομαδοποίησης. Η επιτυχία ενός αλγορίθμου εξαρτάται άμεσα από την προετοιμασία του συνόλου των δεδομένων εισόδου. Στη συνέχεια, ως μέτρο εγγύτητας ορίζουμε κάποιο μέτρο της απόστασης των δεδομένων. Είναι μια απόφαση που εξαρτάται από τον τύπο μίας αντιπροσωπευτικής συστάδας. Ένα πολύ συχνό μέτρο είναι η Ευκλείδεια απόσταση. Παράλληλα, ορίζεται ένα κριτήριο ομαδοποίησης δηλαδή με αυτόν τον όρο εννοούμε το είδος των δομών των συστάδων, που μπορεί να ανακαλυφθούν στο σύνολο των δεδομένων και να είναι για παράδειγμα συμπαγές ή επίμηκες (Theodoridis& Koutroumbas, 2009) . Το κριτήριο ομαδοποίησης είναι το ίδιο με το στάδιο που σύμφωνα με τον Buhmann (2002) περιγράφεται ως μοντελοποίηση. Στη συνέχεια, επιλέγεται ο αλγόριθμος ομαδοποίησης που θα χρησιμοποιηθεί, καθορίζονται οι παράμετροι που απαιτούνται και εκτελείτε η ομαδοποίηση. Ο ορισμός των τιμών των παραμέτρων του

αλγορίθμου συσταδοποίησης συνήθως επιτυγχάνεται μέσω της εκτέλεσης κάποιων δεικτών εγκυρότητας. Ωστόσο, αυτή η διαδικασία δεν πρέπει να συγχέεται με το τελευταίο στάδιο της συνολικής διαδικασία ανάλυσης της ομαδοποίησης ,που είναι η επικύρωση. Σε αυτό το τελικό στάδιο συμβαίνει η αξιολόγηση των αποτελεσμάτων της ομαδοποίησης.

Ειδικότερα, οι αλγόριθμοι ομαδοποίησης δεδομένων (Jain et al. ,1999) μπορούν να χωριστούν σε δυο κατηγορίες:

• **Ιεραρχικοί αλγόριθμοι:**

Οι ιεραρχικοί σχηματίζουν ομάδες με βάση τις ήδη υπάρχουσες σχηματισμένες ομάδες, δηλαδή λειτουργούν επαναληπτικά λαμβάνοντας υπόψη τους σε κάθε επανάληψη τις ομάδες που δημιουργήθηκαν προηγουμένως. Οι ιεραρχικοί αλγόριθμοι μπορούν να σχεδιαστούν με δύο τεχνικές, από πάνω προς τα κάτω ή από κάτω προς τα πάνω. Αυτό σημαίνει πως κάθε στοιχείο του αρχικού συνόλου δεδομένων μπορεί να θεωρηθεί στην αρχή ξεχωριστή ομάδα (από κάτω προς τα πάνω τεχνική), όπου στη συνέχεια εφαρμόζονται συγχωνεύσεις ομάδων. Ακόμα, μπορεί να θεωρηθεί ότι το σύνολο των δεδομένων είναι μία ομάδα (από πάνω προς τα κάτω τεχνική) όπου στη συνέχεια θα διαιρείται.

• **Αλγόριθμοι διαίρεσης:**

Στους αλγόριθμους διαίρεσης οι ομάδες σχηματίζονται χωρίς την επαναληπτική εκτέλεση. Η ιεραρχική ομαδοποίηση δημιουργεί μία ιεραρχία ομάδων. Η συνήθης αναπαράσταση της ιεραρχίας γίνεται με ένα δέντρο όπου στο ένα του άκρο περιέχονται ξεχωριστά αντικείμενα και στο άλλο η ομάδα στην οποία ανήκει το αντικείμενο.

Εικόνα 4.1 *κύρια βήματα αλγόριθμου ομαδοποίησης*

5. Εκτίμηση απόδοσης αλγορίθμων ομαδοποίησης με μετρικές αξιολόγησης

Μετά την ομαδοποίηση των δεδομένων είναι σημαντικό να επικυρωθεί η εγκυρότητα των ομάδων που προέκυψαν, καθώς κάθε αλλαγή στις παραμέτρους του αλγόριθμου και κάθε προσπάθεια επεξεργασίας των δεδομένων πιθανόν να δώσει διαφορετικά αποτελέσματα. Έτσι, είναι απαραίτητο να ελεγχθούν οι ομάδες που προκύπτουν, τόσο με στατιστικά όσο και με βιολογικά κριτήρια για δεδομένα εισόδου μοριακής βιολογίας (Dalton et al., 2009).Τα κριτήρια εγκυρότητας των ομάδων χωρίζονται στις εξής κατηγορίες:

- **Εσωτερικά κριτήρια εγκυρότητας:**
  Αυτά τα κριτήρια βασίζονται στη σύσταση των ίδιων των ομάδων. Πιο συγκεκριμένα, αυτά τα κριτήρια χαρακτηρίζουν καλύτερους τους αλγόριθμους ομαδοποίησης που δίνουν ομάδες των οποίων τα δεδομένα που τις αποτελούν μοιάζουν πολύ μεταξύ τους και έτσι υπάρχει μεγάλη εσωτερική ομοιότητα (ομοιογένεια) και μικρή «εξωτερική» ομοιότητα ανάμεσα στις ομάδες ,ως σύνολα, μεταξύ τους. Παραδείγματα τέτοιων κριτηρίων είναι τα Davies-Bouldin index, Bayesian information criteria, Linear Algebra measure και άλλα.
- **Εξωτερικά κριτήρια εγκυρότητας:**
  Τα εξωτερικά κριτήρια συγκρίνουν τις συστάδες ,που προέκυψαν ως αποτέλεσμα της ομαδοποίησης, με τις πραγματικές ομάδες που αντιστοιχούν στα δεδομένα εισόδου. Ουσιαστικά, τα εξωτερικά κριτήρια εξετάζουν πόσο κοντά είναι οι παραγόμενες ομάδες σε σχέση με τις ομάδες μιας προδιαγεγραμμένης μορφής. Παραδείγματα τέτοιων κριτηρίων είναι τα Rand measure, confusion matrix,Fowlkes-Mallows index,mutual information και άλλα. Στο πρακτικό κομμάτι της παρούσας εργασίας χρησιμοποιούνται εξωτερικά κριτήρια για την αξιολόγηση των scRNA tools.

Στην παρούσα ενότητα αναλύονται οι μετρικές αξιολόγησης ,της ικανότητας ομαδοποίησης, που χρησιμοποιούνται παρακάτω στην συγκεκριμένη πτυχιακή εργασία.

Αρχικά, βασική είναι η κατανόηση του «contingency matrix» πριν την εμβάθυνση στις μετρικές. Ο πίνακας «contingency matrix» για τα προβλήματα ταξινόμησης είναι ένας πίνακας «square confusion matrix» όπου η σειρά των γραμμών και των στηλών αντιστοιχεί σε μια λίστα κλάσεων.

Στον τομέα της μηχανικής μάθησης και συγκεκριμένα στο πρόβλημα της στατιστικής ταξινόμησης, ένας πίνακας confusion matrix, επίσης γνωστός ως πίνακας σφάλματος, είναι μια συγκεκριμένη διάταξη πίνακα που επιτρέπει την οπτικοποίηση της απόδοσης ενός αλγορίθμου. Ένας πίνακας confusion matrix είναι ένας πίνακας με δύο σειρές και δύο στήλες που αναφέρει τον αριθμό των ψευδώς θετικών, ψευδώς αρνητικών, πραγματικών θετικών και πραγματικών αρνητικών. Η ακρίβεια (accuracy , δηλαδή το ποσοστό του συνολικού αριθμού προβλέψεων που είναι σωστές) θα αποφέρει παραπλανητικά αποτελέσματα εάν το σύνολο των δεδομένων δεν είναι ισορροπημένο. Δηλαδή, όταν ο αριθμός των παρατηρήσεων σε διαφορετικές κλάσεις ποικίλλει πολύ. Ορίζοντας ,λοιπόν, ένα πείραμα από positive instances P και negative instances N για κάποια κατάσταση, τα τέσσερα αποτελέσματα μπορούν να διατυπωθούν σε έναν πίνακα 2×2 , confusion matrix ως εξής:

Προβλεπόμενη Κλάση

Πραγματική Κλάση		**ΘΕΤΙΚΗ**	**ΑΡΝΗΤΙΚΗ**	
	ΘΕΤΙΚΗ	True Positive **Αληθινά θετική**	False Negative **TYPE II ERROR** **Λανθασμένα αρνητική**	**Ευαισθησία (sensitivity)** $\frac{TP}{(TP+FN)}$
	ΑΡΝΗΤΙΚΗ	False Positive **TYPE I ERROR** **Λανθασμένα θετική**	True Negative **Αληθινά αρνητική**	**Ειδικότητα (specificity)** $\frac{TN}{(TN+FP)}$
		Ακρίβεια (Precision) $\frac{TP}{(TP+FP)}$	**αρνητική προγνωστική τιμή** $\frac{TN}{(TN+FN)}$	**Ορθότητα (Accuracy)** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Εικόνα 5.1 *: Πίνακας «Confusion matrix» με μετρικές ταξινόμησης*

Ο contingency matrix (sklearn.metrics.cluster.contingency_matrix) αναφέρει την intersection cardinality (καρδικότητα διασταύρωσης) για κάθε αξιόπιστο πραγματικό / προβλεπόμενο ζεύγος συστάδων (true, prediction). Ο πίνακας - contingency matrix προβλέπει επαρκή στατιστικά στοιχεία για όλες τις μετρήσεις ομαδοποίησης όπου τα δείγματα είναι ανεξάρτητα και ταυτόσημα κατανεμημένα και δεν χρειάζεται να ληφθούν υπόψη ορισμένες περιπτώσεις που δεν συγκεντρώνονται.

Ουσιαστικά, ο contingency Matrix επιτρέπει την εξέταση της εξάπλωσης κάθε πραγματικού cluster σε προβλεπόμενες ομάδες και αντίστροφα. Ο υπολογισμός του πίνακα contingency table χρησιμοποιείται συνήθως για τον υπολογισμό μιας στατιστικής ομοιότητας - similarity statistic μεταξύ των δύο ομάδων.

Ωστόσο, ο πίνακας contingency matrix είναι εύκολο να ερμηνευτεί για έναν μικρό αριθμό συστάδων, αλλά γίνεται πολύ δύσκολο να ερμηνευτεί για μεγάλο αριθμό συστάδων. Τέλος, δε δίνει ούτε μία μετρική (metric) για χρήση ως στόχο τη βελτιστοποίηση ομαδοποίησης.

5.1 Adjusted Rand index

Ο Adjusted Rand Index (ARI) χρησιμοποιείται συχνά στην επικύρωση συστάδων (cluster validation), δεδομένου ότι αποτελεί μέτρο συμφωνίας μεταξύ δύο «τμημάτων» όπου το ένα δίνεται από τη διαδικασία ομαδοποίησης και το άλλο ορίζεται από εξωτερικά κριτήρια. Δεδομένης της γνώσης των αληθινών εκχωρήσεων class assignments - labels_true και των αντιστοιχιών του αλγορίθμου ομαδοποίησης των ίδιων δειγμάτων - label_pred, ο adjusted rand index (προσαρμοσμένος δείκτης rand) είναι μια συνάρτηση που μετρά την ομοιότητα των δύο αναθέσεων, αγνοώντας τις παραλλαγές και με τυχαία ομαλοποίηση (normalization).

Τα πλεονεκτήματα[3] του δείκτη ARI είναι τα εξής:

- Οι τυχαίες (ομοιόμορφες) αναθέσεις ετικετών έχουν βαθμολογία ARI κοντά στο 0,0 για οποιαδήποτε τιμή n_clusters και n_samples (κάτι που για παράδειγμα δεν ισχύει για τον ακατέργαστο δείκτη Rand ή το μέτρο V).

[3] https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

- Οριοθετημένο εύρος [-1, 1]: οι αρνητικές τιμές είναι «κακές» (π.χ. για ανεξάρτητες ετικέτες), παρόμοιες ομαδοποιήσεις έχουν θετικό ARI, το 1.0 είναι το τέλειο αποτέλεσμα αγώνα.

Ενώ, το μοναδικό μειονέκτημά[3] του είναι:

- Σε αντίθεση με την αδράνεια, το ARI απαιτεί γνώση των αληθινών κατηγοριών (ground truth classes) , ενώ σχεδόν ποτέ δεν είναι διαθέσιμο στην πράξη ή απαιτεί χειροκίνητη ανάθεση από ανθρώπινους σχολιαστές (human annotators) (όπως στο εποπτευόμενο περιβάλλον μάθησης-supervised learning setting).

  Ωστόσο, το ARI μπορεί επίσης να είναι χρήσιμο σε μια καθαρά χωρίς επίβλεψη ρύθμιση (purely unsupervised setting) ως δομικό στοιχείο (building block) για ένα Consensus Index που μπορεί να χρησιμοποιηθεί για τη συλλογή μοντέλων (clustering model selection) (TODO).

Αναλύωντας μαθηματικά τον δείκτη ARI:

Περιληπτικά,

Εάν το C είναι μια εκχώρηση αληθινής κατηγορίας (ground truth class assignment) και το K ομαδοποίησης (the clustering), ας ορίσουμε a και b ως:

- a, ο αριθμός ζευγών στοιχείων που βρίσκονται στο ίδιο σύνολο στο C και στο ίδιο σύνολο στο K
- b, ο αριθμός ζευγών στοιχείων που βρίσκονται σε διαφορετικά σύνολα στο C και σε διαφορετικά σύνολα στο K

Ο μη προσαρμοσμένος δείκτης Rand - unadjusted Rand index δίνεται στη συνέχεια :

$$RI = \frac{a + b}{C_2^{n_{samples}}}$$

Όπου $C_2^{n_{samples}}$ είναι ο συνολικός αριθμός πιθανών ζευγών στο σύνολο των δεδομένων (χωρίς κατάταξη).

Ο δείκτη Rand index θεωρείται το μέτρο του ποσοστού των σωστών αποφάσεων που λαμβάνονται από τον αλγόριθμο ομαδοποίησης. Ωστόσο,

υπάρχουν μερικά γνωστά προβλήματα με τον RI, όπως το γεγονός ότι η αναμενόμενη τιμή του RI δύο τυχαίων κατατμήσεων δεν παίρνει σταθερή τιμή (πχ το μηδέν) και ότι η στατιστική Rand πλησιάζει το ανώτατο όριο της μονάδας καθώς αυξάνεται ο αριθμός των συστάδων. Δηλαδή, η βαθμολογία RI δεν εγγυάται ότι οι τυχαίες αναθέσεις ετικετών (random label assignments) θα πάρουν μια τιμή κοντά στο μηδέν, για παράδειγμα, εάν ο αριθμός των συστάδων είναι στην ίδια τάξη μεγέθους (order of magnitude) με τον αριθμό των δειγμάτων.

Για να αντιμετωπίσουμε αυτό το αποτέλεσμα μπορούμε να μειώσουμε το αναμενόμενο RI - E[RI] τυχαίων ετικετών καθορίζοντας τον προσαρμοσμένο δείκτη Rand (adjusted Rand index) ως εξής:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

Ο adjusted Rand index είναι η «διορθωμένη» έκδοση του Rand index (Hubert L.& Arabie P., 1985). Η διαδικασία διόρθωσης απαιτεί ένα μοντέλο για τυχαίες ομαδοποιήσεις και τη γνώση για το πώς σχεδιάζονται οι ομαδοποιήσεις από το τυχαίο μοντέλο. Αυτές οι απαιτήσεις μπορεί να έχουν δραματικά αποτελέσματα και επομένως η επιλογή ενός συγκεκριμένου μοντέλου για τυχαίες ομαδοποιήσεις θα πρέπει να αιτιολογείται με βάση την κατανόηση του σεναρίου ομαδοποίησης. Μια κακή επιλογή του τυχαίου μοντέλου μπορεί να μην είναι αρκετά «τυχαία» και να κωδικοποιεί κρίσιμα χαρακτηριστικά των ομαδοποιήσεων σε όλες τις τυχαίες ομαδοποιήσεις, παρέχοντας μια κακή βάση. Ταυτόχρονα, ένα τυχαίο μοντέλο μπορεί να είναι «πολύ τυχαίο» στο οποίο χάνονται κρίσιμα χαρακτηριστικά σε μια θάλασσα τυχαίων ομαδοποιήσεων που δεν είναι αντιπροσωπευτικά του συγκεκριμένου προβλήματος. Ο χαρακτηρισμός τυχαίων μοντέλων είναι ένα σημαντικό θέμα στο κομμάτι της έρευνας (Sethna, 2006; Goldenberg et al., 2010; Mansour, 2012).

Ο δείκτης Rand διορθώθηκε χρησιμοποιώντας το Μοντέλο ομαδοποιήσεων Permutation (Gates &Yong-Yeol, 2017) κατά το οποίο ο αριθμός και το μέγεθος των συστάδων εντός μιας ομαδοποίησης είναι σταθεροί και όλες οι τυχαίες ομαδοποιήσεις δημιουργούνται ανακατεύοντας τα στοιχεία μεταξύ των σταθερών συστάδων δηλαδή χωρίς να υπάρχει κατάταξη. Ωστόσο, οι προϋποθέσεις του μοντέλου συχνά παραβιάζονται καθώς σε πολλές περιπτώσεις ομαδοποίησης ο αριθμός των συστάδων ή/και η κατανομή μεγέθους αυτών των συστάδων ποικίλλουν. Για παράδειγμα, θεωρούμε στον αλγόριθμο συσταδοποίησης K-means ότι

ο αριθμός των συστάδων καθορίζεται από τον «επαγγελματία» , αλλά τα μεγέθη αυτών των συστάδων θα προκύψουν από τα δεδομένα.

Για

$$ARI_{model}(A,B) = \frac{RI(A,B) - E_{model}[RI(A,B)]}{1 - E_{model}[RI(A,B)]}$$

Είναι

$$\mathbb{E}_{\text{perm}}[\text{RI}\ (\mathcal{A},\mathcal{B})] = \frac{\Sigma_i \binom{a_i}{2} \Sigma_j \binom{b_j}{2}}{\binom{N}{2}} + \left(1 - \frac{\Sigma_i \binom{a_i}{2}}{\binom{N}{2}}\right)\left(1 - \frac{\Sigma_j \binom{b_j}{2}}{\binom{N}{2}}\right)$$

Πιο αναλυτικά,

Δίνεται ένα σύνολο S = {O1 ,. . . . ., On} που αποτελείται από n αντικείμενα και δύο διαφορετικά τμήματα του S είναι τα υποσύνολα U ={u1 ,. . . . ., u_R} και V={v1 ,. . . . ., v_C}. Τα υποσύνολα U και V αντιστοιχούν στα C και K που αναφέρθηκαν προηγουμένως.

Όπου:

$$U_{i=1}^{R} u_i = S = U_{j=1}^{C} v_j$$

$$u_i \bigcap u_{i'} = \emptyset = v_j \bigcap v_{j'}$$

$$1 \le i \ne i' \le R$$

$$1 \le j \ne j' \le C$$

Το nij υποδηλώνει τον αριθμό των αντικειμένων που είναι κοινά στις κλάσεις ui και vj και η πληροφορία για την κλάση , που αλληλεπικαλύπτεται μεταξύ των δύο υποσυνόλων U και V , μπορεί να γραφτεί με τη μορφή contingency table με ni. τον αριθμό των αντικειμένων στην κλάση ui (γραμμή i) και n.j που αναφέρεται στον αριθμό των αντικειμένων στην κλάση vj (στήλη j), όπως φαίνεται στον Table 1 που ακολουθεί (Hubert & Arabie , 1985).

		ΔΙΑΜΕΡΙΣΜΑ V				
	ΚΛΑΣΗ	**V1**	**V2**	**...**	**Vc**	**ΑΘΡΟΙΣΜΑΤΑ**
ΔΙΑΜΕΡΙΣΜΑ U	u_1	n_{11}	n_{12}	...	n_{1C}	$n_{1\cdot}$
	u_2	n_{21}	n_{22}		n_{2C}	$n_{2\cdot}$
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
	u_R	n_{R1}	n_{R2}	...	n_{RC}	$n_{R\cdot}$
	ΑΘΡΟΙΣΜΑΤΑ	$n_{\cdot 1}$	$n_{\cdot 2}$	...		$n_{..} = n$

Πίνακας 1 *: (Table 1) Σημείωση για τη σύγκριση δύο διαμερισμάτων (Ο πίνακας «Contingency Table» για τη σύγκριση διαμερισμάτων U και V)*

Η Rand βασίζεται σε μετρήσεις αντιστοιχίας μεταξύ των συνόλων U και V στον τρόπο ταξινόμησης των ζευγών αντικειμένων στον πίνακα R X C contingency table (Rand,1971). Συγκεκριμένα, υπάρχουν τέσσερις διαφορετικοί τύποι "τμημάτων" μεταξύ των $\binom{n}{2}$ διακριτών ζευγών που θα μπορούσαν να βρεθούν:

- Τύπος 1 (TYPE 1):
  i. Αντικείμενα στο ζεύγος τοποθετούνται στην ίδια κλάση στο U και στην ίδια κλάση στο V.
  ii. Ο τύπος αυτός ερμηνεύει τη συμφωνία στην ταξινόμηση σε κλάσεις των αντικειμένων ενός ζεύγους.

iii. Εναλλακτικά ονομάζουμε αυτόν τον τύπο τμημάτων a.

- Τύπος 2 (TYPE 2):
  i. Αντικείμενα στο ζεύγος τοποθετούνται σε διαφορετικές κλάσεις στο U και σε διαφορετικές κλάσεις στο V.
  ii. Ο τύπος αυτός ερμηνεύει τη συμφωνία στην ταξινόμηση σε κλάσεις των αντικειμένων ενός ζεύγους.
  iii. Εναλλακτικά ονομάζουμε αυτόν τον τύπο τμημάτων b.

- Τύπος 3 (TYPE 3):
  i. Αντικείμενα στο ζεύγος τοποθετούνται σε διαφορετικές κλάσεις στο U και στην ίδια κλάση στο V.
  ii. Ο τύπος αυτός ερμηνεύει τη διαφωνία στην ταξινόμηση σε κλάσεις των αντικειμένων ενός ζεύγους.
  iii. Εναλλακτικά ονομάζουμε αυτόν τον τύπο τμημάτων c.

- Τύπος 4 (TYPE 4):
  i. Αντικείμενα στο ζεύγος τοποθετούνται στην ίδια κλάση στο U και σε διαφορετικές κλάσεις στο V.
  ii. Ο τύπος αυτός ερμηνεύει τη διαφωνία στην ταξινόμηση σε κλάσεις των αντικειμένων ενός ζεύγους.
  iii. Εναλλακτικά ονομάζουμε αυτόν τον τύπο τμημάτων d.

Εάν το A αντιπροσωπεύει τον συνολικό αριθμό συμφωνιών και το D τον συνολικό αριθμό διαφωνιών, τότε :

$$A + D = \binom{n}{2}$$

Έτσι, αποδεικνύεται ότι (Brennan & Light, 1974):

$$A = \binom{n}{2} + \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \frac{1}{2}\left(\sum_{i=1}^{R} n_i^2 + \sum_{j=1}^{C} n_{\cdot j}^2\right)$$
$$= \binom{n}{2} + 2\sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} - \left(\sum_{i=1}^{R}\binom{n_i}{2} + \sum_{j=1}^{C}\binom{n\cdot j}{2}\right)$$

(σχέση 1)

όπου ένας διωνυμικός συντελεστής $\binom{m}{2}$ ορίζεται ως 0 όταν m = 0 ή 1.

Το μοντέλο «model for randomness» υποθέτει ότι ο πίνακας contingency table R x C κατασκευάζεται από τη γενικευμένη κατανομή hypergeometric distribution όπου αποδεικνύεται ότι (Hubert 1977, Klastorin 1985, Fowlkes and Mallows 1983):

$$E\left(\sum_{i,j}\binom{n_{ij}}{2}\right) = \sum_{i}\binom{n_{i.}}{2}\sum_{j}\binom{n_{.j}}{2} / \binom{n}{2}$$

(σχέση 2)

Η σχέση (2) δίνει τον αναμενόμενο αριθμό ζευγών αντικειμένων του τύπου “TYPE 1”, δηλαδή ζεύγη στα οποία τα αντικείμενα τοποθετούνται στην ίδια κλάση στο U και στην ίδια κλάση στο V.

Ο αναμενόμενος αριθμός ζευγών αντικειμένων του TYPE 1 που αποδίδεται σε ένα συγκεκριμένο κελί είναι ο αριθμός των ζευγών στη σειρά του επί τον αριθμό των ζευγών στη στήλη του δια του $\binom{n}{2}$,δηλαδή:

$$E\left(\binom{n_{ij}}{2}\right) = \binom{n_{i.}}{2}\binom{n_{.j}}{2} / \binom{n}{2}$$

Όλοι οι τύποι στον πίνακα Table 2 που ακολουθεί είναι σταθεροί γραμμικοί μετασχηματισμοί του : $\sum_{i,j} \frac{n_{ij}}{2}$.

TYPE	FORMULA
(i)	$\frac{1}{2}\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}(n_{ij}-1)$
(ii)	$\frac{1}{2}\left(n^2+\sum_{i=1}^{R}\sum_{j=1}^{c} n_{ij}^2-\left(\sum_{i=1}^{R} n_i^2+\sum_{j=1}^{C} n_{jj}^2\right)\right)$
(iii)	$\frac{1}{2}\left(\sum_{j=1}^{C} n_j^2-\sum_{i=1}^{R}\sum_{j=1}^{c} n_{ij}^2\right)$
(iv)	$\frac{1}{2}\left(\sum_{i=1}^{R} n_i^2-\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2\right)$
(i) +(ii) = A =	$\binom{n}{2}+\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2-\frac{1}{2}\left(\sum_{i=1}^{R} n_i^2+\sum_{j=1}^{C} n_{.j}^2\right)$
(iii) +(iv) = D =	$\frac{1}{2}\left(\sum_{i=1}^{R} n_i^2+\sum_{j=1}^{C} n_{.j}^2\right)-\sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2$

***Πίνακας 2**: (Table 2) Τύπος για τον αριθμό ζευγών αντικειμένων (χωρίς διάταξη) των τεσσάρων τύπων (Hubert & Arabie , 1985)*

Άρα, οι προσδοκώμενες τιμές (expectations) για οποιαδήποτε από αυτές τις εκφράσεις μπορούν να ληφθούν απευθείας από την expectation στη (σχέση 2). Για παράδειγμα, χρησιμοποιώντας αυτό το αποτέλεσμα και το τελευταίο μέρος της σχέσης (σχέση 1), διαπιστώνουμε ότι το μέτρο Rand έχει expectation:

$$E\left(A/\binom{n}{2}\right) = 1 + 2\sum_{i}\left(\frac{n_{i.}}{2}\right)\sum_{j}\left(\frac{n_{.j}}{2}\right)/\binom{n}{2}^2 - \left[\sum_{i}\binom{n_{i.}}{2}+\sum_{j}\binom{n_{.j}}{2}\right]/\binom{n}{2}$$

(σχέση 3)

Έτσι, χρησιμοποιώντας τη γενική μορφή ενός index που διορθώνεται κατά τύχη:

$$\frac{Index - Expected\ Index}{Maximum\ Index - Expected\ Index}$$

(σχέση 4)

που ορίζεται παραπάνω από το 1 και παίρνει την τιμή 0 όταν ο δείκτης ισούται με την αναμενόμενη τιμή του, ο διορθωμένος δείκτης Rand index θα έχει τη μορφή (υποθέτοντας έναν μέγιστο δείκτη Rand index το 1):

$$\frac{\sum_{i,j}\binom{n_{ij}}{2} - \sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2} / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_i}{2} + \sum_j \binom{n_{.j}}{2}\right] - \sum_i \binom{n_i}{2}\sum_j \binom{n_{.j}}{2} / \binom{n}{2}}$$

(σχέση 5)

Γενικά, η κανονικοποιημένη έκφραση στη (σχέση 3) είναι αναλλοίωτη υπό θετικούς γραμμικούς μετασχηματισμούς του αρχικού raw index. Επομένως, λόγω της σταθερής οριακής υπόθεσης στον πίνακα contingency table R x C, η επιλογή κάποιου από τους τέσσερις τύπους ζευγών που αναφέρονται στον πίνακα Table 2 χρησιμοποιείται ως βάση στη γενική μορφή εξίσωσης στη (σχέση 4) και είναι «τυχαία» δεδομένου ότι όλες οι επιλογές στον Πίνακα 2 είναι γραμμικοί μετασχηματισμοί του αριθμού των ζεύγη αντικειμένων (χωρίς διάταξη), $\sum_{i,j}\binom{n_{ij}}{2}$.

Η κανονικοποίηση που δίνεται στη (σχέση 4) προϋποθέτει ότι μεγαλύτερες θετικές τιμές του δείκτη raw index συνεπάγονται μεγαλύτερους βαθμούς αντιστοιχίας τμημάτων (partition correspondence). Εάν το αντίστροφο είναι αλήθεια, δηλαδή, κάποιος έχει δείκτες που βασίζονται σε ζεύγη αντικειμένων τύπων “TYPE 3” και “TYPE 4” , τότε η κατεύθυνση του κανονικοποιημένου δείκτη πρέπει να αντιστραφεί αντικαθιστώντας το μέγιστο στον παρονομαστή από το ελάχιστο. Σε αυτήν την περίπτωση, ωστόσο, η χρήση ζευγών αντικειμένων τύπου TYPE 3 ή TYPE 4 και ένας ελάχιστος δείκτης 0 θα εξακολουθούσε να οδηγεί απευθείας στη (σχέση 5) και έναν κανονικοποιημένο δείκτη βάσει του $\sum_{i,j}\binom{n_{ij}}{2}$.

Συνοψίζοντας, ακολουθεί απλοποιημένος 2 × 2 Contingency Table(Santos & Embrechts ,2009):

Διαμέρισμα U	V	
	Σύζευξη στην ίδια ομάδα	Σύζευξη σε διαφορετική ομάδα
Σύζευξη στην ίδια ομάδα	a	b
Σύζευξη σε διαφορετική ομάδα	c	d

Πίνακας 3: *(Table 3) Απλοποιημένος πίνακας « Contingency Table» 2 × 2 για σύγκριση διαμερισμάτων U και V (Santos & Embrechts ,2009)*

Όπου:

$$a = \sum_{r=1}^{R}\sum_{c=1}^{C}\binom{t_{rc}}{2} = \left(\sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2 - n\right)/2$$

$$b = \sum_{r=1}^{R}\binom{t_{r.}}{2} - a = \left(\sum_{r=1}^{R} t_{r.}^2 - \sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2\right)/2$$

$$c = \sum_{c=1}^{C}\binom{t_{.c}}{2} - a = \left(\sum_{c=1}^{C} t_{.c}^2 - \sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2\right)/2$$

$$d = \binom{n}{2} - a - b - c = \binom{n}{2} - \sum_{r=1}^{R}\binom{t_{r.}}{2} - \sum_{c=1}^{C}\binom{t_{.c}}{2} + a$$

$$= \left(\sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2 + n^2 - \sum_{r=1}^{R} t_{r.}^2 - \sum_{c=1}^{C} t_{.c}^2\right)/2$$

Και όπου t_{rc} αντιπροσωπεύει κάθε στοιχείο του R X C contingency table (Table 1). (Santos & Embrechts ,2009)

Ο Rand Index μπορεί εύκολα πλέον να υπολογιστεί από τη σχέση:

$$RI = \frac{a+d}{a+b+c+d}$$

Ωστόσο, για την αποφυγή προβλημάτων που ήδη αναφέρθηκαν ο Adjusted Rand Index (ARI) προτείνεται από τους Hubert και Arabie ως βελτίωση του RI (Hubert & Arabie , 1985). Ο ARI μπορεί να υπολογιστεί από τον ακόλουθο τύπο με expected value 0 και maximum value 1: (Santos & Embrechts ,2009)

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]}$$

Ή

$$ARI = \frac{\binom{n}{2}\sum_{r=1}^{R}\sum_{c=1}^{C}\binom{t_{rc}}{2} - \left[\sum_{r=1}^{R}\binom{t_{r.}}{2}\sum_{c=1}^{C}\binom{t_{.c}}{2}\right]}{\frac{1}{2}\binom{n}{2}\left[\sum_{r=1}^{R}\binom{t_{r.}}{2} + \sum_{c=1}^{C}\binom{t_{.c}}{2}\right] - \left[\sum_{r=1}^{R}\binom{t_{r\cdot}}{2}\sum_{c=1}^{C}\binom{t_{.c}}{2}\right]}$$

5.2 Mutual Information based scores (Βαθμολογίες αμοιβαίων πληροφοριών)

Γνωρίζοντας τις αληθινές εκχωρήσεις κλάσεων (ground truth class assignments) - labels_true και τις αναθέσεις των αλγορίθμων ομαδοποίησης των ίδιων δειγμάτων - labels_pred, το Mutual Information είναι μια συνάρτηση που μετρά τη συμφωνία των δύο αναθέσεων, αγνοώντας τις παραλλαγές. Διατίθενται δύο διαφορετικές ομαλοποιημένες εκδόσεις αυτού του μέτρου, η Κανονικοποιημένη Αμοιβαία Πληροφορία (Normalized Mutual Information-NMI) και η Προσαρμοσμένη Αμοιβαία Πληροφορία (Adjusted Mutual Information-AMI) . Το NMI χρησιμοποιείται συχνά στη βιβλιογραφία, ενώ το AMI προτάθηκε πιο πρόσφατα και ομαλοποιείται τυχαία (normalized against chance)**.**

Η βαθμολογία ομοιότητας - similarity score υπάρχει στο εύρος [0; 1], όπου το 1 αντιστοιχεί σε πανομοιότυπες ομαδοποιήσεις και το 0 συνεπάγεται ανόμοιες ομαδοποιήσεις στο μέγιστο βαθμό. Ωστόσο, στην πράξη, δεν χρησιμοποιείται με ευκολία το πλήρες εύρος τιμών μεταξύ 0 και 1, με πολλές συγκρίσεις να συγκεντρώνονται κοντά στις ακραίες τιμές (Vinh et al., 2009; Hubert & Arabie, 1985). Αυτό καθιστά δύσκολη την άμεση ερμηνεία των αποτελεσμάτων μιας σύγκρισης.

Ακολουθούν τα πλεονεκτήματα[3] του MI:

- Οι τυχαίες (ομοιόμορφες) αναθέσεις ετικετών (Random (uniform) label assignments) έχουν βαθμολογία AMI κοντά στο 0 για οποιαδήποτε τιμή n_clusters και n_samples. Γεγονός που δεν ισχύει για τις raw Mutual Information ή για το μέτρο V-measure .

- Ανώτερο όριο (Upper bound) το 1 : Οι τιμές που πλησιάζουν το μηδέν υποδεικνύουν δύο αναθέσεις ετικετών που είναι σε μεγάλο βαθμό ανεξάρτητες, ενώ οι τιμές κοντά στο 1 υποδηλώνουν σημαντική «συμφωνία». Όταν AMI είναι ακριβώς 1 δείχνει ότι οι δύο αναθέσεις ετικετών είναι ίσες ,με ή χωρίς παραλλαγή (permutation).

