ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Λογισμικό για μετα-ανάλυση δεδομένων γονιδιακής έκφρασης από μικροσυστοιχίες DNA

Γεώργιος Μανιός

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Μπάγκος Παντελεήμων
Καθηγητής

Λαμία, 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Λογισμικό για μετα-ανάλυση δεδομένων γονιδιακής έκφρασης από μικροσυστοιχίες DNA

Γεώργιος Μανιός

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Μπάγκος Παντελεήμων
Καθηγητής

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: 7/6/2021

Ο Δηλών
Γεώργιος Μανιός

(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.

Λογισμικό για μετα-ανάλυση δεδομένων γονιδιακής έκφρασης από μικροσυστοιχίες DNA

Γεώργιος Μανιός

Τριμελής Επιτροπή:

Μπάγκος Παντελεήμων, Καθηγητής (επιβλέπων)

Πλαγιανάκος Βασίλειος, Καθηγητής

Μπράλιου Γεωργία, Επίκουρη Καθηγήτρια

Στον θείο Μανώλη,
στην γιαγιά Ευαγγελία και στον παππού Γιώργη

Ευχαριστίες

Η παρούσα πτυχιακή εργασία εκπονήθηκε από τον Οκτώβριο του 2020 έως τον Απρίλιο του 2021 στο τμήμα Πληροφορικής με εφαρμογές στη Βιοϊατρική του Πανεπιστημίου Θεσσαλίας.

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου Καθηγητή κ. Παντελεήμονα Μπάγκο, για την συνεχή καθοδήγηση, την στήριξη, την βοήθεια καθώς και τον χρόνο που ήταν πάντα διατεθειμένος να μου αφιερώσει, παρά το υπερβολικά φορτωμένο πρόγραμμα του.

Ακόμη, θα ήθελα να ευχαριστήσω την μεταδιδακτορική ερευνήτρια του τμήματος Πληροφορικής με εφαρμογές στη Βιοϊατρική, κα Κωνσταντίνα Βέννου, για τις υποδείξεις, την καθοδήγηση και τον πολύτιμο χρόνο που αφιέρωσε σε κάθε δυσκολία που αντιμετώπιζα.

Ιδιαίτερες ευχαριστίες θα ήθελα να εκφράσω στους μεταδιδακτορικούς ερευνητές του εργαστηρίου Μοριακής και Υπολογιστικής Βιολογίας και Γενετικής, κα Κοντού Παναγιώτα και κ. Ταμπόση Ιωάννη, για τις παρατηρήσεις και τις υποδείξεις τους τόσο στο λειτουργικό, όσο και στο προγραμματιστικό μέρος του προγράμματος που συνοδεύει την παρούσα πτυχιακή εργασία.

Φυσικά, θα ήθελα να ευχαριστήσω τους γονείς μου και την οικογένεια μου για την αμέριστη αγάπη, την υπομονή και την στήριξη τους σε κάθε μου βήμα.

Κλείνοντας, θα ήθελα να αναφέρω ότι η πτυχιακή εργασία αυτή αφιερώνεται στον θείο μου Εμμανουήλ, στην γιαγιά μου Ευαγγελία και στον παππού μου Γεώργιο Μανιό, που έφυγαν από την ζωή το 2010, 2018 και 2020 αντίστοιχα.

Γεώργιος Μανιός
Λαμία, Ιούνιος 2021

Περιεχόμενα

Κεφάλαιο 1ο
Εισαγωγή

Πρόλογος

Η πτυχιακή εργασία αυτή πραγματεύεται την δημιουργία ενός λογισμικού μετα-ανάλυσης για δεδομένα γονιδιακής έκφρασης που προέρχονται από μικροσυστοιχίες DNA. Το πρόγραμμα αυτό έχει συνταχθεί στη γλώσσα προγραμματισμού Python (Python 3.7.9) και χρησιμοποιεί αρκετές βιβλιοθήκες που βοηθούν στην εξαγωγή αποτελεσμάτων μιας τέτοιας ανάλυσης. Ο χρήστης με αυτό το πρόγραμμα μπορεί να δώσει τις ρυθμίσεις με τις οποίες επιθυμεί να εκτελεστεί η μετα-ανάλυση είτε μέσω ενός αρχείου txt (tab delimited), είτε μέσω ενός γραφικού περιβάλλοντος (GUI). Το πρόγραμμα διαβάζει τις εργασίες και ένα αρχείο που υποδεικνύει σε ποιες στήλες βρίσκονται τα άτομα ελέγχου και τα άτομα αναφοράς. Στην συνέχεια, πραγματοποιείται η μετα-ανάλυση και ανάλογα με την επιλογή του χρήστη, εμφανίζονται τα αποτελέσματα στην οθόνη είτε αποθηκεύονται σε ένα αρχείο txt. Τέλος, εμφανίζονται διαγράμματα για τους δείκτες της μετα-ανάλυσης που περατώθηκε.

1. Εισαγωγή

1.1 Γονιδιακή Έκφραση

Η γονιδιακή έκφραση ορίζεται ως η διαδικασία κατά την οποία χρησιμοποιούνται τα δεδομένα ενός γονίδιου για τη σύνθεση ενός λειτουργικού γονιδιακού προϊόντος και αναφέρεται κυρίως στην ποσότητα του προϊόντος αυτού που παράγεται σε μια συγκεκριμένη χρονική στιγμή μέσα στο κύτταρο. Τα παραγόμενα αυτά προϊόντα μπορεί να αποτελούν είτε πρωτεΐνες είτε διαφόρων ειδών RNAs (tRNAs, miRNAs, snRNAs κ.α.). Γνωρίζουμε ότι σε συγκεκριμένα κύτταρα ή ιστούς εκφράζονται υποσύνολα γονιδίων από ολόκληρο το γονιδίωμα. Οι διαφορετικές εντάσεις των εκφραζόμενων γονιδίων συνιστούν πολύ καθοριστικές για την λειτουργία (Hayes and Meyerson 2005), την ανάπτυξη, είτε την εκδήλωση κάποιας ασθένειας (Lukk, Kapushesky et al. 2010) ενός συγκεκριμένου κυτταρικού τύπου.

Στην σύγχρονη Μοριακή Βιολογία, η ανάλυση της γονιδιακής έκφρασης έχει μείζονα ρόλο. Υπάρχουν πολλές τεχνικές για την ανάλυση και ποσοτικοποίηση της γενετικής έκφρασης. Αξιοσημείωτες είναι οι εξής:

- PCR (Liang and Pardee 1992)
- Σειριακή ανάλυση της γονιδιακής έκφρασης (serial analysis of gene expression, SAGE) (Velculescu, Zhang et al. 1995)
- PCR πραγματικού χρόνου (Real Time, RT)
- Αλληλούχιση RNA (RNA sequencing, RNA-Seq)
- Του στυπώματος τύπου Northern (Alberts, Wilson et al. 2008)

- Μικροσυστοιχίες (Micro-arrays)

1.2 Μικροσυστοιχίες (micro-arrays)

1.2.1 Κατασκευή των μικροσυστοιχιών

Οι μικροσυστοιχίες (μικροσυστοιχίες DNA, DNA chips) αναπτύχθηκαν για την ανάλυση αλληλουχιών, στα τέλη του 1980. Όμως, όταν παρατηρήθηκε η εφαρμογή τους για την ποσοτικοποίηση της γονιδιακής έκφρασης, άρχισαν να γίνονται όλο και πιο γνωστές. Με τις μικροσυστοιχίες, μπορούμε να αναλύσουμε ταυτόχρονα την έκφραση χιλιάδων γονιδίων σε ένα σύνολο δειγμάτων ή σε διαφορετικά στάδια ανάπτυξης. Μας διευκολύνουν στην σύγκριση της έκφρασης φυσιολογικών και παθολογικών καταστάσεων, στον εντοπισμό γονιδίων που εκφράζονται ή καταστέλλονται όταν επιδρούν διάφοροι παράγοντες στο περιβάλλον και στο να αποφανθούμε αν υπάρχει ανταπόκριση των φαρμακευτικών ουσιών ή των θεραπειών που χορηγούνται σε ασθενείς.

1.2.2 Πειραματική διαδικασία των μικροσυστοιχιών

- Για την διεξαγωγή αυτής της πειραματικής διαδικασίας πρέπει αρχικά να τεθεί το βιολογικό ερώτημα. Το βιολογικό ερώτημα απαντά στο τι επιθυμούμε να πετύχουμε με το πείραμα αυτό και ποια υλικά και μέσα πρέπει να χρησιμοποιηθούν.
- Το επόμενο στάδιο, είναι να επιλέξουμε τον κατάλληλο τύπο τσιπ μικροσυστοιχιών
- Στη συνέχεια, γίνεται η παρασκευή του δείγματος
- Έπειτα, η υβριδοποίηση
- Και τέλος η κανονικοποίηση των δεδομένων

1.2.3 Βάσεις δεδομένων μικροσυστοιχιών

Οι βάσεις βιολογικών δεδομένων αποτελούν αρκετά σημαντικό εργαλείο για τους επιστήμονες τόσο στη καταχώρηση των πειραματικών αποτελεσμάτων όσο και στην ανάκτηση των δεδομένων. Για κάθε πείραμα που είναι καταχωρημένο σε αυτές τις βάσεις δεδομένων, δίνονται πληροφορίες σχετικά με το πείραμα. Πιο συγκεκριμένα, παρέχονται πληροφορίες για το είδος των δεδομένων, για τα άτομα που έλαβαν μέρος στη πειραματική διαδικασία, τα γονίδια που μελετήθηκαν, την πλατφόρμα μικροσυστοιχιών που χρησιμοποιήθηκε για το πείραμα, καθώς και άλλες δευτερεύουσες πληροφορίες.

Οι κυριότερες βάσεις μικροσυστοιχιών που χρησιμοποιούνται ευρέως από την επιστημονική κοινότητα είναι οι :

- Gene Expression Omnibus (GEO) https://www.ncbi.nlm.nih.gov/geo/ (Barrett and Edgar 2006)

- Archive of Functional Genomics Data (ArrayExpress) https://www.ebi.ac.uk/arrayexpress/ (Athar, Fullgrabe et al. 2019)
- ONCOMINE https://www.oncomine.org/resource/login.html (Rhodes, Yu et al. 2004)
- Center for Information Biology Gene Expression database (CIBEX) https://bio.tools/cibex (Ikeo, Ishi-i et al. 2003)
- The Cancer Genome Atlas (TCGA) https://cancergenome.nih.gov/
- Cancer Program Datasets http://portals.broadinstitute.org/cgi-bin/cancer/datasets.cgi.

1.3 Μετα – ανάλυση

Τις τελευταίες δεκαετίες τα βιοϊατρικά δεδομένα αυξάνονται εκθετικά. Αυτός ο ρυθμός αύξησης καθιστά πλέον αναγκαίο τον συνδυασμό και την ανάλυσή τους. Η τεχνική της μετα-ανάλυσης έχει αποδειχτεί ως καλή πρακτική για αυτά τα προβλήματα. Η τεχνική της μετα-ανάλυσης εφαρμόστηκε για πρώτη φορά στη Ψυχολογία στα τέλη της δεκαετίας του ’70. Με αυτήν την τεχνική, συνδυάζονται οι μελέτες που μελετάται ένα κοινό ερώτημα και διερευνάται η κοινή ποικιλομορφία τους (Greenland 1998). Είναι ένα αξιόπιστο εργαλείο για να αξιολογηθούν οι επιδράσεις των μελετώμενων γονιδίων σε σύνθετους φαινοτύπους. Στην ιατρική έρευνα, η μετα-ανάλυση εφαρμόστηκε πρώτη φορά στις τυχαιοποιημένες κλινικές δοκιμές (Sacks, Berrier et al. 1987) και σήμερα αποτελεί αρκετά χρήσιμο εργαλείο για τον συνδυασμό μελετών γενετικής συσχέτισης, επιδημιολογικών μελετών, μελετών φαρμακογονιδιωματικής (Trikalinos, Salanti et al. 2008), και άλλων βιοϊατρικών μελετών.

1.3.1 Πλεονεκτήματα της μετα – ανάλυσης

Η μετα-ανάλυση ως στατιστική μεθοδολογία προσφέρει μια αμερόληπτη σύνθεση πολλών μελετών που αφορούν ένα συγκεκριμένο θέμα, προκειμένου να γίνει η διεξαγωγή ενός αντικειμενικού συνολικού συμπεράσματος.

Επίσης, αξιολογείται η μεροληψία των μελετών, βάσει του πλήθους των μελετών που μας στο συμπέρασμα ύπαρξης μη σημαντικής επίδρασης.

Επιπλέον, ένα σημαντικό στοιχείο της μετα-ανάλυσης είναι ότι επιτρέπει πριν την σύνθεση των αποτελεσμάτων, να δοθεί διαφορετικό βάρος στις μελέτες, το οποίο βασίζεται στο μέγεθος του δείγματός τους.

Τέλος, δίνει την δυνατότητα να υπολογιστεί το μέγεθος επίδρασης, δηλαδή την ποσότητα κατά την οποία η εναλλακτική υπόθεση διαφέρει από την από την μηδενική υπόθεση, για να αναδειχθεί κυρίως η σημασία μια διαπίστωσης η οποία είναι στατιστικά σημαντική (Πανάρετος and Ξεκαλάκη 2000).

1.3.2 Μεθοδολογία μετα – ανάλυσης

Το πρώτο βήμα κατά την μετα – ανάλυση είναι να γίνει μια έρευνα πάνω στην ήδη υπάρχουσα βιβλιογραφία για το υπό μελέτη αντικείμενο (Normand 1999). Αφότου πραγματωθεί λοιπόν αυτό το βήμα και έχει συγκεντρωθεί ένας επαρκής αριθμός μελετών (συνήθως περισσότερες από τρείς μελέτες), καταγράφονται τα δεδομένα που θα λάβουν μέρος στην μετα-ανάλυση. Παρόλα αυτά, χρήζει ιδιαίτερης προσοχής η ανάκτηση και η επεξεργασία των δεδομένων που θα εισαχθούν στην μετα-ανάλυση, καθώς η αξιοπιστία της μετα-ανάλυσης επηρεάζεται από την αξιοπιστία των δεδομένων.

1.3.3 Μέγεθος επίδρασης και έλεγχος ετερογένειας

Βασικό στοιχείο της μετα-ανάλυσης αποτελεί το μέγεθος επίδρασης (effect size), το οποίο μπορεί να είναι είτε η διαφορά των μέσων τιμών, είτε ο συντελεστής συσχέτισης, είτε ο σχετικός λόγος πιθανοτήτων (odds ratio), ο οποίος αποδίδει την πιθανότητα ένα γεγονός να συμβεί προς την πιθανότητα να μη συμβεί. Τα δύο μοντέλα που μπορούν να χρησιμοποιηθούν, είναι το μοντέλο των σταθερών επιδράσεων (fixed effect model) και το μοντέλο τυχαίων επιδράσεων (random effects model).

Στο μοντέλο των τυχαίων επιδράσεων, λαμβάνεται υπόψη η ετερογένεια μεταξύ των μελετών. Στο μοντέλο των σταθερών επιδράσεων δεν λαμβάνεται υπόψη η ετερογένεια. Παρόλα αυτά, τα δεδομένα ενδέχεται να προέρχονται από διαφορετικούς πληθυσμούς, πράγμα που μπορεί να προσδώσει ετερογένεια στη μελέτη και να προκύψουν λάθος συμπεράσματα από τα αποτελέσματα της μετα-ανάλυσης.

1.3.4 Μετα – ανάλυση σε δεδομένα μικροσυστοιχιών

Η μετα-ανάλυση έχει εφαρμογή και στη σύνοψη αποτελεσμάτων από πειράματα μελετών συσχέτισης ολόκληρου του γονιδιώματος (Genome Wide Association Study, GWAS) (Zeggini and Ioannidis 2009) και από μελέτες γονιδιακής έκφρασης μικροσυστοιχιών ή αλληλούχισης νέας γενιάς (Next Generation Sequencing, NGS) (Hong and Breitling 2008). Βέβαια, σε αυτά τα πειράματα, ο αριθμός των δειγμάτων είναι μικρός και στις παρατηρούμενες τιμές υπάρχει πολύ υψηλή ποσότητα θορύβου. Γι' αυτόν ακριβώς τον λόγο έχουν προταθεί από την επιστημονική κοινότητα προηγμένες στατιστικές μέθοδοι για την αποτελεσματική αντιμετώπιση αυτών των προβλημάτων και για έναν πιο εύστοχο υπολογισμό των διαστημάτων εμπιστοσύνης (confidence intervals, C.I) και των p-values.

1.3.5 Μέθοδοι διόρθωσης πολλαπλών συγκρίσεων

Για την εύρεση των γονιδίων που είναι στατιστικώς σημαντικά σε μια μετα-ανάλυση μικροσυστοιχιών, χρειάζεται ένας έλεγχος προκειμένου να υπάρχει βεβαιότητα ότι τα γονίδια που βρέθηκαν ως στατιστικώς σημαντικά είναι πράγματι σημαντικά και επιπλέον ότι δεν απορρίφθηκαν γονίδια τα οποία δεν έπρεπε να απορριφθούν. Για παράδειγμα, αν στην μετα-ανάλυση αναλύονται πάνω από 20.000 γονίδια, η πιθανότητα να βρεθούν γονίδια στατιστικώς σημαντικά είναι μεγάλη, ακόμα και αν επιλεχθεί ένα αυστηρό επίπεδο σημαντικότητας (πχ. 0.01), λόγω τύχης.

Με τις μεθόδους διόρθωσης πολλαπλών συγκρίσεων είναι εφικτό να βρεθούν τα γονίδια που είναι όντως στατιστικά σημαντικά σε μια μετα-ανάλυση μικροσυστοιχιών. Αυτές οι μέθοδοι διακρίνονται στις μεθόδους που ελέγχουν το ποσοστό σφάλματος FDR και σε αυτές που ελέγχουν το ποσοστό σφάλματος FWER (family – wise error rate). Οι μέθοδοι διόρθωσης πολλαπλών συγκρίσεων δέχονται ως είσοδο τα p-values που έχουν προκύψει από την μετα-ανάλυση και επιστρέφουν τα διορθωμένα p-values για κάθε γονίδιο της λίστας εισόδου.

1.3.6 Μέθοδοι αναδειγματοληψίας (Resampling Methods)

Όπως αναφέρθηκε, τα πειράματα μικροσυστοιχιών συνήθως δεν διαθέτουν μεγάλο μέγεθος και οι τιμές έκφρασης των γονιδίων δεν ακολουθούν την κανονική κατανομή. Για αυτόν ακριβώς τον λόγο, ο έλεγχος βασισμένος στο t-test ακολουθείται αρκετές φορές από μια μέθοδο αναδειγματοληψίας όπως το Bootstrap (Efron 1982, Efron and Tibshirani 1993), που θα αναλυθεί σε επόμενη ενότητα.

Κεφάλαιο 2ο
Υλικά και μέθοδοι

2.1 Ανάλυση δεδομένων μικροσυστοιχιών

2.1.1 Συλλογή Δεδομένων

Σε αυτό το σημείο, θα ήθελα να ευχαριστήσω την κα Βέννου Κωνσταντίνα, μεταδιδακτορική ερευνήτρια στο τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική του Πανεπιστημίου Θεσσαλίας, για την παραχώρηση των συνόλων δεδομένων που χρησιμοποίησε για την διεκπεραίωση της διδακτορικής της διατριβής.

Τα δεδομένα που χρησιμοποιήθηκαν για τις ανάγκες αυτής της πτυχιακής εργασίας, προήλθαν έπειτα από αναζήτηση στην βιολογική βάση δεδομένων GEO, η οποία όπως αναφέρθηκε περιέχει αποτελέσματα πειραμάτων μικροσυστοιχιών. Για την ανάκτηση των συνόλων δεδομένων έγινε η χρήση σύνθετων ερωτημάτων (queries), προκειμένου να βρεθούν πειράματα μικροσυστοιχιών που αφορούσαν ανθρώπινα γονίδια.

Σε κάθε σύνολο δεδομένων που χρησιμοποιήθηκε, η πρώτη στήλη αφορά το όνομα των γονιδίων και οι υπόλοιπες στήλες αφορούν τις κωδικές ονομασίες των ανθρώπων που έλαβαν μέρος στην πειραματική μελέτη. Αυτό πρακτικά σημαίνει ότι σε κάθε σειρά βλέπουμε την τιμή της έκφρασης ενός γονίδιου σε κάθε συμμετέχοντα της μελέτης.

GENE_SYMBOL	GSM106249	GSM106250	GSM106274	GSM106275	GSM106276	GSM106277	GSM106278	GSM106279
A1BG	0.07175	-0.00749	0.06815	0.063	0.00335	0.0205	0.00099	0.03792
A1BG-AS1	-0.0637	0.182	-0.148	-0.176	-0.28	-0.174	-0.0894	-0.333
A1CF	0.0927	0.117	-0.0838	-0.00781	-0.0689	-0.0132	-0.0774	-0.617
A2M	-0.00156	0.0791	0.0443	-0.0961	-0.251	-0.279	-0.15	-0.217
A2ML1	0.0367	-0.0748	0.0633	0.138	0.104	0.039	0.0184	0.0108
A4GALT	-0.253	0.0246	-0.0369	0.0294	-0.205	-0.115	-0.142	-0.213
A4GNT	0.0103	-0.174	-0.000934	0.168	0.216	0.0965	0.0852	0.115
AAAS	-0.0785	-0.0139	-0.0564	-0.0718	-0.0997	-0.0892	-0.0491	-0.116
AACS	-0.0277	0.0988	-0.0469	-0.159	0.12	0.139	0.0441	0.0506
AADAC	-0.0393	-0.0871	0.163	-0.0743	-0.0301	0.0784	-0.0819	-0.0669
AADACL2	0.0327	0.0431	0.0815	0.0727	0.0769	0.0991	0.092	0.088
AADAT	0.0761	0.0525	-0.079	0.199	0.0124	-0.0215	0.0822	0.104
AAED1	0.234	0.109	-0.0757	0.0351	-0.118	0.032	0.128	0.231
AAGAB	0.0257	-0.0424	0.0194	0.0477	-0.0523	-0.0461	-0.0164	-0.0173
AAK1	0.0042	-0.0449	-0.054615	0.037285	-0.05199	-0.0425	-0.07735	-0.07515
AAMDC	0.0272	0.191545	0.14435	0.1005	0.049705	-0.1145	-0.0314	-0.11945
AAMP	-0.3	0.0348	-0.0539	-0.164	-0.135	0.101	-0.0386	-0.022
AANAT	0.0101	0.0243	-0.0483	-0.0311	-0.0408	0.0787	-0.00321	-0.0423
AAR2	-0.0205	0.0112	-0.000208	-0.0299	0.00483	0.00299	-0.00266	-0.0296
AARS	-0.119	0.129	-0.0697	-0.0955	-0.0713	0.0496	-0.0377	-0.0699
AARS2	0.017	0.0391	0.045	-0.0433	-0.0247	0.0203	-0.0152	-0.0218
AARSD1	-0.0195	-0.057	-0.1092	-0.3725	0.0485	0.13325	0.10925	0.0787
AASDH	0.027233	-0.008567	-0.020667	-0.059267	-0.0854	-0.02298	-0.007567	0.0454
AASDHPPT	-0.077483	0.0027	0.050263	-0.113967	-0.011233	0.023933	0.039433	0.0971
AASS	0.0282	0.159	-0.0444	0.0921	0.0725	0.0339	-0.0384	0.054
AATF	-0.194	0.05195	-0.0706	-0.1425	0.1216	0.1615	0.1068	0.06875
AATK	-0.0705	0.038533	0.05957	0.09471	-0.109333	-0.076233	-0.0568	-0.0691
ABAT	0.10305	0.11255	0.115	0.054	-0.0238	0.05977	-0.0672	0.02175
ABCA1	-0.024275	0.054984	-0.054917	-0.0363	0.028134	0.039584	0.015833	0.021916
ABCA10	-0.0837	-0.106	0.141	-0.198	-0.144	-0.274	-0.163	-0.0815
ABCA11P	-0.0758	-0.119	0.183	-0.173	-0.00636	-0.155	-0.0257	-0.0229
ABCA12	-0.0942	-0.118	0.226	0	0.463	0.432	-0.0696	0.288
ABCA13	0.0587	-0.09	0.0208	-0.349	-0.0442	0.35	-0.0435	0.0918
ABCA2	-0.0597	0.0314	-0.04	0.0327	-0.175	-0.121	-0.175	-0.24
ABCA3	0.0978	0.16	0.0252	0.0286	-0.188	-0.0438	-0.0528	-0.117
ABCA4	0.0202	-0.246	-0.0184	0.239	-0.105	-0.233	0.0339	-0.0801
ABCA5	0.0775	0.005	0.128	0.1142	0.1695	-0.122	-0.073	-0.028595

Εικόνα 1 *Ένα στιγμιότυπο του πίνακα γονιδιακής έκφρασης, από μια μελέτη μικροσυστοιχιών.*

2.1.2 Στατιστικό Μοντέλο

Στη περίπτωση της γονιδιακής έκφρασης τα δεδομένα είναι συνεχή. Στα συνεχή δεδομένα, το μέγεθος επίδρασης (effect size) θ ορίζεται ως η διαφορά των μέσων τιμών των δειγμάτων αναφοράς (cases) ως προς των δειγμάτων ελέγχου (controls) (Normand 1999). Η διαφορά αυτή μπορεί να είναι είτε τυποποιημένη (standardized mean difference) είτε μη τυποποιημένη (un standardized mean difference). Στο λογισμικό που συντάχθηκε η μετα-ανάλυση γίνεται βάσει της τυποποιημένης διαφοράς. Το μέγεθος επίδρασης αποτελεί δηλαδή τον παράγοντα με τον οποίο θα πραγματοποιηθεί η μετα-ανάλυση και η διακύμανση (s_i^2) της κάθε μελέτης.