Ενώ, τα μειονέκτηματά[3] του είναι:

- Σε αντίθεση με την αδράνεια, τα μέτρα που βασίζονται στο MI (MI-based measures) απαιτούν τη γνώση των αληθινών συστάδων (ground truth classes) **,** ενώ σχεδόν ποτέ δεν είναι διαθέσιμα στην πράξη ή απαιτούν χειροκίνητη ανάθεση (manual assignment by human annotators) από τους ανθρώπινους σχολιαστές (όπως στο εποπτευόμενο περιβάλλον μάθησης - supervised learning setting). Ωστόσο, τα μέτρα που βασίζονται σε MI (MI-based measures) μπορούν επίσης να είναι χρήσιμα σε καθαρά χωρίς επίβλεψη ρύθμιση (purely unsupervised setting) ως δομικό στοιχείο (building block) για ένα Consensus Index που μπορεί να χρησιμοποιηθεί για την ομαδοποίηση της επιλογής μοντέλων.
- Τα NMI και MI ουσιαστικά δεν προσαρμόζονται (adjusted) τυχαία .

Αναλύωντας μαθηματικά το Mutual Information based score:

Ένα μέτρο ομοιότητας ομαδοποίησης το οποίο βασίζεται στις πληροφορίες Shannon μεταξύ των πιθανών αναπαραστάσεων κάθε ομαδοποίησης είναι η «εντροπία διαμερισμάτων» (Gates & Ahn, 2017). Αυτές οι κατανομές πιθανότητας υπολογίζονται από τον πίνακα contingency table T . Ας υποθέσουμε δύο αναθέσεις ετικετών U και V από τα ίδια N αντικείμενα . Η εντροπία τους είναι το ποσό της αβεβαιότητας για ένα σύνολο διαμερισμάτων (partition set), που ορίζεται παρακάτω. Έστω η εντροπία διαμερισμάτων H μιας ομαδοποίησης U δίνεται από :

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log\left(P(i)\right)$$

όπου P(i)=|Ui|/N είναι η πιθανότητα από την οποία ένα αντικείμενο επιλέχθηκε τυχαία U εμπίπτει στην κατηγορία Ui. Ομοίως για V με P′(j)=|Vj|/N έχουμε :

$$H(V) = -\sum_{j=1}^{|V|} P'(j)\log\left(P'(j)\right)$$

Όπου contingency table T, ο πίνακας contingency table T για δύο ομαδοποιήσεις A ={A_1,…,A_{KA}} και B ={ B_1,…,B_{KB}}των στοιχείων N,

όπου $n_{ij} = |A_i \cap B_j|$ είναι ο αριθμός των στοιχείων που βρίσκονται και στις δύο ομάδες $A_i \in \mathcal{A}$ και $B_j \in \mathcal{B}$.

Χρησιμοποιώντας την παραπάνω εντροπία, η mutual information MI(A,B) μεταξύ δύο ομάδων A και B δίνονται ως εξής:

$$\text{MI}(\mathcal{A},\mathcal{B}) = H(\mathcal{A}) + H(\mathcal{B}) - H(\mathcal{A},\mathcal{B})$$
$$= \sum_{k,m=1}^{K_{\mathcal{A}},K_{\mathcal{B}}} \frac{n_{km}}{N} \log \frac{n_{km}N}{a_k b_m}$$

Όμοια η mutual information (MI) μεταξύ U και V υπολογίζεται από:

$$\text{MI}(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i,j) \log \left(\frac{P(i,j)}{P(i)P'(j)} \right)$$

όπου P(i,j)=|Ui∩Vj|/N είναι η πιθανότητα ένα αντικείμενο που επιλέγεται τυχαία να πέσει και στις δύο κατηγορίες Ui και Vj.

Μπορεί επίσης να εκφραστεί σε set cardinality formulation :

$$\text{MI}(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \left(\frac{N|U_i \cap V_j|}{|U_i||V_j|} \right)$$

Οι κανονικοποιημένες αμοιβαίες πληροφορίες (normalized mutual information) ορίζονται ως:

$$\text{NMI}(U,V) = \frac{\text{MI}(U,V)}{\text{mean}(H(U),H(V))}$$

Συνεπώς, η τιμή των αμοιβαίων πληροφοριών ,όπως και η κανονικοποιημένη μορφή της, δεν προσαρμόζεται τυχαία και θα τείνει να αυξάνεται καθώς ο αριθμός των διαφορετικών ετικετών ,άρα και συστάδων, αυξάνεται ανεξάρτητα από το πραγματικό ποσό των mutual informations μεταξύ των εκχωρήσεων των ετικετών.

Η αναμενόμενη τιμή (expected value) για τις αμοιβαίες πληροφορίες μπορεί να υπολογιστεί χρησιμοποιώντας την ακόλουθη

εξίσωση (Vinh et. al,2009) . Σε αυτήν την εξίσωση, ai=|Ui| (ο αριθμός των στοιχείων σε Ui) και bj=|Vj| (ο αριθμός των στοιχείων σε Vj).

$$E[\mathrm{MI}(U,V)] = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|}\sum_{n_{ij}=(a_i+b_j-N)^+}^{min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N\cdot n_{ij}}{a_i b_j}\right)\frac{a_i!\,b_j!\,(N-a_i)!\,(N-b_j)!}{N!\,n_{ij}!\,(a_i-n_{ij})!\,(b_j-n_{ij})!\,(N-a_i-b_j+n_{ij})!}$$

Χρησιμοποιώντας την αναμενόμενη τιμή (expected value) , οι προσαρμοσμένες αμοιβαίες πληροφορίες (adjusted mutual information) μπορούν στη συνέχεια να υπολογιστούν χρησιμοποιώντας των ακόλουθο τύπο :

$$\mathrm{AMI} = \frac{\mathrm{MI} - E[MI]}{\mathrm{mean}\,(H(U), H(V) - E[MI])}$$

Για κανονικοποιημένες αμοιβαίες πληροφορίες (normalized mutual information) και προσαρμοσμένες αμοιβαίες πληροφορίες (adjusted mutual information), η τιμή ομαλοποίησης (normalizing value) είναι συνήθως κάποιος *γενικευμένος* μέσος όρος των εντροπίων κάθε ομαδοποίησης. Υπάρχουν διάφορα γενικευμένα μέσα και δεν υπάρχουν αυστηροί κανόνες για την προτίμηση ενός έναντι των άλλων. Η απόφαση είναι σε μεγάλο βαθμό χωριστή βάση. Για παράδειγμα, στην ανίχνευση κοινότητας (community detection) , ο αριθμητικός μέσος όρος είναι πιο συχνός. Κάθε μέθοδος ομαλοποίησης παρέχει «ποιοτικά παρόμοιες συμπεριφορές» (Yang et al., 2016) . Αυτό ελέγχεται από την «average_method» παράμετρο. Οι Vinh et al. ονόμασαν τις παραλλαγές NMI και AMI με βάση την averaging μέθοδο (Vinh et al. 2010). Οι μέσοι όροι «sqrt» και «sum» είναι οι γεωμετρικοί και αριθμητικοί μέσοι όροι αντίστοιχα.

Πιο αναλυτικά, οι αναμενόμενες αμοιβαίες πληροφορίες - expected Mutual Information (Gates & Ahn, 2017) μεταξύ δύο τυχαίων ομαδοποιήσεων στοιχείων A και B στοιχείων N:

$$\mathrm{AMI}_{\mathrm{model}}\,(\mathcal{A},\mathcal{B}) = \frac{\mathrm{MI}\,(\mathcal{A},\mathcal{B}) - \mathbb{E}_{\mathrm{model}}\,[\mathrm{MI}\,(\mathcal{A},\mathcal{B})]}{\max\limits_{\mathrm{model}}[\mathrm{MI}\,(\mathcal{A},\mathcal{B})] - \mathbb{E}_{\mathrm{model}}\,[\mathrm{MI}\,(\mathcal{A},\mathcal{B})]}$$

Οι mutual informations μεταξύ δύο ομαδοποιήσεων έχουν μελετηθεί υπό την προϋπόθεση ότι και οι δύο ομαδοποιήσεις δημιουργήθηκαν τυχαία από το μοντέλο permutation model. Επέκταση του ορισμού των mutual informations δίνουν Eperm ακολουθεί παρακάτω:

Permutation Model (Two-sided = One-sided)

$$\mathbb{E}_{\text{perm}}[\text{MI}(\mathcal{A},\mathcal{B})] = \mathbb{E}_{\text{perm}}[H(\mathcal{A})] + \mathbb{E}_{\text{perm}}[H(\mathcal{B})] - \mathbb{E}_{\text{perm}}[H(\mathcal{A},\mathcal{B})]$$

$$\mathbb{E}_{\text{perm}}[H(\mathcal{A})] = -\sum_{i=1}^{K_A} \frac{a_i}{N} \log \frac{a_i}{N}$$

$$\mathbb{E}_{\text{perm}}[H(\mathcal{A},\mathcal{B})] = \sum_{i=1}^{K_A} \sum_{j=1}^{K_B} \sum_{n} \frac{n}{N} \log \left(\frac{n}{N}\right) \frac{\binom{b_j}{n}\binom{N-b_j}{a_i-n}}{\binom{N}{a_i}}$$

Upper Bound

$$\max_{\text{perm}}[\text{MI}(\mathcal{A},\mathcal{B})] = \text{m}in\{\log H(\mathcal{A}), H(\mathcal{B})\}\text{OR}\ \sqrt{H(\mathcal{A})H(\mathcal{B})}\ \text{OR}\ \frac{1}{2}(H(\mathcal{A}) + H(\mathcal{B}))\text{OR m}\ \{H(\mathcal{A}), H(\mathcal{B})\}$$

Όπου:

$$\mathbb{E}_{\text{perm}}[\text{MI}(\mathcal{A},\mathcal{B})] = \mathbb{E}_{\text{perm}}[H(\mathcal{A})] + \mathbb{E}_{\text{perm}}[H(\mathcal{B})] - \mathbb{E}_{\text{perm}}[H(\mathcal{A},\mathcal{B})] = H(\mathcal{A}) + H(\mathcal{B}) - \mathbb{E}_{\text{perm}}[H(\mathcal{A},\mathcal{B})]$$

προκύπτει από το γεγονός ότι όλα τα μεγέθη συστάδων (και συνεπώς η εντροπία) είναι τα ίδια για κάθε συστάδα στο permutation model. Οι προσαρμοσμένες αμοιβαίες πληροφορίες (AMI) χρησιμοποιούν το permutation model για διόρθωση του τυχαίου MI (Correction for Chance) σύμφωνα με την εξίσωση (Hubert & Arabie, 1985):

$$\frac{s - \mathbb{E}_{\text{model}}[s]}{s_{max} - \mathbb{E}_{\text{model}}[s]}$$

και επιλέγοντας ένα ανώτατο όριο max [MI] από την εξίσωση :

$$min\{H(\mathcal{A}), H(\mathcal{B})\} \leq \sqrt{H(\mathcal{A})H(\mathcal{B})} \leq \frac{H(\mathcal{A}) + H(\mathcal{B})}{2} \leq max\{H(\mathcal{A}), H(\mathcal{B})\} \leq max\{\log K_{\mathcal{A}}, \log K_{\mathcal{B}}\} \leq \log N$$

καθώς οι mutual informations μπορούν να ερμηνευθούν ως αντίστροφο μέτρο ανεξαρτησίας μεταξύ των ομαδοποιήσεων, ή ως μέτρο του όγκου των πληροφοριών που έχει κάθε ομάδα για την άλλη. Μπορεί να ποικίλλει στο εύρος $[0, \min\{H(\mathcal{A}), H(\mathcal{B})\}]$,ωστόσο για τη διευκόλυνση των συγκρίσεων, είναι επιθυμητό να το ομαλοποιηθεί στο εύρος [0; 1].

5.3 Homogeneity, completeness and V-measure

(Ομοιογένεια, πληρότητα και μέτρο - V)

Γνωρίζοντας τις αληθινές αναθέσεις της κάθε ομάδας των δειγμάτων (the ground truth class assignments), είναι δυνατό να οριστεί κάποιος διαισθητικός τρόπος μέτρησης χρησιμοποιώντας τη «μέθοδο ανάλυσης της εντροπίας υπό όρους» (conditional entropy analysis).

Για οποιαδήποτε ανάθεση συστάδας (cluster assignment) δύο επιθυμητοί στόχοι είναι οι εξής (Rosenberg & Hirschberg, 2007):

- Ομοιογένεια (homogeneity) : κάθε ομάδα περιέχει μόνο τα μέλη μίας κλάσης.
- Πληρότητα (completeness) : όλα τα μέλη μιας δεδομένης κλάσης έχουν ανατεθεί στην ίδια ομάδα.

Στην πράξη αυτοί οι δύο «στόχοι» μετατρέπονται σε «σκορ» homogeneity_score και completeness_score. Και τα δύο οριοθετούνται κάτω από το 0,0 και πάνω από το 1,0. Το υψηλότερο είναι και το καλύτερο σκορ.

Ο αρμονικός μέσος όρος τους (harmonic mean) ονομάζεται V-measure (υ) και στην πράξη υπολογίζεται από τη συνάρτηση v_measure_score η οποία συνάρτηση προκύπτει απ' τον τύπο:

$$v = \frac{(1+\beta) \times \text{ homogeneity } \times \text{ completeness}}{(\beta \times \text{ homogeneity } + \text{ completeness })}$$

Το «V» σημαίνει «validity» δηλαδή «εγκυρότητα», ένας κοινός όρος που χρησιμοποιείται για την περιγραφή μιας καλής λύσης ομαδοποίησης. Το V- measure, είναι ένα μέτρο αξιολόγησης των συστάδων με εξωτερική εντροπία και παρέχει λύση σε πολλά προβλήματα που επηρεάζουν τα προηγούμενα μέτρα αξιολόγησης συστάδων, όπως είναι:

1) η εξάρτηση από τον αλγόριθμο ομαδοποίησης ή το σύνολο των δεδομένων

2) το «πρόβλημα της αντιστοίχισης», όπου αξιολογείται η ομαδοποίηση μόνο ενός τμήματος δεδομένων και

3) η ακριβής αξιολόγηση και ο συνδυασμός δύο επιθυμητών στόχων της ομαδοποίησης, της ομοιογένειας και της πληρότητας.

Το V-measure (μέτρο V) είναι στην πραγματικότητα ισοδύναμο με τις normalized mutual informations - (NMI) που αναφέρθηκαν παραπάνω, με τη συνάρτηση συνάθροισης να είναι ο αριθμητικός μέσος - arithmetic mean, όπως αναλύεται παρακάτω (Becker , 2011) .

Τα πλεονεκτήματα[3] των Homogeneity, completeness και V-measure:

- Οριοθετημένες βαθμολογίες (Bounded scores**):** 0 είναι η χειρότερη και 1 είναι η καλύτερη βαθμολογία.

- Διαισθητική ερμηνεία (Intuitive interpretation): η ομαδοποίηση με «κακό» V-measure μπορεί να αναλυθεί ποιοτικά από την άποψη της ομοιογένειας και της πληρότητας για να καταλάβουμε καλύτερα τι «είδους» λάθη γίνονται από την ανάθεση.

- Δεν γίνεται παραδοχή σχετικά με τη δομή του cluster : μπορεί να χρησιμοποιηθεί για τη σύγκριση αλγορίθμων ομαδοποίησης όπως το k-means που υποθέτει ισοτροπικά σχήματα σταγόνων (isotropic blob shapes) με αποτελέσματα αλγορίθμων φασματικής ομαδοποίησης (spectral clustering algorithms) που μπορούν να βρουν ομάδες (clusters) με «διπλωμένα» σχήματα.

Ενώ, τα μειονεκτήματα[3] είναι:

- Οι μετρικές (metrics) που έχουν εισαχθεί στο παρελθόν δεν είναι κανονικοποιημένες όσον αφορά την τυχαία επισήμανση **(**random labeling**) :** αυτό σημαίνει ότι ανάλογα με τον αριθμό των δειγμάτων, των συστάδων και των αληθινών κατηγοριών, μια εντελώς τυχαία επισήμανση δεν θα αποδίδει πάντα τις ίδιες τιμές για ομοιογένεια, πληρότητα και ως εκ τούτου v-measure . Συγκεκριμένα, η τυχαία επισήμανση- random labeling δεν θα αποφέρει μηδενικές βαθμολογίες ειδικά όταν ο αριθμός των συστάδων είναι μεγάλος .

  Αυτό το πρόβλημα μπορεί να αγνοηθεί με ασφάλεια όταν ο αριθμός των δειγμάτων είναι μεγαλύτερος από χίλια και ο αριθμός των

συστάδων είναι μικρότερος από 10. Για μικρότερα μεγέθη δειγμάτων ή μεγαλύτερο αριθμό συστάδων, είναι ασφαλέστερο να χρησιμοποιηθεί προσαρμοσμένος δείκτης (adjusted index) όπως ο Adjusted Rand Index (ARI) .

- Αυτές οι μετρήσεις απαιτούν τη γνώση των αληθινών συστάδων, ενώ σχεδόν ποτέ δεν είναι διαθέσιμες στην πράξη ή απαιτούν (manual assignment by human annotators) χειροκίνητη ανάθεση από ανθρώπινους σχολιαστές (όπως στο εποπτευόμενο περιβάλλον μάθησης).

Αναλύωντας μαθηματικά τα Homogeneity, completeness και V-measure scores:

Υποθέτω ένα σύνολο δεδομένων που περιλαμβάνει N σημεία δεδομένων και δύο τμήματα αυτών: ένα σύνολο κλάσεων C = {ci | i = 1,. . . , n} και ένα σύνολο συστάδων K = {ki | 1,. . . ,M}. Ο A είναι ο πίνακας contingency table που παράγεται από τον αλγόριθμο ομαδοποίησης που αντιπροσωπεύει τη λύση ομαδοποίησης, έτσι ώστε A = {aij} όπου το aij είναι ο αριθμός σημείων δεδομένων που είναι μέλη της κλάσης ci και στοιχεία της συστάδας kj.

Προκειμένου να ικανοποιηθούν τα κριτήρια ομοιογένειας, μια ομαδοποίηση πρέπει να εκχωρήσει μόνο εκείνα τα δεδομένων που είναι μέλη μιας μεμονωμένης κλάσης σε μια συστάδα. Δηλαδή, η κατανομή κλάσης σε κάθε συστάδα θα πρέπει να μετατρέπεται σε μία μόνο κλάση, δηλαδή μέσω μηδενικής εντροπίας. Προσδιορίζουμε πόσο κοντά είναι μια δεδομένη ομαδοποίηση σε αυτό το «ιδανικό» εξετάζοντας την υπό όρους εντροπία της κατανομής της κλάσης των δεδομένων της προτεινόμενης ομαδοποίησης. Στην απόλυτα ομοιογενή περίπτωση, η τιμή H (C | K) είναι 0. Ωστόσο, σε μια ατελή κατάσταση, το μέγεθος αυτής της τιμής (σε bits) εξαρτάται από το μέγεθος του συνόλου των δεδομένων και την κατανομή των μεγεθών της κλάσης. Επομένως, αντί να πάρουμε την «ακατέργαστη εντροπία υπό όρους», ομαλοποιούμε αυτήν την τιμή με τη μέγιστη μείωση της εντροπίας που θα μπορούσαν να παρέχουν οι πληροφορίες ομαδοποίησης, συγκεκριμένα, H (C). (Rosenberg & Hirschberg, 2007)

Το H (C | K) είναι το μέγιστο και ισούται με το H (C), όταν η ομαδοποίηση δεν παρέχει νέες πληροφορίες δηλαδή η κατανομή τάξης σε κάθε συστάδα είναι ίση με τη συνολική κατανομή της κλάσης. Το H (C |

K) είναι 0 όταν κάθε σύμπλεγμα περιέχει μόνο μέλη μιας μεμονωμένης κλάσης, μια τέλεια ομοιογενή ομαδοποίηση. Στην περίπτωση όπου H (C) = 0, όταν υπάρχει μόνο μία κλάση, ορίζουμε την ομοιογένεια ως 1. Για μια τέλεια ομοιογενή λύση, αυτή η ομαλοποίηση, H (C | K) / H (C) ισούται με 0. Έτσι, για να τηρούμε τη σύμβαση ότι ως 1 είναι το «επιθυμητό» αποτέλεσμα και 0 το «ανεπιθύμητο» , ορίζουμε την ομοιογένεια ως εξής:

$$h = \begin{cases} 1 & \text{if } H(C,K) = 0 \\ 1 - \dfrac{H(C \mid K)}{H(C)} & \text{else} \end{cases}$$

όπου H(C|K) είναι η «υπό όρους εντροπία των κλάσεων» καθώς έχουμε τις αναθέσεις σε συστάδες και δίνεται από τον τύπο:

$$H(C \mid K) = -\sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right)$$

και H(C) είναι η εντροπία των κλάσεων και δίνεται από τον τύπο:

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right)$$

με n ο συνολικός αριθμός δειγμάτων, nc και nk ο αριθμός δειγμάτων που ανήκουν αντίστοιχα στην κλάση c και συστάδα k, και τελικά nc,k τον αριθμό των δειγμάτων από την κλάση c εκχωρήθηκε στη συστάδα k.

Η πληρότητα είναι συμμετρική ως προς την ομοιογένεια όπως προαναφέρθηκε (Rosenberg & Hirschberg, 2007). Προκειμένου λοιπόν να ικανοποιηθούν τα κριτήρια της πληρότητας, μια ομαδοποίηση πρέπει να εκχωρήσει όλα αυτά τα σημεία των δεδομένων που είναι μέλη μιας μεμονωμένης κλάσης σε μία συστάδα. Για να αξιολογήσουμε την πληρότητα, εξετάζουμε την κατανομή των αναθέσεων ομάδων σε κάθε κλάση. Σε μια απόλυτα ολοκληρωμένη λύση ομαδοποίησης, κάθε μία από αυτές τις κατανομές θα είναι πλήρως στραμμένη σε μία μόνο συστάδα. Μπορούμε να αξιολογήσουμε τον βαθμό «degree of skew» με τον υπολογισμό της υπό όρους εντροπία της προτεινόμενης κατανομής συστάδας δεδομένων της κλάσης των συνιστωσών δεδομένων, H (K | C). Στην απόλυτα ολοκληρωμένη περίπτωση είναι H (K | C) = 0. Ωστόσο, στη χειρότερη περίπτωση, κάθε κατηγορία αντιπροσωπεύεται από κάθε

συστάδα με κατανομή ίση με την κατανομή των μεγεθών της συστάδας, το H (K | C) είναι μέγιστο και ισούται με H (K). Τέλος, στην περίπτωση όπου H (K) = 0, όταν υπάρχει μία μόνο συστάδα, ορίζουμε την πληρότητα ως 1. Επομένως, αντιστοίχως με παραπάνω, ορίζουμε την πληρότητα ως:

$$c = \begin{cases} 1 & \text{if } H(K,C) = 0 \\ 1 - \dfrac{H(K \mid C)}{H(K)} & \text{else} \end{cases}$$

Η υπό όρους εντροπία των συστάδων εφόσον δίνονται οι συστάδες (conditional entropy of clusters given class) H(K|C)και η εντροπία των συστάδων (entropy of clusters) H(K) ορίζονται με συμμετρικό τρόπο:

$$H(K \mid C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{a_{ck}}{N}\log\frac{a_{ck}}{\sum_{k=1}^{|K|}a_{ck}}$$

$$H(K) = -\sum_{k=1}^{|K|}\frac{\sum_{c=1}^{|C|}a_{ck}}{n}\log\frac{\sum_{c=1}^{|C|}a_{ck}}{n}$$

Το V-measure ορίζεται όπως κ παραπάνω **ως ο** αρμονικός μέσος ομοιογένειας και πληρότητας (harmonic mean of homogeneity and completeness)**.** Δηλαδή, ορίζουμε την απόσταση από μια τέλεια ομαδοποίηση ως τον σταθμισμένο αρμονικό μέσο όρο ομοιογένειας και πληρότητας ως εξής (Rosenberg & Hirschberg, 2007):

$$v = 2 \times \frac{\text{homogeneity} \times \text{completeness}}{(\text{homogeneity} + \text{completeness})}$$

Όπως προαναφέρθηκε οι Normalized Mutual Informations - NMI είναι ισοδύναμες με τον αρμονικό μέσο ομοιογένειας και πληρότητας (harmonic mean of homogeneity and completeness) και έτσι μπορούμε να εξηγήσουμε την ποιότητα της ομαδοποίησης σε όρους κάθε «συστατικού» ξεχωριστά, καθιστώντας έτσι τα αποτελέσματα πιο ερμηνεύσιμα από ό, τι όταν χρησιμοποιώντας μόνο τη συνδυασμένη μέτρηση. Η απόδειξη που ακολουθεί περιλαμβάνει δύο μετρήσεις ποιότητας συστάδων που χρησιμοποιούνται συχνά, οι οποίες επιτρέπουν μια άμεση σύγκριση μεταξύ των αποτελεσμάτων ομαδοποίησης που αξιολογούνται σύμφωνα με αυτές τις μετρήσεις. (Becker , 2011)

Για ένα σύνολο συστάδων $C=\{c_1,\ldots,c_j\}$ και γεγονότων $E=\{e_1,\ldots,e_k\}$, η ομοιογένεια (h) ,η πληρότητα (m) και V-measure ορίζονται ως:

$$h = 1 - \frac{H(E \mid C)}{H(E)}, m = 1 - \frac{H(C \mid E)}{H(C)}, \quad V = \frac{h \cdot m}{h + m}$$

Αντικαθιστώντας τους ορισμούς των h και m έχουμε:

$$V = \frac{\left(1 - \frac{H(E \mid C)}{H(E)}\right) \cdot \left(1 - \frac{H(C \mid E)}{H(C)}\right)}{\left(1 - \frac{H(E \mid C)}{H(E)}\right) + \left(1 - \frac{H(C \mid E)}{H(C)}\right)}$$

που ισοδυναμεί με

$$V = \frac{\left(\frac{H(E) - H(E \mid C)}{H(E)}\right) \cdot \left(\frac{H(C) - H(C \mid E)}{H(C)}\right)}{\left(\frac{H(E) - H(E \mid C)}{H(E)}\right) + \left(\frac{H(C) - H(C \mid E)}{H(C)}\right)}$$

και είναι $H(E) - H(E \mid C) = H(C) - H(C \mid E)$ άρα:

$$V = \frac{\frac{(H(E) - H(E \mid C))^2}{H(E) \cdot H(C)}}{\frac{(H(E) - H(E \mid C)) \cdot (H(E) + H(C))}{H(E) \cdot H(C)}}$$

συνεπώς:

$$V = \frac{H(E) - H(E \mid C)}{H(E) + H(C)}$$

Και είναι $H(E) - H(E \mid C) = I(C, E)$ οπότε:

$$V = \frac{I(C, E)}{H(E) + H(C)} = NMI(C, E)$$

Όπου

$$I(C, E) = \sum_k \sum_j \frac{|e_k \cap c_j|}{n} \log \frac{n \cdot |e_k \cap c_j|}{|e_k| \cdot |c_j|}$$

$$H(C) = -\sum_j \frac{|c_j|}{n} \log \frac{|c_j|}{n}$$

$$H(E) = -\sum_k \frac{|e_k|}{n} \log \frac{|e_k|}{n}$$

5.4 Calinski-Harabasz Index

Εάν οι αληθινές ετικέτες δεν είναι γνωστές, ο δείκτης Calinski-Harabasz (sklearn.metrics.calinski_harabasz_score) ,γνωστός ως κριτήριο αναλογίας διακύμανσης, μπορεί να χρησιμοποιηθεί για την αξιολόγηση του αλγόριθμου ομαδοποίησης. Μια υψηλή βαθμολογία Calinski-Harabasz σχετίζεται με ένα μοντέλο με καλύτερα καθορισμένες ομάδες (clusters) ,δηλαδή καλή ποιότητα ομαδοποίησης.

Ο δείκτης ορίζεται ως λόγος μεταξύ της διασποράς εντός μίας ομάδας (within-cluster) και της διασποράς μεταξύ των ομάδων (between-cluster).Η διασπορά ορίζεται ως το άθροισμα των αποστάσεων των τετραγώνων.

Σε κανονική χρήση, ο δείκτης Calinski-Harabasz εφαρμόζεται στα αποτελέσματα μιας ανάλυσης συστάδων (cluster analysis).

Τα πλεονεκτήματα[3] του δείκτη:

- Η βαθμολογία είναι υψηλότερη όταν τα clusters είναι πυκνά και καλά διαχωρισμένα, το οποίο σχετίζεται με μια τυπική έννοια μιας συστάδας.

- Το σκορ είναι γρήγορο στον υπολογισμό του.

Το μειονέκτημα[3] του δείκτη:

Ο δείκτης Calinski-Harabasz είναι γενικά υψηλότερος για κυρτές συστάδες από άλλες έννοιες συστάδων, όπως συστάδες βάσει πυκνότητας όπως αυτές που λαμβάνονται μέσω του DBSCAN.

Αναλύωντας μαθηματικά τον Calinski-Harabasz Index:

Για ένα σύνολο δεδομένων E του μεγέθους n_E που έχει ομαδοποιηθεί σε k συστάδες, η βαθμολογία Calinski-Harabasz (s) ορίζεται ως ο λόγος του μέσου διασποράς μεταξύ συστάδων και της διασποράς εντός της συστάδας:

$$s = \frac{\mathrm{tr}\,(B_k)}{\mathrm{tr}\,(W_k)} \times \frac{n_E - k}{k - 1}$$

όπου $tr(B_k)$ και $tr(W_k)$ είναι τα ίχνη between-cluster και within-cluster των πινάκων διασποράς αντίστοιχα.

Ορίζονται από:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

με c_q το σύνολο των σημείων στη συστάδα q , c_q το κέντρο της συστάδας q, c_E το κέντρο της E, και n_q τον αριθμό των σημείων στη συστάδα q.

6. Εργαλεία ομαδοποίησης δεδομένων της αλληλουχίας RNA ενός κυττάρου

Η βάση δεδομένων, των εργαλείων ομαδοποίησης δεδομένων της αλληλουχίας RNA ενός κυττάρου scRNA-tools[4] (single-cell RNA tools), είναι μία λίστα σύγχρονων πακέτων λογισμικού για την ανάλυση των δεδομένων αλληλουχίας RNA ενός κυττάρου. Παρακάτω αναλύονται τα tools PARC , treeSne , Snn-cliq , GraphSCC , scziDesk , Giniclust3 και hypercluster για την πραγματοποίηση ομαδοποίησης (clustering) δεδομένων (datasets). Στο πρακτικό μέρος της παρούσας εργασίας χρησιμοποιούνται δεδομένα εισόδου που διατίθενται σε matlab μορφή αρχείων (.mat) τα οποία επεξεργάζονται ώστε στα ακόλουθα εργαλεία να δίνονται οι πραγματικές κλάσεις (true labels) των δεδομένων στα ορίσματα εισόδου τους. (ενότητα 7; παράρτημα 9.1)

[4] https://www.scrna-tools.org/table

6.1 PARC (phenotyping by accelerated refined community-partitioning)

PARC (phenotyping by accelerated refined community-partitioning) , είναι μια γρήγορη, αυτοματοποιημένη, συνδυαστική graph-based προσέγγιση ομαδοποίησης, η οποία ενσωματώνει την ιεραρχική δομή γραφημάτων (HNSW) και το graph-pruning , με τον νέο αλγόριθμο ανίχνευσης κοινοτήτων Leiden. Ο PARC αποτελεί έναν επεκτάσιμο αλγόριθμο ομαδοποίησης βάσει γραφημάτων για δεδομένα κυττάρων (single-cell) μεγάλης κλίμακας και συγκεκριμένα μεγέθους μεγαλύτερου του ενός εκατομμύριου κύτταρα. Χρησιμοποιώντας μεγάλη μονοκύτταρη ροή και κυτταρομετρία μάζας, RNA-sequencing και βιοφυσικά δεδομένα με βάση την απεικόνιση παρατηρείται ότι ο PARC υπερτερεί έναντι πολλών αλγορίθμων ομαδοποίησης , χωρίς να πραγματοποιείται υποδειγματοληψία κυττάρων, από την άποψη της ταχύτητας αλλά και της ικανότητας ανίχνευσης δυναμικών σπάνιων κυττάρων. Παρουσιάζεται ,λοιπόν, ένας επεκτάσιμος αλγόριθμος για την αντιμετώπιση ολοένα και μεγαλύτερης κλίμακας μονοκύτταρης ανάλυσης (large-scale single-cell analysis). (Stassen et al. , 2020)

Το PARC χρησιμοποιεί τρία μεγάλα βήματα για να επιτρέψει την κλιμάκωση και βάσει των δεδομένων την ομαδοποίηση δεδομένων ενός κυττάρου. Αρχικά το πρώτο στάδιο είναι η επιταχυνόμενη κατασκευή γραφήματος του πλησιέστερου γείτονα με ιεραρχική πλοήγηση HNSW όπου ο κάθε κόμβος είναι ένα μεμονωμένο κύτταρο συνδεδεμένο σε μια «γειτονιά» με παρόμοιά του κύτταρα. Το δεύτερο στάδιο είναι η αποκοπή των «ακραίων» δεδομένων με βάση την κατανομή των βαρών-άκρου και σε τοπικό επίπεδο δηλαδή κόμβου προς κόμβου (node-by-node), αλλά και σε γενικό επίπεδο του συνολικού δικτύου (global network). Το τρίτο και τελευταίο στάδιο είναι ο εντοπισμός συστάδας με βάση τον αλγόριθμο Leiden (Malkov & Yashunin, 2016; Traag et al., 2011). Με τον αλγόριθμο αυτό διαχειρίζονται αποτελεσματικότερα οι συστάδες ,που περιέχουν ένα δεδομένο σημείο και οι οποίες έχουν προκύψει από τη διαδικασία της αποκοπής – «κλάδεμα».