Η τυποποιημένη διαφορά μέσων τιμών δίνεται από τον ακόλουθο τύπο

$$d_i = \frac{\bar{X}_{1i} - \bar{X}_{2i}}{Sp_i} \qquad (1)$$

και η θεωρούμενη κοινή τυπική απόκλιση (pooled standard deviation) δίνεται από τον τύπο

$$Sp_i = \sqrt{\frac{(n1_i - 1)S1_i{}^2 + (n2_i - 1)S2_i{}^2}{n1_i + n2_i - 2}} \qquad (2)$$

όπου $\bar{X}_{1i}$ η μέση τιμή του δείγματος ελέγχου και $\bar{X}_{2i}$ η μέση τιμή του δείγματος αναφοράς, $n1_i$ και $n2_i$ το μέγεθος δείγματος κάθε ομάδας και $S1_i$ και $S2_i$ η τυπική απόκλιση της γονιδιακής έκφρασης κάθε ομάδας.

2.1.3 Διόρθωση μεγέθους επίδρασης

Η εκτίμηση της τυποποιημένης διαφοράς των μέσων τιμών αναφέρεται ως d του Cohen (Cohen's d , Cohen 1988). Λόγω του μικρού μεγέθους του δείγματος τιμών και της μη κανονικής κατανομής των τιμών που χαρακτηρίζει τις περισσότερες μελέτες μικροσυστοιχιών, οι εκτιμήσεις του d υπερεκτιμούν την απόλυτη τιμή. Αρκετές στατιστικές μέθοδοι έχουν προταθεί από την επιστημονική κοινότητα για την αποτελεσματική αντιμετώπιση αυτών των προβλημάτων. Οι δύο μέθοδοι που χρησιμοποιήθηκαν στο προγραμματιστικό μέρος της εργασίας για μια πιο ακριβή εκτίμηση, ήταν η διόρθωση του Hedges και η μέθοδος αναδειγματοληψίας Bootstrap που απαιτεί μεγάλο υπολογιστικό χρόνο.

Η διόρθωση g του Hedges διορθώνει την μεροληψία του d δίνοντας μια πιο αμερόληπτη εκτίμηση μετατρέποντας το d se g. Αυτό επιτυγχάνεται με την χρήση του συντελεστή J όπως φαίνεται στον ακόλουθο τύπο.

$$g_i = Jd_i \ (3)$$

Ο συντελεστής g συναντάται στη βιβλιογραφία με προσεγγιστικούς τύπους , με πιο γνωστό τον $J(v_i) \approx \frac{3}{4v_i-1}$ (4), όπου $v_i = n_{1_i} + n_{2_i} - 2$, αλλά ο πιο ακριβής τύπος είναι ο ακόλουθος :

$$J(v_i) = \frac{\Gamma(\frac{v_i}{2})}{\sqrt{\frac{v_i}{2}}\,\Gamma(\frac{v_{i-1}}{2})} \qquad (5)$$

Όταν γίνεται η διόρθωση του Hedges η διακύμανση θα αλλάξει επίσης σε :

$$var\,(g_i) = s_i^2 = (J\,(v_i))^2\; var\,(d_i) \quad (6)$$

2.1.4 Μοντέλο τυχαίων επιδράσεων (Random effects model)

Σε αυτό το μοντέλο επιδράσεων θεωρείται ότι τα δείγματα της μελέτης προέρχονται από πληθυσμούς που υπάρχει ετερογένεια. Η κάθε ανεξάρτητη μελέτη έχει ένα συγκεκριμένο μέγεθος επίδρασης θ_i και μια τιμή διακύμανσης s_i^2. Το κάθε μέγεθος επίδρασης είναι δείγμα που προέρχεται από ένα σύνολο τιμών εκτιμητών με μέση τιμή θ και διακύμανση τ^2. Παρατηρείται λοιπόν ότι σε αυτό το μοντέλο εισάγεται μια νέα μεταβλητή τ^2 , η οποία είναι η παρατηρούμενη ετερογένεια ανάμεσα στις μελέτες. Το μοντέλο των τυχαίων επιδράσεων ακολουθεί τον παρακάτω τύπο.

$$Y_i \sim N\,(\theta_i, s_i^2) \quad (7) \quad και \quad \theta_i \sim N(\theta,\tau^2) \quad (8)\; για \quad i=1,2,\ldots k$$

2.1.5 Έλεγχός ετερογένειας

Στο μοντέλο των τυχαίων επιδράσεων λαμβάνεται υπόψη και η ετερογένεια μεταξύ των μελετών.

Από τον έλεγχο *Q* του Cochran (Cochrane's *Q* test), μπορεί να προσδιοριστεί η ετερογένεια μεταξύ των μελετών. Το *Q* υπολογίζεται από τον ακόλουθο τύπο.

$$Q = \sum_{i=1}^{k} W_i\,(Y_i - D)^2 \quad (9)$$

$$\text{με } D = \frac{\sum_{i=1}^{k} w_i\; d_i}{\sum_{i=1}^{k} w_i} \quad (10) \qquad \text{και} \qquad w_i = \frac{1}{var(d_i)} \quad (11)$$

Το Q ακολουθεί κατανομή (x^2) με *k*-1 βαθμούς ελευθερίας, όπου *k* είναι το πλήθος των μελετών. Ο έλεγχος Q του Cochran είναι πιο αποδοτικός όταν εισαχθούν στην μετα-ανάλυση πολλές μελέτες (Villanueva and Zavarsek 2004).

Για τον έλεγχο της ετερογένειας, χρησιμοποιείται επίσης και ο έλεγχος I^2, ο οποίος κάνει χρήση του ελέγχου Q. Το I^2 υπολογίζεται από τον τύπο:

$$I^2 = \max\left(0, \frac{Q-(k-1)}{Q}\right) x\ 100\ \% \qquad (12)$$

Το I^2 λαμβάνει τιμές από 0% έως 100%. Αν το I^2 λάβει τιμές μικρότερες του 25% , η ετερογένεια θεωρείται αμελητέα, ενώ αν λάβει τιμές μεγαλύτερες του 50%, τότε η ετερογένεια χαρακτηρίζεται υψηλή και αποτελεί πρόβλημα για την μετα-ανάλυση.

Τέλος, ένας ακόμη έλεγχος της ετερογένειας είναι ο τ^2. Ο εκτιμητής αυτός υπολογίζει τη μεταβλητότητα μεταξύ των μελετών. Ο εκτιμητής τ^2 ορίστηκε αρχικά από τους DerSimonian και Laird με τον παρακάτω τύπο :

$$\tau^2 = \frac{Q-(k-1)}{\sum W_i - \frac{\sum {w_i}^2}{\sum w_i}} \qquad (13)$$

Όταν αυτοί οι δείκτες ετερογένειας έχουν μεγάλες τιμές, τότε φανερώνουν ότι υπάρχει ετερογένεια στις μελέτες, ενώ όταν είναι κοντά στο μηδέν συμπεραίνεται ότι δεν υπάρχει ετερογένεια μεταξύ των μελετών.

Οπότε, όταν διερευνάται η ύπαρξη ετερογένειας, εξετάζονται οι ακόλουθες υποθέσεις:

- Ως μηδενική υπόθεση (H_0) ορίζεται η υπόθεση ότι τα δείγματα είναι ομοιογενή.

- Ενώ ως εναλλακτική υπόθεση (H_1) ορίζεται η υπόθεση ότι τα δείγματα δεν είναι ομοιογενή, είναι δηλαδή ανομοιογενή.

2.1.6 Μέθοδοι διόρθωσης πολλαπλών ελέγχων

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, οι μέθοδοι διόρθωσης πολλαπλών συγκρίσεων χρησιμοποιούνται για να βρεθούν τα γονίδια που είναι στατιστικώς σημαντικά.

Οι μέθοδοι αυτοί λαμβάνουν ως είσοδο μια λίστα με p-values που έχουν προκύψει από την μετα-ανάλυση και στην συνέχεια επιστρέφουν τα p-values διορθωμένα. Οι μέθοδοι διόρθωσης πολλαπλών ελέγχων διακρίνονται σε δυο κατηγορίες , τις μεθόδους που ελέγχουν το ποσοστό σφάλματος FWER και τις μεθόδους που ελέγχουν το ποσοστό σφάλματος FDR. Η πιο γνωστή μέθοδος διόρθωσης είναι η μέθοδος Bonferroni, στην οποία το επίπεδο σημαντικότητας υπολογίζεται διαιρώντας το FWER με το πλήθος των αριθμών ελέγχων (με το πλήθος των μοναδικών γονιδίων που συμμετέχουν στην μετα-ανάλυση). Επομένως, η διορθωμένη p-value δίνεται από τον τύπο :

$$p_{cor(i)} = \frac{p_{(i)}}{n} \qquad (14)$$

Αν η διορθωμένη τιμή ενός γονιδίου βρίσκεται κάτω από το επίπεδο σημαντικότητας που έχει τεθεί (πχ. $p_{cor(i)} < 0.05$), το γονίδιο αυτό θα επισημανθεί ως στατιστικά σημαντικό.

Υπάρχουν κι άλλες μέθοδοι διόρθωσης FWER, όπως η Bonferroni Step-Down ή Holm (Holm 1979), η διόρθωση Holland,(Holland and Copenhaver 1987) και η Sidak (Šidák 1967).

Με τις μεθόδους ελέγχου FDR, επιτυγχάνεται η μείωση του σφάλματος τύπου II και έχουν μεγαλύτερη στατιστική ισχύ. Όμως, το κόστος αυτών μεθόδων είναι η αύξηση του σφάλματος τύπου I. Με τις μεθόδους FDR , ταξινομείται η λίστα τον p-values των γονιδίων σε αύξουσα σειρά. Η μεγαλύτερη τιμή p-value τιμή μένει ίδια, ενώ οι υπόλοιπες διορθώνονται με τον τύπο των Benjamini – Hochberg :

$$p_{cor(i)} = \frac{i}{n} p_{(i)} \quad (15)$$

2.1.7 Μέθοδοι αναδειγματοληψίας

Όπως έχει ήδη αναφερθεί στο προηγούμενο κεφαλαίο, το μικρό μέγεθος και η μη κανονική κατανομή στις τιμές έκφρασης των πειραμάτων μικροσυστοιχιών αποτελούν πρόβλημα. Για να αντιμετωπιστεί λοιπόν αυτό το πρόβλημα, ακολουθείται ο έλεγχος βασισμένος στο t-test επαναληπτικά από μια μέθοδο αναδειγματοληψίας.

Η μέθοδος αναδειγματοληψίας που χρησιμοποιείται σε αυτά τα πειράματα, είναι η μέθοδος Bootstrap. Κατά αυτή την μέθοδο, δημιουργούνται νέα δείγματα με την μέθοδο της τυχαίας επαναληπτικής επανάθεσης. Στην περίπτωση των πειραμάτων μικροσυστοιχιών, οι στήλες που καταγράφουν τις τιμές έκφρασης σε κάθε άτομο που συμμετείχε στο πείραμα, επιλέγονται σε κάθε βήμα επανάληψης τυχαία. Όταν τελειώσουν τα βήματα επανάληψης, δημιουργείται ένα νέο τεχνητό σύνολο δεδομένων ίδιου μεγέθους με το αρχικό. Από αυτό το σύνολο δεδομένων που θα προκύψει, υπολογίζεται πιο αξιόπιστα το τυπικό σφάλμα και το διάστημα εμπιστοσύνης και γίνεται η μετα-ανάλυση. Ως προς το πλήθος των επαναλήψεων που απαιτούνται από για την εκτίμηση των τυπικών σφαλμάτων, οι 1000 επαναλήψεις παράγουν εκτιμήσεις με μεγάλη ακρίβεια, αλλά έχει παρατηρηθεί επίσης ότι από 200 έως και 500 επαναλήψεις λαμβάνονται αρκετά αξιόπιστες εκτιμήσεις.

2.2 Στατιστικά πακέτα μετα-ανάλυσης

Σε αυτή την ενότητα, θα αναφερθούν τα στατιστικά πακέτα μετα-ανάλυσης που χρησιμοποιούνται ευρέως από την επιστημονική κοινότητα, καθώς και το πακέτο που χρησιμοποιήθηκε για την υλοποίηση του προγραμματιστικού μέρους της πτυχιακής αυτής.

2.2.1 R (meta, MAMA, metaMA, GeneMeta)

Η R είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για την στατιστική ανάλυση των δεδομένων, τόσο λόγω της ευκολίας που έχει στην σύνταξη των εντολών, όσο και λόγω της πληθώρας των βιβλιοθηκών που περιέχει.

Η R περιέχει αρκετά πακέτα μετα-ανάλυσης, με πιο γνωστό το πακέτο meta (Schwarzer, 2007; Balduzzi et al., 2019). Το πακέτο αυτό είναι και το πιο γενικό, καθώς δίνει στον χρήστη πάρα πολλές επιλογές για την μετα-ανάλυση στην επιλογή του μοντέλου των επιδράσεων, στα παραγόμενα διαγράμματα κλπ. Για δεδομένα που προέρχονται από μικροσυστοιχίες, τα *πακέτα MAMA, metaMA* και *GeneMeta* καλύπτουν σχεδόν όλες τις προαναφερθείσες μεθοδολογίες. Τα πακέτα metaMA και MAMA πέρα από την πληθώρα επιλογών που έχουν στις μεθόδους μετα-ανάλυσης, προσφέρουν την δυνατότητα χρήσης μεθόδων διόρθωσης πολλαπλών συγκρίσεων και την δυνατότητα παραγωγής διαγραμμάτων Venn και την περεταίρω ανάλυσή τους. Το πακέτο GeneMeta δίνει έμφαση στον συνδυασμό συνόλων δεδομένων σε συγκρίσεις δύο δειγμάτων, εστιάζοντας περισσότερο στον έλεγχο t (t-test) (Gentleman, Ruschhaupt et al. 2008).

2.2.2 Stata

Το Stata είναι ένα στατιστικό λογισμικό που χρησιμοποιείται σε διάφορους τομείς έρευνας όπως της οικονομίας, της βιοστατιστικής, της επιδημιολογίας κ.α. Παρόλο που το Stata δεν προσφέρει κάποια έτοιμη εντολή για μετα-ανάλυση σε δεδομένα που προέρχονται από μικροσυστοιχίες, είναι εφικτό να δημιουργηθεί ένας κώδικας που να διεξάγει μια τέτοια δοκιμασία επιτυχώς, όπως και υλοποίησε η ομάδα του εργαστηρίου Μοριακής και Υπολογιστικής Βιολογίας και Γενετικής, με την χρήση του μοντέλου τυχαίων επιδράσεων, με μέγεθος επίδρασης το g του Hedges και με την χρήση της μεθόδου επαναδειγματοληψίας Bootstrap.

2.2.3 Python (PyMARE, statsmodels.stats.meta_analysis, PythonMeta)

Η Python είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται κατά κόρον για την ανάλυση δεδομένων. Η ευκολία στην σύνταξη των εντολών της και το μεγάλο μέγεθος που έχει λάβει η κοινότητα της τα τελευταία χρόνια, αποτέλεσαν κινητήριες δυνάμεις για να δημιουργηθούν βιβλιοθήκες, σχεδόν για κάθε πτυχή της ανάλυσης δεδομένων σε πολύ μικρό διάστημα.

Οι βιβλιοθήκες PyMARE, statsmodels.stats.meta_analysis και PythonMeta καλύπτουν πλήρως όλες τις δυνατότητες που ικανοποιούν και οι αντίστοιχες βιβλιοθήκες της R που αναφέραμε. Ωστόσο, στην παρούσα πτυχιακή εργασία έγινε η χρήση της βιβλιοθήκης PythonMeta, λόγω της πληθώρας των επιλογών που είχε σε μεθόδους μετα-ανάλυσης και για τα παραγόμενα διαγράμματα της μετα-ανάλυσης. Για να ικανοποιηθεί ο σκοπός της εργασίας, χρειάστηκε να τροποποιηθεί ένα σημείο στην βιβλιοθήκη της PythonMeta, ώστε τα μεγέθη επίδρασης να υπολογίζονται βάσει της διόρθωσης g του Hedges με τον πιο ακριβή τύπο (τύπος 5), έναντι του προσεγγιστικού που υλοποιούσε η βιβλιοθήκη.

2.3 Χρήση βιβλιοθηκών της Python

Εκτός από την PythonMeta, χρησιμοποιήθηκαν και άλλες βιβλιοθήκες της Python. Η κάθε βιβλιοθήκη βοήθησε σε διάφορα στάδια του αλγορίθμου του προγράμματος.

2.3.1 Pandas

Είναι μια βιβλιοθήκη για χειρισμό και ανάλυση δεδομένων. Συγκεκριμένα, προσφέρει πληθώρα δομών δεδομένων και έχει πολλές συναρτήσεις για τον χειρισμό αριθμητικών πινάκων.

2.3.2 NumPy

Είναι μια βιβλιοθήκη που τρέχει πίσω από την *Pandas* και προσφέρει πολλές δυνατότητες στην επεξεργασία πολυδιάστατων συστοιχιών και πινάκων. Επίσης υλοποιεί μαθηματικές συναρτήσεις που καθιστούν την επεξεργασία των πινάκων εξαιρετικά απλή.

2.3.3 statistics

Η βιβλιοθήκη αυτή υλοποιεί διάφορα στατιστικά μέτρα και συναρτήσεις για πίνακες και λίστες

2.3.4 SciPy

Με αυτή την βιβλιοθήκη μπορούν να εφαρμοστούν συναρτήσεις για οπτικοποίηση, στατιστική, επεξεργασία εικόνας και για τεχνητή νοημοσύνη. Κατά την συγγραφή του προγράμματος, χρησιμοποιήθηκαν κάποιες στατιστικές συναρτήσεις από αυτή την βιβλιοθήκη.

2.3.5 Matplotlib

Είναι μια βιβλιοθήκη απεικόνισης δεδομένων. Παρέχει στον χρήστη την δυνατότητα να επεξεργαστεί περαιτέρω τα παραγόμενα γραφήματα, μορφοποιώντας διάφορα στοιχεία τους (πχ. υπομνήματα, ετικέτες κλπ.)

2.3.6 Seaborn

Είναι και αυτή μια βιβλιοθήκη απεικόνισης δεδομένων η οποία πρακτικά συμπληρώνει την *Matplotlib,* καθώς περιέχει περισσότερα διαγράμματα και επιλογές απεικόνισης.

Κεφάλαιο 3ο
Αποτελέσματα

3.1 Εκτέλεση του προγράμματος

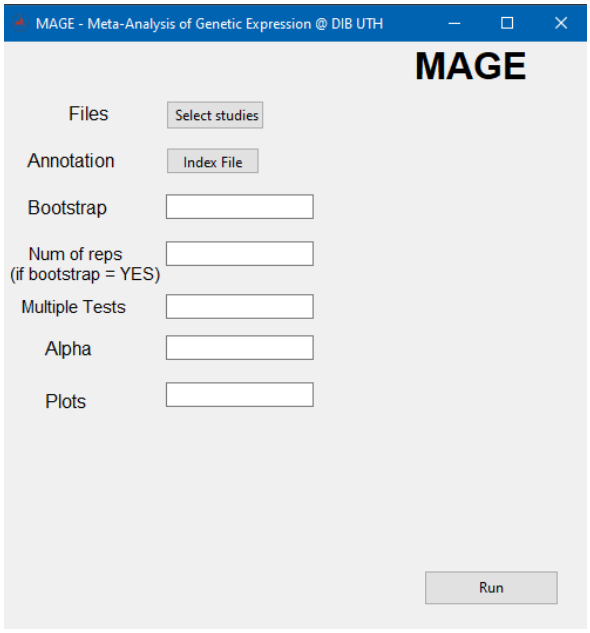
Βάσει όλων των παραπάνω, συντάχθηκε ένα πρόγραμμα σε Python που υλοποιεί όλες τις μεθοδολογίες που αναφέραμε. Το πρόγραμμα που συνοδεύει την παρούσα πτυχιακή εργασία, δίνει στον χρήστη την επιλογή να το εκτελέσει είτε μέσω ενός αρχείου κειμένου, είτε μέσω ενός γραφικού περιβάλλοντος (GUI) που περιλαμβάνει τις ρυθμίσεις εκτέλεσης της μετα-ανάλυσης.

3.1.1 Εισαγωγή αρχείων

Οι ενότητες *Files* και *Annotation* αφορούν τις εργασίες που θέλουμε να εισάγουμε και το αρχείο που περιέχει τις θέσεις των ατόμων ελέγχου και ατόμων αναφοράς,. Τα αρχεία πρέπει να αποτελούνται από στοιχεία που είναι tab delimited για να τα δεχτεί το πρόγραμμα. Αν ο χρήστης επιλέξει να εκτελέσει το πρόγραμμα μέσω του αρχείου ρυθμίσεων, θα πρέπει να εισάγει τα πλήρη μονοπάτια των αρχείων. Με το γραφικό περιβάλλον η διαδικασία εισαγωγής αρχείων είναι πιο απλή, πατώντας τα αντίστοιχα κουμπιά πάνω στην φόρμα.

```
1 Files   study1.txt,study2.txt,study3.txt,study4.txt,study5.txt
2 Annotation  "mage_teams.txt"
3 Bootstrap   NO
4 Num of reps   0
5 Multiple Comparisons    one_step_methods
6 Level of Significance of Mulitple Comparisons   0.05
7 Output to file  NO
8 Output filename mage_results.txt
9 Plots   YES
```

Εικόνα 2- Ρυθμίσεις αρχείου



Εικόνα 3- Το γραφικό περιβάλλον του προγράμματος

3.1.2 Εκτέλεση μετα-ανάλυσης

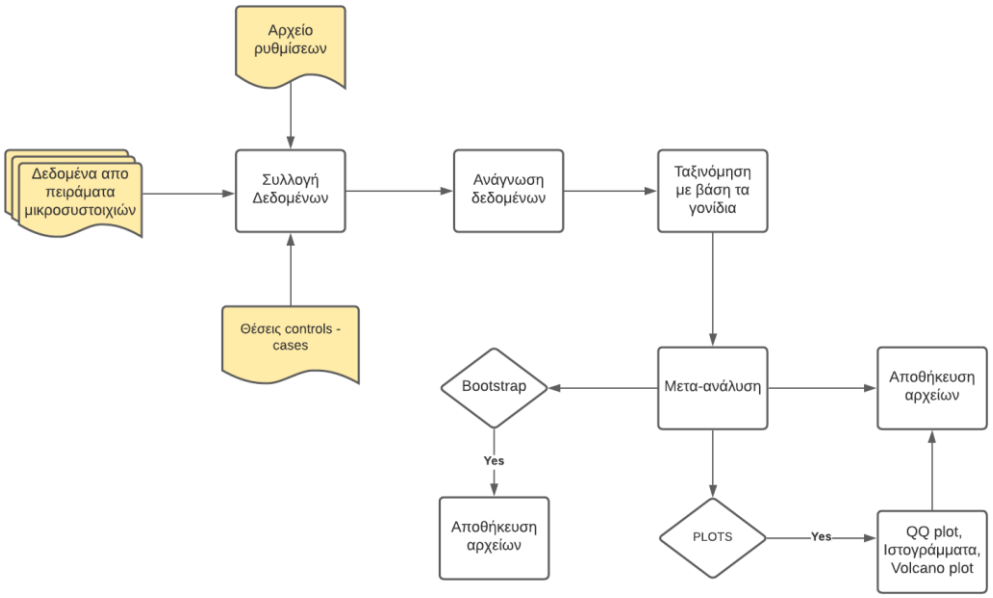
Για να γίνει η εκτέλεση της απλής μετα-ανάλυσης, θα πρέπει στα πεδία *Bootstrap* και *Num of reps* να εισαχθούν οι τιμές *'NO'* και *'0'* αντίστοιχα. Στη συνέχεια, ο χρήστης μπορεί να επιλέξει ποια μέθοδος πολλαπλών ελέγχων θέλει να συνοδέψει τα τελικά αποτελέσματα (*'one_step','step_down','step_up'* ή *'-'* για να συμπεριληφθούν όλες) και με ποιο επίπεδο σημαντικότητας (0.05 ή 0.01). Αν ο χρήστης επιθυμεί να βγουν διαγράμματα μπορεί να πληκτρολογήσει στο πεδίο *Plots 'YES'* ή *'NO'*. Τέλος, πατώντας το κουμπί *Run*, γίνεται η μετα-ανάλυση και βγαίνουν τα αποτελέσματα.