Τα στάδια αυτά είναι ενσωματωμένα με τέτοιον τρόπο που η απόδοση του PARC δεν καθορίζεται από κάθε βήμα ξεχωριστά, αλλά από την ανάδραση μεταξύ των σταδίων αυτών. Συγκεκριμένα, η διαδικασία «κλαδέματος» στο PARC μειώνει το μέγεθος του δείγματος των άκρων

και βελτιώνει την παράσταση του γραφήματος K-NN (k-nearest neighbor graph) των υποκείμενων δεδομένων, αυξάνει την ταχύτητα και την ευρωστία του εντοπισμού των ομάδων που ακολουθεί. Το γεγονός αυτό συμβάλει στην ανίχνευση διακριτών κυττάρων. Συνεπώς, το PARC επεκτείνεται σε μεγάλα δεδομένα κυτταρομετρίας ενός κυττάρου, αναγνωρίζει σπάνιους πληθυσμούς στα δεδομένα αυτά της κυτταρομετρίας, τμηματοποιεί ετερογενή δεδομένα single-cell RNA-sequencing και τέλος καταφέρνει και ομαδοποιεί 1,1 εκατομμύρια εικόνες κυττάρων χωρίς να προϋπάρχουν ετικέτες που να προσδιορίζουν τα δεδομένα αυτά. (Stassen et al. , 2020)

Ακολουθεί ο ψευδοκώδικας[5] του αλγόριθμου του single-cell RNA εργαλείου PARC με σκοπό την πλήρη κατανόηση των σταδίων που αναφέρθηκαν παραπάνω:

```
PARC (array X_data, float local_Euc, float global_Jac, integer small_cluster) {
    // κατασκευή του HNSW δείκτη
    hnsw_index <- X_data.construct_HNSW()
    // query 30-NN (nearest neighbor). Weight edges από ευκλείδειες αποστάσεις
    graph <- hnsw_index.extract_neighbors()
    // «κλάδεμα» τοπικού κόμβου
    // αφαιρούνται οι άκρες local_Euc std.dev. πάνω από το μέσο βάρος του κόμβου_i
    for each node_i in graph:
        node_i.local_prune()
    //Jaccard weight graph
    graph.jaccard_weight()
    // συλλογικό (global) «κλάδεμα» για την αφαίρεση των άκρων global_jaccard std. dev
    //μέσο egde weight σε επίπεδο γραφήματος
    graph.global_prune()
```

[5] https://github.com/ShobiStassen/PARC

//συνδυάζονται οι άκρες (egdes) αθροίζοντας τα βάρη και αφαιρώντας την κατεύθυνση

graph.simplify()

clusters <- graph.Leiden()

// διεκπεραιωτικός χειρισμός για συστάδες μικρότερες από τον πληθυσμό small_cluster

for each small_cluster in clusters:

for each cell in small_cluster:

// επανεκχώρηση σε συστάδα που περιέχει την πλειοψηφία των αρχικών 30NN κυττάρων

If cell.validNN() : do cell.reassign()

return clusters // επιστρέφονται οι συστάδες }

6.2 tree-Sne (tree of t-Distributed Stochastic Neighbor Embedding)

Το t-SNE και η ιεραρχική ομαδοποίηση είναι δημοφιλείς μέθοδοι διερευνητικής ανάλυσης δεδομένων κυρίως στον κλάδο της βιολογίας. Με βάση την πρόσφατη εξέλιξη του t-SNE και την επίτευξη «λεπτότερης» δομής, δημιουργήθηκε ο αλγόριθμος tree-SNE[6] (Robinson & Pierce-Hoffman ,2020). Ένας αλγόριθμος ιεραρχικής ομαδοποίησης και οπτικοποίησης «στοιβαγμένων» μονοδιάστατων ενσωματώσεων t-SNE.

Πιο συγκεκριμένα, το t-SNE είναι μια ευρέως χρησιμοποιούμενη μέθοδος οπτικοποίησης δεδομένων υψηλής διάστασης σε χαμηλά επίπεδα διαστάσεις (van der Maaten & Hinton 2008). Είναι κίνητρο ελαχιστοποιώντας την απόκλιση Kullback-Leibler μεταξύ των κατανομών κατά ζεύγη συγγένεια μεταξύ των παρατηρήσεων στους χώρους υψηλής διαστάσεων και χαμηλών διαστάσεων. Οι προκάτοχοι του SNE, Hinton και Roweis (2002), χρησιμοποίησαν έναν πυρήνα Gauss για να μετατρέψει τις αποστάσεις χαμηλών διαστάσεων , ενώ το t-SNE χρησιμοποιεί μία ουρά t-διανομής με έναν βαθμό ελευθερίας. Όπως σημείωσαν οι van der Maaten και Hinton (2008), το βάρος των ουρών της κατανομής t σε

[6] https://github.com/isaacrob/treesne

σύγκριση με την κατανομή Gaussian βοηθούν στο να διακρίνεται εφόσον υπάρχει το πρόβλημα της «συσσώρευσης». Δεδομένου ότι το t-SNE δίνει καλύτερα διαχωρισμένες ομάδες από το SNE, χρησιμοποιήθηκε μετασχηματισμός Fourier (FFT) προσεγγίζοντας τον αλγόριθμο T-SNE με βάση την παρεμβολή (FIt-SNE) από τους Linderman et al. (2019) για την εφαρμογή μιας γρήγορης έκδοσης του t-SNE για μικρότερους κλασματικούς βαθμούς ελευθερίας.

Συνεπώς, πρακτικά για την εγκατάσταση του tree-SNE ώστε να εφαρμοστεί στην παρούσα εργασία προϋπόθεση ήταν η εγκατάσταση του FFT-accelerated Interpolation-based t-SNE (Fit-SNE[7]) και μετά η επισύναψη (import sys; sys.path.append('/…path…/FIt-SNE/')) στο πρόγραμμα ηλεκτρονικού υπολογιστή ,Python script, πριν κληθεί η βιβλιοθήκη του εργαλείου tree-SNE (import tree_sne) ώστε να μπορεί να εφαρμοστεί στα δεδομένα εισόδου που διατίθενται .

Ο αλγόριθμος tree-SNE αποκαλύπτει ιεραρχικές δομές σε δεδομένα υψηλών διαστάσεων δημιουργώντας πολλές ενσωματώσεις t-SNE με αλλαγές στην «σύγχυση» (perplexity) και τους βαθμούς ελευθερίας της κατανομής t. Καθώς τόσο το perplexity όσο και οι βαθμοί ελευθερίας μειώνονται, τόσες περισσότερες συστάδες παράγονται στην ενσωμάτωση. Αρχικά, μια τυπική μονοδιάστατη ενσωμάτωση t-SNE με υψηλό perplexity , είναι από προεπιλογή η τετραγωνική ρίζα του αριθμού των σημείων των δεδομένων. Το υψηλό perplexity εκκίνησης αυξάνει τον αποτελεσματικό αριθμό γειτόνων που χρησιμοποιεί το t-SNE που σημαίνει ότι οι μεγαλύτερες ομάδες τείνουν να σχηματίζονται, καταγράφοντας περισσότερες καθολικές δομές στα δεδομένα. Το tree-SNE μπορεί να εμφανίσει ολόκληρο φάσμα οργάνωσης δεδομένων, από καθολικές δομές στη βάση του δέντρου και πολύ λεπτές δομές στην κορυφή του. Η προεπιλεγμένη αρχική τιμή perplexity του $\sqrt{N}$ εμπνεύστηκε από την ανάλυση της «σύγχυσης» στο t-SNE από τον Oskolkov (2019) και δίνει ευνοϊκά αποτελέσματα στην πράξη. Η ακριβής αρχική τιμή του perplexity δεν είναι σημαντική, αρκεί να είναι υψηλή, καθώς το t-SNE είναι αρκετά ανθεκτικό σε μικρές αλλαγές σε αυτή. Κατά τη δημιουργία κάθε επιπέδου, η ενσωμάτωση t-SNE αρχικοποιείται με την ενσωμάτωση από το προηγούμενο επίπεδο, αντί για ένα τυπική τυχαία εκκίνηση. Έτσι κάθε επίπεδο είναι μια βελτίωση της ομαδοποίησης που βρέθηκε στο προηγούμενο επίπεδο, με μεγαλύτερες συστάδες να σπάζουν σε μικρότερες στα επόμενα επίπεδα του δέντρου. Ως αποτέλεσμα, το

[7] https://github.com/KlugerLab/FIt-SNE

μονοπάτι μίας ομάδα παρατηρήσεων μπορεί να εντοπιστεί κάθετα μέσω της απεικόνισης δέντρου που παράγεται.

Παράλληλα με τη μέθοδο t-SNE, η ομαδοποίηση άλφα, συνιστά τη βέλτιστη ανάθεση συστάδων, χωρίς προηγούμενη γνώση του αριθμού των συστάδων, με βάση τη σταθερότητα μίας ομάδας σε πολλές κλίμακες. Το Alpha-clustering λειτουργεί αναζητώντας μια αντιστοίχιση συστάδας για τα δεδομένα που είναι στο μέγιστο αμετάβλητα σε πολλαπλάσια επίπεδα ομαδοποίησης. Καθώς το μονοδιάστατο t-SNE τείνει να δημιουργεί εύκολα διακριτές ομάδες δεδομένων, το tree-SNE αναγνωρίζει και μετράει τις συστάδες σε κάθε στρώμα του δέντρου και στη συνέχεια ορίζει την ομαδοποίηση άλφα ως ομαδοποίηση για τις συστάδες στις οποίες ο αριθμός τους δεν αλλάζει στο μεγαλύτερο εύρος τιμών.

Η αποτελεσματικότητα του tree-SNE και της άλφα συσταδοποίησης παρουσιάζεται τόσο σε εικόνες χειρόγραφων ψηφίων, όσο δεδομένων κυτταρομετρίας μάζας (CyTOF) από κύτταρα αίματος αλλά και μονοκύτταρα δεδομένα αλληλουχίας RNA (scRNA-seq) από αμφιβληστροειδή κύτταρα. Για να αποδειχθεί η εγκυρότητα της οπτικοποίησης, χρησιμοποιούμε άλφα-ομαδοποίηση για να ληφθούν μη εποπτευόμενα αποτελέσματα ομαδοποίησης ανταγωνιστικά με την τελευταία λέξη της τεχνολογίας σε πολλά άλλα σύνολα δεδομένων.

Το tree-SNE παρέχει έναν τρόπο διερεύνησης της ομαδοποίησης των δεδομένων σε πολλαπλά επίπεδα ευαισθησίας, γεγονός το οποίο δεν είναι διαθέσιμο σε πολλές άλλες τεχνικές οπτικοποίησης. Συγχρόνως, η μέθοδος alpha-clustering έχει δημιουργηθεί με βάση την tree-SNE ενσωμάτωση και παρέχει ποσοτική επικύρωση ότι το δέντρο tree-SNE αποκαλύπτει σημαντικές δομές από τα δεδομένα. Επιτυγχάνει επίσης ανταγωνιστικά ποσοστά επιτυχίας στις προσεγγίσεις ομαδοποίησης τελευταίας τεχνολογίας, χωρίς να προαπαιτείτε προσδιορισμός του αριθμού των συστάδων. Τέλος, αποδεικνύεται η βιωσιμότητα αυτής της προσέγγισης σε διαφορετικά σύνολα δεδομένων του πραγματικού κόσμου όπως αναφέρθηκαν και παραπάνω.

6.3 Snn-cliq (shared nearest neighbor clique)

Ο SNN-Cliq (shared nearest neighbor) είναι ένας νέος αλγόριθμος που ονομάστηκε έτσι λόγω της έκφρασης «κοινός πλησιέστερος γείτονας» και

ομαδοποιεί μεταγραφώματα ενός κυττάρου (single-cell transcriptomes) (Chen Xu & Zhengchang Su, 2015). Το SNN-Cliq χρησιμοποιεί την έννοια του κοινού πλησιέστερου γείτονα που δείχνει πλεονεκτήματα στο χειρισμό δεδομένων υψηλής διαστάσεως. Όταν αξιολογείται σε μια ποικιλία συνθετικών και αλλά και πραγματικών πειραματικών συνόλων δεδομένων, το SNN-Cliq ξεπερνά άλλα εξελιγμένες μεθόδους που δοκιμάστηκαν. Τα αποτελέσματα ομαδοποίησης του SNN-Cliq ξεχωρίζουν αντικατοπτρίζουν τους τύπους ή/και την προέλευση των κυττάρων με υψηλή ακρίβεια.

Με τον αλγόριθμο snn-cliq αρχικά υπολογίζεται ένας πίνακα ομοιότητας χρησιμοποιώντας την ευκλείδεια απόσταση μεταξύ σημείων ως ζεύγη δεδομένων. Για παράδειγμα, ένα σημείο είναι ένα κύτταρο και υπολογίζεται η απόσταση μεταξύ των σημείων χρησιμοποιώντας τους φορείς των επιπέδων γονιδιακής έκφρασης στα κύτταρα αυτά. Στη συνέχεια, για κάθε σημείο δεδομένων xi, παραθέτουμε τον k-πλησιέστερο γείτονα (KNN) χρησιμοποιώντας ο πίνακας ομοιότητας, με το ίδιο το xi ως την πρώτη καταχώρηση στη λίστα. Για την κατασκευή ενός γραφήματος SNN, για ένα ζευγάρι σημείων xi και xj, εκχωρούμε μία ακμή e (xi, xj) μόνο εάν xi και xj έχουν τουλάχιστον ένα κοινό γείτονα KNN. Ο Το βάρος του άκρου e (xi, xj) ορίζεται ως η διαφορά μεταξύ του k και την υψηλότερη μέση τιμή του κοινού KNN.

Συνεπώς, το γράφημα SNN καταγράφει την ομοιότητα μεταξύ δύο κόμβων όσον αφορά τη συνδεσιμότητά τους στη γειτονιά ,δηλαδή η ομοιότητα μεταξύ δύο κόμβων πρέπει να επιβεβαιωθεί από την εγγύτητά τους με τους άλλους κόμβους (κοινοί πλησιέστεροι γείτονες). Η λογική πίσω από τον αλγόριθμο SNN είναι ότι η κατάταξη των κόμβων είναι συνήθως σημαντική στις υψηλές διαστάσεις χώρου αν και η πρωταρχική ομοιότητα μπορεί να μην μας προδιαθέτει για το τί θα ακολουθήσει. Η κατάταξη των κοινών γειτόνων δύο κόμβων σε μία συστάδα αναμένεται να είναι υψηλή, οδηγώντας έτσι σε ένα πολύ «ζυγισμένο» άκρο. Αντίθετα, η κατάταξη των κοινών γειτόνων δύο κόμβων από διαφορετικές συστάδες αναμένεται να είναι χαμηλή, με αποτέλεσμα μια χαμηλή στάθμιση στο άκρο. Τα γραφήματα SNN είναι συνήθως αραιά, επιτρέποντας έτσι την απαραίτητη κλιμάκωση στα μεγάλα σύνολα δεδομένων.

Αναγνωρίζονται συστάδες στο γράφημα SNN συνδυάζοντας επαναληπτικά σημαντικά επικαλυπτόμενα υπο-γραφήματα που ξεκινούν με τις προβλεπόμενες ψευδο-συστάδες (quasi-cliques). Η μέθοδος ομαδοποίησης γραφημάτων SNN-cliq αποτελείται από δύο βασικά βήματα. Αρχικά, εξάγονται τοπικά μεγάλα quasi-cliques που σχετίζονται με κάθε κόμβο στο υπο-γράφημα που δημιουργείται από έναν κόμβο και στη συνέχεια, κατασκευάζονται συστάδες συγχωνεύοντας τις quasi-cliques και εκχωρώντας κόμβους σε μοναδικές συστάδες. Άρα, αφού βρεθούν οι quasi-cliques στο γράφημα SNN, προσδιορίζονται οι συστάδες

συγχωνεύοντας τις quasi-cliques και τέλος εκχωρούνται οι κόμβοι σε μοναδικές ομάδες.

Στην ανάλυση του μεταγραφώματος ενός κυττάρου, συχνά, ομαδοποιούνται κύτταρα με βάση τα επίπεδα έκφρασης των γονιδίων τους. Έτσι, κάθε ομάδα αντιστοιχεί σε έναν τύπο κυττάρου με συγκεκριμένες λειτουργίες. Μια τέτοια ανάλυση θα μπορούσε να βοηθήσει στον χαρακτηρισμό των κυτταρικών συνθέσεων σε ιστούς και να διακρίνει τα αναπτυξιακά τους στάδια, οδηγώντας σε καλύτερη κατανόηση της φυσιολογίας και της παθολογίας των ιστών και της αναπτυξιακής τους διαδικασίας. Μια ιδανική μέθοδος ομαδοποίησης αποτελεί εκείνη που θα αναφερόταν σε όλο το γονιδίωμα. Με αυτόν τον στόχο, παρουσιάστηκε η ομαδοποίηση με τον αλγόριθμο SNN-Cliq. Το SNN-Cliq έχει εξαιρετική απόδοση όπως προαναφέρθηκε τόσο σε συνθετικά όσο και σε πραγματικά πειραματικά σύνολα δεδομένων. Ο αλγόριθμος SNN-cliq δεν πραγματοποιεί υποθέσεις σχετικά με τη δομή των συστάδων, συνεπώς μπορεί να χειριστεί δεδομένα σε διάφορα σχήματα αλλά και πυκνότητες. Τέλος, τα σύνολα δεδομένων που προκύπτουν από την αξιολόγηση σε σχέση με την ανάλυση της αλληλουχίας RNA ενός κυττάρου (single-cell RNA sequencing), καταδεικνύουν ότι το SNN-Cliq θα μπορούσε να δημιουργήσει τις επιθυμητές απαντήσεις με υψηλή ακρίβεια, ευαισθησία και απόδοση σε σχέση με άλλους παλαιότερους αλγόριθμους που δοκιμάστηκαν. (Chen Xu & Zhengchang Su, 2015)

6.4 GraphSCC

Οι πρόσφατες εξελίξεις στις τεχνολογίες αλληλουχίας RNA single cell, scRNA-seq , παρέχουν μια μεγάλη ευκαιρία για μελέτη της γονιδιακή έκφρασης σε κυτταρική ανάλυση. Επίσης, τα δεδομένα scRNA-seq έχουν αποκαλύψει την ετερογένεια και την ποικιλομορφία των κυττάρων. Ένα κρίσιμο βήμα για τις αναλύσεις scRNA-seq (Kolodziejczyk et al.,2015) είναι να συγκεντρωθούν οι ίδιοι τύποι κυττάρων και για τον σκοπό αυτό έχουν αναπτυχθεί πολλές μέθοδοι για την ομαδοποίηση των κυττάρων. Ωστόσο, οι υπάρχουσες μέθοδοι ομαδοποίησης περιορίζονται αγνοώντας τις δομικές σχέσεις υψηλής τάξης μεταξύ των κυττάρων. Η νέα «αυτο-εποπτευόμενη» μέθοδος GraphSCC με τη βοήθεια των «Graph Convolutional Networks» είναι κατάλληλη για τη δημιουργία καλύτερων συστάδων σύμφωνα με τις ανάλογες βάσεις δεδομένων scRNA-seq. Εκτεταμένα πειράματα δείχνουν ότι το μοντέλο GraphSCC υπερτερεί των προηγμένων μεθόδων σε διάφορες αξιολογήσεις ,τόσο σε προσομοιωμένα

,όσο και σε πραγματικά σύνολα δεδομένων. Επιπλέον οπτικοποιήσεις δείχνουν ότι το GraphSCC παρέχει αναπαραστάσεις για καλύτερη και πιο «συμπαγή» μορφή εντός της ομάδας και καλό διαχωρισμό μεταξύ των συστάδων. (Yuansong et al. ,2020)

Το συνολικό προτεινόμενο δίκτυο GraphSCC αποτελείται από τρία στοιχεία: Denoising Autoencoder (DAE), Graph Convolutional Network (GCN) και Dual Self-supervised Module (DSM).

Το δίκτυο DAE χρησιμοποιείται για τη λήψη ισχυρών χαμηλών διαστάσεων αναπαράστασης που θα μπορούσαν να επηρεάσουν και τελικά να ανακατασκευάσουν τα δεδομένα εισόδου. Το GCN στοχεύει στη βελτιστοποίηση των αναπαραστάσεων με βάση πάντα τους περιορισμούς από τις δομικές πληροφορίες που δίνονται μεταξύ των κυττάρων. Το γράφημα αυτό έχει αρχικοποιηθεί μέσω του αλγόριθμου του K-πλησιέστερου γείτονα (KNN). Τέλος, το DSM στοχεύει στο διαχωρισμό των κυττάρων σύμφωνα με τις εκφρασμένες αναπαραστάσεις από τα DAE και GCN.

Πιο συγκεκριμένα, το δίκτυο DAE ασχολείται με την κωδικοποίηση του πίνακα έκφρασης ενός γονιδίου σε σχέση με ένα διάνυσμα που αναπαρίσταται με σταθερό μέγεθος. Το DAE είναι μια παραλλαγή του αυτόματου κωδικοποιητή που εισάγονται τα κατεστραμμένα δεδομένα και εξάγονται τα κατάλληλα δεδομένα. Το γράφημα GCN αρχικοποιείται από τον αλγόριθμο KNN, όπου τα παρόμοια σημεία του γραφήματος που αντιστοιχούν σε κάθε κύτταρο επιλέγονται ως γείτονες για την κατασκευή του πίνακα γειτονίας για το δίκτυο GCN. Ειδικότερα, το δίκτυο GCN κατασκευάστηκε για να εντοπίζει δομικές πληροφορίες μεταξύ κυττάρων , οι οποίες είχαν αγνοηθεί από το δίκτυο DAE. Η αυτο-εποπτευόμενη ενότητα DSM σχεδιάστηκε για να καθοδηγήσει την εκμάθηση όταν πραγματοποιείται σε χαμηλές διαστάσεις αναπαραστάσεων, από τα DAE και GCN. Με την εισαγωγή των δεδομένων από το DAE, τα κύτταρα ομαδοποιούνται σε c συστάδες που αντιστοιχούν σε κέντρα συμπλέγματος μέσω του αλγορίθμου K-μέσων (k - means). Στη συνέχεια, οι συστάδες βελτιώνονται επαναληπτικά μέχρι τη σύγκλιση, μαθαίνοντας από τις αναθέσεις υψηλής εμπιστοσύνης, χρησιμοποιώντας βοηθητικά την κατανομή στόχου (target distribution). Έτσι, πραγματοποιείται καλύτερη ομαδοποίηση των κυττάρων. (Yuansong et al. ,2020)

Συνεπώς, ενσωματώνοντας τις δομικές σχέσεις μεταξύ των κυττάρων στην scRNA-seq ομαδοποίηση (Kolodziejczyk et al., 2015) από το γραφικό νευρικό δίκτυο, δημιουργήθηκε το δομικό μοντέλο βαθιάς

ομαδοποίησης GraphSCC. Το μοντέλο αυτό αποτελείται από τα δύκτυα GCN, DAE και τα DSM modules και έτσι το GraphSCC είναι σε θέση να παράγει αποτελεσματικά τις σχέσεις μεταξύ των κυττάρων αλλά και τα χαρακτηριστικά των δεδομένων μέσω της εκμάθησης αναπαραστάσεων. Το DSM εφαρμόστηκε σε συστάδες βελτιστοποιώντας επαναληπτικά τη συνάρτηση αντικειμενικής ομαδοποίησης με μη εποπτευόμενο τρόπο. Επίσης, η απόδοση της ομαδοποίησης του GraphSCC, στη διδιάστατη οπτικοποίηση που πραγματοποιήθηκε, ξεπέρασε πολλές ανταγωνιστικές μεθόδους τόσο σε προσομοιωμένα όσο και σε πραγματικά σύνολα δεδομένων και τέλος παρείχε πιο συμπαγείς συστάδες αλλά και διαχωριστικότητα μεταξύ των ομάδων. Το GrahpSCC θεωρείται ένα «πολύτιμο εργαλείο για τη σύλληψη της κυτταρικής ετερογένειας» (Yuansong et al. ,2020).

6.5 scziDesk

Η αλληλουχία RNA ενός κυττάρου (scRNA-seq) επιτρέπει στους ερευνητές να μελετήσουν την ετερογένεια που υφίσταται σε κυτταρικό επίπεδο. Ένα κρίσιμο βήμα για την ανάλυση των δεδομένων scRNA-seq είναι η ομαδοποίηση των κυττάρων σε υπο-πληθυσμούς για τη διευκόλυνση της επόμενης ανάλυσης. Ωστόσο, το αυξανόμενο μέγεθος των δεδομένων scRNA-seq καθιστούν δύσκολη την ομαδοποίηση τόσο υψηλών διαστάσεων. Παρόλο που ορισμένοι υπάρχοντες αλγόριθμοι συνδυάζουν την μείωση των διαστάσεων των ομάδων, αγνοούν τους περιορισμούς απόστασης και συγγένειας μεταξύ παρόμοιων κυττάρων και τέλος κάνουν κάποιες πρόσθετες λανθάνουσες παραδοχές χώρου, όπως η κατανομή Gauss, τελικά, αποτυγχάνουν. Ως εκ τούτου, συνδυάζοντας την τεχνική βαθιάς μάθησης μαζί με τη χρήση ενός αυτόματου κωδικοποιητή «denoising» τελικά καταφέρνουν να χαρακτηρισθούν τα δεδομένα scRNA-seq ,ενώ προτείνεται ένας αλγόριθμος «αυτο-εκπαίδευσης» K-μέσων για τη συσσώρευση του κυτταρικού πληθυσμού στον λανθάνοντα χώρο. Η διαδικασία αυτό-εκπαίδευσης μπορεί να συγκεντρώσει αποτελεσματικά τα παρόμοια κύτταρα και να επιδιώξει περισσότερο λανθάνοντα χώρο. Τελικά, η μέθοδος «scziDesk» εκτελεί εναλλακτικά τη συμπίεση των δεδομένων, την ανακατασκευή τους αλλά και την ομαλή επαναλαμβανόμενη ομαδοποίησή τους. Τα αποτελέσματα δείχνουν εξαιρετική συμβατότητα και αντοχή τόσο σε προσομοιωμένα όσο και σε πραγματικά δεδομένα. Η μέθοδος αυτή έχει εξαιρετική επεκτασιμότητα

ανάλογα με το μέγεθος των κυττάρων σε σύνολα δεδομένων μεγάλης κλίμακας.(Liang Chen et al., 2020)

Συνοπτικά, τα στάδια του αλγόριθμου scziDesk είναι : η προ-επεξεργασία των αρχικών δεδομένων, η χρήση «Zero-inflated» αρνητικού διωνυμικού αυτόματου κωδικοποιητή και τέλος η αυτό-εκπαίδευση σε συστάδες με τον σταθμισμένο αλγόριθμο K-μέσων (k means).

Πιο αναλυτικά, για την προ-επεξεργασία των δεδομένων, αρχικά συλλέγουμε τον πίνακα δεδομένων μέτρησης scRNA-seq μέσω μίας διαδικασίας ποιοτικού ελέγχου. Απορρίπτονται τα γονίδια που έχουν τιμές έκφρασης σε λιγότερα από ένα κύτταρα και στη συνέχεια φιλτράρονται τα κύτταρα χωρίς γονιδιακή έκφραση. Θεωρώντας απαραίτητη τη σταθερότητα αριθμητικής βελτιστοποίησης του νευρωνικού δικτύου, υποχρεωτικά μετατρέπονται τα διακριτά σε συνεχή ομαλά δεδομένα. Συγκεκριμένα, ομαλοποιούμε πρώτα τον πίνακα των δεδομένων μέτρησης διαιρώντας κάθε σειρά του με το άθροισμα της σειράς και πολλαπλασιάζοντας από τη μέση τιμή των συνολικών εκφράσεων όλων των κυττάρων. Έτσι, παίρνουμε έναν φυσικό λογαριθμικό μετασχηματισμό στα δεδομένα. Τέλος, μεταμορφώνουμε τα λογαριθμικά δεδομένα σε δεδομένα βαθμολογίας-z (z-score), που σημαίνει ότι κάθε επιλεγμένο γονίδιο έχει μηδενική μέση τιμή και διακύμανση που ισούται με τη μονάδα. Αυτός ο κανονικοποιημένος πίνακας δεδομένων χρησιμοποιείται για την είσοδο νευρωνικού δικτύου και ο αρχικός πίνακας χρησιμοποιείται για τη μοντελοποίηση των δεδομένων.

Στη συνέχεια, χρησιμοποιείται ο κωδικοποιητής αυτόματης κωδικοποιήσης για να εκπαιδευτούν τα κατεστραμμένα δεδομένα scRNA-seq και να ανακτήσουν τα υποκείμενα μη παραμορφωμένα μεταγραφικά προφίλ τους. Σε αντίθεση με την παραδοσιακή ρύθμιση του αυτόματου κωδικοποιητή που έχει ένα επίπεδο εξόδου, τώρα υπάρχουν τρία επίπεδα εξόδου για να εκτιμήσουν τα τρία σύνολα παραμέτρων. Πιο συγκεκριμένα, το ποσοστό εγκατάλειψης, τη μέση τιμή και τη διασπορά του ZINB (Zero-inflated negative binomial) μοντέλου ξεχωριστά. Το μέγεθος της βιβλιοθήκης είναι κρίσιμο για τη μελέτη δεδομένων scRNA-seq γι’ αυτό και ομαλοποιούμε το άθροισμα της εξόδου στο αρχικό συνολικό μέγεθος μέτρησης για κάθε κελί για να αποφευχθούν οι υπερβολές κατά την «τοποθέτηση» των δεδομένων.

Τελικά, παρατηρούμε ότι οι παράμετροι του ZINB μοντέλου διανέμονται στις χαμηλές διαστάσεις που περιέχουν και τη λανθάνουσα μεταβλητής Z. Επομένως, αντί για εκτέλεση ομαδοποίησης στον χώρο των

δεδομένων εφαρμόζεται ο αλγόριθμος ομαδοποίησης στον «εκμαθημένο» χώρο ενσωμάτωσης. Πρέπει να υπάρχουν K συστάδες με κέντρα υ_r (όπου $1 \leq r \leq K$) στον λανθάνον χώρο. Στη συνέχεια, εφαρμόζεται άμεσα ο αλγόριθμος K-μέσων για τη συσταδοποίηση της λανθάνουσας αναπαράστασης Z.

Σε γενικές γραμμές, η επιλογή των πολύ μεταβλητών γονιδίων και η στρατηγική αυτό-εκπαίδευσης βελτιώνουν σημαντικά την απόδοση της ομαδοποίησης. Η ανάπτυξη λοιπόν μιας μεθόδου διάκρισης διαφορετικών τύπων κυττάρων από τα δεδομένα αλληλουχίας RNA είναι εξαιρετικά σημαντική. Η ομαδοποίηση βάσει του μοντέλου scziDesk για δεδομένα scRNA-seq, συνδυάζει την μοντελοποίηση της πιθανότητας των δεδομένων με τον αλγόριθμο αυτο-εκπαίδευσης K - means.

Η μέθοδος scziDesk έχει κάποια κύρια πλεονεκτήματα. Αρχικά, εφαρμόζοντας μια βαθιά τεχνική αυτόματης κωδικοποίησης με στατιστική μοντελοποίηση για την εκτίμηση της έκφρασης ενός κυττάρου, τα δεδομένα αποδίδουν ενώ παράλληλα εκτελείται μη γραμμική μείωση των διαστάσεων στα δεδομένα. Επίσης, επιλέγονται κορυφαία εξαιρετικά μεταβλητά γονίδια και χρησιμοποιείται η έκφραση μόνο αυτών των γονιδίων για να εκτελεσθεί η ομαδοποίηση των κυττάρων. Έτσι, αφαιρούνται τα περιττά γονίδια χαμηλής έκφρασης που μπορεί να επηρεάζουν αρνητικά τα αποτελέσματα της συσταδοποίησης και τελικά με τον τρόπο αυτό αυξάνεται η ταχύτητα της ομαδοποίησης δραματικά. Ακόμη, βάσει του αλγόριθμου ομαδοποίησης K-μέσων, η προσαρμοσμένη στρατηγική αυτο-εκπαίδευσης ενισχύει τη σχέση μεταξύ των γειτονικών κυττάρων, δυναμικά. Πραγματοποιήθηκαν πολλά πειράματα και μέσα από αυτά αποδεικνύεται η αξία της μεθόδου scziDesk σε πολλές πτυχές της (Liang Chen et al., 2020).

6.6 Giniclust3

Το GiniClust[4] είναι μια μέθοδος ομαδοποίησης ειδικά σχεδιασμένη για ανίχνευση σπάνιων τύπων κυττάρων. Χρησιμοποιεί τον δείκτη Gini για να εντοπίσει γονίδια που σχετίζονται με σπάνια είδη κυττάρων χωρίς προηγούμενη γνώση για εκείνα. Αυτό διαφέρει από τις παραδοσιακές μεθόδους ομαδοποίησης χρησιμοποιώντας πολύ μεταβλητά γονίδια. Χρησιμοποιείται επίσης μια προσέγγιση ομαδοποίησης που βασίζεται σε μια συστάδα. Έτσι, συνδυάζονται τα αποτελέσματα από το δείκτη Gini και

τη συγκέντρωση Fano factor και εντοπίζονται τόσο κοινοί όσο και σπάνιοι τύποι κυττάρων. Σε αυτήν τη νέα έκδοση «GiniClust3», έχει αυξηθεί σημαντικά η ταχύτητα και έχει μειωθεί η χρήση μνήμης , προκειμένου να καλυφθεί η ανάγκη για μεγάλο μέγεθος δεδομένων εισόδου (Dong & Yuan , 2020).

Πιο συγκεκριμένα, ο αλγόριθμος GiniClust3 ακολουθεί τρία βασικά στάδια. Πρώτον, πραγματοποιείται ομαδοποίηση των κυττάρων χρησιμοποιώντας λειτουργίες που βασίζονται στον δείκτη Gini. Δεύτερον, πραγματοποιείται ομαδοποίηση κυττάρων χρησιμοποιώντας λειτουργίες βάσει του Fano factor. Και τέλος, συνδυάζονται οι συστάδες από τα προηγούμενα βήματα μέσω μιας σταθμισμένης συναινετικής ομαδοποίησης που οδηγεί σε μία αποτελεσματική προσέγγιση του σωστού αποτελέσματος.

Ειδικότερα, αρχικά, πραγματοποιείται υπολογισμός και ομαλοποίηση του δείκτη Gini. Μετά την προ-επεξεργασία δεδομένων εισόδου , ο δείκτης Gini για κάθε γονίδιο υπολογίζει το διπλάσιο της περιοχής μεταξύ της διαγώνιου και της καμπύλη Lorenz (Jiang et al.,2016) . Η ταυτοποίηση των συστάδων των κυττάρων υλοποιείται με τον αλγόριθμο Leiden που είναι γνωστός για τη βελτιωμένη αριθμητική απόδοσή του. Εναλλακτικά, οι χρήστες μπορούν επίσης να επιλέξουν τον αλγόριθμο συσταδοποίησης Louvain ορίζοντας την παράμετρο της μεθόδου αναλόγως πριν την εκτέλεση του GiniClust3.

Στη συνέχεια, υλοποιείται ομαδοποίηση κυττάρων χρησιμοποιώντας λειτουργίες βάσει του Fano factor και τα πολύ μεταβλητά γονίδια αναγνωρίζονται χρησιμοποιώντας τα εργαλεία της βιβλιοθήκης Scanpy. Αυτά τα γονίδια χρησιμοποιούνται για την ταυτοποίηση κοινών συστάδων κυττάρων χρησιμοποιώντας τη μέθοδο της ανάλυσης βασικών συστατικών (PCA) ακολουθούμενη από τον Leiden ή Louvain αλγόριθμο ομαδοποίησης , χρησιμοποιώντας τις προεπιλεγμένες ρυθμίσεις στο Scanpy. Το μέγεθος της γειτονικής συστάδας το ορίζουμε μέσω του Fano factor.