3.1.2 Εκτέλεση **Bootstrap** *μετα-ανάλυσης*

Η ενότητα Bootstrap παίρνει τιμές 'YES' ή 'NO'. Αν επιλεχθεί η επιλογή 'YES' από τον χρήστη, τότε στην ενότητα Num of reps πρέπει να εισαχθεί ένας ακέραιος αριθμός για το πλήθος των επαναλήψεων που θα εκτελεστεί η μέθοδος Bootstrap (200, 500 ή 1000 επαναλήψεις).

3.1.3 Αποτελέσματα από την εκτέλεση

Μετά το πέρας των μετα-αναλύσεων, τα αποτελέσματα τυπώνονται είτε στην οθόνη, είτε αποθηκεύονται σε αρχείο κειμένου, ανάλογα με την επιλογή του χρήστη. Τα αποτελέσματα που τυπώνονται, είναι το διορθωμένο μέγεθος επίδρασης *g* του γονιδίου, το τυπικό σφάλμα, τους δείκτες ετερογένειας *Q*, I^2,Tau^2 και φυσικά, οι τιμές *z* και *p-value* των γονιδίων. Τα δεδομένα τυπώνονται ως *Pandas Data Frames*, τα οποία είναι πιο κοντά στην μορφή του πίνακα δεδομένων. Στο παράρτημα της εργασίας είναι καταχωρημένα δύο στιγμιότυπα από τα αποτελέσματα των αναλύσεων (μετα-ανάλυση και bootstrap μετα-ανάλυση).



Εικόνα 4 Διάγραμμα ροής του προγράμματος

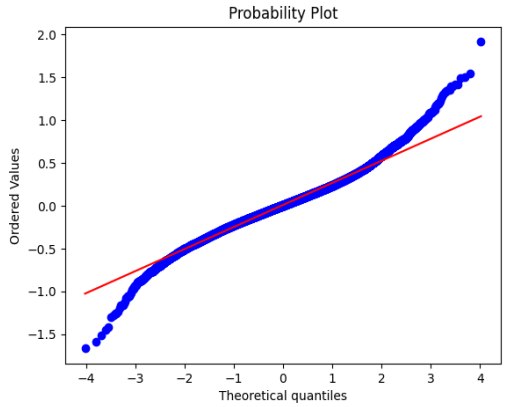
3.1.4 Διαγράμματα (Plots)

Τα διαγράμματα που παράγονται, εφόσον το επιλέξει ο χρήστης, είναι τα ακόλουθα:

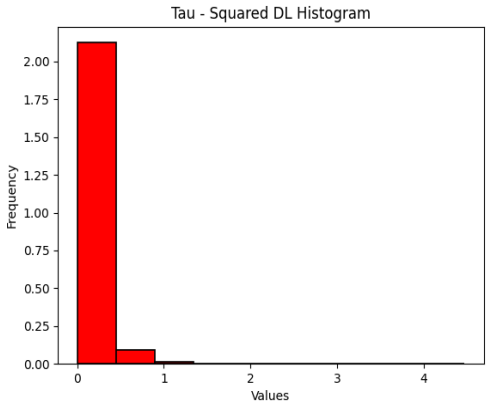
- *Q-Q plot ή Probability Plot:* Σε αυτά τα διαγράμματα απεικονίζονται οι κατανομές πιθανότητας ανάμεσα στα μεγέθη επίδρασης των γονίδιων, ως ένα διάγραμμα διασποράς. Με αυτό το διάγραμμα απεικονίζεται η κατανομή πιθανότητας των μεγεθών επίδρασης σε σχέση με την πιθανότητα κατανομής μιας θεωρητικής κατανομής.

- *Ιστογράμματα :* Στο πρόγραμμα παράγονται ιστογράμματα για τους δείκτες ετερογένειας της μετα-ανάλυσης *Q*, I^2 και Tau^2.

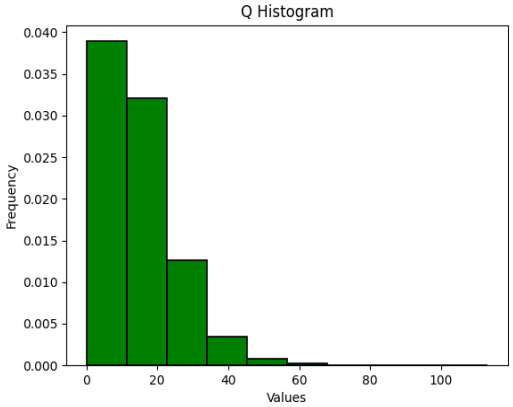
- *Διαγράμματα Ηφαιστείου (Volcano Plots) :* Είναι ένα διάγραμμα διασποράς ανάμεσα στα μεγέθη επίδρασης και στον αρνητικό δεκαδικό λογάριθμο των p-values [$-\log_{10}$(p -values)]. Τα στατιστικώς σημαντικά γονίδια απεικονίζονται με κόκκινο χρώμα, ενώ τα μη στατιστικώς σημαντικά με πράσινο (ανάλογα με το επίπεδο σημαντικότητας *Alpha* που θα επιλεχθεί).



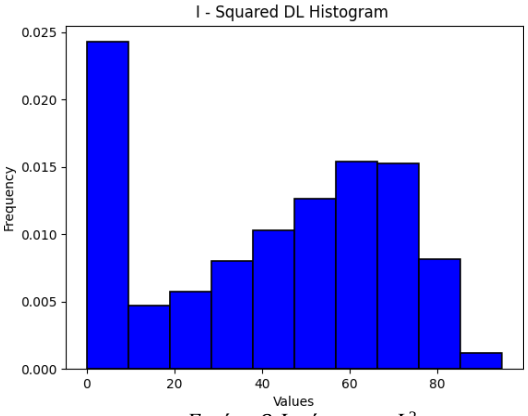
Εικόνα 5 QQ plot



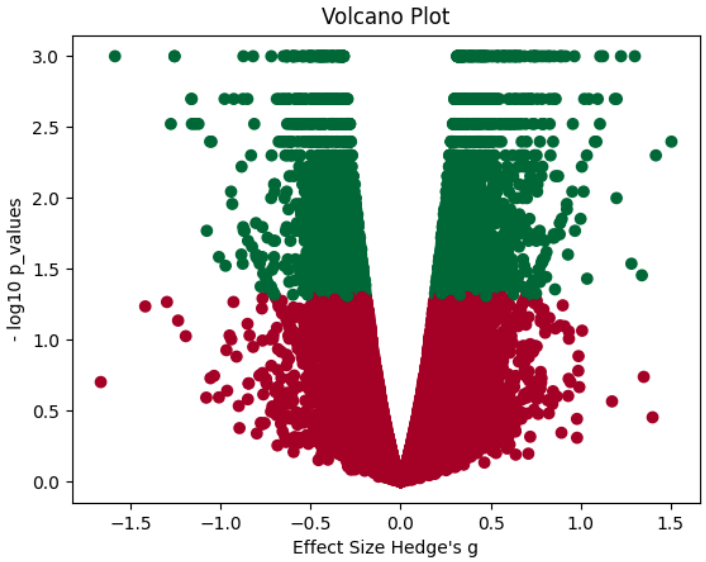
Εικόνα 6 Ιστόγραμμα Tau^2



Εικόνα 7 Ιστόγραμμα Q



Εικόνα 8 Ιστόγραμμα I^2



Εικόνα 9 - Διάγραμμα Ηφαιστείου (Volcano plot)

3.2 Χρόνος εκτέλεσης

Οι χρόνοι που καταγράφονται στον πίνακα που ακολουθεί υπολογίστηκαν σε προσωπικό φορητό υπολογιστή (laptop PC) με τα εξής τεχνικά χαρακτηριστικά :

Λειτουργικό σύστημα (OS) : Windows 10
Επεξεργαστής (CPU) : Intel i7, 7th Gen. (7500U), 2.70 GHz και 2.90 GHz
Μνήμη RAM : 16 GB
Μνήμη ROM : 256 GB (SSD)
Κάρτα γραφικών (GPU): AMD Radeon 530

3.2.1 Σύγκριση με αντίστοιχο πρόγραμμα στο STATA

Για τις ίδιες αναλύσεις είχε γραφτεί και ένας κώδικας στο STATA, στα πλαίσια της διδακτορικής διατριβής της κυρίας Κωνσταντίνας Βέννου. Με τον παρακάτω πίνακα συγκρίνονται οι χρόνοι εκτέλεσης των δύο προγραμμάτων πάνω στο ίδιο σύνολο δεδομένων. Το σύνολο δεδομένων αποτελούνταν από 10 εργασίες, όπου η κάθε εργασία συμπεριλάμβανε γονιδιακές εκφράσεις 25.000 γονιδίων από 15 άτομα περίπου.

Λειτουργία	**Χρόνος εκτέλεσης στο M.A.G.E**	**Χρόνος εκτέλεσης στο STATA**
Μετα-ανάλυση	27 seconds	1.5 hours
Bootstrap (200 επαναλήψεις)	4.5 hours	~ 9 days
Bootstrap (500 επαναλήψεις)	~ 8 hours	N/A

Πίνακας 1- Συγκριτικός πίνακας των χρόνων εκτέλεσης των δύο προγραμμάτων που συντάχθηκαν σε Python και STATA

Κεφάλαιο 4ο

Συζήτηση – Συμπεράσματα

4.1 Μελλοντικά πλάνα

Οι αναλύσεις που προσφέρει αυτό το πρόγραμμα είναι ιδιαίτερα χρήσιμες για τις μελέτες που διεξάγονται στο εργαστήριο Μοριακής Υπολογιστικής Βιολογίας και Γενετικής και για την ευρύτερη επιστημονική κοινότητα. Το MAGE αποδείχτηκε αρκετά γρήγορο και αποτελεσματικό για την διεκπεραίωση αυτών των αναλύσεων.

Στο μέλλον, οι αλγόριθμοι του προγράμματος θα πρέπει να βελτιστοποιηθούν, ώστε οι αναλύσεις να εκτελούνται σε λιγότερο χρόνο. Ακόμη, θα πρέπει να προστεθεί στο πρόγραμμα η λειτουργία της πολυμεταβλητής μετα-ανάλυσης. Η μέθοδος της πολυμεταβλητής μετα-ανάλυσης έχει εφαρμογή σε μελέτες γενετικής συσχέτισης και σε συγκρίσεις διαφορετικών θεραπειών. Επιπλέον, θα πρέπει να προστεθούν και Μπεϋζιανές μέθοδοι μετα-ανάλυσης

Επίσης, ήδη γίνεται η προσπάθεια πάνω στην ενοποίηση του MAGE με το GISU, ένα εργαλείο που είχε συνταχθεί στο παρελθόν στο εργαστήριο το οποίο βρίσκει τους ανιχνευτές από τα πειράματα των μικροσυστοιχιών και τους ομαδοποιεί σε γονίδια. Το GISU χρειάζεται να συνταχθεί ξανά σε Python, καθώς έχει το ένα σκέλος του γραμμένο σε R και το άλλο σε php. Αυτά τα δύο εργαλεία θα αποτελούν μια ενιαία πλατφόρμα η οποία θα τρέχει και από γραμμή εντολών και από γραφικό περιβάλλον ως εφαρμογή.

Τέλος, μια διαδικτυακή server-based εφαρμογή του συνολικού πακέτου αυτού, θα ήταν μια σημαντική προσθήκη στα online διαθέσιμα εργαλεία που χρησιμοποιούνται από την επιστημονική κοινότητα των βιοπληροφορικών.

Βιβλιογραφία

[1] Βέννου, Κωνσταντίνα Ε. (2020). *Μεθοδολογίες υπολογιστικής ανάλυσης και μετα-ανάλυσης δεδομένων γονιδιακής έκφρασης* [Διδακτορική Διατριβή, Πανεπιστήμιο Θεσσαλίας]. Εθνικό Αρχείο Διδακτορικών Διατριβών.

[2] Kontou, Panagiota I., Athanasia Pavlopoulou, and Pantelis G. Bagos. "Methods of analysis and meta-analysis for identifying differentially expressed genes." *Genetic Epidemiology*. Humana Press, New York, NY, 2018. 183-210.

[3] Vennou, K. E., Piovani, D., Kontou, P. I., Bonovas, S., & Bagos, P. G. (2020). Methods for multiple outcome meta-analysis of gene-expression data. *MethodsX*, *7*, 100834.

[4] Hayes, D. N. and M. Meyerson (2005). Microarray Approaches to Gene Expression Analysis. Molecular Diagnostics: For the Clinical Laboratorian. W. B. Coleman and G. J. Tsongalis. Totowa, NJ, Humana Press: 121-148.

[5] Lukk, M., M. Kapushesky, J. Nikkilä, H. Parkinson, A. Goncalves, W. Huber, E. Ukkonen and A.Brazma (2010). "A global map of human gene expression." Nature biotechnology 28(4): 322-324.

[6] Liang, P. and A. B. Pardee (1992). "Differential display of eukaryotic messenger RNA by means of the polymerase chain reaction." Science 257(5072): 967-971.

[7] Velculescu, V. E., L. Zhang, B. Vogelstein and K. W. Kinzler (1995). "Serial analysis of gene expression." Science 270(5235): 484-487.

[8] Alberts, B., J. H. Wilson, A. Johnson, T. Hunt, J. Lewis, K. Roberts, M. Raff and P. Walter (2008). Molecular Biology of the Cell, Garland Science.

[9] Greenland, S. (1998). Meta-analysis. Modern Epidemiology. K. J. Rothman and S. Greenland, Lippincott Williams & Wilkins: 643-673.

[10] Sacks, H. S., J. Berrier, D. Reitman, V. A. Ancona-Berk and T. C. Chalmers (1987). "Metaanalyses of randomized controlled trials." N Engl J Med 316(8): 450-455.

[11] Trikalinos, T. A., G. Salanti, E. Zintzaras and J. P. Ioannidis (2008). "Meta-analysis methods." Adv Genet 60: 311-334.

[12] Πανάρετος, I. and Ε. Ξεκαλάκη (2000). Εισαγωγή στη στατιστική σκέψη. Αθήνα.

[13] Normand, S. L. (1999). "Meta-analysis: formulating, evaluating, combining, and reporting." StatMed 18(3): 321-359.

[14] Zeggini, E. and J. P. Ioannidis (2009). "Meta-analysis in genome-wide association studies."Pharmacogenomics 10(2): 191-201.

[15] Hong, F. and R. Breitling (2008). "A comparison of meta-analysis methods for detecting differentially expressed genes in microarray experiments." Bioinformatics 24(3): 374-382.

[16] Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences. 2nd edn. Hillsdale, New Jersey: L, Erlbaum.

[17] Hedges, L. V. (1981). "Distribution Theory for Glass's Estimator of Effect Size and Related Estimators." Journal of Educational Statistics 6(2): 107-128.

[18] Villanueva, E. V. and S. Zavarsek (2004). "Evaluating heterogeneity in cumulative metaanalyses." BMC Medical Research Methodology 4(1): 18.

[19] Holland, B. and M. Copenhaver (1987). "An Improved Sequentially Rejective Bonferroni Test Procedure." Biometrics 43(2): 417.

[20] Holm, S. (1979). "A simple sequentially rejective multiple test procedure." Scandinavian Journal of Statistics 6: 65-70.

[21] Šidák, Z. (1967). "Rectangular Confidence Regions for the Means of Multivariate Normal Distributions." Journal of the American Statistical Association 62(318): 626-633.

[22] Gentleman, R., M. Ruschhaupt, W. Huber and L. Lusa (2008). Meta-analysis for microarray experiments, Bioconductor.

[23] Borenstein, M., Hedges, L., & Rothstein, H. (2020). Meta-analysis: fixed effect vs. random effects. Metaanalysis. com. 2007.

[24] PythonMeta, https://pypi.org/project/PythonMeta/

[25] Pandas, https://pandas.pydata.org

[26] NumPy, https://numpy.org

[27] Statistics, https://docs.python.org/3.7/library/statistics.html

[28] SciPy, https://www.scipy.org

[29] Matplotlib, https://matplotlib.org

[30] Seaborn, https://seaborn.pydata.org

Παράρτημα

Αποτελέσματα μετα-ανάλυσης

Genes	Effect size (Hedge's g)	Standard_Error	Q	I_Squared	Tau_Squared	p_Q_value	z_test_value	p_value	bonferroni	sidak	holm	holland	hochberg	simes
a1bg	-0.187483812	0.124675674	10.65026	24.88	0.033005045	0.225	1.503772199	0.139	0	0	0	0	0	0
a1bg_as1	-0.464371052	0.23264274	0.790917	0	0	0.851	1.996069387	0.046	0	0	0	0	0	0
a1cf	-0.05634002	0.175202881	23.94549	62.41	0.172685007	0.004	0.321570169	0.748	0	0	0	0	0	0
a2m	-0.361338851	0.17602856	23.95134	62.42	0.174403049	0.004	2.052728544	0.04	0	0	0	0	0	1
a2ml1	0.100899543	0.13155626	11.64866	31.32	0.045366632	0.173	0.766968765	0.445	0	0	0	0	0	0
a4galt	0.071338351	0.125804837	12.96911	30.6	0.045016243	0.17	0.567055706	0.572	0	0	0	0	0	0
a4gnt	0.150358024	0.103690355	8.481419	5.68	0.005826049	0.389	1.450067596	0.154	0	0	0	0	0	0
aaas	-0.182881964	0.201491671	31.56084	71.48	0.264667053	0	0.907640315	0.366	0	0	0	0	0	0
aacs	0.054171429	0.097786	5.796878	0	0	0.67	0.553979391	0.58	0	0	0	0	0	0
aadac	0.087611513	0.158216439	16.68008	52.04	0.106148997	0.034	0.553744688	0.58	0	0	0	0	0	0
aadacl2	0.212328147	0.109663242	6.10259	1.68	0.001639487	0.413	1.936183376	0.053	0	0	0	0	0	0
aadat	0.081504594	0.258347902	35.8873	80.49	0.399745526	0	0.315483861	0.753	0	0	0	0	0	0
aaed1	-0.181036976	0.273535618	28.4666	78.92	0.376581099	0	0.661840594	0.508	0	0	0	0	0	0
aagab	0.078658842	0.151484908	15.3819	47.99	0.08993425	0.052	0.519252005	0.604	0	0	0	0	0	1

Στιγμιότυπο από τα αποτελέσματα της μετα – ανάλυσης

Genes	Effect size (Hedge's g)	Standard_Error	Q	I_Squared	Tau_Squared	p_Q_value	z_test_value	p_value	bonferron	sidak	holm	holland	hochberg	simes
a1bg	-0.270365932	0.225305068	13.54452	0.409355278	0.156849549	0.09443737	-1.199999338	0.1150698	0	0	0	0	0	0
a1bg_as1	-0.275999625	0.07096059	1.722968	0	0	0.631839594	-3.889477586	5.02E-05	1	1	1	1	1	1
a1cf	-0.064441277	0.263324266	4.070296	0	0	0.906725164	-0.244722135	0.40333579	0	0	0	0	0	0
a2m	-0.564222299	0.215974076	5.529703	0	0	0.785907309	-2.612453814	0.00449474	0	0	0	0	1	1
a4galt	0.13492063	0.133116122	5.27828	0	0	0.809406538	1.013555895	0.15539736	0	0	0	0	0	1
a2ml1	0.250370349	0.130344065	3.769596	0	0	0.805900052	1.920841957	0.02737582	0	0	0	0	0	0
a3galt2	-0.162275301	0.234139566	1.447446	0	0	0.835908205	-0.693070819	0.24413255	0	0	0	0	0	0
a4gnt	0.11371831	0.184276835	0.938388	0	0	0.918989964	0.617105836	0.26858246	0	0	0	0	0	0
aaas	0.298812872	0.345949062	5.506862	0.273633556	0.162582798	0.239126934	0.863748179	0.19386312	0	0	0	0	0	0
aacs	-0.061163755	0.134981454	0.186193	0	0	0.911105417	-0.453127097	0.32522861	0	0	0	0	0	0
aadac	-0.520299784	0.284312121	0.876712	0	0	0.349103825	-1.83003026	0.03362271	0	0	0	0	0	0
aagab	0.255000465	0.354433571	0.082363	0	0	0.774119676	0.719459119	0.23592904	0	0	0	0	0	0

Στιγμιότυπο από τα αποτελέσματα της bootstrap μετα – ανάλυσης (200 επαναλήψεις)