Τέλος, για να βελτιωθεί η υπολογιστική αποδοτικότητα, κρατιέται ένα κύτταρο για να αντιπροσωπεύσουμε κύτταρα με τα ίδια αποτελέσματα από ομαδοποιήσεις Gini και Fano. Έτσι, η υπολογιστική απόδοση συνδέεται με τους Gini και Fano αριθμούς συστάδων και όχι με τους αριθμούς των κυττάρων. Στη συνέχεια, υπολογίζουμε τον πίνακα συναίνεσης με βάση τα κύτταρα από τις διαφορετικές ομάδες Gini και Fano. Εάν δύο κύτταρα

ομαδοποιούνται στην ίδια ομάδα, η συνδεσιμότητά τους θεωρείται 1, διαφορετικά η συνδεσιμότητά τους είναι 0.

Συνεπώς, το GiniClust3 χαρακτηρίζεται από ακρίβεια αλλά και εξαιρετική επεκτασιμότητα ως μέθοδος για την ανίχνευση σπάνιων τύπων κυττάρων από υπέρογκα σύνολα δεδομένων RNA-seq. Το GiniClust3 θα μπορούσε να εντοπίσει τόσο έναν κοινό όσο και έναν σπάνιο πληθυσμό κυττάρων και να χειριστεί ένα μεγάλο σύνολο δεδομένων που περιέχει περισσότερα από ένα εκατομμύριο κύτταρα, με αποτελεσματικό τρόπο. Αυτή η ιδιότητα είναι σημαντική για τον πλήρη προσδιορισμό των τύπων των κυττάρων σε μεγάλα σύνολα δεδομένων και μπορεί να είναι ιδιαίτερα χρήσιμη και για άλλα σύνολα δεδομένων στο μέλλον (Dong & Yuan, 2020).

6.7 Hypercluster

Η μη εποπτευόμενη ομαδοποίηση είναι ένα κοινό και εξαιρετικά χρήσιμο εργαλείο για μεγάλα βιολογικά σύνολα δεδομένων. Επίσης, η μη εποπτευόμενη ομαδοποίηση είναι μια τεχνική για την ομαδοποίηση παρόμοιων δειγμάτων σε ένα σύνολο δεδομένων. Είναι μια εξαιρετικά κοινή μέθοδος όταν αναλύονται μεγάλα δεδομένα από τον τομέα της υγείας ή κατά την εκτέλεση τεχνικών υψηλής απόδοσης όπως η διαδικασία RNA-seq ενός κυττάρου. Ωστόσο, η ομαδοποίηση απαιτεί εκ των προτέρων αλγόριθμο και επιλογή υπερπαραμέτρων, η οποία μπορεί να επηρεάσει την πρόβλεψη των τελικών συστάδες ομαδοποίησης. Επομένως, συνιστάται η λήψη μιας σειράς αποτελεσμάτων ομαδοποίησης από πολλά μοντέλα και υπερπαραμέτρους. Η διαδικασία αυτή ,λοιπόν, πιθανόν να είναι χρονοβόρα. Για τον εξορθολογισμό αυτής της διαδικασίας, παρουσιάζεται το hypercluster, ένα πακέτο για οπτικοποίηση της ομαδοποίησης με χρήση της βιβλιοθήκης sklearn σε γλώσσα προγραμματισμού python. Το πακέτο hypercluster υπολογίζει τα αποτελέσματα από πολλαπλές υπερπαραμέτρους χρησιμοποιώντας έναν ή πολλούς αλγόριθμους και, στη συνέχεια, εύκολα υπολογίζει και οπτικοποιεί τις μετρικές αξιολόγησης για κάθε αποτέλεσμα (Blumenberg & Ruggles , 2020)

Ο αλγόριθμος hypercluster ακολουθεί τα εξής στάδια. Αρχικά, οι αλγόριθμοι ομαδοποίησης και οι αντίστοιχες υπερπαραμέτρους τους καθορίζονται από το χρήστη του πακέτου. Στη συνέχεια, το hypercluster

χρησιμοποιεί αυτούς τους συνδυασμούς για να δημιουργήσει όλες τις πιθανές διαμορφώσεις και εφόσον δωθεί η εντολή από τον χρήστη τελικά επιλέγεται ένα τυχαίο υποσύνολο. Έπειτα, το Snakemake χρησιμοποιείται για τη διανομή κάθε υπολογισμού συσταδοποίησης σε ξεχωριστές εργασίες του αλγόριθμου. Παράλληλα, κάθε σύνολο ετικετών που χαρακτηρίζουν τις συστάδες αξιολογείται ως ξεχωριστή εργασία από μια λίστα μετρικών που επιλέγονται από τον χρήστη. Τέλος, όλα τα αποτελέσματα ομαδοποίησης και τα αποτελέσματα της αξιολόγησης οπτικοποιούνται σε πίνακες. Οι καλύτερες ετικέτες μπορούν επίσης να επιλεγούν από μια μετρική που καθορίζεται από το χρήστη και να παρουσιαστούν ευκολότερα.

Η εφαρμογή των συσταδοποιήσεων στον κλάδο της υγείας καθιστά απαραίτητο το γεγονός η μη εποπτευόμενη ομαδοποίηση να είναι ακριβής και να μην επηρεάζεται από την αυθαίρετη επιλογή υπερπαραμέτρων από τους χρήστες του πακέτου hypercluster. Τα εξαιρετικά υψηλής ποιότητας εργαλεία ανοιχτού κώδικα, όπως για παράδειγμα το scikit-learning, το οποίο και χρησιμοποιείται στο παρόν πακέτο, καθιστούν προσιτή τη συσταδοποίηση χωρίς την απαίτηση επίβλεψης. Με το hypercluster κάθε βήμα ανάλυσης δεδομένων απαιτεί επιλογή υπερπαραμέτρων, κατά την οποία είτε η «προκατειλημμένη» είτε η αυθαίρετη επιλογή παραμέτρων μπορεί να επηρεάσει σημαντικά τα αποτελέσματα. Στη συνέχεια, η προεπεξεργασία των δεδομένων, που περιλαμβάνει το φιλτράρισμα του συνόλου των δεδομένων εισόδου πραγματοποιεί την αφαίρεση δειγμάτων είτε χαρακτηριστικών χαμηλής ποιότητας είτε χαμηλής κάλυψης. Για παράδειγμα, αφαιρούνται γονίδια με πολύ λίγες αναγνώσεις στη διαδικασία RNA sequencing, επηρεάζοντας έτσι σημαντικά τα αποτελέσματα της ομαδοποίησης. Το εργαλείο hypercluster παρέχει μια ροή εργασιών που επιτρέπει την ολοκληρωμένη αξιολόγηση πολλαπλών υπερπαραμέτρων και αλγορίθμων συσταδοποίησης ταυτόχρονα. Εκτός από τις απλές λειτουργίες της γραμμής εντολών που διαθέτει, χρησιμοποιήσει επίσης το «SnakeMake pipeline» , ένα παράλληλο σύστημα διαχείρισης της ροής εργασιών. Το πακέτο hypercluster εκτός από την παροχή βοήθειας στους ερευνητές στην επιλογή υπερπαραμέτρων, βοηθά τους επιστήμονες που συγκρίνουν νέους αλγόριθμους ομαδοποίησης, με μετρικές αξιολόγησης και βήματα πριν ή/και μετά την επεξεργασία των δεδομένων (Liu et al.,2019). Συμπερασματικά, το hypercluster χρησιμοποιεί τη μέθοδο της μη εποπτευόμενης ομαδοποίησης για να αντλήσει βιολογικές δομές από τα δεδομένα.

7 Πειραματική διαδικασία και αποτελέσματα

Στο πρακτικό μέρος της παρούσας πτυχιακής εργασίας εγκαταστάθηκαν, τροποποιήθηκαν προγραμματιστικά (ενότητα 9.3) και εν τέλει χρησιμοποιήθηκαν τα πιο σύγχρονα single-cell RNA sequencing εργαλεία ομαδοποίησης (scRNA tools) των οποίων η μεθοδολογία που ακολουθούν αναλύθηκε παραπάνω στην ενότητα 6. Οι αλγόριθμοι αυτοί εφαρμόστηκαν σε πολλαπλά δεδομένα εισόδου (datasets) matlab μορφής αρχείων (.mat files) και στη συνέχεια υλοποιήθηκε ένα πακέτο λογισμικών στο οποίο πιο αναλυτικά περιλαμβάνονται οι αλγόριθμοι της ενότητας 9.1 αρχικά για την αξιολόγηση των μεθόδων ομαδοποίησης των παραπάνω εργαλείων με τη βοήθεια των μετρικών αξιολόγησης, οι οποίες αναλύθηκαν στην ενότητα 5. Έπειτα, τα αποτελέσματα οπτικοποιήθηκαν με τη δημιουργία ανάλογων «πινάκων συγκρίσεων» και «διαγραμμάτων συγκρίσεων» (bar-plots) που πραγματοποιήθηκε αναπτύσσοντας τα ακόλουθα προγράμματα ηλεκτρονικού υπολογιστή σε γλώσσα προγραμματισμού Python και χρησιμοποιώντας το περιβάλλον Eclipse[8] . Για την λειτουργία των tools εγκαταστάθηκαν οι απαιτούμενες βιβλιοθήκες (python libraries) από τον διερμηνέα γραμμής εντολών Anaconda prompt [9]. Η τεκμηρίωση των προγραμμάτων του πακέτου που δημιουργήθηκε επισυνάπτετε στην ενότητα 9.1 σε μορφή εγχειριδίου και αναλυτικότερα ολόκληρος ο κώδικας που υλοποιήθηκε στην ενότητα 9.2.

7.1 Υλοποίηση και αξιολόγηση αλγορίθμων ομαδοποίησης σε single-cell δεδομένα

Σκοπός της συγκεκριμένης εργασίας, είναι η σύγκριση των πιο σύγχρονων αλγορίθμων ομαδοποίησης single-cell δεδομένων γονιδιακής έκφρασης . Μέσα από αυτή τη σύγκριση, ήταν επιθυμητό να δημιουργηθεί ένα χρηστικό πακέτο λογισμικού το οποίο θα ανιχνεύει το πιο αποδοτικό εργαλείο ομαδοποίησης σύμφωνα με τις προαναφερθείσες μετρικές αξιολόγησης. Χρησιμοποιήθηκαν λοιπόν 4 διακριτά σύνολα δεδομένων, τα οποία είναι διαθέσιμα μέσω της βάσης δεδομένων Gene Expression Omnibus (GEO) του NCBI (National Center for Biotechnology

[8] https://www.eclipse.org/
[9] https://www.anaconda.com/

Information) [10]. Τα δεδομένα αυτά, προέχονται και στις 4 περιπτώσεις, από πειράματα που έγιναν για την ανάλυση ενός κυττάρου (single cell resolution). Πιο συγκεκριμένα τα δεδομένα που χρησιμοποιήθηκαν ήταν τα GSE103334,GSE52583,GSE59739 που αντιστοιχούν στον οργανισμό Mus musculus και το GSE86469 που αντιστοιχεί στον οργανισμό Homo sapiens.

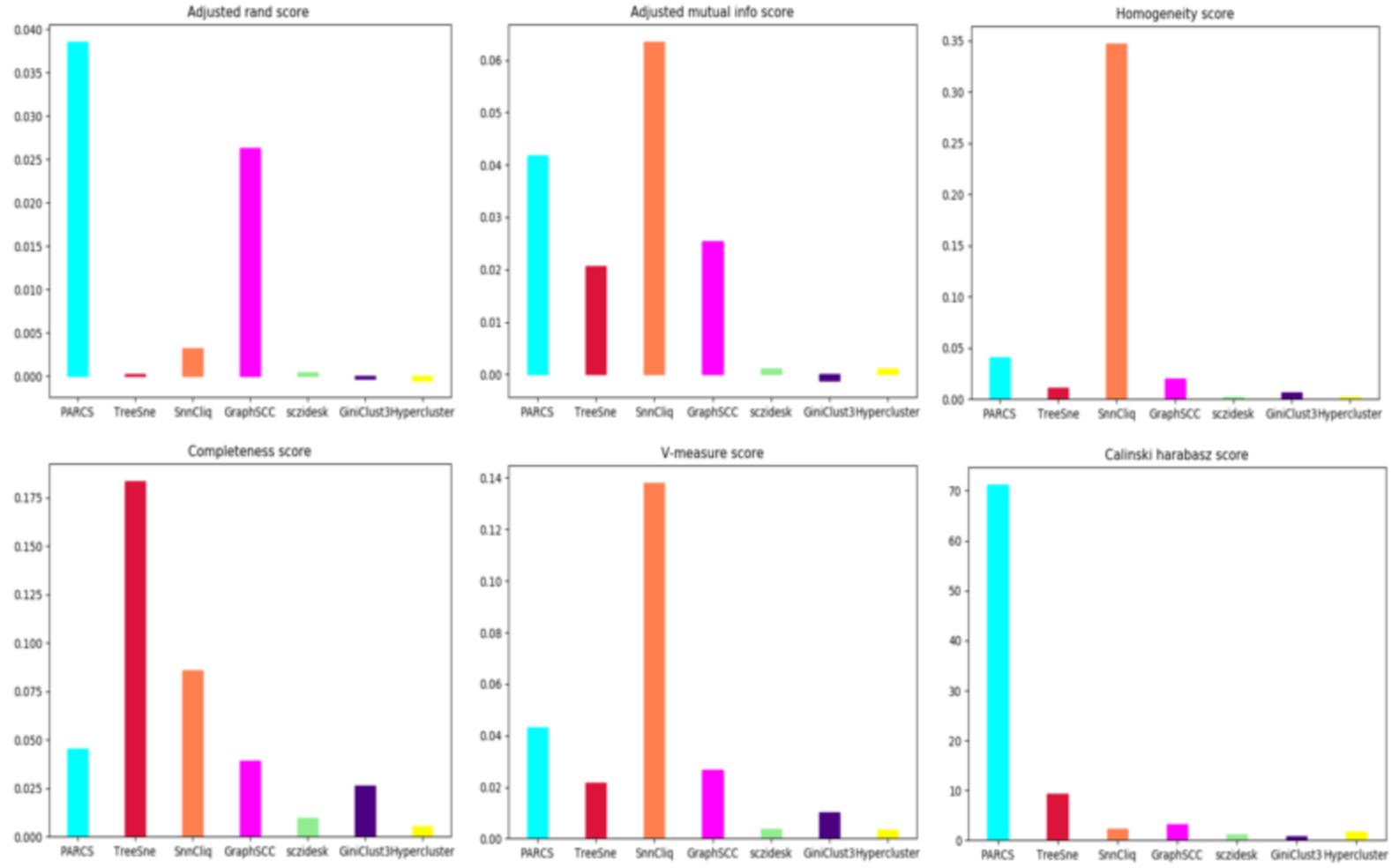
Τα δεδομένα είσοδου της πειραματικής διαδικασίας που ακολουθήθηκε παρέχονται σε matlab μορφή αρχείων τα οποία περιέχουν δύο συσχετισμένους εννοιολογικά πίνακες. Διαβάζοντας τα δεδομένα μέσω της πλατφόρμας Matlab [11] παρατηρούμε αρχικά πως ο ένας πίνακας ονομάζεται «data» και περιέχει τα δεδομένα της κυτταρικής ανάλυσης οργανωμένη στη μορφή ενός διδιάστατου πίνακα gene x cell (γονίδια x κύτταρα). Παράλληλα ο δεύτερος πίνακας που διατίθεται ονομάζεται «class» και αποτελεί ένα διάνυσμα στήλης όπου περιέχει τις κλάσεις ,δηλαδή τις πραγματικές συστάδες, στις οποίες ανήκουν τα δεδομένα του «data». Αρχικά λοιπόν η πρώτη υλοποίηση του πακέτου λογισμικών που δημιουργήθηκε είναι ο διαχωρισμός των «data» από τα «class» δεδομένα και η εισαγωγή τους σε comma-separated value (.csv files) αρχεία προς χρήση μετέπειτα.

Στη συνέχεια, της πειραματικής διαδικασίας, εφόσον γνωρίζουμε τις πραγματικές κλάσεις (ομάδες) στις οποίες ανήκουν τα δεδομένα εισόδου που χρησιμοποιήθηκαν είναι δυνατή η αξιολόγηση των αποτελεσμάτων των ομαδοποιήσεων που πραγματοποιούν τα scRNA εργαλεία PARCS,Tree-SNE,snn-Cliq,GraphSCC,sczidesk,GiniClust3 και hypercluster αντίστοιχα. Οι αλγόριθμοι των εργαλείων αυτών παράγουν τις προβλεπόμενες συστάδες στις οποίες αντιστοιχούν τα γονιδιακά δεδομένα και έτσι μέσω του λογισμικού που υλοποιήθηκε οι κλάσεις αυτές αποθηκεύονται και συγκρίνονται με τις πραγματικές. Χρησιμοποιώντας τις απαραίτητες μεθόδους της δωρεάν βιβλιοθήκης μηχανικής εκμάθησης λογισμικού Scikit-learn [12] για τη γλώσσα προγραμματισμού Python πραγματοποιήθηκε η αξιολόγηση των αλγόριθμων ομαδοποίησης των παραπάνω εργαλείων με τη βοήθεια των μετρικών αξιολόγησης adjusted rand score , mutual information based score, homogeneity, completeness, v-measure και Calinski-Harabasz index. Τέλος, με το παρόν λογισμικό οπτικοποιούνται τ’αποτελέσματα που προκύπτουν.

[10] https://www.ncbi.nlm.nih.gov/geo/
[11] https://www.mathworks.com/products/matlab.html
[12] https://scikit-learn.org/stable/index.html

7.2 Οπτικοποίηση αποτελεσμάτων σε bar-plots & tables

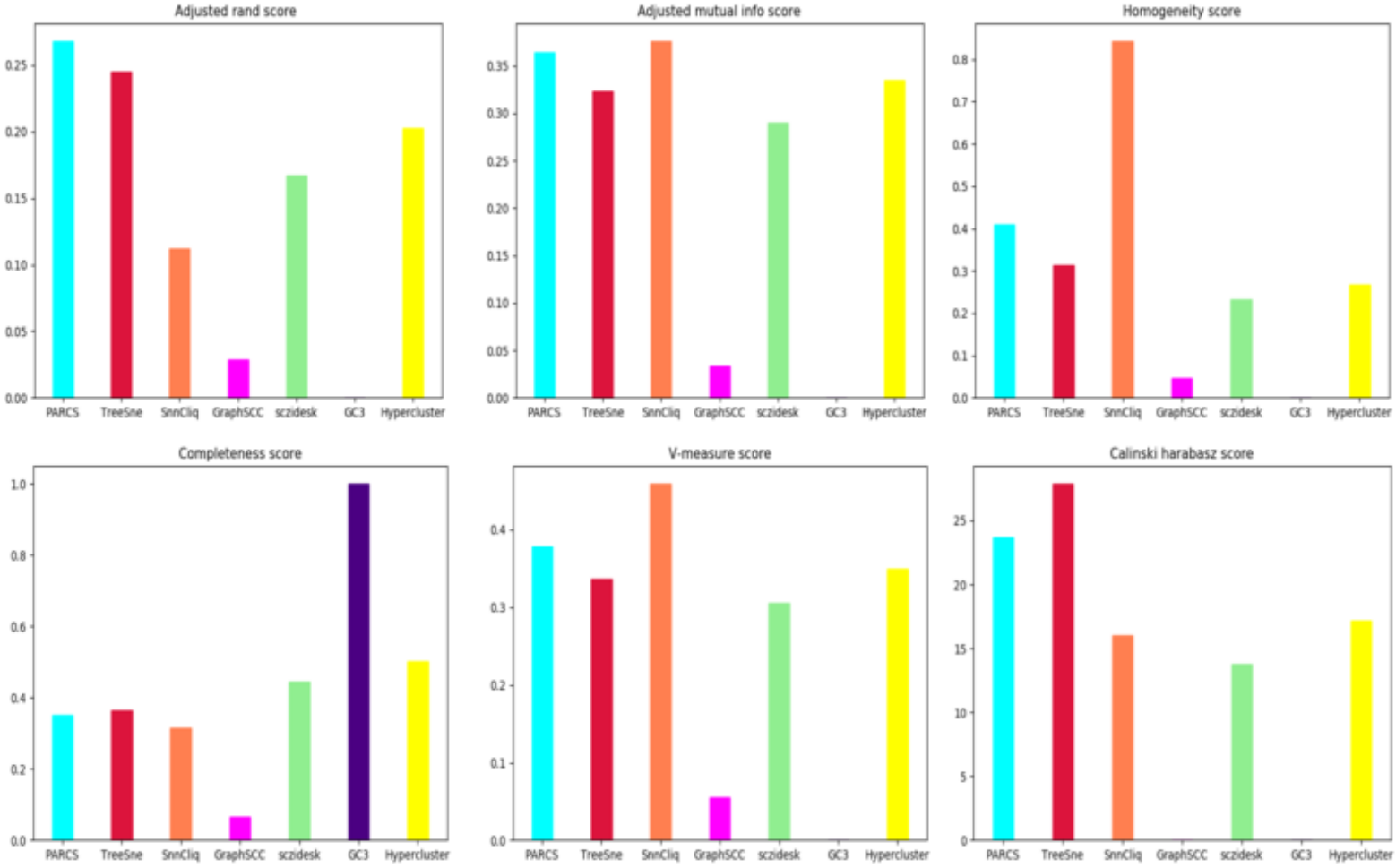
Τα αποτελέσματα της αξιολόγησης των αλγόριθμων ομαδοποίησης που παράγονται για την παρούσα πειραματική διαδικασία τελικά οπτικοποιούνται τόσο σε μορφή διαγραμμάτων όσο και σε μορφή πινάκων. Παρακάτω παρουσιάζονται τα αποτελέσματα ,του πακέτου που πραγματοποιήθηκε, σε μορφή αυτόματα παραγόμενων διαγραμμάτων τα οποία δημιουργούνται με τη βοήθεια της Matplotlib [13] βιβλιοθήκης σχεδίασης.



Εικόνα 7.1: *Διαγράμματα για το 1ο dataset GSE103334.mat*

[13] https://matplotlib.org/

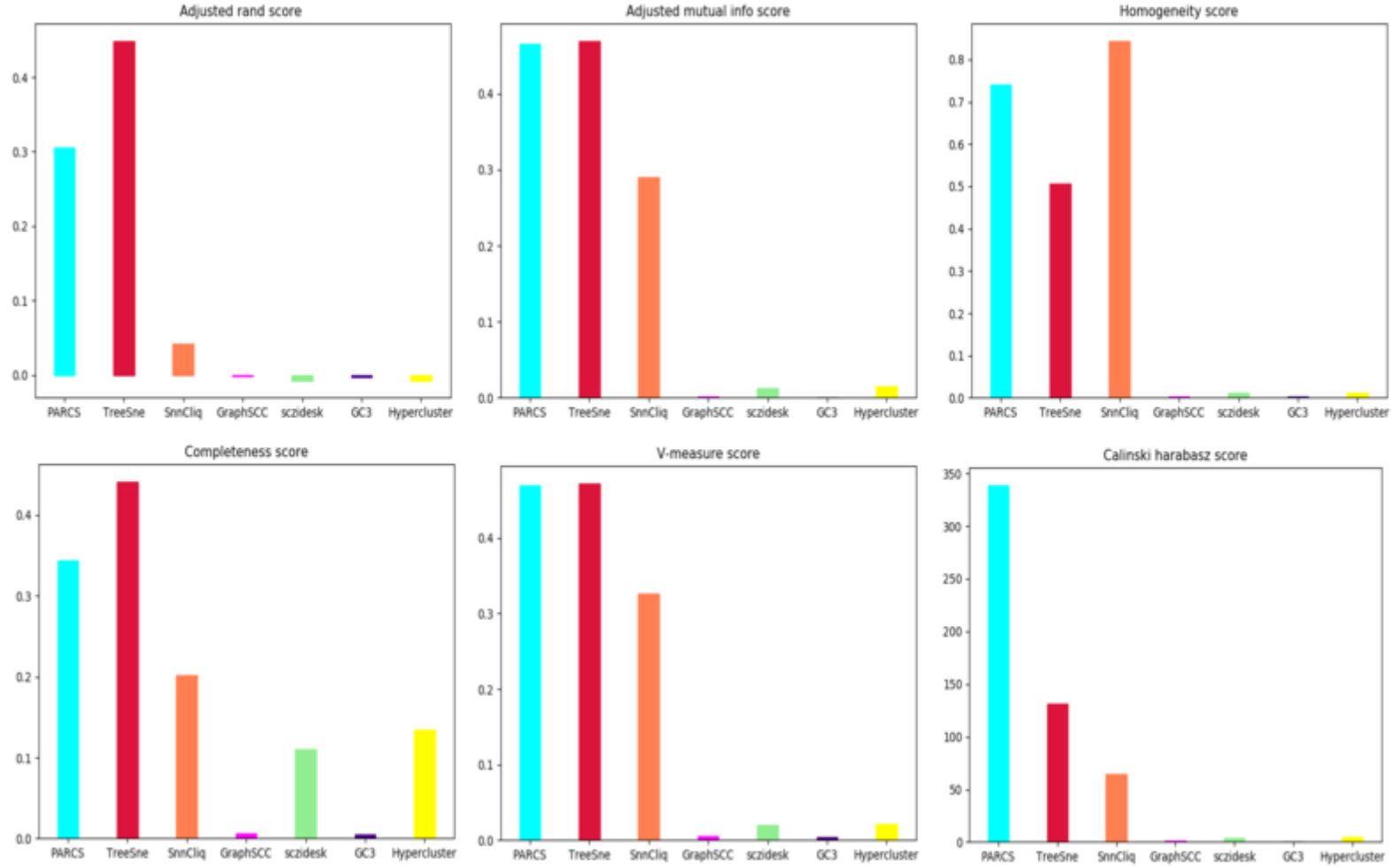
Στο πρώτο σύνολο δεδομένων εισόδου που χρησιμοποιήθηκε παρατηρούμε ότι καλύτερη απόδοση στους 3 (adjusted mutual info score,homogeneity score,v-measure score) από τους 6 δείκτες αξιολόγησης είχε το εργαλείο SnnCliq , στους 2 (adjusted rand score, calinski harabasz score) από τους 6 υπερίσχυσε ο αλγόριθμος PARCS και σε 1 (completeness score) από τις 6 μετρικές το TreeSne .



***Εικόνα 7.2**: Διαγράμματα για το 2ο dataset GSE52583.mat*

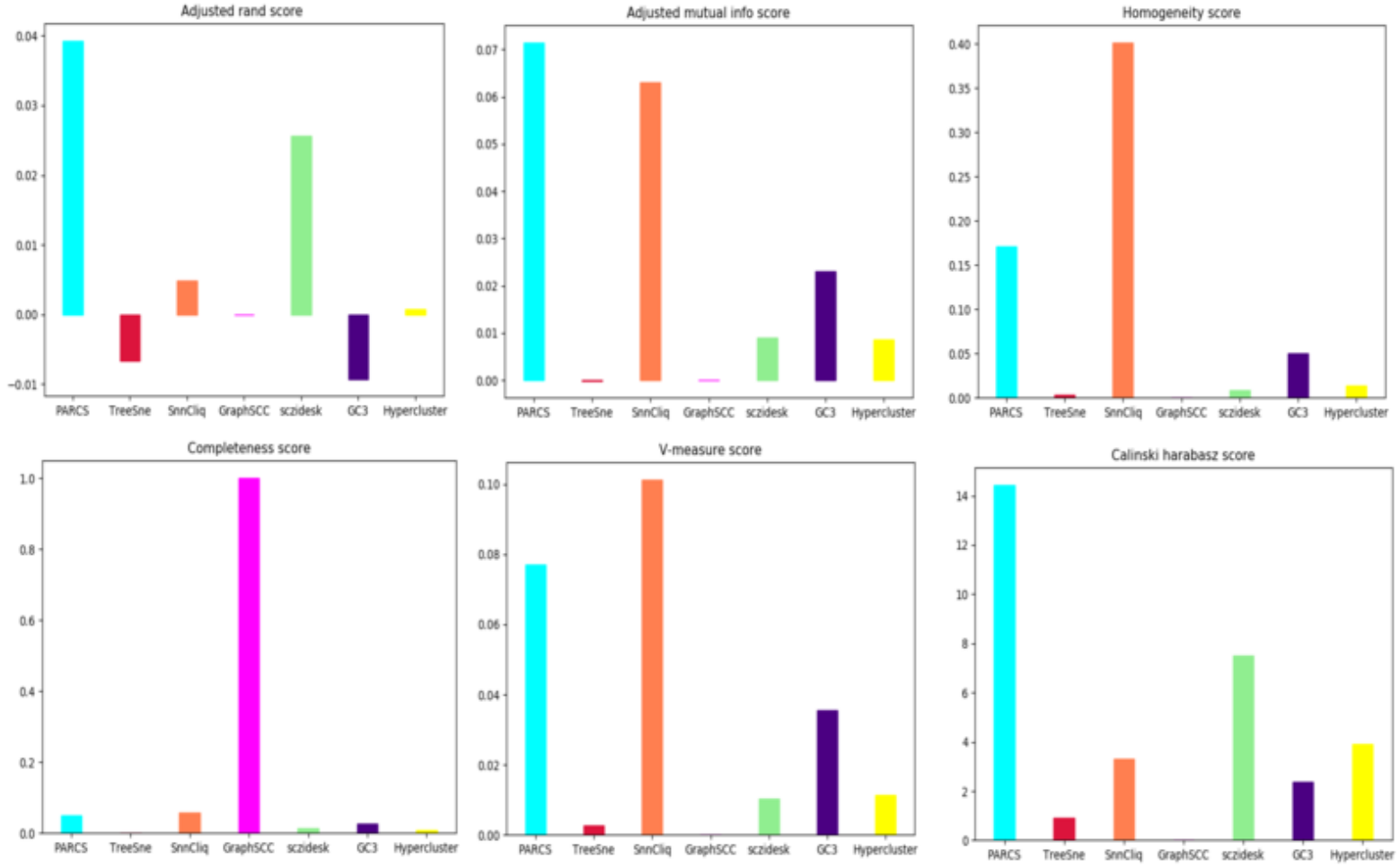
Στο δεύτερο σύνολο δεδομένων εισόδου που χρησιμοποιήθηκε παρατηρούμε ότι καλύτερη απόδοση στους 3 (adjusted mutual info score,homogeneity score,v-measure score) από τους 6 δείκτες αξιολόγησης είχε το εργαλείο SnnCliq. Σύμφωνα με τη μετρική adjusted rand score καλύτερη επίδοση είχε ο αλγόριθμος PARCS, με τη μετρική

completeness score υπερισχύει ο GiniClust3 και με το calinski harabasz score ο αλγόριθμος TreeSne.



***Εικόνα 7.3**: Διαγράμματα για το 3ο dataset GSE59739.mat*

Στο τρίτο σύνολο δεδομένων εισόδου που χρησιμοποιήθηκε παρατηρούμε ότι καλύτερη απόδοση στους δείκτες adjusted rand score και completeness score είχε το λογισμικό TreeSne ενώ στις μετρικές adjusted mutual info score και v-measure score το TreeSne και το PARCS σχεδόν ισοβαθμούν. Με το calinski harabasz score ο αλγόριθμος PARCS επικρατεί ενώ με το homogeneity score νικά το SNNCliq.



***Εικόνα 7.4**: Διαγράμματα για το 4ο dataset GSE86469b.mat*

Στο τέταρτο σύνολο δεδομένων εισόδου που χρησιμοποιήθηκε παρατηρούμε ότι καλύτερη απόδοση στα v-measure score και homogeneity score είχε το SNNCliq. Στα 3 από τα 6 σκορ συγκεκριμένα στα adjusted mutual info score, adjusted rand score και calinski harabasz score υπερισχύει ο PARCS. Τέλος με βάση τη μετρική completeness score νικά το λογισμικό GraphSCC.

Παρακάτω παρουσιάζονται τα αποτελέσματα ,του πακέτου που δημιουργήθηκε, σε μορφή αυτόματα παραγόμενων πινάκων συγκρίσεων των tools ανά δύο μεταξύ τους στη μορφή που παρουσιάζονται παρακάτω. Για τον σκοπό αυτό χρησιμοποιήθηκε το πακέτο ανοιχτού λογισμικού TableIt [14].

	PARCS	TreeSne	SnnCliq	GraphSCC	sczidesk	GiniClust3	Hypercluster
PARCS	-	3	0	3	3	3	3
TreeSne	1	-	0	0	3	3	3
SnnCliq	4	4	-	3	3	3	3
GraphSCC	1	4	1	-	3	3	3
sczidesk	1	1	1	1	-	3	0
GiniClust3	1	1	1	1	1	-	0
Hypercluster	1	1	1	1	4	4	-

Εικόνα 7.5: *πίνακας για σύγκριση των tools με βάση τη μετρική adjusted mutual info score*

	PARCS	TreeSne	SnnCliq	GraphSCC	sczidesk	GiniClust3	Hypercluster
PARCS	-	3	3	3	3	3	3
TreeSne	1	-	0	0	0	3	3
SnnCliq	1	4	-	0	3	3	3
GraphSCC	1	4	4	-	3	3	3
sczidesk	1	4	1	1	-	3	3
GiniClust3	1	1	1	1	1	-	3
Hypercluster	1	1	1	1	1	1	-

Εικόνα 7.6: *πίνακας για σύγκριση των tools με βάση τη μετρική adjusted rand score*

	PARCS	TreeSne	SnnCliq	GraphSCC	sczidesk	GiniClust3	Hypercluster
PARCS	-	3	3	3	3	3	3
TreeSne	1	-	3	3	3	3	3
SnnCliq	1	1	-	0	3	3	3
GraphSCC	1	1	4	-	3	3	3
sczidesk	1	1	1	1	-	3	0
GiniClust3	1	1	1	1	1	-	0
Hypercluster	1	1	1	1	4	4	-

Εικόνα 7.7: *πίνακας για σύγκριση των tools με βάση τη μετρική calinski harabasz score*

[14] https://github.com/SuperMaZingCoder/TableItS

	PARCS	TreeSne	SnnCliq	GraphSCC	sczidesk	GiniClust3	Hypercluster
PARCS	-	0	0	3	3	3	3
TreeSne	4	-	3	3	3	3	3
SnnCliq	4	1	-	3	3	3	3
GraphSCC	1	1	1	-	3	3	3
sczidesk	1	1	1	1	-	0	3
GiniClust3	1	1	1	1	4	-	3
Hypercluster	1	1	1	1	1	1	-

Εικόνα 7.8: *πίνακας για σύγκριση των tools με βάση τη μετρική completeness score*

	PARCS	TreeSne	SnnCliq	GraphSCC	sczidesk	GiniClust3	Hypercluster
PARCS	-	3	0	3	3	3	3
TreeSne	1	-	0	0	3	3	3
SnnCliq	4	4	-	3	3	3	3
GraphSCC	1	4	1	-	3	3	3
sczidesk	1	1	1	1	-	0	0
GiniClust3	1	1	1	1	4	-	3
Hypercluster	1	1	1	1	4	1	-

Εικόνα 7.9: *πίνακας για σύγκριση των tools με βάση τη μετρική homogeneity score*

	PARCS	TreeSne	SnnCliq	GraphSCC	sczidesk	GiniClust3	Hypercluster
PARCS	-	3	0	3	3	3	3
TreeSne	1	-	0	0	3	3	3
SnnCliq	4	4	-	3	3	3	3
GraphSCC	1	4	1	-	3	3	3
sczidesk	1	1	1	1	-	0	3
GiniClust3	1	1	1	1	4	-	3
Hypercluster	1	1	1	1	1	1	-

Εικόνα 7.10: *πίνακας για σύγκριση των tools με βάση τη μετρική V-measure score*

Αναλυτικότερα, σύμφωνα με τη μετρική αξιολόγησης adjusted mutual info score παρατηρούμε ότι ο αλγόριθμος PARCS ήταν καλύτερος σε 3 από τα 4 datasets σε σχέση με τον TreeSne, 0/4 σε σχέση με τον SnnCliq, 3/4 σε σχέση με τους GraphSCC,sczidesk,GiniClust3 και Hypercluster αντίστοιχα. Με ανάλογο τρόπο βλέπουμε τις νίκες των tools που βρίσκονται στην πρώτη στήλη του πίνακα σε σχέση με τα tools της πρώτης γραμμής του πίνακα. Συνεπώς με τον τρόπο αυτό μπορούμε εύκολα να συγκρίνουμε ένα επιθυμητό λογισμικό σε σχέση με ένα άλλο ώστε να επιλεχθεί εκείνο με την καλύτερη απόδοση σε σχέση με τις δημοφιλέστερες μετρικές αξιολόγησης που παρατίθενται.