Κώδικας Python

mage.py

```
import PythonMeta as PMA
import pandas as pd

global genes_p_values, meta_analysis_df
genes_p_values = []
meta_analysis_df = []

file_list=['study1.txt','study2.txt','study3.txt','study4.txt','study5.txt','study6.txt','study7.txt','study8.txt','study9.txt','study10.txt']
index_file = 'mage_teams.txt'
# With this function we load our data. We need a list which will contain
# the filepaths of each study (txt and tab delimeted)
# and a file which will contain the indices of our control and cases on each study
def load_mage_data(file_list, index_file):
    # This function separates the cases from the controls from the annotation file
    def indices_of_teams():
        # 'mage_teams.txt'
        with open(index_file) as f:
            lines = [line.rstrip().split("\t") for line in f]

        team1 = 0
        team2 = 0
        study_counter = 0

        positions_team1 = []
        positions_team2 = []

        positions_array_team1 = []
        positions_array_team2 = []

        for i in range(len(lines)):
            study_counter += 1
            for j in range(len(lines[i])):  # in each line
                if lines[i][j] == "0":
                    team1 += 1
                    positions_team1.append(j + 1)


                elif lines[i][j] == "1":
                    team2 += 1
                    positions_team2.append(j + 1)
            positions_array_team1.append(positions_team1)
            positions_array_team2.append(positions_team2)
            positions_team1 = []

            positions_team2 = []

            team1 = 0
            team2 = 0

        return positions_array_team1, positions_array_team2

    global ind1, ind2, all_genes, means1_table, means2_table
    ind1, ind2 = indices_of_teams()

    expressions_team1 = []
    expressions_team2 = []
    all_genes = []
    means1 = []
    means2 = []
    means1_table = []
    means2_table = []
    expr1_arr = []
    expr2_arr = []
    all_list = []

    global num_of_studies
    num_of_studies = len(file_list)

    # Read study files
    for i, file in enumerate(file_list):
        data = pd.read_csv(file, sep="\t", skiprows=[0], header=None, low_memory=False, )
        genes = data.iloc[:, 0]

        n1 = len(ind1[i])
        n2 = len(ind2[i])

        genes = pd.DataFrame({"GENES": genes})
        data_team1 = data.iloc[:, ind1[i]]
        data_team2 = data.iloc[:, ind2[i]]

        expr1_arr.append(data_team1)
        expr2_arr.append(data_team2)

        data_team1 = pd.concat([genes, data_team1], axis=1)
        data_team2 = pd.concat([genes, data_team2], axis=1)

        y1 = data_team1.mean(axis=1)
        y2 = data_team2.mean(axis=1)

        means1.append(y1)
        means2.append(y2)

        y1 = pd.DataFrame(y1, columns=['MEANS TEAM 1'])
        y2 = pd.DataFrame(y2, columns=['MEANS TEAM 2'])
```

```
        z1 = data_team1.std(axis=1)
        z2 = data_team2.std(axis=1)

        z1 = pd.DataFrame(z1, columns=['STANDARD DEVIATIONS TEAM 1'])
        z2 = pd.DataFrame(z2, columns=['STANDARD DEVIATIONS TEAM 2'])

        n1_arr = [n1] * len(y1)
        n2_arr = [n2] * len(y2)

        n1 = pd.DataFrame(n1_arr, columns=["GROUP SIZE"])
        n2 = pd.DataFrame(n2_arr, columns=["GROUP SIZE"])

        means1_arr = pd.concat([genes, y1, z1, n1], axis=1)
        means2_arr = pd.concat([y2, z2, n2], axis=1)
        all = pd.concat([means1_arr, means2_arr], axis=1)

        expressions_team1.append(data_team1)
        expressions_team2.append(data_team2)

        all_genes.append(genes)

        means1_table.append(means1_arr)
        means2_table.append(means2_arr)
        all_list.append(all)

    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    pd.set_option('display.width', None)

    pd.options.display.float_format = "{:,.8f}".format

    return expressions_team1, expressions_team2, means1_table, means2_table


# With this function we can conduct a meta-analysis (Random models, IV-Heg,and SMD)
# WARNING! : First we have to load our data (with the load_mage_data function) in order to execute this function
def meta_analysis(expressions_team1, expressions_team2):
    global study
    study = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
             'V', 'W',
             'X', 'Y', 'Z']
    all_list2 = []
    import numpy as np
    for i in range(len(expressions_team1)):
        #

        genes = expressions_team1[i].iloc[:, 0]
        genes = pd.DataFrame(genes, columns=["GENES"])
        x1 = expressions_team1[i].iloc[:, 1:].mean(axis=1)
        x2 = expressions_team1[i].iloc[:, 1:].std(axis=1)

        n1 = len(ind2[i])
        n1 = [n1] * len(x1)

        n1 = pd.DataFrame(n1, columns=["GROUP SIZE"])

        x1 = pd.DataFrame(x1, columns=['MEANS TEAM 1'])
        x2 = pd.DataFrame(x2, columns=['STANDARD DEVIATIONS TEAM 1'])

        all_1 = pd.concat([genes, x1, x2, n1], axis=1)

        y1 = expressions_team2[i].iloc[:, 1:].mean(axis=1)
        y2 = expressions_team2[i].iloc[:, 1:].std(axis=1)

        n2 = len(ind1[i])
        n2 = [n2] * len(y1)

        n2 = pd.DataFrame(n2, columns=["GROUP SIZE"])

        y1 = pd.DataFrame(y1, columns=['MEANS TEAM 2'])
        y2 = pd.DataFrame(y2, columns=['STANDARD DEVIATIONS TEAM 2'])

        all_2 = pd.concat([y1, y2, n2], axis=1)

        all = pd.concat([all_1, all_2], axis=1)
        all_list2.append(all)
    all_info = []
    all_genes2 = []
    for i in all_list2:
        all_info.append(i.values.tolist())
    for i in all_genes:
        all_genes2.append(i.values.tolist())

    items = []

    for j in all_info:
        items.append(j)
    items = list(np.concatenate(
        items))  # now everything is in the desired dataFrame,but we need to make the genes as keys in hash tables
    items = np.array(items)

    hash_table = []
    for item in items:
        hash_table.append({item[0]: item[1:]})
    hash_table = np.array(hash_table)
    # hash_table=list(hash_table.flat)
    from collections import defaultdict
    res = defaultdict(list)
    for sub in hash_table:
        for key in sub:
            res[key].append(sub[key])
```

```
    hash_table = dict(res)
    res = {k: v for k, v in hash_table.items() if len(v) >= 2}
    hash_table = res

    # print(hash_table)

    x = pd.DataFrame(hash_table.values(), index=hash_table.keys()).T

    text = []
    gene_names = x.columns
    # print(list(gene_names)) # Every column is a gene

    global counter, index

    # print("Gene\tEffect size (Hedge's g)\tStandard_Error\tQ\tI_Squared\tTau_Squared\tp_Q_value\tz_test_value\tp_value\n")
    for i in range(len(x.columns)):
        counter = gene_names[i]
        index = i
        column = x.iloc[:, i].dropna()
        # print(column)
        for i, row in enumerate(column):
            temp = ",".join(str(x) for x in row)
            text.append(study[i] + "," + temp)
            # print(text)

        settings = {"datatype": "CONT",  # for CONTinuous data
                    "models": "Random",  # models: Fixed or Random
                    "algorithm": "IV-Heg",  # algorithm: IV
                    "effect": "SMD"}  # effect size: MD, SMD
        main(text, settings)
        # print(text)
        # print('\n')
        text = []
    return pd.DataFrame(meta_analysis_df)


# This function helps in the execution for the meta_analysis function,and calls the showresults() function
def main(stys, settings):
    d = PMA.Data()  # Load Data class
    m = PMA.Meta()  # Load Meta class
    # f = PMA.Fig()   #Load Fig class

    # You should always tell the datatype first!!!
    d.datatype = settings["datatype"]  # set data type, 'CATE' for binary data or 'CONT' for continuous data
    studies = d.getdata(stys)  # load data
    # studies = d.getdata(d.readfile("studies.txt"))  #get data from a data file, see examples of data files
    # print(showstudies(studies,d.datatype))          #show studies

    m.datatype = d.datatype  # set data type for meta-analysis calculating
    m.models = settings["models"]  # set effect models: 'Fixed' or 'Random'
    m.algorithm = settings["algorithm"]  # set algorithm, based on datatype and effect size
    m.effect = settings["effect"]  # set effect size:RR/OR/RD for binary data; SMD/MD for continuous data
    results = m.meta(studies)  # performing the analysis

    # print (showresults(results))                    #show results table
    showresults(results)


# This function helps in the execution for the meta_analysis function and is called by the main function()
def showresults(rults):
    text = "%-10s %-6s  %-18s %-10s" % ("Study ID", "n", "ES[95% CI]", "Weight(%)\n")
    global effect_size_list
    # gene                    #es                         #se                     #Q                              # i2                        #ta
score              p_value
    text1 = str(counter) + "\t" + str(rults[0][1]) + "\t" + str(abs(rults[0][1] / rults[0][10])) + "\t" + str(
        rults[0][7]) + "\t" + str(round(rults[0][9], 2)) + "%\t" + str((rults[0][12])) + "\t" + str(
        (rults[0][8])) + "\t" + str(rults[0][10]) + "\t" + str(rults[0][11])
    new_row = {'Genes': counter, 'p_value': rults[0][11]}
    new_row2 = {'Genes': counter, "Effect size (Hedge's g)": rults[0][1],
                'Standard_Error': abs(rults[0][1] / rults[0][10]), 'Q': rults[0][7], 'I_Squared': round(rults[0][9], 2),
                'Tau_Squared': rults[0][12], 'p_Q_value': rults[0][8], 'z_test_value': rults[0][10],
                'p_value': rults[0][11]}
    genes_p_values.append(new_row)
    meta_analysis_df.append(new_row2)
    # return meta_analysis_df
    return text1


#
def altmeta(y1, s2):
    from scipy.stats import norm
    import numpy as np
    from scipy.stats.distributions import chi2

    n = len(y1)
    w = [(1 / x) for x in s2]
    mu_bar = sum(a * b for a, b in zip(w, y1)) / sum(w)

    Q = sum(a * b for a, b in zip(w, [(x - mu_bar) ** 2 for x in y1]))
    p_Q = chi2.sf(Q, len(y1) - 1)
    H = np.sqrt(Q / (n - 1))
    I2 = (Q - (len(s2) - 1)) / Q
    I2 = max(0, I2)
    tau2_DL = (Q - n + 1) / (sum(w) - sum([x ** 2 for x in w]) / sum(w))
    tau2_DL = max(0, tau2_DL)

    # re adjustment of the weights
    w = [(1 /(x+tau2_DL) ) for x in s2]
    mu_bar = sum(a * b for a, b in zip(w, y1)) / sum(w)
```

```
    se = np.sqrt(1 / sum(w))
    z = abs(mu_bar / se)
    p = norm.cdf(1- abs(z))
    return (Q, I2, tau2_DL, p_Q, se, z, mu_bar, p)

# With this function we can conduct a bootstrapped meta_analysis
# With this function we can conduct a bootstrapped meta_analysis
def bootstrap_analysis(expressions_team1, expressions_team2, means1_table, means2_table, n):
    import random
    import numpy as np
    from math import sqrt

    import gc

    import statistics as st

    random_cols = []
    n_of_res = n

    def bootstrap(expressions_team):
        bootstrap_data = []
        for i in range(len(expressions_team)):  # for each study
            g = expressions_team[i]["GENES"]  # For each gene
            expr = expressions_team[i].iloc[:, 1:]
            # for each expressions set
            col_list = list(expr.columns)
            #            print(len(col_list) )
            col_list_new = []

            for _ in range(n_of_res):  # change it for bigger iterations
                boot_df = pd.DataFrame()

                cols_new = [random.choice(col_list) for _ in range(len(col_list))]
                random_cols.append(cols_new)
                for k in range(len(cols_new)):
                    boot_df = pd.concat([boot_df, expr[cols_new[k]]], axis=1)

                new_df = pd.concat([g, boot_df], axis=1)

                bootstrap_data.append(new_df)

                del boot_df
        return bootstrap_data

    et1 = np.array(bootstrap(expressions_team1), dtype=object).flatten()
    et2 = np.array(bootstrap(expressions_team2), dtype=object).flatten()

    # for step in range(0,len(et1),5):   # change it for bigger iterations
    et1_sliced = [et1[i:i + n_of_res] for i in range(0, len(et1), n_of_res)]
    et2_sliced = [et2[i:i + n_of_res] for i in range(0, len(et2), n_of_res)]

    # print(len(et1_sliced),len(et1_sliced[0]))
    # print((et1_sliced[0][0].std(axis=1)))

    def get_means_and_std(et1_sliced):
        std_boot = []
        means_boot = []
        for i in range(len(et1_sliced)):
            for j in range(len(et1_sliced[0])):
                n = len(et1_sliced[i][j].columns) - 1  # giati exoyme kai ta genes opote meion 1 sthlh

                n = [n] * len(et1_sliced[i][j]['GENES'])
                n = pd.DataFrame(n, columns=['N'])

                means_boot.append(
                    pd.concat([et1_sliced[i][j]['GENES'], et1_sliced[i][j].mean(axis=1), et1_sliced[i][j].std(axis=1),
                               n], axis=1))
        return means_boot, std_boot

    means_boot1, std_boot1 = get_means_and_std(et1_sliced)
    means_boot2, std_boot2 = get_means_and_std(et2_sliced)

    rename_cols1 = ['GENES', 'MEANS1', 'STD1', 'N1']
    rename_cols2 = ['GENES2', 'MEANS2', 'STD2', 'N2']

    boot_data_all = []
    for i in range(len(means_boot1)):
        means_boot1[i].columns = rename_cols1
        means_boot2[i].columns = rename_cols2

    effect_sizes_d = []
    effect_sizes_g = []
    test_g = []
    test_gene = []
    # print(means_boot1)