8. Συμπεράσματα

8.1 Γενικά συμπεράσματα

Συνοψίζοντας, οι σύγχρονες ερευνητικές διαδικασίες πολλών επιστημονικών κλάδων με τις διαρκώς αυξανόμενες απαιτήσεις τους βρίσκουν απαραίτητη την εκμετάλλευση των δεδομένων μεγάλου όγκου τα οποία παρεμποδίζουν την εξαγωγή σαφώς ορισμένων και πιστοποιημένα ορθών αποτελεσμάτων όλο και περισσότερο. Έτσι, η επιστήμη της μηχανικής μάθησης έρχεται διαρκώς στο προσκήνιο των εξελίξεων καθώς λειτουργεί βοηθητικά στην προσπάθεια αυτή. Οι νέες ανακαλύψεις και η διαρκής πρόοδος στη δημιουργία μεθόδων αναγνώρισης προτύπων, βοηθούν στη δημιουργία μοντέλων πρόβλεψης αλλά και λήψης αναγκαίων αποφάσεων σε μικρό χρόνο και με το δυνατόν μικρότερο κόστος. Για να πραγματοποιηθούν τα παραπάνω αναγκαία είναι αρχικά η εξόρυξη των δεδομένων αυτών και στη συνέχεια η ομαδοποίησή τους **.** Αναλύονται τελικά όλα αυτά τα μεγάλου όγκου βιολογικά και όχι μόνο δεδομένα δίνοντας λύσεις σε προβλήματα πολλών κλάδων. (Baxevanis & Ouellette, 2016 ; Phil , 2013)

Με το RNA sequencing συγκεκριμένα αναλύονται βιολογικά δεδομένα με αποτέλεσμα την αδιαμφισβήτητη χρησιμότητα της τεχνικής αυτής για την καλύτερη εξερεύνηση του γονιδιώματος των έμβιων όντων. (Evans et al. , 2018) Το εγγύς μέλλον της εξατομικευμένης ιατρικής βασίζεται στους ερευνητές που εντοπίζουν ισχυρούς μη επιτηρούμενους τύπους ασθενειών βασιζόμενοι στην ομαδοποίηση. Ο καθορισμός των μοριακά παρόμοιων δειγμάτων σε ομάδες ,τα οποία δείγματα για παράδειγμα έχουν ληφθεί από έναν ασθενή, είναι η λύση στην εξατομικευμένη ιατρική πρόγνωση, τη διάγνωση αλλά και τη δημιουργία νέων στρατηγικών θεραπείας. Έτσι, η μη εποπτευόμενη ομαδοποίηση καθίσταται ένα σημαντικό εργαλείο για τους ερευνητές που προωθούν την εξατομικευμένη ιατρική (Liu et al.,2019). Συνεπώς, μέσω της διευκόλυνσης των επιστημών, πραγματοποιούνται καθημερινά ,αλλά και θα υλοποιούνται μελλοντικά καινούργιες ανακαλύψεις, βρίσκοντας λύσεις σε προβλήματα που θα διευκολύνουν την καθημερινότητα των ανθρώπων, παρέχοντας μια καλύτερη ποιότητα ζωής και το σημαντικότερο λειτουργώντας ως αρωγός στη μακροβιότητά του. Το γεγονός αυτό το

αποδεικνύουν οι ως τώρα εφαρμογές της μηχανικής μάθησης και της εξόρυξης δεδομένων στον κλάδο της υγείας.

8.2 Επόμενα βήματα

Έπειτα από μελέτη ενός μεγάλου μέρους της βιβλιογραφίας που αναφέρεται στην ομαδοποίηση των υπέρογκων δεδομένων ,αλλά και την αναγκαιότητα δημιουργίας χρηστικών εργαλείων που να εξυπηρετούν τις αυξανόμενες απαιτήσεις των επιστημονικών κλάδων, ακολουθούν ορισμένα προσωπικά συμπεράσματα αλλά και ιδέες για μελλοντικές επεκτάσεις του θέματος που πραγματεύεται η προγραμματιστική μελέτη της παρούσας εργασίας.

Η εφαρμογή των εργαλείων ομαδοποίησης κατά τη διάρκεια της διεκπεραίωσης της συγκεκριμένης εργασίας έδειξε ορισμένα αδύναμα σημεία αυτών των εργαλείων. Η ανάγκη βελτιστοποίησης των υπαρχόντων αλλά και δημιουργίας νέων αποδοτικότερων εργαλείων ομαδοποίησης και γενικότερα ανάλυσης μεγάλου όγκου δεδομένων είναι γεγονός. Παράλληλα, παρατηρώντας τις δυσκολίες αποπεράτωσης της ερευνητικής εργασίας υπήρξαν κολλήματα στην εγκατάσταση αλλά και την εκτέλεση των προγραμμάτων των scRNA εργαλείων και χρειάστηκαν ορισμένες εξειδικευμένες προγραμματιστικές λύσεις για την επίλυσή τους αλλά και την προσαρμογή τους στο υπολογιστικό σύστημα που εγκαταστάθηκαν ,όμως και στα διαφορετικά δεδομένα εισόδου που χρησιμοποιήθηκαν. Επίσης, καθότι τα δεδομένα εισόδου είναι εξ ορισμού υπέρογκα τα προγράμματα ομαδοποίησης χρειάστηκαν αρκετό χρόνο ώστε να εκτελεσθούν. Συνεπώς, με βάση τα σύγχρονα εργαλεία ομαδοποίησης που χρησιμοποιήθηκαν κατανοούμε ότι επιδέχονται βελτιστοποίηση. Αρχικά, η προσβασιμότητα από μη εξειδικευμένους χρήστες αλλά και η διαθεσιμότητα σε διαφορετικά υπολογιστικά συστήματα είναι ορισμένοι τομείς που μπορούν να βελτιωθούν. Ακόμη, ο χρόνος ανταπόκρισης των προγραμμάτων μπορεί να μεγιστοποιηθεί με τη χρήση υπολογιστικού νέφους (Cloud computing) καθώς διατίθενται υπολογιστικοί πόροι μέσω διαδικτύου, από κεντρικά συστήματα που βρίσκονται απομακρυσμένα από τον τελικό χρήστη, και εξυπηρετούν , αυτοματοποιώντας διαδικασίες και παρέχοντας την απαραίτητη ευελιξία. Ο τρόπος αξιολόγησης των αποτελεσμάτων των ομαδοποιήσεων με τις πιο δημοφιλής μετρικές μέσω των python scripts της συγκεκριμένης πτυχιακής

εργασίας δείχνει αναλυτικά την απόδοση των scRNA εργαλείων που χρησιμοποιήθηκαν. Με την πιστοποίηση λοιπόν της ορθής ομαδοποίησης των δεδομένων εισόδου ,μέσω των κατάλληλων μετρικών αξιολόγησης, επιτυγχάνεται τόσο η αναβάθμιση των παλιών όσο και η ανακάλυψη νέων αποδοτικότερων scRNA εργαλείων.

Σχετικά με το πακέτο λογισμικών που υλοποιήθηκε υπάρχουν ορισμένες επεκτάσεις που θα μπορούσαν μελλοντικά να πραγματοποιηθούν για τη βελτίωσή του. Αρχικά, θα μπορούσαν να προστεθούν ακόμη περισσότερες μετρικές αξιολόγησης και άλλων εργαλείων ανάλυσης δεδομένων μεγάλου όγκου. Παράλληλα, μπορούν να αυτοματοποιηθούν και άλλες διαδικασίες όπως για παράδειγμα η μεταφορά ορισμένων παραγόμενων αρχείων τα οποία την συγκεκριμένη χρονική στιγμή πρέπει να μεταφερθούν από τον χρήστη στον σωστό φάκελο και ο ίδιος να παρέχει το όνομά τους προς χρήση από τα υποπρογράμματα για να μετατραπούν στην κατάλληλη μορφή ώστε να αξιοποιηθούν για να προκύψουν τα τελικά αποτελέσματα. Συνεπώς, μελλοντικά θα μπορούσε να δημιουργηθεί η κατάλληλη διεπαφή (interface) ολοκληρωμένης εφαρμογής για μία εύκολη εμπειρία πλοήγησης και διαχείρισης από χρήστες εξειδικευμένους και μη, πολλαπλών επιστημονικών κλάδων, οι οποίοι χρειάζονται το παρόν πακέτο λογισμικών.

9. Παράρτημα

9.1 Εγχειρίδιο

9.1.1 Τεκμηρίωση των python scripts

Η τρέχουσα ενότητα χρησιμεύει ως οδηγός και τεκμηρίωση για τους κώδικες που παρέχονται παρακάτω (στην ενότητα «παράρτημα») μαζί με την παρούσα πτυχιακή εργασία για να προκύψουν τα συμπεράσματα του πρακτικού μέρους της.

Παρακάτω για τα ονόματα των μεταβλητών των προγραμμάτων ,που αντιστοιχούν σε μετρικές αξιολόγησης ,όπου ametrics, mutual1, h, c, vm, ch αναφέρονται αντίστοιχα στις μετρικές adjusted rand score, adjusted mutual info score, homogeneity score, completeness score, v-measure score , calinski harabasz score .

Documentation

Module: metPlots.py

Functions:

main()

Χρησιμοποιώντας τις παρακάτω συναρτήσεις πραγματοποιείται η κατασκευή διαγραμμάτων της μορφής «bar plots» όπου στον άξονα των y τοποθετούνται σε αύξουσα σειρά οι τιμές των μετρικών αξιολόγησης των tools και στον άξονα των x τοποθετούνται τα ονόματα των tools που αντιστοιχούν σε κάθε μπάρα. Έτσι η «ψηλότερη» μπάρα υποδεικνύει και το βέλτιστο tool, δηλαδή εκείνο που έδωσε τιμές πιο κοντά στην πραγματικότητα ,συγκρίνοντας τα αποτελέσματα με τη βοήθεια των μετρικών αξιολόγισης.

loadMetLists(csv_file_name)

Parameters: ***csv_file_name:*** όρισμα τύπου string - το όνομα του csv file (μαζί με την κατάληξη του αρχείου .csv)

Returns: -

Δέχεται τις τιμές των μετρικών αξιολόγησης κάθε tool διαβάζοντας το αντίστοιχο csv αρχείο και στη συνέχεια αποθηκεύει τις τιμές τοποθετώντας τις στο τέλος των αντίστοιχων global lists του προγράμματος. Δηλαδή μετά από πολλαπλές κλήσεις της συνάρτησης loadMetLists μέσα στη main οι λίστες ametrics,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs περιέχουν απ΄την 1η μέχρι την 7η θέση τους τις τιμές της κάθε μετρικής για τα tools PARCS,TreeSne,SnnCliq,GraphSCC,sczidesk,GC3 και Hypercluster αντίστοιχα.

convertStrToFloat(lst_name)

Parameters: ***lst_name:*** όρισμα τύπου list – το όνομα λίστας

Returns: -

Μετατρέπει τα στοιχεία της δοθείσας λίστας από τύπου string σε τύπο float.

plotting(x,met,width,values,mytitle)

Parameters:

x : οι τοποθεσίες των ετικετών του άξονα των x (δηλαδή των ονομάτων των Tools) πάνω στο διάγραμμα πιο συγκεκριμένα αυτή η παράμετρος είναι η ακολουθία των οριζόντιων συντεταγμένων της κάθε μπάρας.

met : η global list που αντιστοιχεί στη μετρική που συμμετέχει ως αξιολόγηση των tools στο συγκεκριμένο παραγόμενο διάγραμμα

width : το πλάτος της κάθε μπάρας του bar plot

values : λίστα από string values που περιέχει τα ονόματα των tools που παρουσιάζονται στον άξονα των x

mytitle : όρισμα τύπου string - Ο τίτλος που εμφανίζεται στην κορυφή στο παράθυρο του plot

Returns: -

Διακόπτει τη ροή του προγράμματος και εμφανίζει νέα παράθυρο με bar plots που αντιστοιχούν σε κάθε μετρική που χρησιμοποιήθηκε για την αξιολόγηση των tools.

Module: tablePlots.py

Functions:

main()

Χρησιμοποιεί τα csv files που προέκυψαν με τη βοήθεια του metrics.py (που βλέπουμε παρακάτω) για 10 διαφορετικά datasets και με τη βοήθεια των ακόλουθων συναρτήσεων δημιουργούνται και τυπώνονται έξι «πίνακες σύγκρισης» όσες και οι μετρικές που χρησιμοποιήθηκαν για την αξιολόγηση των tools. Κάθε πίνακας έχει σε γραμμές και στήλες τα ονόματα των επτά Tools και τα πεδία του γεμίζουν με τιμές από το 0 έως το 10. Για παράδειγμα το πεδίο που αντιστοιχεί στη γραμμή του PARCS και στη στήλη του TreeSne αν έχει για παράδειγμα την τιμή 8 σημαίνει ότι το tool PARCS στα 8 datasets από τα συνολικά είχε καλύτερη απόδοση από το TreeSne. Έτσι, χωρίς περαιτέρω υπολογισμούς καταλαβαίνουμε ότι στο πεδίο που αντιστοιχεί στη γραμμή του TreeSne και στη στήλη του PARCS η τιμή θα είναι ο αριθμός 2 (10-8=2). Δηλαδή 2 «νίκες» του TreeSne έναντι του PARCS. Ουσιαστικά, στη main() εφόσον καλείται η συνάρτηση listOftables με τη σειρά της αλυσιδωτά καλεί τις υπόλοιπες συναρτήσεις όπως αναλύεται παρακάτω και παράγει μία τελική λίστα lstOfTables[ ] με 6 θέσεις που η κάθε μία περιέχει τον τελικό πίνακα σύγκρισης κάθε μετρικής. Τέλος με τη βοήθεια της open source library TableIt [https://github.com/SuperMaZingCoder/TableItS] και με τη συνάρτηση TableIt.printTable() τελικά τυπώνονται στην επιθυμητή μορφολογία (Εικόνα 1) ένας-ένας οι πίνακες αξιολόγησης που δημιουργήθηκαν.

loadMetLists(csv_file_name)

Parameters: ***csv_file_name:*** όρισμα τύπου string - το όνομα του csv file (μαζί με την κατάληξη του αρχείου .csv)

Returns: -

Δέχεται τις τιμές των μετρικών αξιολόγησης κάθε tool διαβάζοντας το αντίστοιχο csv αρχείο και στη συνέχεια αποθηκεύει τις τιμές τοποθετώντας τις στο τέλος των αντίστοιχων global lists του προγράμματος. Δηλαδή μετά από πολλαπλές κλήσεις της συνάρτησης loadMetLists μέσα στη main οι λίστες ametrics,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs περιέχουν απ΄την 1η μέχρι την 7η θέση τους τις τιμές της κάθε μετρικής για τα tools PARCS,TreeSne,SnnCliq,GraphSCC,sczidesk,GC3 και Hypercluster αντίστοιχα.

convertStrToFloat(lst_name)

Parameters: ***lst_name:*** όρισμα τύπου list – το όνομα λίστας

Returns: -

Μετατρέπει τα στοιχεία της δοθείσας λίστας από τύπου string σε τύπο float.

dtset(files)

Parameters: ***files:*** λίστα με string values – κάθε στοιχείο της αντιστοιχεί στο path που οδηγεί στο csv file της επιθυμητής μετρικής του επιθυμητού dataset.

Returns: ***lst_metrics :*** λίστα που τα στοιχεία της είναι οι global λιστες ametrics,mutual1,h,c,vm,ch που περιέχουν τις τιμές των μετρικών που χρειάζομαι στην παρούσα πτυχιακή εργασία.

Καλέι τη συνάρτηση LoadMetLists ώστε να περάσει τις τιμές των μετρικών στις global lists και στη συνέχεια καλεί τη συνάρτηση

convertStrToFloat ώστε οι τιμές στις λίστες να είναι float μορφής για να χρησιμοποιηθούν παρακάτω. Τέλος επιστρέφει μία λίστα με τις επιθυμητές λίστες μετρικών.

comp(all_lsts,x,a,b)

Parameters:

all_lsts : λίστα που αποτελείται από λίστες οι οποίες περιέχουν τα δεδομένα των μετρικών από κάθε dataset (αναλυτικότερα παρακάτω)

x : δέχεται τιμές από 0 έως 5 που αντιστοιχούν στις μετρικές (σύμφωνα με τη συνάρτηση comparison) ως εξής: 0 = ametrics , 1=mutual1 , 2 = h , 3=c, 4=vm και 5=ch

a , b : δέχονται τιμές από το 0 έως το 6 που βοηθούν στην αντιστοιχιση θέσεων μνήμης με τα δεδομένα που αντιστοιχούν στο επιθυμητό ζευγάρι Tools το οποίο θα συγκριθεί όπως αναλύεται παρακάτω. Πιο συγκεκριμένα οι τιμές 0 έως 6 αντίστοιχα αναφέρονται στα ακόλουθα tools με τη σειρά που δίνονται δηλαδή PARCS, TreeSne, SnnCliq, GraphSCC, sczidesk ,GC3 και Hypercluster.

Returns: ***tmp :*** είναι το πλήθος των «νικών» δηλαδή το πλήθος από το σύνολο των datasets στο οποίο υπερισχύει ένα tool από το ζευγάρι των tools που συγκρίνεται. (Παίρνει τιμές από 0 έως 10)

Η συγκεκριμένη συνάρτηση υπολογίζει τις «νίκες» ενός tool έναντι ενός άλλου και έτσι προκύπτει μία τιμή από 0 έως 10 «νίκες» που αργότερα τοποθετείται στο ανάλογο πεδίο του παραγόμενου πίνακα συγκρίσεων. Ειδικότερα, πραγματοποιείται η σύγκριση του all_lists[i][x][a] και του all_lists[i][x][b] ,όπου το i παίρνει τιμές από 0 έως 9 και αντιστοιχούν σε κάθε ένα από τα 10 datasets και τα x,a,b όπως αναλύθηκαν παραπάνω. Έτσι, για παράδειγμα για να δημιουργήσω τον πρώτο πίνακα όπου η συγκρίσεις γίνονται με βάση τη μετρική adjusted rand score έχω all_lists[0][0][a] και all_lists[0][0][b], all_lists[1][0][a] και all_lists[1][0][b],…, all_lists[9][0][a] και all_lists[9][0][b] που αναφέρονται σε όλες τις ametrics λίστες(0) από όλα τα datasets (0-9) όπου a,b παίρνουν τιμές 0 έως 6. Για παράδειγμα λοιπόν, όταν έχω all_lists[0][0][0] και all_lists[0][0][1] γίνεται ο έλεγχος PARCS (0) vs TreeSne (1) για τον 1^{ο} πίνακα - adjusted rand score (0) του 1^{ου} dataset(0).

Όμοια αν τοποθετήσω όπου x=1 γίνονται συγκρίσεις για τη μετρική adjusted mutual info score, κ.ο.κ Όμοια για τις υπόλοιπες περιπτώσεις.

comparison(all_lists,metric)

Parameters:

all_lists : λίστα που αποτελείται από λίστες οι οποίες περιέχουν τα δεδομένα των μετρικών από κάθε dataset (αναλυτικότερα παρακάτω)

metric : όρισμα με string value δέχεται τις τιμές : "ametrics" , "mutual1" , "h" , "c", "vm" και "ch" και έτσι αναφερόμαστε στον παραγόμενο πίνακα για την αντίστοιχη μετρική.

Returns:

lst1 : λίστα με τις τελικές τιμές – «νίκες» σε σειρά ώστε να αντιστοιχούν στα PARCS, TreeSne, SnnCliq, GraphSCC, sczidesk ,GC3 και Hypercluster

lst2 : λίστα με «συμμετρικές» τιμές της lst1. (αντίστοιχη τιμή της lst2 = το πλήθος των datasets (10) – τις νίκες δηλαδή αντίστοιχη τιμή της lst1). Είναι επίσης σε σειρά ώστε να αντιστοιχούν στα PARCS, TreeSne, SnnCliq, GraphSCC, sczidesk ,GC3 και Hypercluster.

Αναλόγως το όρισμα metric για τις τιμές : "ametrics" , "mutual1" , "h" , "c", "vm" ή "ch" αντίστοιχα η μεταβλητή x παίρνει τις τιμές 0,1,… 4 ή 5 και συμπληρώνεται το όρισμα x της comp που καλείται κάθε φορά από τη συνάρτηση comparison. Έτσι, ξεκινά να παράγεται πίνακας για την αντίστοιχη μετρική π.χ. για "ametrics" πίνακας για τη μετρική adjusted rand score.Οι πολλαπλές κλήσεις της comp επιστρέφουν τις τιμές που θα γεμίσουν τα πεδία του κάθε πίνακα συγκρίσεων. Οι τελικές τιμές όπως αναφέρθηκε και παραπάνω αποθηκεύονται στις lst1, lst2 οι οποίες στη συνέχεια χρησιμοποιούνται για να τυπώσουμε τους τελικούς πίνακες μέσω της συνάρτησης create_table(lst1,lst2).

create_table(lst1,lst2)

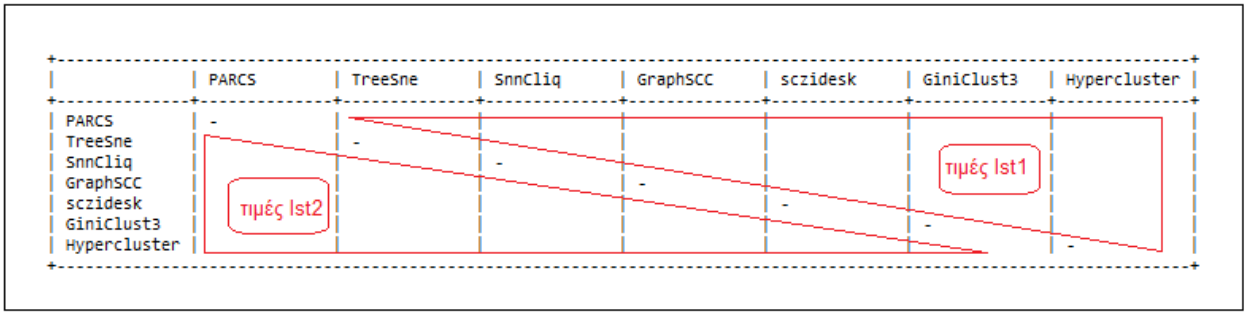
Parameters:

lst1 : λίστα με τις τελικές τιμές – «νίκες»

lst2 : λίστα με «συμμετρικές» τιμές της lst1

Returns: ***met_table :*** πίνακας συγκρίσεων μίας μετρικής

Η συνάρτηση αυτή δημιουργεί με τη χρήση εμφωλευμένων λιστών τη μορφολογία ενός πίνακα συγκρίσεων και γεμίζει τα πεδία του με τις τιμές των lst1 και lst2.



Εικόνα 9.1 *Μορφολογία πίνακα συγκρίσεων*

listOftables(all_ds_lists)

Parameters: ***all_ds_lists :*** λίστα από λίστες που περιέχουν τα δεδομένα – μετρικών των 10 datasets

Returns: ***lst_tables :*** λίστα που περιέχει τους 6 πίνακες που δημιουργήθηκαν απ' τη συγκεκριμένη συνάρτηση

Η συγκεκριμένη συνάρτηση καλεί την comparison για να παράξει τις lst1,lst2 για κάθε μετρική και έπειτα καλεί την create_table για να δημιουργήσει τον πίνακα, 6 φορές όσες και οι μετρικές αξιολόγησης που χρησιμοποιούνται στην παρούσα εργασία. Επίσης, η listOftables δίνει στην comparison το όρισμα all_ds_lists η οποία είναι μία λίστα που περιέχει λίστες

all_ds_lists =[list1,list2,list3,list4,list5,list6,list7,list8,list9,list10]

οι οποίες λίστες list1,…list10 αποτελούν τα δεδομένα που προκύπτουν για κάθε ένα από τα datasets 1 έως 10 μέσω της συνάρτησης dtset π.χ. list1=dtset(files1) όπου files1=['dt1/parcmet.csv','dt1/treemet.csv','dt1/snnmet.csv','dt1/graphSccMetrics.csv','dt1/zid_met.csv','dt1/ginimet.csv','dt1/hypMet.csv'] με dt1 τον φάκελο με τα αποτελέσματα των μετρικών για το 1ο dataset. Όμοια και για τις υπόλοιπες περιπτώσεις. (Κάθε φάκελος dt1,dt2,…dt10 περιέχει 7 csv files που αντιστοιχούν στ' αποτελέσματα των 7 tools από κάθε dataset. Το 1ο αντιστοιχεί στο GSE103334.mat, κλπ.)

Module: metrics.py

Functions:

main()

Δέχεται ως είσοδο το διάνυσμα των predicted labels σε txt αρχείο όπως και το .mat αρχείο, που αντιστοιχεί σε ένα dataset, από το οποίο εξάγονται οι αληθινές κλάσεις ,δηλαδή οι true labels των δεδομένων. Έτσι, παράγονται επτά αρχεία csv ,όσα και τα tools που χρησιμοποιήθηκαν στην παρούσα εργασία, καθώς εκεί αποθηκεύονται όλες οι γνωστές μετρικές που υπολογίστηκαν (διαχωρισμένες με κόμμα). Δημιουργείτε ένα csv file για κάθε ένα απ' τα επτά tools. Τα παραγόμενα αρχεία είναι που χρησιμοποιούνται απ' τα MetPlots.py και TablePlots.py για την αξιολόγιση των tools.

Module: extra.py

Functions:

main()

Εισάγεται δυναμικά το όνομα του .mat αρχείου που περιέχει τα δεδομένα που χρησιμοποιούνται στην είσοδο των tools και στη συνέχεια δημιουργείται ένα data.csv και ένα class.csv αρχείο διαχωρίζοντας τα δεδομένα εισόδου. Αυτά τα νέα αρχεία χρησιμοποιούνται ανάλογα τις ανάγκες από τα tools ,όπως στην είσοδο του giniclust3.

Module: snncliq.py

Functions:

main()

Το snncliq.py δημιουργήθηκε για να παράγει το τελικό .csv file με τις μετρικές που αξιολογούν το tool SnnCliq. Αρχικά, Τρέχουμε σε περιβάλλον matlab το αρχείο SNN.m δίνοντας στα ορίσματα της συνάρτησης SNN( ) τον «πίνακα» data ,με τα δεδομένα του αντίστοιχου dataset για το οποίο διεκπεραιώνουμε τον αλγόριθμο, και ένα όνομα .txt αρχείου στο οποίο αποθηκεύονται τ'αποτελέσματα του matlab προγράμματος (*ενότητα 9.2*). Στη συνέχεια στο Anaconda prompt εκτελείται ο κώδικας clip.py δίνοντας ως παράμετρο – i το όνομα του παραπάνω αρχείου και ως -o ένα νέο όνομα .txt file όπου τελικά αποθηκεύονται τα αποτελέσματα της ομαδοποίησης του συγκεκριμένου scRna tools. Το τελικό αυτό αρχείο χρησιμοποιείται απ' το παρόν «module». (*ενότητα 9.1*).

9.1.2 Ο στόχος και η σχέση μεταξύ των python scripts

Στόχος της υλοποίησης των παραπάνω python modules είναι η συνεργασία τους με σκοπό τη δημιουργία τόσο των bar-plots όσο και των πινάκων συγκρίσεων και έτσι επιτυγχάνεται η αξιολόγηση των scRNA tools αλλά και η οπτικοποίηση των συγκρίσεων των clustering predicted tools της παρούσας εργασίας με βάση τις παραπάνω συγκεκριμένες μετρικές αξιολόγησης.

Πιο συγκεκριμένα, ο κώδικας **Metrics.py** δέχεται τα predicted labels (δηλαδή τις κλάσεις που προέβλεψαν τα tools για τα δεδομένα εισόδου) από .txt αρχείο, υπολογίζει τις μετρικές αξιολόγησης χρησιμοποιώντας τις true labels (δηλαδή τις πραγματικές κλάσεις των δεδομένων) και δημιουργεί .csv αρχείο με τ' αποτελέσματα των μετρικών ώστε να χρησιμοποιηθεί από το πρόγραμμα **metPlots.py** τυπώνοντάς τα σε διαγράμματα bar-plots και από το **TablePlots.py** για να δημιουργηθούν

οι πίνακες συγκρίσεων των tools .Τα GraphScc, GiniClust3 και sczidesk προετοιμάζονται μέσω του **metrics.py** για τη δημιουργία των bar plots απ'το **metPlots.py** και των πινάκων από το **TablePlots.py**. Στα tools PARC, treeSNE, Snn-Cliq δημιουργούνται απευθείας τα csv files με τα δεδομένα.

Επιπλέον, δημιουργήθηκε το **extra.py** για να πραγματοποιήσει την προετοιμασία των δεδομένων που χρησιμοποιούνται απ' το tool GiniClust3 στο gini.py . Συγκεκριμένα, παράγεται αρχείο data.csv το οποίο και χρησιμοποιείται από το gini.py.

Τελος, για το hypercluster παράγεται αυτόματα ξεχωριστό διάγραμμα μετρικών αξιολόγησης από το ίδιο το tool :

Εικόνα 9.2 *Διάγραμμα μετρικών Hypercluster*

Ωστόσο, τροποποιώντας τον κώδικά του πραγματοποιήθηκε λήψη των απαραίτητων δεδομένων για την οπτικοποίηση στο λογισμικό της παρούσας εργασίας. Παρατηρώντας βαθύτερα τον κώδικα του Hypercluster παρατηρείται πως το attribute, .labels_ του clusterer δίνει τις predicted labels ,που χρειάζομαι ,σε μορφή dataframe. Έτσι, κάνοντας ανακατεύθυνση του output σε αρχείο out.txt και print, παρουσιάζονται τα επιθυμητά δεδομένα με την εντολή print(getattr(clusterer,'labels_')). Το tool hypercluster δίνει πολλαπλά πιθανά διανύσματα predicted labels ανάλογα τον αριθμό των clusters. Για τα παραπάνω αποτελέσματα που φαίνονται στα bar plots και τους πινάκες της *ενότητας 7* επιλέγεται η στήλη με τα predicted labels που αντιστοιχεί στην ομαδοποίηση των δεδομένων στον προβλεπόμενο αριθμό clusters.

Name: 4, dtype: object

labels																		
n_clusters	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	2
0	0	0	0	0	4	0	6	8	0	0	8	11	5	0	9	6	14	5	1
1	0	0	2	3	2	2	0	0	9	6	5	2	12	6	5	12	17	9	5
2	0	0	0	2	0	4	2	5	7	4	6	3	10	14	2	13	0	12	1
3	0	0	2	3	2	2	0	3	8	6	0	4	0	4	7	12	17	3	5
4	0	0	0	2	0	4	2	3	8	6	0	4	0	4	7	12	0	3	5
5	0	0	0	0	4	0	6	3	8	0	8	11	5	0	2	13	8	12	1
6	0	0	0	2	0	4	2	5	7	4	6	3	10	14	2	13	8	12	1
7	0	0	0	2	0	4	2	3	8	6	0	4	0	4	7	12	0	3	5
8	0	0	2	3	2	2	0	3	8	6	0	4	0	4	7	12	17	3	5
9	0	0	2	3	2	2	0	0	9	6	0	4	0	4	7	12	17	3	5
10	0	0	2	3	2	2	0	0	9	6	0	4	0	4	7	12	17	3	5
11	0	0	0	2	0	4	2	5	7	4	6	3	10	14	2	13	0	12	1
12	0	0	2	3	2	2	0	0	8	6	0	4	0	4	7	12	17	3	5
13	0	0	2	3	2	2	0	0	9	3	5	2	12	6	5	9	5	9	1
14	0	0	0	2	0	4	6	3	8	6	0	4	0	4	2	13	0	3	1
15	0	0	2	3	2	2	0	0	9	3	5	2	12	6	5	9	5	9	1
16	0	0	0	2	0	4	2	5	7	4	6	3	10	14	2	13	0	3	1
17	0	0	0	2	0	4	6	3	8	6	0	4	0	4	2	13	0	12	1
18	0	0	0	2	0	4	2	3	8	6	0	4	0	4	7	12	0	3	5
19	0	0	0	0	4	0	6	3	8	0	0	11	5	0	2	13	8	12	1

Εικόνα 9.3 *Out.txt αρχείο*

Οπότε, τέλος για λόγους απλούστευσης ακολουθούμε την εξής παραδοχή. Αποθηκεύεται για παράδειγμα το διάνυσμα 2208x1 (*Εικόνα 9.3*) για το πλήθος των 4 clusters ,σύμφωνα με τις προβλέψεις των προηγούμενων scRNA tools, για τα δεδομένα εισόδου GSE103334.mat και μεταφέρεται σε ξεχωριστό .txt file του οποίου τα στοιχεία στη συνέχεια μέσω του **metrics.py** παράγουν ,όπως αναλύθηκε παραπάνω, το csv file μετρικών. Με το **metplots.py** προκύπτουν τα παρακάνω bar plots (*ενότητα 7*) και από το **TablePlots.py** οι πίνακες συγκρίσεων των tools (*ενότητα 7*). Όμοια διαδικασία ακολουθείται και για τα υπόλοιπα .mat datasets.

Ειδικότερα, για τους πίνακες συγκρίσεων που δημιουργούνται τα εργαλεία ομαδοποίησης τρέχουν τουλάχιστον 3 φορές και παράγουν 3 .csv αρχεία με τιμές των μετρικών αξιολόγησης. Από τα αρχεία αυτά υπολογίζεται ο μέσος όρος των δεδομένων τους και τα τελικά αποτελέσματα αποθηκεύονται σε ένα νέο .csv αρχείο. Η διαδικασία αυτή πραγματοποιείται με τη βοήθεια προγράμματος που υλοποιήθηκε με τη γλώσσα προγραμματισμού java στο περιβάλλον Eclipse. Το πρόγραμμα CsvCalc.java επισυνάπτεται στο παρόν παράρτημα στην ενότητα.

9.2 Υλοποίηση αλγορίθμων σε python scripts

metPlots.py

```
import matplotlib.pyplot as plt
import csv
import numpy as np


ametrics=[]
mutual1=[]
mutual2=[]
mutual3=[]
h=[]
c=[]
vm=[]
vmbs=[]
vmbg=[]
fm=[]
sc=[]
ch=[]
dbs=[]



def loadMetLists(csv_file_name):
#loads the lists of metrics
    with open(csv_file_name) as csvfile:
        readCSVv = csv.reader(csvfile, delimiter=',')
        readCSV=list(readCSVv)


    arsmetric = readCSV[0]
    metricm1  = readCSV[1]
    metricm2  = readCSV[2]
    metricm3  = readCSV[3]
    hmetric  = readCSV[4]
    cmetric  = readCSV[5]
    vmmetric  = readCSV[6]
    vmbsmetric  = readCSV[7]
    vmbgmetric  = readCSV[8]
    fmmetric  = readCSV[9]
    scmetric  = readCSV[10]
    chmetric  = readCSV[11]
    dbsmetric  = readCSV[12]



    ametrics.append(arsmetric[0])
    mutual1.append(metricm1[0])
    mutual2.append(metricm2[0])
    mutual3.append(metricm3[0])
    h.append(hmetric[0])
    c.append(cmetric[0])
    vm.append(vmmetric[0])
    vmbs.append(vmbsmetric[0])
    vmbg.append(vmbgmetric[0])
    fm.append(fmmetric[0])
    sc.append(scmetric[0])
    ch.append(chmetric[0])
    dbs.append(dbsmetric[0])
```

```
def convertStrToFloat(lst_name):
    # converting string to float elements in the lists
    for i in range(0, len(lst_name)):
        lst_name[i] = float(lst_name[i])


def plotting(x,met,width,values,mytitle):

    fig, ax = plt.subplots()
    rects = ax.bar(x , met, width, label='')
    rects[0].set_color('cyan')
    rects[1].set_color('crimson')
    rects[2].set_color('coral')
    rects[3].set_color('magenta')
    rects[4].set_color('lightgreen')
    rects[5].set_color('indigo')
    rects[6].set_color('yellow')

    # Add some text for labels, title and custom x-axis tick labels, etc.
    ax.set_title(mytitle)
    ax.set_xticks(x)
    ax.set_xticklabels(values)

    fig.tight_layout()
    plt.show()

def main():

    loadMetLists('parcmet.csv')
    loadMetLists('treemet.csv')
    loadMetLists('snnmet.csv')
    loadMetLists('graphSccMetrics.csv')
    loadMetLists('zid_met.csv')
    loadMetLists('ginimet.csv')
    loadMetLists('hypMet.csv')


    convertStrToFloat(ametrics)
    convertStrToFloat(mutual1)
    convertStrToFloat(mutual2)
    convertStrToFloat(mutual3)
    convertStrToFloat(h)
    convertStrToFloat(c)
    convertStrToFloat(vm)
    convertStrToFloat(vmbs)
    convertStrToFloat(vmbg)
    convertStrToFloat(fm)
    convertStrToFloat(sc)
    convertStrToFloat(ch)
    convertStrToFloat(dbs)


    #Print the lists

    print('ametrics:')
    print(ametrics)

    print('mutual1:')
    print(mutual1)

    print('mutual2:')
    print(mutual2)

    print('mutual3:')
    print(mutual3)

    print('h:')
    print(h)

    print('c:')
```

```
    print(c)

    print('vm:')
    print(vm)

    print('vmbs:')
    print(vmbs)

    print('vmbg:')
    print(vmbg)

    print('fm:')
    print(fm)

    print('sc:')
    print(sc)

    print('ch:')
    print(ch)

    print('dbs:')
    print(dbs)

    values = ['PARCS','TreeSne','SnnCliq','GraphSCC','sczidesk','GC3','Hypercluster']
    x = np.arange(len(values))  # the label locations
    width = 0.35  # the width of the bars

    # 6 final plots
    #ars
    plotting(x,ametrics,width,values,'Adjusted rand score ')
    #mutual1
    plotting(x,mutual1,width,values,'Adjusted mutual info score')
    #h
    plotting(x,h,width,values,'Homogeneity score')
    #c
    plotting(x,c,width,values,'Completeness score')
    #vm
    plotting(x,vm,width,values,'V-measure score')
    #ch
    plotting(x,ch,width,values,'Calinski harabasz score')



if __name__ == "__main__":
                        main()
```

tablePlots.py

```
# Tables "victories of tools"  for every metric
import csv
import TableIt

ds_no=10 # number of datasets # na to diorthwsw na dinetai dunamika?


#initialization
ametrics=[]
mutual1=[]
mutual2=[]
mutual3=[]
h=[]
c=[]
vm=[]
vmbs=[]
vmbg=[]
fm=[]
sc=[]
ch=[]
dbs=[]

def loadMetLists(csv_file_name):
#loads the lists of metrics
    with open(csv_file_name) as csvfile:
        readCSVv = csv.reader(csvfile, delimiter=',')
        readCSV=list(readCSVv)


    arsmetric = readCSV[0]
    metricm1  = readCSV[1]
    metricm2  = readCSV[2]
    metricm3  = readCSV[3]
    hmetric  = readCSV[4]
    cmetric  = readCSV[5]
    vmmetric  = readCSV[6]
    vmbsmetric  = readCSV[7]
    vmbgmetric  = readCSV[8]
    fmmetric  = readCSV[9]
    scmetric  = readCSV[10]
    chmetric  = readCSV[11]
    dbsmetric  = readCSV[12]



    ametrics.append(arsmetric[0])
    mutual1.append(metricm1[0])
    mutual2.append(metricm2[0])
    mutual3.append(metricm3[0])
    h.append(hmetric[0])
    c.append(cmetric[0])
    vm.append(vmmetric[0])
    vmbs.append(vmbsmetric[0])
    vmbg.append(vmbgmetric[0])
    fm.append(fmmetric[0])
    sc.append(scmetric[0])
    ch.append(chmetric[0])
    dbs.append(dbsmetric[0])
```

```
def convertStrToFloat(lst_name):
    # converting string to float elements in the lists
    for i in range(0, len(lst_name)):
        lst_name[i] = float(lst_name[i])

def dtset(files):

    #take the metrics from the csv files

    loadMetLists(files[0])
    loadMetLists(files[1])
    loadMetLists(files[2])
    loadMetLists(files[3])
    loadMetLists(files[4])
    loadMetLists(files[5])
    loadMetLists(files[6])


    # converting string to float elements in the lists

    convertStrToFloat(ametrics)
    convertStrToFloat(mutual1)
    convertStrToFloat(mutual2)
    convertStrToFloat(mutual3)
    convertStrToFloat(h)
    convertStrToFloat(c)
    convertStrToFloat(vm)
    convertStrToFloat(vmbs)
    convertStrToFloat(vmbg)
    convertStrToFloat(fm)
    convertStrToFloat(sc)
    convertStrToFloat(ch)
    convertStrToFloat(dbs)

    # Apo ta data twn arxeiwn xreiazomai mono tis parakatw listes (6 metrikes gia thn ptuxiakh)
    lst_metrics=[ametrics,mutual1,h,c,vm,ch]
    return lst_metrics

def comp(all_lsts,x,a,b):

    tmp=0

    for i in range(0,9):
        if all_lsts[i][x][a] > all_lsts[i][x][b]:
            tmp=tmp+1

    return tmp

def comparison(all_lists,metric):

    x=-1 #initialization

    if metric=="ametrics":
        x=0
    elif metric=="mutual1":
        x=1
    elif metric=="h":
        x=2
    elif metric=="c":
        x=3
    elif metric=="vm":
        x=4
    elif metric=="ch":
        x=5
    else:
        print("Wrong metric argument! Choose one of the following {ametrics,mutual1,h,c,vm,ch}")
```

```
            x1=comp(all_lists,x,0,1)
            x2=comp(all_lists,x,0,2)
            x3=comp(all_lists,x,0,3)
            x4=comp(all_lists,x,0,4)
            x5=comp(all_lists,x,0,5)
            x6=comp(all_lists,x,0,6)
            y1=comp(all_lists,x,1,2)
            y2=comp(all_lists,x,1,3)
            y3=comp(all_lists,x,1,4)
            y4=comp(all_lists,x,1,5)
            y5=comp(all_lists,x,1,6)
            z1=comp(all_lists,x,2,3)
            z2=comp(all_lists,x,2,4)
            z3=comp(all_lists,x,2,5)
            z4=comp(all_lists,x,2,6)
            k1=comp(all_lists,x,3,4)
            k2=comp(all_lists,x,3,5)
            k3=comp(all_lists,x,3,6)
            l1=comp(all_lists,x,4,5)
            l2=comp(all_lists,x,4,6)
            m1=comp(all_lists,x,5,6)

            x_1=ds_no-x1
            x_2=ds_no-x2
            x_3=ds_no-x3
            x_4=ds_no-x4
            x_5=ds_no-x5
            x_6=ds_no-x6
            y_1=ds_no-y1
            y_2=ds_no-y2
            y_3=ds_no-y3
            y_4=ds_no-y4
            y_5=ds_no-y5
            z_1=ds_no-z1
            z_2=ds_no-z2
            z_3=ds_no-z3
            z_4=ds_no-z4
            k_1=ds_no-k1
            k_2=ds_no-k2
            k_3=ds_no-k3
            l_1=ds_no-l1
            l_2=ds_no-l2
            m_1=ds_no-m1



            lst1=[x1,x2,x3,x4,x5,x6,y1,y2,y3,y4,y5,z1,z2,z3,z4,k1,k2,k3,l1,l2,m1]
            lst2=[x_1,x_2,x_3,x_4,x_5,x_6,y_1,y_2,y_3,y_4,y_5,z_1,z_2,z_3,z_4,k_1,k_2,k_3,l_1,l_2,m_1]


            return lst1,lst2

        def create_table(lst1,lst2):



            x1=lst1[0]
            x2=lst1[1]
            x3=lst1[2]
            x4=lst1[3]
            x5=lst1[4]
            x6=lst1[5]
            y1=lst1[6]
            y2=lst1[7]
            y3=lst1[8]
            y4=lst1[9]
            y5=lst1[10]
            z1=lst1[11]
            z2=lst1[12]
            z3=lst1[13]
            z4=lst1[14]
            k1=lst1[15]
            k2=lst1[16]
            k3=lst1[17]
```

```
        l1=lst1[18]
        l2=lst1[19]
        m1=lst1[20]



        x_1=lst2[0]
        x_2=lst2[1]
        x_3=lst2[2]
        x_4=lst2[3]
        x_5=lst2[4]
        x_6=lst2[5]
        y_1=lst2[6]
        y_2=lst2[7]
        y_3=lst2[8]
        y_4=lst2[9]
        y_5=lst2[10]
        z_1=lst2[11]
        z_2=lst2[12]
        z_3=lst2[13]
        z_4=lst2[14]
        k_1=lst2[15]
        k_2=lst2[16]
        k_3=lst2[17]
        l_1=lst2[18]
        l_2=lst2[19]
        m_1=lst2[20]

        #table of a metric
        met_table = [
            ["", "PARCS","TreeSne","SnnCliq","GraphSCC","sczidesk","GiniClust3","Hypercluster"],
            ["PARCS", "-", x1,x2,x3,x4,x5,x6],
            ["TreeSne", x_1, "-",y1,y2,y3,y4,y5],
            ["SnnCliq", x_2, y_1,"-",z1,z2,z3,z4],
            ["GraphSCC", x_3, y_2,z_1,"-",k1,k2,k3],
            ["sczidesk", x_4, y_3,z_2,k_1,"-",l1,l2],
            ["GiniClust3", x_5, y_4,z_3,k_2,l_1,"-",m1],
            ["Hypercluster", x_6, y_5,z_4,k_3,l_2,m_1,"-"]
        ]


        return met_table

    #gia 1o dataset


    def listOftables(all_ds_lists):

        lst1,lst2=comparison(all_ds_lists,"ametrics")

        #adjusted rand score
        ars_table = create_table(lst1,lst2)

        lst1,lst2=comparison(all_ds_lists,"mutual1")

        #adjusted mutual info score
        amis_table = create_table(lst1,lst2)

        lst1,lst2=comparison(all_ds_lists,"h")

        #homogeneity score
        hs_table = create_table(lst1,lst2)

        lst1,lst2=comparison(all_ds_lists,"c")

        #completeness score
        cs_table  = create_table(lst1,lst2)

        lst1,lst2=comparison(all_ds_lists,"vm")

        #v-measure score
        vms_table = create_table(lst1,lst2)

        lst1,lst2=comparison(all_ds_lists,"ch")
```

```
    #calinski harabasz score
    chs_table = create_table(lst1,lst2)


    lst_tables=[ars_table,amis_table,hs_table,cs_table,vms_table,chs_table]

    return lst_tables

def main():



files1=['dt1/parcmet.csv','dt1/treemet.csv','dt1/snnmet.csv','dt1/graphSccMetrics.csv','dt1/zi
d_met.csv','dt1/ginimet.csv','dt1/hypMet.csv']
    #gia 2o dataset

files2=['dt2/parcmet.csv','dt2/treemet.csv','dt2/snnmet.csv','dt2/graphSccMetrics.csv','dt2/zi
d_met.csv','dt2/ginimet.csv','dt2/hypMet.csv']
    #gia 3o dataset

files3=['dt3/parcmet.csv','dt3/treemet.csv','dt3/snnmet.csv','dt3/graphSccMetrics.csv','dt3/zi
d_met.csv','dt3/ginimet.csv','dt3/hypMet.csv']
    #gia 4o dataset

files4=['dt4/parcmet.csv','dt4/treemet.csv','dt4/snnmet.csv','dt4/graphSccMetrics.csv','dt4/zi
d_met.csv','dt4/ginimet.csv','dt4/hypMet.csv']
    #gia 5o dataset

files5=['dt5/parcmet.csv','dt5/treemet.csv','dt5/snnmet.csv','dt5/graphSccMetrics.csv','dt5/zi
d_met.csv','dt5/ginimet.csv','dt5/hypMet.csv']
    #gia 6o dataset

files6=['dt6/parcmet.csv','dt6/treemet.csv','dt6/snnmet.csv','dt6/graphSccMetrics.csv','dt6/zi
d_met.csv','dt6/ginimet.csv','dt6/hypMet.csv']
    #gia 7o dataset

files7=['dt7/parcmet.csv','dt7/treemet.csv','dt7/snnmet.csv','dt7/graphSccMetrics.csv','dt7/zi
d_met.csv','dt7/ginimet.csv','dt7/hypMet.csv']
    #gia 8o dataset

files8=['dt8/parcmet.csv','dt8/treemet.csv','dt8/snnmet.csv','dt8/graphSccMetrics.csv','dt8/zi
d_met.csv','dt8/ginimet.csv','dt8/hypMet.csv']
    #gia 9o dataset

files9=['dt9/parcmet.csv','dt9/treemet.csv','dt9/snnmet.csv','dt9/graphSccMetrics.csv','dt9/zi
d_met.csv','dt9/ginimet.csv','dt9/hypMet.csv']
    #gia 10o dataset

files10=['dt10/parcmet.csv','dt10/treemet.csv','dt10/snnmet.csv','dt10/graphSccMetrics.csv','d
t10/zid_met.csv','dt10/ginimet.csv','dt10/hypMet.csv']


    list1=dtset(files1)
    list2=dtset(files2)
    list3=dtset(files3)
    list4=dtset(files4)
    list5=dtset(files5)
    list6=dtset(files6)
    list7=dtset(files7)
    list8=dtset(files8)
    list9=dtset(files9)
    list10=dtset(files10)

    all_ds_lists=[list1,list2,list3,list4,list5,list6,list7,list8,list9,list10]


    lstOfTables=listOftables(all_ds_lists)


    print("\nAdjusted rand score table for 10 datasets:\n")
    TableIt.printTable(lstOfTables[0], useFieldNames=True, color=None)
    print("\nAdjusted mutual info score table for 10 datasets:\n")
    TableIt.printTable(lstOfTables[1], useFieldNames=True, color=None)
```

```
  print("\nHomogeneity score table for 10 datasets:\n")
    TableIt.printTable(lstOfTables[2], useFieldNames=True, color=None)
    print("\nCompleteness score table for 10 datasets:\n")
    TableIt.printTable(lstOfTables[3], useFieldNames=True, color=None)
    print("\nV-measure score table for 10 datasets:\n")
    TableIt.printTable(lstOfTables[4], useFieldNames=True, color=None)
    print("\nCalinski Harabasz score table for 10 datasets:\n")
    TableIt.printTable(lstOfTables[5], useFieldNames=True, color=None)



if __name__ == "__main__":
    main()
```

extra.py

```
from scipy.io.matlab.mio import loadmat
import numpy as np
#for the input of giniclust3 tool

def main():
    print('Enter your .mat file-name:')
    filenm = input()
    print('Your file is: ' + filenm)

    dt=loadmat(filenm) # for example GSE103334.mat

    #  array object
    arr =dt['data']
    cl =dt['class']

    l=[int(x) for x in cl] #convert to float items for the tool's function
    classarray=np.array(l) #convert to array

    dataarray=arr #convert to array

    # Saving the array
    #.csv Converter
    np.savetxt("data.csv", dataarray, delimiter=",")
    np.savetxt("class.csv", classarray, delimiter=",")


if __name__ == "__main__":
    main()
```

metrics.py

```
import numpy as np
from scipy.io.matlab.mio import loadmat
from sklearn.metrics.cluster import
adjusted_rand_score,adjusted_mutual_info_score,mutual_info_score,normalized_mutual_info_score,
homogeneity_score,completeness_score,v_measure_score,homogeneity_completeness_v_measure,fowlke
s_mallows_score,silhouette_score,calinski_harabasz_score,davies_bouldin_score,contingency_matr
ix

#Autos o kwdikas dexetai to dianusma twn predicted labels ws arxeio kai dhmiourgei .csv arxeio
me tis metrikes wste na xrhsimopoihthei apo to metPlots2.py
#Gia ta tools: GraphSCC (out_res.txt), scziDesk(out_pred_zid.txt),GiniClust3(out_pred_gini.txt)


def main():

    print('Enter your .mat file-name:')
    filenm = input()
    print('Your file is: ' + filenm)

    #for example filenm ='GSE103334.mat'
    dt=loadmat(filenm)

    # array object - class
    classarray=dt["class"]


    lst2=[float(x) for x in classarray] #convert to float items for the PARC() function
    classarray=np.array(lst2) #convert to array

    #change the numbering of classes

    for n  in range(0,len(classarray)):
        classarray[n] = classarray[n] - 1

    #then changed array classarray starts from 0

    labels_true1=classarray

    print('Enter the name of predict. classes - file :')
    filenm2 = input()


    labels_pred1 = []
    with open(filenm2, "r") as f:
        for line in f:
            labels_pred1.append(int(line.strip()))



    mutual1=adjusted_mutual_info_score(labels_true1, labels_pred1)
    mutual2=mutual_info_score(labels_true1, labels_pred1)
    mutual3=normalized_mutual_info_score(labels_true1, labels_pred1)
    ars=adjusted_rand_score(labels_true1, labels_pred1)
    h=homogeneity_score(labels_true1, labels_pred1)
    c=completeness_score(labels_true1, labels_pred1)
    vm=v_measure_score(labels_true1, labels_pred1) #beta=1.0
    hcvm=homogeneity_completeness_v_measure(labels_true1, labels_pred1)
    vmbs=v_measure_score(labels_true1, labels_pred1, beta=0.6)
    vmbg=v_measure_score(labels_true1, labels_pred1, beta=1.8)
    fm=fowlkes_mallows_score(labels_true1, labels_pred1)
    conarray=contingency_matrix(labels_true1, labels_pred1)

    labels_true1=labels_true1.reshape(-1, 1) # reshape the dim. for the above metrics



    sc=silhouette_score(labels_true1, labels_pred1)
```

```
    ch=calinski_harabasz_score(labels_true1, labels_pred1)
    dbs=davies_bouldin_score(labels_true1, labels_pred1)

    print("adjusted_rand_score: ",ars)
    print("adjusted_mutual_info_score: ",mutual1)
    print("mutual_info_score: ",mutual2)
    print("normalized_mutual_info_score: ",mutual3)
    print("homogeneity_score: ",h)
    print("completeness_score: ",c)
    print("v_measure_score: ",vm)
    print("homogeneity_completeness_v_measure: ",hcvm)
    print("v_measure_score, beta=0.6: ",vmbs)
    print("v_measure_score, beta=1.8: ",vmbg)
    print("fowlkes_mallows_score: ",fm)
    print("contingency_matrix:")
    print(conarray)
    print("silhouette_score: ",sc)
    print("calinski_harabasz_score: ",ch)
    print("davies_bouldin_score: ",dbs)


    metrics = [ars,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs]

    print('Give a name to final metric file :')
    filenm3 = input()


    # using the savetxt
    # from the numpy module
    np.savetxt(filenm3,
               metrics,
               delimiter =", ",
               fmt ='% s')



if __name__ == "__main__":
    main()
```

sncliq.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.matlab.mio import loadmat
from                    sklearn.metrics.cluster                    import
adjusted_rand_score,adjusted_mutual_info_score,mutual_info_score,normalized
_mutual_info_score,homogeneity_score,completeness_score,v_measure_score,hom
ogeneity_completeness_v_measure,fowlkes_mallows_score,silhouette_score,cali
nski_harabasz_score,davies_bouldin_score,contingency_matrix
import pandas as pd

def main():

        print('Enter your .mat file-name:')
        filenm = input()

        print('Your file is: ' + filenm)
```

```
dt=loadmat(filenm)
#  array object - data
X =dt["data"]

# array object - class
classarray=dt["class"]


lst2=[float(x) for x in classarray] #convert to float items for the
PARC() function
classarray=np.array(lst2) #convert to array

#change the numbering of classes

for n  in range(0,len(classarray)):
    classarray[n] = classarray[n] - 1

#then changed array classarray starts from 0

labels_true1=classarray

#convert txt to csv

# readinag given csv file
# and creating dataframe
dataframe1 = pd.read_csv("mydataoutout.txt")

# storing this dataframe in a csv file
dataframe1.to_csv('pred.csv',index = None)
# Reading the csv into an array
labels_pred1 = np.genfromtxt("pred.csv", delimiter=",")

theclasses=set([int(x) for x in labels_pred1])
print("The predicted classes: ",theclasses)  # prints the different
types of predicted classes


theclasses2=set([int(x) for x in labels_true1])
print("The true classes: ",theclasses2)  # prints the different types
of true classes


mutual1=adjusted_mutual_info_score(labels_true1, labels_pred1)
mutual2=mutual_info_score(labels_true1, labels_pred1)
mutual3=normalized_mutual_info_score(labels_true1, labels_pred1)
ars=adjusted_rand_score(labels_true1, labels_pred1)
h=homogeneity_score(labels_true1, labels_pred1)
c=completeness_score(labels_true1, labels_pred1)
vm=v_measure_score(labels_true1, labels_pred1) #beta=1.0
hcvm=homogeneity_completeness_v_measure(labels_true1, labels_pred1)
vmbs=v_measure_score(labels_true1, labels_pred1, beta=0.6)
vmbg=v_measure_score(labels_true1, labels_pred1, beta=1.8)
fm=fowlkes_mallows_score(labels_true1, labels_pred1)
conarray=contingency_matrix(labels_true1, labels_pred1)
```

```
        labels_true1=labels_true1.reshape(-1, 1) # reshape the dim. for the
        above metrics


        sc=silhouette_score(labels_true1, labels_pred1)
        ch=calinski_harabasz_score(labels_true1, labels_pred1)
        dbs=davies_bouldin_score(labels_true1, labels_pred1)

        print("adjusted_rand_score: ",ars)
        print("adjusted_mutual_info_score: ",mutual1)
        print("mutual_info_score: ",mutual2)
        print("normalized_mutual_info_score: ",mutual3)
        print("homogeneity_score: ",h)
        print("completeness_score: ",c)
        print("v_measure_score: ",vm)
        print("homogeneity_completeness_v_measure: ",hcvm)
        print("v_measure_score, beta=0.6: ",vmbs)
        print("v_measure_score, beta=1.8: ",vmbg)
        print("fowlkes_mallows_score: ",fm)
        print("contingency_matrix:")
        print(conarray)
        print("silhouette_score: ",sc)
        print("calinski_harabasz_score: ",ch)
        print("davies_bouldin_score: ",dbs)



        fig = plt.figure()
        ax = fig.add_subplot(111)
        values    =    ['ars',    'adj.mutual',    'mutual',    'norm.mutual',
        'homogeneity','completeness','v
        measure','vm,b=0.6','vm,b=1.8','fowl.mall.','silhouette','calin.hara
        b.','dav.boul.']
        metrics = [ars,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs]
        ax.bar(values,metrics)
        ax.set_title("SNN-Cliq")
        plt.show()

        np.savetxt("snnmet.csv",
                   metrics,
                   delimiter =", ",
                   fmt ='% s')

if __name__ == "__main__":
    main()
```

9.3 Προσαρμογές στα scripts των tools

GraphSCC

graphSCC.py

```
from __future__ import print_function, division
import sys
import argparse
import math
import random
import numpy as np
from sklearn.cluster import KMeans
from   sklearn.metrics.cluster   import   normalized_mutual_info_score   as
nmi_score
from sklearn.metrics import adjusted_rand_score as ari_score
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.nn.parameter import Parameter
from torch.optim import Adam, RMSprop
from torch.utils.data import DataLoader
from torch.nn import Linear
from utils import load_graph, load_data_origin_data
from GNN import GCNII
from evaluation import eva,eva2
from torch.utils.data import Dataset
from collections import Counter
import h5py
import copy
from scipy.io.matlab.mio import loadmat
import matplotlib.pyplot as plt



from sklearn.metrics import pairwise_distances as pair
from sklearn.preprocessing import normalize
from pre_processing import pre_processing_single
from sklearn import preprocessing
from scipy.stats import pearsonr
from calcu_graph import construct_graph_kmean
torch.set_num_threads(2)
seed=666
import torch.backends.cudnn as cudnn
cudnn.deterministic = True
cudnn.benchmark = True
import random
random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed_all(seed)
```

```
        np.random.seed(seed)
        import pandas as pd


        class AE(nn.Module):

            def __init__(self, n_enc_1, n_enc_2, n_enc_3, n_dec_1, n_dec_2, n_dec_3,
                         n_input, n_z):
                super(AE, self).__init__()

                self.enc_1 = Linear(n_input, n_enc_1)
                self.enc_2 = Linear(n_enc_1, n_enc_2)
                self.enc_3 = Linear(n_enc_2, n_enc_3)
                #
                self.z_layer = Linear(n_enc_3, n_z)
                #
                self.dec_1 = Linear(n_z, n_dec_1)
                self.dec_2 = Linear(n_dec_1, n_dec_2)
                self.dec_3 = Linear(n_dec_2, n_dec_3)


                self.x_bar_layer = Linear(n_dec_3, n_input)


            def forward(self, x):
                enc_h1 = F.relu(self.enc_1(x))
                enc_h2 = F.relu(self.enc_2(enc_h1))
                enc_h3 = F.relu(self.enc_3(enc_h2))

                z = self.z_layer(enc_h3)

                dec_h1 = F.relu(self.dec_1(z))
                dec_h2 = F.relu(self.dec_2(dec_h1))
                dec_h3 = F.relu(self.dec_3(dec_h2))
                x_bar = self.x_bar_layer(dec_h3)
                return x_bar, enc_h1, enc_h2, enc_h3, z


        class GraphSCC(nn.Module):

            def __init__(self, n_enc_1, n_enc_2, n_enc_3, n_dec_1, n_dec_2, n_dec_3,
                         n_input, n_z, n_clusters, v=1):
                super(GraphSCC, self).__init__()

                # autoencoder for intra information
                self.ae = AE(
                    n_enc_1=n_enc_1,
                    n_enc_2=n_enc_2,
                    n_enc_3=n_enc_3,
                    n_dec_1=n_dec_1,
                    n_dec_2=n_dec_2,
                    n_dec_3=n_dec_3,
                    n_input=n_input,
                    n_z=n_z)



                self.gcn_model = GCNII(nfeat=n_input,
                               nlayers=args.nlayers,
```

```
                        nhidden=args.nhidden,
                        nclass=n_clusters,
                        dropout=0,
                        lamda=0.3,
                        alpha=0.2,
                        variant=False)

        # cluster layer
        self.cluster_layer = Parameter(torch.Tensor(n_clusters, n_z))
        torch.nn.init.xavier_normal_(self.cluster_layer.data)

        # degree
        self.v = v

    def add_noise(self, inputs):
        return inputs + (torch.randn(inputs.shape) * args.noise_value)

    def pretrain_ae(self, dataset):
        train_loader = DataLoader(dataset, batch_size=args.pre_batch_size,
shuffle=True)
        optimizer = Adam(self.ae.parameters(), lr=args.pre_lr)
        for epoch in range(args.pre_epoch):
            for batch_idx, (x, _) in enumerate(train_loader):
                x_noise = self.add_noise(x)
                x_noise = x_noise.cuda()
                x = x.cuda()

                x_bar, _, _, _, z = self.ae(x_noise)
                loss = F.mse_loss(x_bar, x)

                optimizer.zero_grad()
                loss.backward()
                optimizer.step()

    def forward(self, x, adj):
        # DNN Module
        x_bar, tra1, tra2, tra3, z = self.ae(x)

        predict = self.gcn_model(x, adj)

        # Dual Self-supervised Module
        q = 1.0 / (1.0 + torch.sum(torch.pow(z.unsqueeze(1) -
self.cluster_layer, 2), 2) / self.v)
        q = q.pow((self.v + 1.0) / 2.0)
        q = (q.t() / torch.sum(q, 1)).t()

        return x_bar, q, predict, z


def target_distribution(q):
    weight = q**2 / q.sum(0)
    return (weight.t() / weight.sum(1)).t()


class LoadDataset(Dataset):
    def __init__(self, data):
        self.x = data

    def __len__(self):
```

```
            return self.x.shape[0]

        def __getitem__(self, idx):
            return torch.from_numpy(np.array(self.x[idx])).float(), \
                   torch.from_numpy(np.array(idx))


    def train_graphscc(dataset):

        model = GraphSCC(512, 256, 64, 64, 256, 512,
                    n_input=args.n_input,
                    n_z=args.n_z,
                    n_clusters=args.n_clusters,
                    v=1.0).to(device)
        print(model)

        model.pretrain_ae(LoadDataset(dataset.x))
        optimizer = Adam(model.parameters(), lr=args.lr)
        data = torch.Tensor(dataset.x).to(device)
        y = dataset.y

        with torch.no_grad():
            xbar, _, _, _, z = model.ae(data)
        print("hey")
        print(args.n_clusters)
        kmeans = KMeans(n_clusters=args.n_clusters, n_init=20, random_state=666)
        y_pred = kmeans.fit_predict(z.data.cpu().numpy())
        model.cluster_layer.data                                                =
    torch.tensor(kmeans.cluster_centers_).to(device)
        y_pred_last = y_pred

        pae_acc,                                                    pae_nmi,
    pae_ari,pae_mutual1,pae_mutual2,pae_h,pae_c,pae_vm,pae_hcvm,pae_vmbs,pae_vm
    bg,pae_fm,pae_conarray,pae_sc,pae_ch,pae_dbs   =   eva(y,   y_pred,   'pae',
    pp=False)
        print(":pae_acc {:.4f}".format(pae_acc))
        print(", pae_nmi {:.4f}".format(pae_nmi))
        print(", pae_ari {:.4f}".format(pae_ari))
        print(", pae_mutual1 {:.4f}".format(pae_mutual1))
        print(", pae_mutual2 {:.4f}".format(pae_mutual2))
        print(", pae_h {:.4f}".format(pae_h))
        print(", pae_c {:.4f}".format(pae_c))
        print(", pae_vm {:.4f}".format(pae_vm))
        print(", pae_hcvm : "),print(pae_hcvm)
        print(", pae_vmbs {:.4f}".format(pae_vmbs))
        print(", pae_vmbg {:.4f}".format(pae_vmbg))
        print(", pae_fm {:.4f}".format(pae_fm))
        print(", pae_conarray : "),print(pae_conarray)
        print(", pae_sc {:.4f}".format(pae_sc))
        print(", pae_ch {:.4f}".format(pae_ch))
        print(", pae_dbs {:.4f}".format(pae_dbs))


        features = z.data.cpu().numpy()
        error_rate = construct_graph_kmean(args.name, features.copy(),y, y,
                                           load_type='csv',         topk=args.k,
    method='ncos')
        adj = load_graph(args.name, k=args.k, n=dataset.x.shape[0])
        adj = adj.cuda()
```

```
    patient=0
    series=False
    for epoch in range(args.train_epoch):
        if epoch % 1 == 0:
        # update_interval
            xbar, tmp_q, pred, z = model(data, adj)
            tmp_q = tmp_q.data
            p = target_distribution(tmp_q)

            res1 = tmp_q.cpu().numpy().argmax(1)       #Q
            res2 = pred.data.cpu().numpy().argmax(1)   #Z
            res3 = p.data.cpu().numpy().argmax(1)      #P
            r2=set([int(x) for x in res2])
            cnt=sum(1 for num in r2)
            Q_acc,                                                          Q_nmi,
Q_ari,Q_mutual1,Q_mutual2,Q_h,Q_c,Q_vm,Q_hcvm,Q_vmbs,Q_vmbg,Q_fm,Q_conarray
,Q_sc,Q_ch,Q_dbs = eva(y, res1, str(epoch) + 'Q', pp=False)
            P_acc,                                                          P_nmi,
p_ari,P_mutual1,P_mutual2,P_h,P_c,P_vm,P_hcvm,P_vmbs,P_vmbg,P_fm,P_conarray
,P_sc,P_ch,P_dbs = eva(y, res3, str(epoch) + 'P', pp=False)
            if (cnt<2):
                Z_acc,                                                      Z_nmi,
Z_ari,Z_mutual1,Z_mutual2,Z_h,Z_c,Z_vm,Z_hcvm,Z_vmbs,Z_vmbg,Z_fm,Z_conarray
= eva2(y, res2, str(epoch) + 'Z', pp=False)
                Z_sc=-1
                Z_ch=-1
                Z_dbs=-1
            else:
                Z_acc,                                                      Z_nmi,
Z_ari,Z_mutual1,Z_mutual2,Z_h,Z_c,Z_vm,Z_hcvm,Z_vmbs,Z_vmbg,Z_fm,Z_conarray
,Z_sc,Z_ch,Z_dbs = eva(y, res2, str(epoch) + 'Z', pp=False)
            print(epoch)
            print(':Q_acc {:.5f}'.format(Q_acc))
            print(', Q_nmi {:.5f}'.format(Q_nmi))
            print(', Q_ari {:.5f}'.format(Q_ari))
            print(', Q_mutual1 {:.5f}'.format(Q_mutual1))
            print(', Q_mutual2 {:.5f}'.format(Q_mutual2))
            print(', Q_h {:.5f}'.format(Q_h))
            print(', Q_c {:.5f}'.format(Q_c))
            print( ', Q_vm {:.5f}'.format(Q_vm))
            print(', Q_hcvm : '),print(Q_hcvm)
            print( ', Q_vmbs {:.5f}'.format(Q_vmbs))
            print( ', Q_vmbg {:.5f}'.format(Q_vmbg))
            print( ', Q_fm {:.5f}'.format(Q_fm))
            print( ', Q_conarray : '),print(Q_conarray)
            print( ', Q_sc {:.5f}'.format(Q_sc))
            print( ', Q_ch {:.5f}'.format(Q_ch))
            print( ', Q_dbs {:.5f}'.format(Q_dbs))
            if (epoch==(args.train_epoch-1)):
                fig = plt.figure()
                ax = fig.add_subplot(111)
                values                                                      =
['Q_acc','Q_nmi','Q_ari','Q_mutual1','Q_mutual2','Q_h','Q_c','Q_vm','Q_vmbs
','Q_vmbg','Q_fm','Q_sc','Q_ch','Q_dbs']
                metrics                                                     =
[Q_acc,Q_nmi,Q_ari,Q_mutual1,Q_mutual2,Q_h,Q_c,Q_vm,Q_vmbs,Q_vmbg,Q_fm,Q_sc
,Q_ch,Q_dbs]
                ax.bar(values,metrics)
```

```
                    ax.set_title("GraphSCC")
                    plt.show()
                print(epoch)
                print(':Z_acc {:.5f}'.format(Z_acc))
                print(', Z_nmi {:.5f}'.format(Z_nmi))
                print(', Z_ari {:.5f}'.format(Z_ari))
                print(', Z_mutual1 {:.5f}'.format(Z_mutual1))
                print(', Z_mutual2 {:.5f}'.format(Z_mutual2))
                print(', Z_h {:.5f}'.format(Z_h))
                print(', Z_c {:.5f}'.format(Z_c))
                print(', Z_vm {:.5f}'.format(Z_vm))
                print(', Z_hcvm : '), print(Z_hcvm)
                print(', Z_vmbs {:.5f}'.format(Z_vmbs))
                print(', Z_vmbg {:.5f}'.format(Z_vmbg))
                print(', Z_fm {:.5f}'.format(Z_fm))
                print(', Z_conarray : '),print(Z_conarray)
                print(', Z_sc {:.5f}'.format(Z_sc))
                print(', Z_ch {:.5f}'.format(Z_ch))
                print(', Z_dbs {:.5f}'.format(Z_dbs))
                if (epoch==(args.train_epoch-1)):
                    fig = plt.figure()
                    ax = fig.add_subplot(111)
                    values                                                  =
['Z_acc','Z_nmi','Z_ari','Z_mutual1','Z_mutual2','Z_h','Z_c','Z_vm','Z_vmbs
','Z_vmbg','Z_fm','Z_sc','Z_ch','Z_dbs']
                    metrics                                                 =
[Z_acc,Z_nmi,Z_ari,Z_mutual1,Z_mutual2,Z_h,Z_c,Z_vm,Z_vmbs,Z_vmbg,Z_fm,Z_sc
,Z_ch,Z_dbs]
                    ax.bar(values,metrics)
                    ax.set_title("GraphSCC")
                    plt.show()
                print(epoch)
                print(':P_acc {:.5f}'.format(P_acc))
                print(', P_nmi {:.5f}'.format(P_nmi))
                print(', p_ari {:.5f}'.format(p_ari))
                print(', P_mutual1 {:.5f}'.format(P_mutual1))
                print(', P_mutual2 {:.5f}'.format(P_mutual2))
                print(', P_h {:.5f}'.format(P_h))
                print(', P_c {:.5f}'.format(P_c))
                print(', P_vm {:.5f}'.format(P_vm))
                print(', P_hcvm : '),print(P_hcvm)
                print(', P_vmbs {:.5f}'.format(P_vmbs))
                print(', P_vmbg {:.5f}'.format(P_vmbg))
                print(', P_fm {:.5f}'.format(P_fm))
                print(', P_conarray : '),print(P_conarray)
                print(', P_sc {:.5f}'.format(P_sc))
                print(', P_ch {:.5f}'.format(P_ch))
                print(', P_dbs {:.5f}'.format(P_dbs))
                if (epoch==(args.train_epoch-1)):
                    fig = plt.figure()
                    ax = fig.add_subplot(111)
                    values                                                  =
['P_acc','P_nmi','P_ari','P_mutual1','P_mutual2','P_h','P_c','P_vm','P_vmbs
','P_vmbg','P_fm','P_sc','P_ch','P_dbs']
                    metrics                                                 =
[P_acc,P_nmi,p_ari,P_mutual1,P_mutual2,P_h,P_c,P_vm,P_vmbs,P_vmbg,P_fm,P_sc
,P_ch,P_dbs]
                    ax.bar(values,metrics)
                    ax.set_title("GraphSCC")
```

```
                plt.show()

            delta_label = np.sum(res2 != y_pred_last).astype(np.float32) /
res2.shape[0]
            y_pred_last = res2
            if epoch > 0 and delta_label < 0.0001:
                if series:
                    patient+=1
                else:
                    patient = 0
                series=True
                if patient==300:
                   print('Reached tolerance threshold. Stopping training.')
                   print("Z_acc:         {}".format(Z_acc),         "Z_nmi:
{}".format(Z_nmi),"Z_ari:                         {}".format(Z_ari),"Z_ari:
{}".format(Z_ari),"Z_mutual1:              {}".format(Z_mutual1),"Z_mutual2:
{}".format(Z_mutual2),"Z_h:   {}".format(Z_h),"Z_c:   {}".format(Z_c),"Z_vm:
{}".format(Z_vm),"Z_hcvm:                        {}".format(Z_hcvm),"Z_vmbs:
{}".format(Z_vmbs),"Z_vmbg:                        {}".format(Z_vmbg),"Z_fm:
{}".format(Z_fm),"Z_conarray:                   {}".format(Z_conarray),"Z_sc:
{}".format(Z_sc),"Z_ch: {}".format(Z_ch),"Z_vmbg: {}".format(Z_dbs))

                   break
            else:
                series=False


        x_bar, q, pred, _ = model(data, adj)
        kl_loss = F.kl_div(q.log(), p, reduction='batchmean')
        ce_loss = F.kl_div(pred.log(), p, reduction='batchmean')
        re_loss = F.mse_loss(x_bar, data)

        loss = args.kl_loss * kl_loss + args.ce_loss * ce_loss + re_loss
        print(epoch,    ':kl_loss    {:.5f}'.format(kl_loss),    ',    ce_loss
{:.5f}'.format(ce_loss),
              ',    re_loss    {:.5f}'.format(re_loss),    ',    total_loss
{:.5f}'.format(loss))

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()


    Q_acc,                                                              Q_nmi,
Q_ari,Q_mutual1,Q_mutual2,Q_h,Q_c,Q_vm,Q_hcvm,Q_vmbs,Q_vmbg,Q_fm,Q_conarray
,Q_sc,Q_ch,Q_dbs = eva(y, res1, str(epoch) + 'Q', pp=False)
    P_acc,                                                              P_nmi,
p_ari,P_mutual1,P_mutual2,P_h,P_c,P_vm,P_hcvm,P_vmbs,P_vmbg,P_fm,P_conarray
,P_sc,P_ch,P_dbs = eva(y, res3, str(epoch) + 'P', pp=False)
    r2=set([int(x) for x in res2])



    print("! ---------------------------------------")


    with open("out_res.txt", "w") as f:
        for s in res2:
            f.write(str(s) +"\n")
```

```
    cnt=sum(1 for num in r2)
    if (cnt<2):
        Z_acc,                                                    Z_nmi,
Z_ari,Z_mutual1,Z_mutual2,Z_h,Z_c,Z_vm,Z_hcvm,Z_vmbs,Z_vmbg,Z_fm,Z_conarray
= eva2(y, res2, str(epoch) + 'Z', pp=False)
        Z_sc=-1
        Z_ch=-1
        Z_dbs=-1
    else:
        Z_acc,                                                    Z_nmi,
Z_ari,Z_mutual1,Z_mutual2,Z_h,Z_c,Z_vm,Z_hcvm,Z_vmbs,Z_vmbg,Z_fm,Z_conarray
,Z_sc,Z_ch,Z_dbs = eva(y, res2, str(epoch) + 'Z', pp=False)
    print(':Q_acc {:.4f}'.format(Q_acc))
    print(', Q_nmi {:.4f}'.format(Q_nmi))
    print(', Q_ari {:.4f}'.format(Q_ari))
    print(', Q_mutual1 {:.4f}'.format(Q_mutual1))
    print(', Q_mutual2 {:.4f}'.format(Q_mutual2))
    print(', Q_h {:.4f}'.format(Q_h))
    print(', Q_c {:.4f}'.format(Q_c))
    print( ', Q_vm {:.4f}'.format(Q_vm))
    print(', Q_hcvm : '),print(Q_hcvm)
    print( ', Q_vmbs {:.4f}'.format(Q_vmbs))
    print( ', Q_vmbg {:.4f}'.format(Q_vmbg))
    print( ', Q_fm {:.4f}'.format(Q_fm))
    print( ', Q_conarray : '),print(Q_conarray)
    print( ', Q_sc {:.4f}'.format(Q_sc))
    print( ', Q_ch {:.4f}'.format(Q_ch))
    print( ', Q_dbs {:.4f}'.format(Q_dbs))
    if (epoch==(args.train_epoch-1)):
        fig = plt.figure()
        ax = fig.add_subplot(111)
        values                                                    =
['Q_acc','Q_nmi','Q_ari','Q_mutual1','Q_mutual2','Q_h','Q_c','Q_vm','Q_vmbs
','Q_vmbg','Q_fm','Q_sc','Q_ch','Q_dbs']
        metrics                                                   =
[Q_acc,Q_nmi,Q_ari,Q_mutual1,Q_mutual2,Q_h,Q_c,Q_vm,Q_vmbs,Q_vmbg,Q_fm,Q_sc
,Q_ch,Q_dbs]
        ax.bar(values,metrics)
        ax.set_title("GraphSCC")
        plt.show()
    print(epoch)
    print(':Z_acc {:.4f}'.format(Z_acc))
    print(', Z_nmi {:.4f}'.format(Z_nmi))
    print(', Z_ari {:.4f}'.format(Z_ari))
    print(', Z_mutual1 {:.4f}'.format(Z_mutual1))
    print(', Z_mutual2 {:.4f}'.format(Z_mutual2))
    print(', Z_h {:.4f}'.format(Z_h))
    print(', Z_c {:.4f}'.format(Z_c))
    print(', Z_vm {:.4f}'.format(Z_vm))
    print(', Z_hcvm : '),print(Z_hcvm)
    print(', Z_vmbs {:.4f}'.format(Z_vmbs))
    print(', Z_vmbg {:.4f}'.format(Z_vmbg))
    print(', Z_fm {:.4f}'.format(Z_fm))
    print(', Z_conarray : '),print(Z_conarray)
    print(', Z_sc {:.4f}'.format(Z_sc))
    print(', Z_ch {:.4f}'.format(Z_ch))
```

```
            print(', Z_dbs {:.4f}'.format(Z_dbs))
            if (epoch==(args.train_epoch-1)):
                fig = plt.figure()
                ax = fig.add_subplot(111)
                values                                                    =
['Z_acc','Z_nmi','Z_ari','Z_mutual1','Z_mutual2','Z_h','Z_c','Z_vm','Z_vmbs
','Z_vmbg','Z_fm','Z_sc','Z_ch','Z_dbs']
                metrics                                                   =
[Z_acc,Z_nmi,Z_ari,Z_mutual1,Z_mutual2,Z_h,Z_c,Z_vm,Z_vmbs,Z_vmbg,Z_fm,Z_sc
,Z_ch,Z_dbs]
                ax.bar(values,metrics)
                ax.set_title("GraphSCC")
                plt.show()
            print(epoch)
            print(':P_acc {:.4f}'.format(P_acc))
            print(', P_nmi {:.4f}'.format(P_nmi))
            print(', p_ari {:.4f}'.format(p_ari))
            print(', P_mutual1 {:.4f}'.format(P_mutual1))
            print(', P_mutual2 {:.4f}'.format(P_mutual2))
            print(', P_h {:.4f}'.format(P_h))
            print(', P_c {:.4f}'.format(P_c))
            print(', P_vm {:.4f}'.format(P_vm))
            print(', P_hcvm : '),print(P_hcvm)
            print(', P_vmbs {:.4f}'.format(P_vmbs))
            print(', P_vmbg {:.4f}'.format(P_vmbg))
            print(', P_fm {:.4f}'.format(P_fm))
            print(', P_conarray : '),print(P_conarray)
            print(', P_sc {:.4f}'.format(P_sc))
            print(', P_ch {:.4f}'.format(P_ch))
            print(', P_dbs {:.4f}'.format(P_dbs))
            if (epoch==(args.train_epoch-1)):
                fig = plt.figure()
                ax = fig.add_subplot(111)
                values                                                    =
['P_acc','P_nmi','P_ari','P_mutual1','P_mutual2','P_h','P_c','P_vm','P_vmbs
','P_vmbg','P_fm','P_sc','P_ch','P_dbs']
                metrics                                                   =
[P_acc,P_nmi,p_ari,P_mutual1,P_mutual2,P_h,P_c,P_vm,P_vmbs,P_vmbg,P_fm,P_sc
,P_ch,P_dbs]
                ax.bar(values,metrics)
                ax.set_title("GraphSCC")
                plt.show()
            print(args)


        if __name__ == "__main__":
            parser = argparse.ArgumentParser(
                description='train',
                formatter_class=argparse.ArgumentDefaultsHelpFormatter)
            parser.add_argument('--name', type=str, default='klein')
            parser.add_argument('--k', type=int, default=10)
            parser.add_argument('--lr', type=float, default=1e-5)
            parser.add_argument('--pre_lr', type=float, default=1e-4)
            parser.add_argument('--n_clusters', default=5, type=int)
            parser.add_argument('--n_z', default=10, type=int)
            parser.add_argument('--load_type', type=str, default='csv')
            parser.add_argument('--kl_loss', type=float, default=0.1)
            parser.add_argument('--ce_loss', type=float, default=0.01)
            parser.add_argument('--similar_method', type=str, default='ncos')
```

```
        parser.add_argument('--pre_batch_size', type=int, default=32)
        parser.add_argument('--pre_epoch', type=int, default=400)
        parser.add_argument('--train_epoch', type=int, default=1000)
        parser.add_argument('--noise_value', type=float, default=1)
        parser.add_argument('--nlayers', type=int, default=5)
        parser.add_argument('--nhidden', type=int, default=256)
        parser.add_argument('--device', type=int, default=0)


        args = parser.parse_args()
        args.cuda = torch.cuda.is_available()
        torch.cuda.set_device(args.device)
        print("use cuda: {}".format(args.cuda))
        device = torch.device("cuda" if args.cuda else "cpu")
        #file_path  =  "C:\\Users\\ThanasisEleni\\Desktop\\GraphSCC-master\\"  +
    args.name +".csv"
        #print(args.name)
        dataset = load_data_origin_data(args.name, args.load_type,scaling=True)
        print("ok")
        args.k = int(len(dataset.y)/100)
        if args.k < 5:
            args.k = 5
        if args.k > 20:
            args.k = 20

        theclasses=set([int(k) for k in dataset.y])
        print(theclasses)
        numclasses=sum(1 for num in theclasses)
        print("The num of classes is: ")
        print(numclasses)
        #args.n_clusters = len(np.unique(dataset.y))
        args.n_clusters =numclasses
        print(args.n_clusters)
        args.n_input = dataset.x.shape[1]

        train_graphscc(dataset)
```

GiniClust3

gini.py

```
import scanpy as sc
import numpy as np
import giniclust3 as gc
import anndata
import pandas as pd
from sklearn.metrics.cluster import
adjusted_rand_score,adjusted_mutual_info_score,mutual_info_score,normalized
_mutual_info_score,homogeneity_score,completeness_score,v_measure_score,hom
ogeneity_completeness_v_measure,fowlkes_mallows_score,silhouette_score,cali
nski_harabasz_score,davies_bouldin_score,contingency_matrix
import matplotlib.pyplot as plt


#Read single cell file

adataSC=sc.read_csv("firstArray12.csv",first_column_names=True)
#Filter expression matrix

sc.pp.filter_cells(adataSC,min_genes=3)
sc.pp.filter_genes(adataSC,min_cells=20)
#Format expression matrix

###example csv file is col:cells X row:genes. Skip this step if the input
matrix is col:genes X row:cells
#adataSC=anndata.AnnData(X=adataRaw.X.T,obs=adataRaw.var,var=adataRaw.obs)
#Normalization

sc.pp.normalize_per_cell(adataSC, counts_per_cell_after=1e4)
#Perform GiniIndexClust

gc.gini.calGini(adataSC) ###Calculate Gini Index
adataGini=gc.gini.clusterGini(adataSC,neighbors=3) ###Cluster based on Gini
Index
#Perform FanoFactorClust

gc.fano.calFano(adataSC) ###Calculate Fano factor
adataFano=gc.fano.clusterFano(adataSC) ###Cluster based on Fano factor
#ConsensusClust

consensusCluster={}
consensusCluster['giniCluster']=np.array(adataSC.obs['rare'].values.tolist(
))
consensusCluster['fanoCluster']=np.array(adataSC.obs['fano'].values.tolist(
))
gc.consensus.generateMtilde(consensusCluster) ###Generate consensus matrix
gc.consensus.clusterMtilde(consensusCluster) ###Cluster consensus matrix
np.savetxt("final.txt",consensusCluster['finalCluster'],
delimiter="\t",fmt='%s')
```

```
        #UMAP visualization

        adataGini.obs['final']=consensusCluster['finalCluster']
        adataFano.obs['final']=consensusCluster['finalCluster']
        print("ok")
        gc.plot.plotGini(adataGini)
        gc.plot.plotFano(adataFano)

        #convert txt file to csv
        # readinag given csv file
        # and creating dataframe
        dataframe1 = pd.read_csv("final.txt")

        # storing this dataframe in a csv file
        dataframe1.to_csv('final.csv',index = None)

        finalarray = np.genfromtxt("final.csv", delimiter=",") #reading csv into
        array
        classarray = np.genfromtxt("array.csv", delimiter=",") #reading csv into
        array

        lst=[int(x) for x in finalarray] #convert to float items
        finalarray=np.array(lst) #convert to array

        lst2=[int(x) for x in classarray] #convert to float items
        classarray=np.array(lst2) #convert to array

        labels_true1=classarray
        labels_pred1=finalarray

        print("! ----------------------------------------")


        with open("out_pred_gini.txt", "w") as f:
            for s in labels_pred1:
                f.write(str(s) +"\n")

        mutual1=adjusted_mutual_info_score(labels_true1, labels_pred1)
        mutual2=mutual_info_score(labels_true1, labels_pred1)
        mutual3=normalized_mutual_info_score(labels_true1, labels_pred1)
        ars=adjusted_rand_score(labels_true1, labels_pred1)
        h=homogeneity_score(labels_true1, labels_pred1)
        c=completeness_score(labels_true1, labels_pred1)
        vm=v_measure_score(labels_true1, labels_pred1) #beta=1.0
        hcvm=homogeneity_completeness_v_measure(labels_true1, labels_pred1)
        vmbs=v_measure_score(labels_true1, labels_pred1, beta=0.6)
        vmbg=v_measure_score(labels_true1, labels_pred1, beta=1.8)
        fm=fowlkes_mallows_score(labels_true1, labels_pred1)
        conarray=contingency_matrix(labels_true1, labels_pred1)

        labels_true1=labels_true1.reshape(-1, 1) # reshape the dim. for the above
        metrics


        sc=silhouette_score(labels_true1, labels_pred1)
        ch=calinski_harabasz_score(labels_true1, labels_pred1)
        dbs=davies_bouldin_score(labels_true1, labels_pred1)

        print("adjusted_rand_score: ",ars)
```

```
print("adjusted_mutual_info_score: ",mutual1)
print("mutual_info_score: ",mutual2)
print("normalized_mutual_info_score: ",mutual3)
print("homogeneity_score: ",h)
print("completeness_score: ",c)
print("v_measure_score: ",vm)
print("homogeneity_completeness_v_measure: ",hcvm)
print("v_measure_score, beta=0.6: ",vmbs)
print("v_measure_score, beta=1.8: ",vmbg)
print("fowlkes_mallows_score: ",fm)
print("silhouette_score: ",sc)
print("calinski_harabasz_score: ",ch)
print("davies_bouldin_score: ",dbs)
print("contingency_matrix:")
print(conarray)

fig = plt.figure()
ax = fig.add_subplot(111)
values     =     ['ars',     'adj.mutual',     'mutual',     'norm.mutual',
'homogeneity','completeness','v
measure','vm,b=0.6','vm,b=1.8','fowl.mall.','silhouette','calin.harab.','da
v.boul.']
metrics = [ars,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs]
ax.bar(values,metrics)
ax.set_title("GiniClust3")
plt.show()

print("The end")
```

sczidesk

zidpkm.py

```
      from preprocess import *
from network import *
from utils import *
import argparse
from                    sklearn.metrics.cluster                    import
adjusted_rand_score,adjusted_mutual_info_score,mutual_info_score,normalized
_mutual_info_score,homogeneity_score,completeness_score,v_measure_score,hom
ogeneity_completeness_v_measure,fowlkes_mallows_score,silhouette_score,cali
nski_harabasz_score,davies_bouldin_score,contingency_matrix
import tensorflow as tf
from scipy.io.matlab.mio import loadmat
import matplotlib.pyplot as plt
```

```
        def cluster_acc(y_true, y_pred):
            y_true = y_true.astype(np.int64)
            assert y_pred.size == y_true.size
            D = max(y_pred.max(), y_true.max()) + 1
            w = np.zeros((D, D), dtype=np.int64)
            for i in range(y_pred.size):
                w[y_pred[i], y_true[i]] += 1
            from sklearn.utils.linear_assignment_ import linear_assignment
            ind = linear_assignment(w.max() - w)
            return sum([w[i, j] for i, j in ind]) * 1.0 / y_pred.size

        if __name__ == "__main__":
            random_seed = [1111, 2222, 3333, 4444, 5555, 6666, 7777, 8888, 9999,
        10000]

            parser = argparse.ArgumentParser(description="train",
        formatter_class=argparse.ArgumentDefaultsHelpFormatter)
            parser.add_argument("--dataname", default = "Quake_10x_Trachea", type =
        str)
            parser.add_argument("--distribution", default = "ZINB")
            parser.add_argument("--self_training", default = True)
            parser.add_argument("--dims", default = [499, 256, 64, 32])
            parser.add_argument("--highly_genes", default = 500)
            parser.add_argument("--alpha", default = 0.001, type = float)
            parser.add_argument("--gamma", default = 0.001, type = float)
            parser.add_argument("--learning_rate", default = 0.0001, type = float)
            parser.add_argument("--random_seed", default = random_seed)
            parser.add_argument("--batch_size", default = 256, type = int)
            parser.add_argument("--update_epoch", default = 10, type = int)
            parser.add_argument("--pretrain_epoch", default = 1000, type = int)
            parser.add_argument("--funetrain_epoch", default = 2000, type = int)
            parser.add_argument("--t_alpha", default = 1.0)
            parser.add_argument("--noise_sd", default = 1.5)
            parser.add_argument("--error", default = 0.001, type = float)
            parser.add_argument("--gpu_option", default = "0")

            args = parser.parse_args()


            filenm=args.dataname
            #dt=loadmat('GSE103334.mat')
            dt=loadmat(filenm)
            #  array object - data
            X =dt["data"]

            # array object - class
            classarray=dt["class"]


            lst2=[int(x) for x in classarray] #convert to float items for the PARC()
        function
            classarray=np.array(lst2) #convert to array

            #change the numbering of classes

            for n  in range(0,len(classarray)):
                classarray[n] = classarray[n] - 1

            #then changed array classarray starts from 0
```

```
        Y = classarray

        print("X")
        # Printing type of arr object
        print("Array is of type: ", type(X))

    # Printing array dimensions (axes)
        print("No. of dimensions: ", X.ndim)
    # Printing shape of array
        print("Shape of array: ", X.shape)

    # Printing size (total number of elements) of array
        print("Size of array: ", X.size)

    # Printing type of elements in array
        print("Array stores elements of type: ", X.dtype)

        print("Y")
            # Printing type of arr object
        print("Array is of type: ", type(Y))

    # Printing array dimensions (axes)
        print("No. of dimensions: ", Y.ndim)
    # Printing shape of array
        print("Shape of array: ", Y.shape)

    # Printing size (total number of elements) of array
        print("Size of array: ", Y.size)

    # Printing type of elements in array
        print("Array stores elements of type: ", Y.dtype)


        X = np.ceil(X).astype(np.int)
        count_X = X

        adata = sc.AnnData(X)
        adata.obs['Group'] = Y
        adata = normalize(adata, copy=True, highly_genes=args.highly_genes,
    size_factors=True, normalize_input=True, logtrans_input=True)
        X = adata.X.astype(np.float32)
        Y = np.array(adata.obs["Group"])
        high_variable = np.array(adata.var.highly_variable.index, dtype=np.int)
        count_X = count_X[:, high_variable]
        size_factor = np.array(adata.obs.size_factors).reshape(-1,
    1).astype(np.float32)
        cluster_number = int(max(Y) - min(Y) + 1)

        result1 = []
        result2 = []

        for seed in args.random_seed:
            np.random.seed(seed)
            tf.reset_default_graph()
            chencluster = autoencoder(args.dataname, args.distribution,
    args.self_training, args.dims, cluster_number, args.t_alpha,
                                      args.alpha, args.gamma,
    args.learning_rate, args.noise_sd)
```

```
        chencluster.pretrain(X,   count_X,   size_factor,   args.batch_size,
args.pretrain_epoch, args.gpu_option)
        chencluster.funetrain(X,   count_X,   size_factor,   args.batch_size,
args.funetrain_epoch, args.update_epoch, args.error)
        kmeans_accuracy = np.around(cluster_acc(Y, chencluster.kmeans_pred),
5)
        kmeans_ARI            =            np.around(adjusted_rand_score(Y,
chencluster.kmeans_pred), 5)
        kmeans_NMI       =       np.around(normalized_mutual_info_score(Y,
chencluster.kmeans_pred), 5)
        mutual1=np.around(adjusted_mutual_info_score(Y,
chencluster.kmeans_pred), 5)
        mutual2=np.around(mutual_info_score(Y, chencluster.kmeans_pred), 5)
        h=np.around(homogeneity_score(Y, chencluster.kmeans_pred), 5)
        c=np.around(completeness_score(Y, chencluster.kmeans_pred), 5)
        vm=np.around(v_measure_score(Y,     chencluster.kmeans_pred),     5)
#beta=1.0
        hcvm=np.around(homogeneity_completeness_v_measure(Y,
chencluster.kmeans_pred), 5)
        vmbs=np.around(v_measure_score(Y,          chencluster.kmeans_pred,
beta=0.6), 5)
        vmbg=np.around(v_measure_score(Y,          chencluster.kmeans_pred,
beta=1.8), 5)
        fm=np.around(fowlkes_mallows_score(Y, chencluster.kmeans_pred), 5)
        conarray=np.around(contingency_matrix(Y,   chencluster.kmeans_pred),
5)
        Ynew=Y.reshape(-1, 1) # reshape the dim. for the above metrics
        sc=np.around(silhouette_score(Ynew, chencluster.kmeans_pred), 5)
        ch=np.around(calinski_harabasz_score(Ynew,
chencluster.kmeans_pred), 5)
        dbs=np.around(davies_bouldin_score(Ynew,   chencluster.kmeans_pred),
5)

        print("! -----------------------------------------")



        pred=chencluster.Y_pred


        with open("out_pred_zid.txt", "w") as f:
            for s in pred:
                f.write(str(s) +"\n")

        accuracy = np.around(cluster_acc(Y, chencluster.Y_pred), 5)
        ARI = np.around(adjusted_rand_score(Y, chencluster.Y_pred), 5)
        NMI = np.around(normalized_mutual_info_score(Y, chencluster.Y_pred),
5)
        nmutual1=np.around(adjusted_mutual_info_score(Y,
chencluster.Y_pred), 5)
        nmutual2=np.around(mutual_info_score(Y, chencluster.Y_pred), 5)
        nh=np.around(homogeneity_score(Y, chencluster.Y_pred), 5)
        nc=np.around(completeness_score(Y, chencluster.Y_pred), 5)
        nvm=np.around(v_measure_score(Y, chencluster.Y_pred), 5) #beta=1.0
        nhcvm=np.around(homogeneity_completeness_v_measure(Y,
chencluster.Y_pred), 5)
        nvmbs=np.around(v_measure_score(Y, chencluster.Y_pred, beta=0.6), 5)
        nvmbg=np.around(v_measure_score(Y, chencluster.Y_pred, beta=1.8), 5)
        nfm=np.around(fowlkes_mallows_score(Y, chencluster.Y_pred), 5)
```

```
        nconarray=np.around(contingency_matrix(Y, chencluster.Y_pred), 5)
        nsc=np.around(silhouette_score(Ynew, chencluster.Y_pred), 5)
        nch=np.around(calinski_harabasz_score(Ynew, chencluster.Y_pred), 5)
        ndbs=np.around(davies_bouldin_score(Ynew, chencluster.Y_pred), 5)


        result1.append([args.dataname,     kmeans_accuracy,     kmeans_ARI,
kmeans_NMI,mutual1,mutual2,h,c,vm,hcvm,vmbs,vmbg,fm,conarray,sc,ch,dbs])
        result2.append([args.dataname,            accuracy,            ARI,
NMI,nmutual1,nmutual2,nh,nc,nvm,nhcvm,nvmbs,nvmbg,nfm,nconarray,nsc,nch,ndb
s])

    output1 = np.array(result1)
    output1   =   pd.DataFrame(output1,   columns=["dataset   name",   "kmeans
accuracy",               "kmeans               ARI",               "kmeans
NMI","mutual1","mutual2","h","c","vm","hcvm","vmbs","vmbg","fm","conarray",
"sc","ch","dbs"])
    print(output1)

    output2 = np.array(result2)
    output2  =  pd.DataFrame(output2,  columns=["dataset  name",  "accuracy",
"ARI",
"NMI","nmutual1","nmutual2","nh","nc","nvm","nhcvm","nvmbs","nvmbg","nfm","
nconarray","nsc","nch","ndbs"])
    print(output2)

fig = plt.figure()
ax = fig.add_subplot(111)
values     =     ["kmeans     accuracy",     "kmeans     ARI",     "kmeans
NMI","mutual1","mutual2","h","c","vm","vmbs","vmbg","fm","sc","ch","dbs"]
metrics             =             [kmeans_accuracy,             kmeans_ARI,
kmeans_NMI,mutual1,mutual2,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs]
ax.bar(values,metrics)
ax.set_title("zidpkm - Output1")
plt.show()

fig = plt.figure()
ax = fig.add_subplot(111)
values                =                ["accuracy",                "ARI",
"NMI","nmutual1","nmutual2","nh","nc","nvm","nvmbs","nvmbg","nfm","nsc","nc
h","ndbs"]
metrics                 =                 [accuracy,                 ARI,
NMI,nmutual1,nmutual2,nh,nc,nvm,nvmbs,nvmbg,nfm,nsc,nch,ndbs]
ax.bar(values,metrics)
ax.set_title("zidpkm - Output2")
plt.show()
```

PARC

mydataparc.py

```
import parc
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.matlab.mio import loadmat
from                    sklearn.metrics.cluster                    import
adjusted_rand_score,adjusted_mutual_info_score,mutual_info_score,normalized
_mutual_info_score,homogeneity_score,completeness_score,v_measure_score,hom
ogeneity_completeness_v_measure,fowlkes_mallows_score,silhouette_score,cali
nski_harabasz_score,davies_bouldin_score,contingency_matrix

print('Enter your .mat file-name:')
filenm = input()
print('Your file is: ' + filenm)

#dt=loadmat('GSE103334.mat')
dt=loadmat(filenm)


#  array object - data
X =dt["data"]

# array object - class
classarray=dt["class"]


lst2=[float(x) for x in classarray] #convert to float items for the PARC()
function
classarray=np.array(lst2) #convert to array

#change the numbering of classes

for n  in range(0,len(classarray)):
    classarray[n] = classarray[n] - 1

#then changed array classarray starts from 0

labels_true1=classarray

#in PARC() labels true and pred has float items
parc1 = parc.PARC(X,true_label=labels_true1,jac_std_global=0.15, random_seed
=1, small_pop = 190) # instantiate PARC / y=None label
# small_pop = smallest cluster population to be considered a community
#(setting small_pop to 50 cleans up some of the smaller clusters, but can
also be left at the default 10)
#egw to auksanw se small_pop = 190 gia na exw apotelesma pio konta sthn
pragmatikothta
```

```
parc1.run_PARC() # run the clustering

parc_labels = parc1.labels
plot_parc_labels = parc1.labels #plots in the end

plt.scatter(X[:, 0], X[:, 1], c=plot_parc_labels) #plot
plt.show()

labels_pred1=np.array([float(x) for x in np.array(parc_labels)]) #convert
parc_labels to array and the items to float, saved in labels_pred1 array

theclasses=set([int(x) for x in labels_pred1])
print("The predicted classes: ",theclasses)  # prints the different types of
predicted classes


theclasses2=set([int(x) for x in labels_true1])
print("The true classes: ",theclasses2)  # prints the different types of true
classes


mutual1=adjusted_mutual_info_score(labels_true1, labels_pred1)
mutual2=mutual_info_score(labels_true1, labels_pred1)
mutual3=normalized_mutual_info_score(labels_true1, labels_pred1)
ars=adjusted_rand_score(labels_true1, labels_pred1)
h=homogeneity_score(labels_true1, labels_pred1)
c=completeness_score(labels_true1, labels_pred1)
vm=v_measure_score(labels_true1, labels_pred1) #beta=1.0
hcvm=homogeneity_completeness_v_measure(labels_true1, labels_pred1)
vmbs=v_measure_score(labels_true1, labels_pred1, beta=0.6)
vmbg=v_measure_score(labels_true1, labels_pred1, beta=1.8)
fm=fowlkes_mallows_score(labels_true1, labels_pred1)
conarray=contingency_matrix(labels_true1, labels_pred1)

labels_true1=labels_true1.reshape(-1, 1) # reshape the dim. for the above
metrics


sc=silhouette_score(labels_true1, labels_pred1)
ch=calinski_harabasz_score(labels_true1, labels_pred1)
dbs=davies_bouldin_score(labels_true1, labels_pred1)

print("adjusted_rand_score: ",ars)
print("adjusted_mutual_info_score: ",mutual1)
print("mutual_info_score: ",mutual2)
print("normalized_mutual_info_score: ",mutual3)
print("homogeneity_score: ",h)
print("completeness_score: ",c)
print("v_measure_score: ",vm)
print("homogeneity_completeness_v_measure: ",hcvm)
print("v_measure_score, beta=0.6: ",vmbs)
print("v_measure_score, beta=1.8: ",vmbg)
print("fowlkes_mallows_score: ",fm)
print("contingency_matrix:")
print(conarray)
print("silhouette_score: ",sc)
print("calinski_harabasz_score: ",ch)
print("davies_bouldin_score: ",dbs)
```

```
fig = plt.figure()
ax = fig.add_subplot(111)
values     =     ['ars',     'adj.mutual',     'mutual',     'norm.mutual',
'homogeneity','completeness','v
measure','vm,b=0.6','vm,b=1.8','fowl.mall.','silhouette','calin.harab.','da
v.boul.']
metrics = [ars,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs]
ax.bar(values,metrics)
ax.set_title("PARC")
plt.show()


# using the savetxt
# from the numpy module
np.savetxt("parcmet.csv",
           metrics,
           delimiter =", ",
           fmt ='% s')
```

tree-SNE

treeSne.py

```
import  sys;  sys.path.append('C:\\Users\\ThanasisEleni\\source\\repos\\FIt-
SNE')

from tree_sne import TreeSNE
from display_tree import display_tree_mnist
import numpy as np
from scipy.io.matlab.mio import loadmat
from                    sklearn.metrics.cluster                    import
adjusted_rand_score,adjusted_mutual_info_score,mutual_info_score,normalized
_mutual_info_score,homogeneity_score,completeness_score,v_measure_score,hom
ogeneity_completeness_v_measure,fowlkes_mallows_score,silhouette_score,cali
nski_harabasz_score,davies_bouldin_score,contingency_matrix
import matplotlib.pyplot as plt

# to reproduce our sample image
SEED = 103
np.random.seed(SEED)

# load the sample dataset
#dt = loadmat("GSE103334.mat")
print('Enter your .mat file-name:')
filenm = input()
print('Your file is: ' + filenm)
dt=loadmat(filenm)
```

```
X = dt["data"]
X = X.reshape(X.shape[0], -1)
#------------------------------S
# array object - class
classarray=dt["class"]


lst2=[float(x) for x in classarray] #convert to float items for the TreeSNE
function
classarray=np.array(lst2) #convert to array

#change the numbering of classes

for n  in range(0,len(classarray)):
    classarray[n] = classarray[n] - 1

#then changed array classarray starts from 0

labels_true1=classarray

#---------------------------------

# seed the tree with the random seed
tree = TreeSNE(rand_state = SEED)
# fit the tree with 30 layers
embeddings, layer_clusters, best_clusters = tree.fit(X, n_layers = 30)


labels_pred1=best_clusters



print("labels_pred1 size is ")
print(len(labels_pred1))
print(labels_pred1.size)
print("type: ",type(labels_pred1))
print("shape",labels_pred1.shape)
print(labels_pred1.ndim)
print(labels_pred1.dtype)


print("labels_true1 size is ")
print(len(labels_true1))
print(labels_true1.size)
print("type labels_true1: ",type(labels_true1))
print("shape",labels_true1.shape)
print(labels_true1.ndim)
print(labels_true1.dtype)

labels_pred1=np.array([float(x) for x in np.array(best_clusters)]) #convert
treesne labels to array and the items to float, saved in labels_pred1 array

print("NOW labels_pred1 size is ")
print(len(labels_pred1))
print(labels_pred1.size)
print("type: ",type(labels_pred1))
print("shape",labels_pred1.shape)
```

```
print(labels_pred1.ndim)
print(labels_pred1.dtype)

theclasses=set([int(x) for x in labels_pred1])
print("The predicted classes: ",theclasses)  # prints the different types of
predicted classes


theclasses2=set([int(x) for x in labels_true1])
print("The true classes: ",theclasses2)  # prints the different types of true
classes

mutual1=adjusted_mutual_info_score(labels_true1, labels_pred1)
mutual2=mutual_info_score(labels_true1, labels_pred1)
mutual3=normalized_mutual_info_score(labels_true1, labels_pred1)
ars=adjusted_rand_score(labels_true1, labels_pred1)
h=homogeneity_score(labels_true1, labels_pred1)
c=completeness_score(labels_true1, labels_pred1)
vm=v_measure_score(labels_true1, labels_pred1) #beta=1.0
hcvm=homogeneity_completeness_v_measure(labels_true1, labels_pred1)
vmbs=v_measure_score(labels_true1, labels_pred1, beta=0.6)
vmbg=v_measure_score(labels_true1, labels_pred1, beta=1.8)
fm=fowlkes_mallows_score(labels_true1, labels_pred1)
conarray=contingency_matrix(labels_true1, labels_pred1)

labels_true1=labels_true1.reshape(-1, 1)

sc=silhouette_score(labels_true1, labels_pred1)
ch=calinski_harabasz_score(labels_true1, labels_pred1)
dbs=davies_bouldin_score(labels_true1, labels_pred1)

print("adjusted_rand_score: ",ars)
print("adjusted_mutual_info_score: ",mutual1)
print("mutual_info_score: ",mutual2)
print("normalized_mutual_info_score: ",mutual3)
print("homogeneity_score: ",h)
print("completeness_score: ",c)
print("v_measure_score: ",vm)
print("homogeneity_completeness_v_measure: ",hcvm)
print("v_measure_score, beta=0.6: ",vmbs)
print("v_measure_score, beta=1.8: ",vmbg)
print("fowlkes_mallows_score: ",fm)
print("silhouette_score: ",sc)
print("calinski_harabasz_score: ",ch)
print("davies_bouldin_score: ",dbs)
print("contingency_matrix:")
print(conarray)

# display the tree
display_tree_mnist(embeddings,best_clusters,None,list(np.unique(best_cluste
rs)),None, distinct = True)

fig = plt.figure()
ax = fig.add_subplot(111)
values     =     ['ars',     'adj.mutual',     'mutual',     'norm.mutual',
'homogeneity','completeness','v
measure','vm,b=0.6','vm,b=1.8','fowl.mall.','silhouette','calin.harab.','da
v.boul.']
metrics = [ars,mutual1,mutual2,mutual3,h,c,vm,vmbs,vmbg,fm,sc,ch,dbs]
```

```
ax.bar(values,metrics)
ax.set_title("treeSne")
plt.show()


# using the savetxt
# from the numpy module
np.savetxt("treemet.csv",
           metrics,
           delimiter =", ",
           fmt ='% s')
```

Snn-Cliq

cliq.py

```
        #!/usr/bin/python

###############################################################
# developed in Python 2.7.3 (tested on Python 3.5.0)       #
# by Chen Xu  cxu3@uncc.edu                                #
# Department of Bioinformatics and Genomics, UNC-Charlotte #
# Aug 2014
# newest update Feb 2017                                                  #
###############################################################

import re, os, sys, getopt
from collections import defaultdict
import itertools

### step4. format output
### infile: one line for one clique
### outfile: cluster ID in a column; line Number is the cell index
def output(clique_list=None, outfile=None, number_cells=None):
    if (clique_list is None) or (outfile is None) or (number_cells is None):
        sys.stderr.write("wrong argument number from output()")
        sys.exit(1)
    cell_clqID={}

    clqID=1
    for line in clique_list:
        cells=line.split(' ')
        cells=list(map(int, cells))  # index
        if len(cells)<3:
            for c in cells:
                cell_clqID[c]="-1"
        else:
            for c in cells:
                cell_clqID[c]=str(clqID)
            clqID+=1
    try:
        outfn=open(outfile, 'w')
        for c in range(1,number_cells+1):
            if c in cell_clqID:
                outfn.write(cell_clqID[c]+'\n')
            else:   # some cells have not been covered because no clique found for them.
                outfn.write('-1\n')
        outfn.close()
```

```
    except IOError:
        sys.stderr.write("cannot open outfile\n")
        sys.exit(1)
# end 4


### 3. delete overlap. After merging, if one cell appears in multiple cliques (and the cliques
do not satisfy merging), select one cliq for the cell to be in. At the same time, delete the
cell from the other cliques.
### criteria: select the cliq who has bigger average link weights to the node (by checking the
SNN graph)
def uniq(clique_list=None, edgeFile=None, outfile=None):
    # clique_list: clique output from merge()
    # edgeFile: pair of nodes and weigh of edge (3 column file)
    # outfile: clique output file, each line is a clique
    if (clique_list is None) or (edgeFile is None):
        sys.stderr.write("wrong argument number from uniq()")
        sys.exit(1)

    # read in the edge file (node1\tnode2\tweight)
    A=defaultdict(dict)
    try:
        infn=open(edgeFile,'r')
        for line in infn:
            line=line.rstrip('\r\n')
            lst=line.split()
            A[lst[0]][lst[1]]=lst[2]
            A[lst[1]][lst[0]]=lst[2]
        infn.close()
    except IOError:
        sys.stderr.write("cannot open "+edgeFile+"\n")
        sys.exit(1)

    # parse the clique list
    cell_cliq=defaultdict(list)
    cliq_cell=defaultdict(list)
    cliqNum=1
    for line in clique_list:
        cells=line.split(' ')
        for c in cells:
            cell_cliq[c].append(str(cliqNum))
            cliq_cell[str(cliqNum)].append(c)
        cliqNum+=1

#------------------- unique assign-----------------------------------------
    for c in list(cell_cliq.keys()): # for each node
        if len(cell_cliq[c])>1:  # if it is in more than one clique
            # count links in each clique associated with the cell
            count_link=defaultdict(list) # key: cliq, value: weighted link in the cliq to the
node
            for connect_node in A[c].keys(): # each neighbor of node c
                for cl in cell_cliq[connect_node]: # for each clique that node c in
                    if cl in cell_cliq[c]:
                        count_link[cl].append(A[c][connect_node])
            # calculate the average of link weights connect from the cliq to the node c
            max_link={}
            for cl in count_link.keys():
                max_link[cl]=sum(list(map(float,count_link[cl])))/len(count_link[cl])
            # select the clique to assign the node c
            assign_cliq=max(max_link, key=max_link.get)
            #print c+" in "+" ".join(cell_cliq[c])+" assign: "+assign_cliq

            # assign to one and delete in other cliqs
            for cl in cell_cliq[c]:
                if not cl==assign_cliq:
                    cliq_cell[cl].remove(c)

#-------------- output------------------------------------------
    uniqCliq_list=[]
    for cl in sorted(list(map(int,cliq_cell.keys()))):
        if len(cliq_cell[str(cl)])>0:
            uniqCliq_list.append(" ".join(cliq_cell[str(cl)])+'\n')

    if outfile:
```

```
        try:
            outfn=open(outfile,'w')
            outfn.write("\n".join(uniqCliq_list))
            outfn.close()
        except IOError:
            sys.stderr.write("cannot open "+outfile+"\n")
            sys.exit(1)

    return uniqCliq_list

## end 3.


### step2. merge quasi-clique. Some cluster are long and extended. Need to merge quasi-cliques
to find such cluster.
### be aware that after merging, the new clusters generated are bigger and have the potential
to be merged again.
### cutoff: the shared must exceed the cutoff percent in one clique to be merged
def merge(clique_list=None, cutoff=None):
    # clique_list: clique output from findquasiclig(). one line is space deliminated indices of
a cluster.
    # cutoff: threshold of overlapping rate for merging
    if (clique_list is None):
        sys.stderr.write("wrong argument number from merge()")
        sys.exit(1)
    if cutoff is None:    cutoff=0.5 #
    cell_cliq=defaultdict(list) # key: cell ID, value: a list of cliq ID(str) that the cell
belongs to
    cliq_cell=defaultdict(list) # key: cliq ID(str), value: a list of cells ID in the cliq
    cliqNum=1

    for line in clique_list:
        cells=line.split(' ')
        for c in cells:
            cell_cliq[c].append(str(cliqNum))
            cliq_cell[str(cliqNum)].append(c)
        cliqNum+=1


#------------- inner merge cycle---------------------------
    merged_bl=mergeOnline(cutoff, cell_cliq, cliq_cell)
    while merged_bl:
        #for cl in cliq_cell.keys():
        #    print str(cl)+" cliq: "+" ".join(cliq_cell[cl])
        #for c in cell_cliq.keys():
        #    print c+" cell: "+" ".join(list(map(str,cell_cliq[c])))
        #print "------ new merge begins ------"
        merged_bl=mergeOnline(cutoff, cell_cliq, cliq_cell)

#------------- output -------------------------------------
    mergedCliq_list=[]
    for cl in cliq_cell.keys():
        mergedCliq_list.append(" ".join(cliq_cell[cl]))
    return mergedCliq_list

# 2a. on-line merging algorithm. refresh the cliques and re-do the candidate searching after
each merge happens.
# sort the candidate of merging by the size of cliques, this will allow bigger clique join small
cliques first, instead of small cliques join first.
def mergeOnline(cutoff=None, cell_cliq=None, cliq_cell=None):
    # the function will make one possible merge and return

    overlapCliq={} # pairs of cliques that have overlap in between
    # overlapCliq: key is pair of clique names, value is Number of cells overlapping between
them
    for c in cell_cliq.keys():
        if len(cell_cliq[c])>1:
            if len(cell_cliq[c])>2: ## pair-wise if 1 cell appear in more than 2 cliques.
                s=list(itertools.combinations(cell_cliq[c],2))
                for pair in s:
                    grp=" ".join(pair)
                    if grp in overlapCliq:
                        overlapCliq[grp]+=1
                    else:
```

```
                        overlapCliq[grp]=1
                else:
                    grp=" ".join(cell_cliq[c])
                    if grp in overlapCliq:
                        overlapCliq[grp]+=1
                    else:
                        overlapCliq[grp]=1


        #select overlapped cliques to merge
        # Since one cliq can overlap with multiple cliqs, create a score to sort all possible
    mergings
        # score is the size of the cliques to merge
        cliq2Merge={}
        for grp in overlapCliq.keys():
            #print "overlap "+grp+": "+str(overlapCliq[grp])
            clqs=grp.split(' ')
            ol=overlapCliq[grp]
            bl=0 # boolean to see if it qualify a merge
            ss=0 # sum of score (sum of overlap proportion in each cliq of the group)
            sum_size=0
            for cl in clqs:
                sum_size+=len(cliq_cell[cl])
                # algo2: overlapping rate only need to exceed cutoff in one cliq
                score=float(ol)/len(cliq_cell[cl])
                ss+=score
                if score>=cutoff:
                    bl=1
            if bl==1:
                cliq2Merge[grp]=sum_size

        # online merge: merge only one candidate group, then refresh.
        # merge the one with the biggest score
        currCliqNum=max(list(map(int, cliq_cell.keys())))+1
        if len(cliq2Merge.keys()):
            grp=max(cliq2Merge, key=cliq2Merge.get)
            #print "merge "+grp+": "+str(cliq2Merge[grp])
            new={}
            for cl in grp.split(' '):
                for c in cliq_cell[cl]:
                    new[int(c)]=''
                    # refresh the cell_cliq for on-line learning
                    index=cell_cliq[c].index(cl)
                    del cell_cliq[c][index]   # delete old cliq index
                    if str(currCliqNum) not in cell_cliq[c]: # add new cliq index
                        cell_cliq[c].append(str(currCliqNum))
                del cliq_cell[cl]
            cliq_cell[str(currCliqNum)]=list(map(str,sorted(new.keys())))

        return len(cliq2Merge.keys())

    # end of mergeInner
    ## end 2. merge()

    ### step 1. find quasi-clique
    # find a r-quasi-clique associated with each node greedily
    # inFile: pair-wise similarity #format: "node_index node_index edge_similarity\n"
    # r: cutoff for quasi-clique
    def findQuasiCliq(infile=None, r=None):
        if infile is None:
            sys.stderr.write("wrong argument number for findQuasiCliq()")
            sys.exit(1)
        if r is None:    r=0.7 #
        minCliqueSize=3 # minimum acceptable clique size

        nodehash=defaultdict(list)
        try:
            infn=open(infile,'r')
            for line in infn:
                line=line.rstrip('\r\n')
                lst=line.split()
                if len(lst)<3 or (not lst[0].isdigit()) or (not lst[1].isdigit()):
                    sys.stderr.write("Error: input file not in the right format:\nIndex_of_node1
    Index_of_node2 Weight_of_edge\n")
```

```
                sys.exit(1)
            try:
                float(lst[2])
            except ValueError:
                sys.stderr.write("Error: input file not in the right format:\nIndex_of_node1
Index_of_node2 Weight_of_edge\n")
                sys.exit(1)
            if float(lst[2])>0:
                nodehash[lst[0]].append(lst[1])
                nodehash[lst[1]].append(lst[0])
        infn.close()
    except IOError:
        sys.stderr.write("Error: cannot open input file "+infile+"\n")
        sys.exit(1)

#-----------------find clique for each node -------------------
    cliques={}

    for node in sorted(nodehash):
        neighList=list(nodehash[node]) ### get all neighbors
        while 1:
            if len(neighList)<(minCliqueSize-1):
                break
            degreeHash={} ### each neighbor connect to how many of the other neighbors
            for neighNode in neighList:
                neighNeighList=list(nodehash[neighNode])
                degree=len(set(neighNeighList) & set(neighList))+1
                degreeHash[neighNode]=degree

            ### greedy: start from the neighbor with the least connection to other neighbors
            minDegNeigh=min(degreeHash, key=degreeHash.get)
            ## %NeighHash length is the number of all neighbors to the Node (including the
minDegNeigh itself
            if degreeHash[minDegNeigh]>=int(len(neighList)*r+1):
                ###  the local cluster satisfy to be a quasi-clique
                break
            else:
                ### does not satisfy, delete the neighbor
                neighList.remove(minDegNeigh)

        # quasi-cliq found
        if len(neighList)>=(minCliqueSize-1):
            neighList.append(node)
            neighList.sort(key=int)
            cliques[" ".join(neighList)]=len(neighList)

    #for clq in cliques.keys():
    #    print clq

#------------------ delete redundant clique ----------------------
### delete cliques that are part of other cliques
    cliques_clean={}
    for currCliq in sorted(cliques, key=cliques.get, reverse=True):
        bl=1 # bl=1: unique ; bl=0: redundant
        for existCliq in sorted(cliques_clean, key=cliques_clean.get, reverse=True):
            if (cliques_clean[existCliq] <= cliques[currCliq]):
                break
            else:
                bl=0
                currCliqAry=currCliq.split(" ")
                existCliqAry=existCliq.split(" ")
                for win in currCliqAry:
                    if win not in existCliqAry:
                        bl=1  # unique
                        break
                if bl==0:
                    break
        if bl:
            cliques_clean[currCliq]=cliques[currCliq]

    max_ID=max(list(map(int, nodehash.keys())))
    return (cliques_clean.keys(),max_ID)
# end 1.
################       end of algorithm         #########
```

```
def usage():
    help_msg="\n  usage: SNNgraph_clustering.py -i <edge_file> -o <out_file> [options]\n\
  -i,--input\tinput file path and name\n\
  -o,--output\toutput file path and name\n\
  optional arguments:\n\
  -r,--r-quasi-cliq\tquasi-clique parameter. A number in the range of (0  1]. Default is 0.7.\n\
  -m,--merging\tmerging parameter. A number in the range of (0  1]. Default is 0.5.\n\
  -n,--number\tnumber of objects. Default is the maximum index in the input file.\n\
  -h,--help\tprint help message.\n"
    print(help_msg)
    sys.exit(0)

def main(argv):

    edgeFile=None
    outfile=None
    merge_cutoff=0.5
    r_cutoff=0.7
    number_cells=None

    try:
        opts, args=getopt.getopt(argv, "i:o:m:r:n:h", ["input=","output=","merging=","r-quasi-
cliq=","number=","help"])
    except getopt.GetoptError as err:
        print(str(err))
        usage()
        sys.exit(0)


    for opt, arg in opts:
        if opt in ('-h', '--help'):
            usage()
        elif opt in ('-i','--input'):
            edgeFile=arg
        elif opt in ('-o', '--output'):
            outfile=arg
        elif opt in ('-m','--merging'):
            merge_cutoff=float(arg)
        elif opt in ('-r','--r-quasi-cliq'):
            r_cutoff=float(arg)
        elif opt in ('-n','--number'):
            number_cells=int(arg)
        else:
            print("Error in arguments: unknown option\n")
            usage()
            sys.exit(0)

    if (edgeFile is None) or (outfile is None):
        sys.stderr.write("Error in arguments: must specify -i -o\n")
        usage()
    else:
        if not os.path.exists(edgeFile):
            sys.stderr.write("Error: "+edgeFile+" does not exists.\n")
            sys.exit(0)
        try:
            outfn=open(outfile, 'w')
            outfn.close()
        except IOError:
            sys.stderr.write("Error: file path not exists "+outfile+"\n")
            sys.exit(1)
    if (r_cutoff>1) or (r_cutoff<=0) or (merge_cutoff>1) or (merge_cutoff<=0):
        sys.stderr.write("Error: parameters are not in the right range.\n")
        usage()
        sys.exit(0)
    cliques=[]
    results=findQuasiCliq(edgeFile, r_cutoff)
    cliques=results[0]
    max_ID=results[1]
    if number_cells is None:
        number_cells=max_ID
    print("input file "+edgeFile)
    print("find "+str(len(cliques))+" quasi-cliques")
    merge_cliques=[]
```

```
    merge_cliques=merge(cliques, merge_cutoff)
    print("merged into "+str(len(merge_cliques))+" clusters")
    uniq_cliques=[]
    uniq_cliques=uniq(merge_cliques, edgeFile)
    print("unique assign done \n")



    output(uniq_cliques, outfile, number_cells)

if __name__=="__main__":
    main(sys.argv[1:])
```

SNN.m

```
function [A]=SNN(d, outfile, k, distance)
% data input. row: object/sample; column: attribute.
% outfile: strength output
% k: nearest neighbor
% distance: methods for calculation distance (euclidean, correlation ...)

switch nargin
    case 1
        error('not enough arguments given: data, output file name.')
    case 2
        k=3
        distance='euclidean';
    case 3
        distance='euclidean';
end

numSpl=size(d,1);
%%
IDX=knnsearch(d, d, 'K', k, 'Distance', distance);
strength=zeros(numSpl,numSpl);
for i=1:numSpl
for j=(i+1):numSpl
    nni=IDX(i,:);
    nnj=IDX(j,:);
    shared=intersect(nni,nnj);
    % the closeness depend on the rank of the shared knn in both list
    s=[0];
    for l=1:length(shared)
        s=[s, k-0.5*(find(nni==shared(l))+find(nnj==shared(l)))];
    end
    % the score means how close the two points are.
    strength(i,j)=max(s);
    strength(j,i)=strength(i,j);
end
end
A=strength;
```

```
fn=fopen(outfile,'w');
for i=1:length(A)
    for j=1:i
        if(A(i,j)>0)
            fprintf(fn,'%d\t%d\t%f\r\n',i,j,A(i,j));
        end
    end
end
fclose(fn);

return
```

Hypercluster

hyper.py

```
import pandas as pd
from scipy.io.matlab.mio import loadmat
import hypercluster
import matplotlib.pyplot as plt
import numpy as np
import sys

#display full Dataframe - print
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', -1)


Path='/content/drive/MyDrive/Dataset/'
print('Enter your .mat file-name:')
filenm = Path + input()
print('Your file is: ' + filenm)
dt=loadmat(filenm)
# array object
firstarray =dt["data"]

#_________________

# array object - class
classarray=dt["class"]
```

```
lst2=[float(x) for x in classarray] #convert to float items for the hyper
function
classarray=np.array(lst2) #convert to array

#change the numbering of classes

for n  in range(0,len(classarray)):
    classarray[n] = classarray[n] - 1

#then changed array classarray starts from 0

labels_true1=classarray

#_________________



data = pd.DataFrame(firstarray)
#labels=None
labels=labels_true1

print("Run with a single clustering algorithm")
clusterer = hypercluster.AutoClusterer()
#methods                                                                =
hypercluster.constants.need_ground_truth+hypercluster.constants.inherent_me
trics
predd1= clusterer.labels_

methods = hypercluster.constants.inherent_metrics
gold_standard = labels
clusterer.fit(data).evaluate(methods,None,gold_standard)

orig_stdout = sys.stdout
f = open('out.txt', 'w')
sys.stdout = f

print (getattr(clusterer,'evaluation_'))
print("labels for 4 clusters")
lab=getattr(clusterer,'labels_')


print (lab[4])
print("labels")
print (getattr(clusterer,'labels_'))
sys.stdout = orig_stdout
f.close()

print(predd1)
print("done!1")
clusterer.visualize_evaluations() #Returns (List[matplotlib.axes.Axes]):
plt.show()

print("Run with a range of algorithms")
clusterer = hypercluster.MultiAutoClusterer()
#methods                                                                =
hypercluster.constants.need_ground_truth+hypercluster.constants.inherent_me
trics
```

```
methods = hypercluster.constants.inherent_metrics
gold_standard = labels
#predd2= clusterer.labels_

clusterer.fit(data).evaluate(methods,None,gold_standard)


orig_stdout = sys.stdout
f = open('out2.txt', 'w')
sys.stdout = f

print (getattr(clusterer,'evaluation_'))
print("labels")
print (getattr(clusterer,'labels_'))

sys.stdout = orig_stdout
f.close()

print("done!2")
nc=hypercluster.additional_metrics.number_of_clusters(None,labels)
print("The no. of clusters: ")
print(nc)
clusterer.visualize_evaluations()
plt.show()
```

9.4 Εργαλείο υπολογισμού μέσων όρων csv αρχείων

CsvCalc.java

```
import java.io.File;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

/**
 * @author Eleni Papaleftheri
 */
public class CsvCalc
{
    public static void main(String[] args) throws IOException
    {
        List<List<String>> data = readCsvs("parcmet_52583_2", "parcmet_52583_3",
"parcmet_52583");
        saveSum("parcmet_52583_sum", data);
    }

    private static void saveSum(String filePath, List<List<String>> data) throws IOException
    {
        final var file = createFile(filePath + ".csv");
        final List<Double> lines = new ArrayList<>();

        for (var i = 0; i < data.get(0).size(); i++)
        {
            double currentValue = 0L;
            for (var j = 0; j < data.size(); j++)
                currentValue += Double.valueOf(data.get(j).get(i));

            lines.add(currentValue / data.size());
        }

        writeTo(file, lines);
    }

    private static void writeTo(File file, List<Double> lines) throws IOException
    {
        final var out = Paths.get(file.getAbsolutePath());
        Files.write(out, lines.stream().map(op -> op.toString()).collect(Collectors.toList()),
Charset.defaultCharset());
    }

    private static File createFile(String filePath) throws IOException
    {
        final var file = new File(filePath);
        if (!file.exists())
            file.createNewFile();
        return file;
    }

    private static List<List<String>> readCsvs(String... filePaths) throws IOException
```

```
        {
            final List<List<String>> contents = new ArrayList<>();
            for (final var filePath : filePaths)
                contents.add(getCsvContent(filePath));

            return contents;
        }

        private static List<String> getCsvContent(String filePath) throws IOException
        {
            final var url = CsvCalc.class.getResource(filePath + ".csv");
            final var f = new File(url.getPath());

            return Files.readAllLines(Paths.get(f.getAbsolutePath()), StandardCharsets.UTF_8);
        }
    }
```

10. Αναφορές-Βιβλιογραφία

I. Baxevanis A. D. and Ouellette F. B. F. , (2016), Bioinformatics: A Practical Guide to the analysis of genes and Proteins, 3rd edition
II. Becker Hila, (2011), Identication and Characterization of Events in Social Media, PhD Thesis.
III. Blumenberg L. , Ruggles K. V., (2020) , “Hypercluster: a python package and SnakeMake pipeline for flexible, parallelized unsupervised clustering optimization”
IV. Brennan R.L. and Light R.J., (I974), "Measuring Agreement When Two Observers Classify People into Categories not Defined in Advance," British Journal of Mathematical and Statistical Psychology, 27, 154-163.
V. Buhmann, J. M. (2002). The Handbook of Brain Theory and Neural Networks.
VI. Chen Xu and Zhengchang Su, (2015), “Identification of cell types from single-cell transcriptomes using a novel clustering method”
VII. Costa-Silva J., Domingues D., and Lopes F. M., “RNA-Seq differential expression analysis: An extended review and a software tool,” PLoS ONE, vol. 12, no. 12. Public Library of Science, Dec. 01, 2017, doi: 10.1371/journal.pone.0190152.
VIII. Dalton, L., Ballarin, V., Brun, M. (2009). Clustering algorithms: on learning, validation, performance, and applications to genomics. Curr. Genomics., 10, 430-45.
IX. Dong R, Yuan GC., 2020, “GiniClust3: a fast and memory-efficient tool for rare cell type identification. BMC Bioinformatics.”
X. Edwards Y. J. K., Bioinformatics and Functional Genomics, vol. 3, no. 2. 2004.
XI. Evans C., Hardin J. and Stoebel D. M., “Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions,” Briefings in bioinformatics, vol. 19, no. 5, pp. 776–792, Sep. 2018, doi: 10.1093/bib/bbx008.
XII. Fowlkes, E.B., and Mallows, C.L., (1983), "A Method for Comparing Two Hierarchical Clusterings", Journal of the American Statistical Association, 78, 553-569.
XIII. Gates Alexander J and Yong-Yeol Ahn (2017), "The Impact of Random Models on Clustering Similarity" , Journal of Machine Learning Research 18: 1–28.

XIV. Goldenberg Anna, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi.,(2010), “A survey of statistical network models.”, Foundations and Trends in Machine Learning, 2(2):129-233, 2.

XV. Han J., Kamber M., Pei J. (2012). Data mining- Concepts and Techniques, 3rd Edition, Morgan Kaufmann Publishers, Elsevier, USA

XVI. Heilkenbrinker A., Mutz Kai-Oliver, Lönne M., Walter J.-G., and Stahl F., “Transcriptome analysis using next-generation sequencing,” Curr. Opin. Biotechnol., vol. 24, no. 1, pp. 22–30, 2013.

XVII. Hinton, G. & Roweis, S. Stochastic neighbor embedding. In Proceedings of the 15th International Conference on Neural Information Processing Systems (NIPS’02), in print. MIT Press 857–864 (2002).

XVIII. Hubert L., Arabie P., (1985), "Comparing Partitions", Journal of Classification, 2:193-218.

XIX. Hubert, L.J., (1977), "Nominal Scale Response Agreement as a Generalized Correlation",British Journal of Mathematical and Statistical Psychology, 30, 98-103.

XX. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data Clustering: A Review. ACM Computing Surveys.

XXI. Jiang L, Chen H, Pinello L, Yuan GC. GiniClust: detecting rare cell types from single-cell gene expression data with Gini

XXII. index. Genome Biol. (2016)

XXIII. Klastorin, T.D., (1985), "The p-Median Problem for Cluster Analysis: A Comparative Test Using the Mixture Model Approach", Management Science, 31, 84-95.

XXIV. Koboldt D. C., Steinberg K. M., Larson D. E., Wilson R. K., and Mardis E. R., “The next-generation sequencing revolution and its impact on genomics,” Cell, vol. 155, no. 1. Cell Press, p. 27, Sep. 26, 2013, doi: 10.1016/j.cell.2013.09.006.

XXV. Kodama, Y., Shumway, M., Leinonen, R., International Nucleotide Sequence Database Collaboration (2012). The Sequence Read Archive: explosive growth of sequencing data. Nucleic Acids Res. 40, D54-56. Mansour T.,(2012),”Combinatorics of Set Partitions”, CRC Press.

XXVI. Kolodziejczyk A. A., J. K. Kim, V. Svensson, J. C. Marioni, and S. A. Teichmann, "The technology and biology of single-cell RNA

sequencing," Mol Cell, vol. 58, no. 4, pp. 610-20, May 21 2015, doi: 10.1016/j.molcel.2015.04.005.

XXVII. Liang Chen, Weinan Wang, Yuyao Zhai, Minghua Deng, (2020), "Deep soft K-means clustering with self-training for single-cell RNA sequence data"

XXVIII. Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S. & Kluger, Y. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. Nat. Methods 16, 243 (2019).

XXIX. Liu X, Song W, Wong BY, Zhang T, Yu S, Lin GN, et al., (2019), " A comparison framework and guideline of clustering methods for mass cytometry data. "

XXX. Malkov,Y. and Yashunin D., (2016), Efficient and Robust Approximate Nearest Neighbor Search using Hierarchical Navigable Small World Graphs. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2018.2889473.

XXXI. McGettigan P. A, "Transcriptomics in the RNA-seq era," Curr. Opin. Chem. Biol., vol. 17, no. 1, pp. 4–11, (2013).

XXXII. Oskolkov N. , "How to tune hyperparameters of tSNE . Towards Data Science", (2019).

XXXIII. Phil Simon, (2013), "Too Big to Ignore: The Business Case for Big Data. Wiley".

XXXIV. Rand, W.M., (1971), "Objective Criteria for the Evaluation of Clustering Methods," Journal of the American Statistical Association, 66, 846-850.

XXXV. Robinson Isaac and Emma Pierce-Hoffman ,(2020) ,TREE-SNE: hierarchical clustering and visualization using t-SNE

XXXVI. Rosenberg Andrew, Hirschberg Julia, (2007),V-Measure: A conditional entropy-based external cluster evaluation measure .

XXXVII. Santos Jorge M. and Embrechts Mark, (2009),"On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification", ISEP - Instituto Superior de Engenharia do Porto, Portugal, Rensselaer ,Polytechnic Institute, Troy, New York, USA

XXXVIII. Sethna James,(2006), Statistical mechanics: entropy, order parameters, and complexity, volume 14 Oxford University Press.

XXXIX. Sharma, Asuda. (2018), Cluster Analysis of Biological Networks

XL. Skourtaniotis Athanasios, (2016) Python based study of clustering algorithms

XLI. Shobana V. Stassen, Dickson M. D. Siu1, Kelvin C. M. Lee1, JoshuaW. K. Ho,Hayden K. H. So and Kevin K. Tsia,(2020), "PARC: ultrafast and accurate clustering of phenotypic data of millions of single cells"

XLII. Theodoridis, S., & Koutroumbas, K. (2009). Pattern Recognition. Elsevier.

XLIII. Traag,V.A. et al., (2011) , Narrow scope for resolution-limit-free community detection. Phys. Rev. E, 84, 016114.

XLIV. Van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. J. Mach. Learning Res. 9, 2579–2605 (2008).

XLV. Vinh Nguyen Xuan, Epps Julien, Bailey James , (2009), "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance", Australia.

XLVI. Vinh, Epps, and Bailey, (2010), "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance".

XLVII. Yang, Algesheimer, and Tessone, (2016), "A comparative analysis of community detection algorithms on artificial networks"

XLVIII. Yuansong Zeng, Xiang Zhou1, Jiahua Rao, Yutong Lu1, Yuedong Yang, (2020) "Accurately Clustering Single-cell RNA-seq data by Capturing Structural Relations between Cells through Graph Convolutional Network"