    import dask.dataframe as dd
    import math
    for i in range(len(means_boot1)):
        for j in range(len(means_boot1[i])):
            list_of_items1 = list(means_boot1[i].iloc[j])
            list_of_items2 = list(means_boot2[i].iloc[j])
            # print(list_of_items1)
            N = list_of_items1[3] + list_of_items2[3]
            df = N - 2
            J = (math.gamma(df / 2) / (math.sqrt(df / 2) * math.gamma((df - 1) / 2)))
            Sp = sqrt(
                #    (*n1-1)*s1^2                                                (*n2-1)*s2^2
                ((list_of_items1[3] - 1) * (list_of_items1[2] * list_of_items1[2]) + (list_of_items2[3] - 1) * (
                        list_of_items2[2] * list_of_items2[2])) / (list_of_items1[3] + list_of_items2[3] - 2))
            d = (list_of_items1[1] - list_of_items2[1]) / Sp  # effect sizes _d
            g = J * d  # effect size corrected  with Hedge's g
```

```
        gc.collect()
        effect_sizes_d.append({list_of_items1[0]: d})
        effect_sizes_g.append({list_of_items1[0]: g})
        test_gene.append(list_of_items1[0])
        test_g.append(g)

# print(test_g)
# print(test_gene)
df = pd.DataFrame(test_g, index=test_gene)
df = df.sort_index()
# print(df)
genes = list(dict.fromkeys(test_gene))  # unique

temp_df_list = []

for gene in genes:
    x = df.loc[gene]

    temp_df_list.append(list(x[0]))

import itertools
flat_list = list(itertools.chain(*list(n * [np.arange(0, num_of_studies)])))

# print(flat_list)
temp_df = pd.DataFrame(temp_df_list, index=genes, columns=flat_list)
new_cols = np.arange(0, len(list(temp_df.columns)))
temp_df.columns = new_cols

# temp_df=pd.DataFrame(effect_sizes_g).T         # or effect_sizes_d
# # b=db.from_sequence(np.array(effect_sizes_g).T,npartitions=1)
# # temp_df=b.to_dataframe()

# print(temp_df)
# print(temp_df)
genes_boot = genes
# print(genes_boot)
std_err_boot = []
for gene in genes_boot:
    tmp = (list(temp_df.loc[gene, :].dropna()))
    tmp_sliced = [tmp[i:i + n_of_res] for i in range(0, len(tmp), n_of_res)]
    # print(tmp_sliced)
    for i in range(len(tmp_sliced)):
        # for j in range (len(tmp_sliced[0])):
        std_err = st.stdev(tmp_sliced[i])
        std_err_boot.append({'Gene': gene, 'std_err': std_err})
# print(pd.DataFrame(std_err_boot))
# print(std_err_boot)

std_err_tmp = pd.DataFrame(std_err_boot, columns=['Gene', 'std_err'])
std_err_tmp = std_err_tmp.set_index('Gene')

std_err_list_boot = []
for gene in genes:
    x = std_err_tmp.loc[gene]

    std_err_list_boot.append(list(x.values.flatten()))

import itertools
flat_list = list(itertools.chain(*list([np.arange(0, num_of_studies)])))

df = pd.DataFrame(std_err_list_boot, index=genes, columns=flat_list)

boot_df = pd.DataFrame()
for gene in genes:
    # print(gene)
    # print(df.T.loc[gene,:].dropna())
    tmp = list(df.loc[gene, :].dropna())
    new_row = {'GENES': gene, "SE": tmp}
    # append row to the dataframe
    boot_df = boot_df.append(new_row, ignore_index=True)  # Boot DF has the standard_errors

pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)

genes_uniq = (boot_df['GENES'])

es_df = pd.DataFrame()

for i in range(len(means1_table)):
    for gene in means1_table[i].index:
        # print(list(means2_table[i].iloc[gene]))
        m1 = (list(means1_table[i].iloc[gene])[1])
        m2 = (list(means2_table[i].iloc[gene])[0])

        # print(list_of_items1)
        n1 = list(means1_table[i].iloc[gene])[3]
        n2 = list(means2_table[i].iloc[gene])[2]
        N = n1 + n2
        df = N - 2
        J = (math.gamma(df / 2) / (math.sqrt(df / 2) * math.gamma((df - 1) / 2)))
        Sp = sqrt(
            #    (*n1-1)*s1^2                                                (*n2-1)*s2^2
            ((n1 - 1) * (list(means1_table[i].iloc[gene])[2] * list(means1_table[i].iloc[gene])[2]) + (n2 - 1) * (
                    list(means2_table[i].iloc[gene])[1] * list(means2_table[i].iloc[gene])[1])) / (N - 2))
        d = (m1 - m2) / Sp  # effect sizes _d
        g = J * d  # effect size corrected  with Hedge's g
        var_d = N / (n1 * n2) + (d * d) / (2 * N)
        var_g = J * J * var_d
        new_row = {'GENES': (list(means1_table[i].iloc[gene])[0]), 'd': d, 'g': g, 'var_d': var_d, 'var_g': var_g}
        es_df = es_df.append(new_row, ignore_index=True)
```

```
    # print(es_df)
    es_list = pd.DataFrame()
    for gene in genes_uniq:
        es_df_index = es_df[es_df['GENES'] == gene]
        es_of_studies = list(es_df_index['g'])  # or d
        new_row = {"GENES": gene, "List_of_ES": es_of_studies}
        es_list = es_list.append(new_row, ignore_index=True)
        # print(es_df_index)
    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    pd.set_option('display.width', None)
    # print(es_list)
    pd.set_option('display.max_rows', None)
    pd.set_option('display.max_columns', None)
    pd.set_option('display.width', None)
    # print(boot_df)

    total_df = pd.DataFrame()

    list_of_boot = []
    # for each unique gene
    # Take each line  from the  to es_list and from  boot_df and do the meta-analysis
    # print('Genes'+'\t'+"Effect Size (Hedge's g)"+'\t'+'Standard Error '+'\t'+'Q'+'\t'+'I - Squared'+'\t'+'Tau - Squared'+'\t'+'p_Q'+'\t'+'\t'+'z-
    num_of_chunks = 10
    es_list_chunks = [es_list[i:i + num_of_chunks] for i in range(0, len(es_list), num_of_chunks)]
    boot_df_chunks = [boot_df[i:i + num_of_chunks] for i in range(0, len(boot_df), num_of_chunks)]
    for es_list, bood_df in zip(es_list_chunks, boot_df_chunks):
        for index in es_list.index:
            gene = (es_list.loc[index, 'GENES'])
            d = (es_list.loc[index, 'List_of_ES'])
            se = boot_df.loc[index, 'SE']

            if (len(se) > 1):
                # Gene Effect size (Hedge's g)    Q  I - Squared    Tau - Squared  p_Q value  z-test value   p-value
                # Genes'+'\t'+"Effect Size"+'\t'+'Q'+'\t'+'p_Q'+'\t'+'I - Squared'+'\t'+'z-score'+'\t'+'p-value'

                Q, I2, tau2, p_Q, st_e, z, e_size, p = altmeta(d, se)

                # print(gene+"\t"+str(e_size)+"\t"+str(st_e)+"\t"+str(Q)+"\t"+str(I2)+"\t"+str(tau2)+"\t"+str(p_Q)+"\t"+str(z)+"\t"+str(p))
                #
                new_row2 = {'Genes': gene, "Effect size (Hedge's g)": e_size, 'Standard Error': st_e, 'Q': Q,
                            'I_Squared': I2, 'Tau_Squared': tau2, 'p_Q_value': p_Q, 'z_test_value': z, 'p_value': p}
                list_of_boot.append(new_row2)

    # print(total df)
    return pd.DataFrame(list_of_boot)


# With this function we can get the  one step methods (Bonferroni and Sidak)
def get_one_step_methods(meta_analysis_df,
                         alpha):  # This function takes the Sidak and the Bonferoni multiple test method
    import numpy as np
    m = len(meta_analysis_df['p_value'])
    p_values = (np.array(meta_analysis_df['p_value'], dtype=float))
    bonf = []
    sidak = []
    exponent = 1 / m
    for i in range(m):  # a = 0.05 due to the 95 % CI
        if p_values[i] >= (1 - (1 - alpha) ** exponent):
            sidak.append(0)  # 0 ---> no significant difference
        else:
            sidak.append(1)  # 1 --->   significant  difference

        if p_values[i] >= alpha / m:
            bonf.append(0)

        else:
            bonf.append(1)

    one_step_methods = pd.DataFrame(list(zip(list(meta_analysis_df['Genes']), p_values, bonf, sidak)),
                                    columns=['Genes', 'p_value', 'bonferroni', 'sidak'])
    # print(one_step_methods)
    return (one_step_methods)


# With this function we can get the  step down methods (Holm and Holland)
def get_step_down_methods(meta_analysis_df,
                          alpha):  # This function takes the Sidak and the Bonferoni multiple test method
    import numpy as np
    m = len(meta_analysis_df['p_value'])
    meta_analysis_df = meta_analysis_df.sort_values(by=['p_value'])
    # print(genes_and_p_values)
    p_values = (np.array(meta_analysis_df['p_value'], dtype=float))

    holm = []
    holland = []
    for i in range(m):
        if p_values[i] >= (alpha / (m - (i) + 1)):

            holm.append(0)
        else:
            holm.append(1)

        exponent = 1 / (m - (i) + 1)
        if p_values[i] >= (1 - (1 - alpha) ** exponent):

            holland.append(0)

        else:
            holland.append(1)

    step_down_methods = pd.DataFrame(list(zip(list(meta_analysis_df['Genes']), p_values, holm, holland)),
```

```
                                    columns=['Genes', 'p_value', 'holm', 'holland'])
    # print(step_down_methods)
    return (step_down_methods)


# With this function we can get the step up methods (Simes and Hochberg)
def get_step_up_methods(meta_analysis_df,
                        alpha):  # This function takes the Sidak and the Bonferoni multiple test method
    import numpy as np
    m = len(meta_analysis_df['p_value'])
    meta_analysis_df = meta_analysis_df.sort_values(by=['p_value'], ascending=False)  # sort by p_value
    # print(genes_and_p_values)
    p_values = (np.array(meta_analysis_df['p_value'], dtype=float))

    hochberg = []
    simes = []
    for i in range(m):

        if p_values[i] >= (alpha / (m - (i) + 1)):
            hochberg.append(0)
        else:
            hochberg.append(1)

        if p_values[i] >= ((i * alpha) / m):
            simes.append(0)
        else:
            simes.append(1)

    step_down_methods = pd.DataFrame(list(zip(list(meta_analysis_df['Genes']), p_values, hochberg, simes)),
                                    columns=['Genes', 'p_value', 'hochberg', 'simes'])
    # print(step_down_methods)
    return (step_down_methods)


# call  all the Multiple - tests functions (Bonferroni,Sidak,Holm,Holland,Simes and Hochberg from the upper functions)
def all_multiple_tests(meta_analysis_df, alpha):
    import pandas as pd

    d1 = get_one_step_methods(meta_analysis_df, alpha)

    d2 = get_step_down_methods(meta_analysis_df, alpha)

    d3 = get_step_up_methods(meta_analysis_df, alpha)

    total_df = pd.concat([d1, d2, d3], axis=1)
    return total_df



import time

t0 = time.time()
file_list=['study1.txt','study2.txt','study3.txt','study4.txt','study5.txt','study6.txt','study7.txt','study8.txt','study9.txt','study10.txt']
index_file = 'mage_teams.txt'
boot_msg='YES'
num_of_reps= 2
mult_tests='-'
alpha=0.05

def run_mage(file_list, index_file, boot_msg, num_of_reps, mult_tests, alpha):
    num_of_reps = int(num_of_reps)

    alpha = float(alpha)


    ## load_data: Load data contains the file list and the index file
    # #contains the indices positions of cases and controls

    expressions_team1, expressions_team2, means1_table, means2_table = load_mage_data(file_list, index_file=index_file)

    # Conduct the meta-analysis of the genes (WARNING !!! Expressions 2 is for cases and Expressions 1 is for Controls)

    meta_analysis_df = meta_analysis(expressions_team2, expressions_team1)

    # # This function conducts the Bootstrap trial
    if boot_msg == 'YES':
        boot_df = bootstrap_analysis(expressions_team2, expressions_team1, means1_table, means2_table, n=num_of_reps)
        step_down = get_step_down_methods(boot_df, alpha)
        step_up = get_step_up_methods(boot_df, alpha)
        one_step = get_one_step_methods(boot_df, alpha)

        # #   all multiple tests  (bonferroni, sidak, holm, holland, hochberg and simes methods)
        # #   0 --> No statistical significance difference
        # #   1 --> Statistical significance difference

        all_tests = all_multiple_tests(boot_df, alpha)

        # if output =='YES':
        if mult_tests == "one_step":
            tests = one_step
        elif mult_tests == "step_up":
            tests = step_up

        elif mult_tests == "step_down":
            tests = step_down

        else:
            tests = all_tests

        meta_an = pd.concat([boot_df, all_tests], axis=1)
        print(meta_an)
        meta_an.to_csv("results_bootstrap.txt", sep='\t', mode='w')
```

```
        print('file created')



    else:

        # Each function of these functions, conducts the multiple test functions  and returns a dataframe.
        step_down = get_step_down_methods(meta_analysis_df, alpha)
        step_up = get_step_up_methods(meta_analysis_df, alpha)
        one_step = get_one_step_methods(meta_analysis_df, alpha)

        # #   all multiple tests  (bonferroni, sidak, holm, holland, hochberg and simes methods)
        # #   0 --> No statistical significance difference
        # #   1 --> Statistical significance difference

        all_tests = all_multiple_tests(meta_analysis_df, alpha)

        # print(boot_df)
        t1 = time.time()
        total = t1 - t0

        if mult_tests == "one_step":
            tests = one_step
        elif mult_tests == "step_up":
            tests = step_up

        elif mult_tests == "step_down":
            tests = step_down

        else:
            tests = all_tests

        meta_an = pd.concat([meta_analysis_df, tests], axis=1)
        meta_an.to_csv("results.txt", sep='\t', mode='w')
        print(meta_an)
        print('File created')

        t1 = time.time()
        total = t1 - t0
        print( str(total) + '\t' + '  seconds')


run_mage(file_list, index_file, boot_msg, num_of_reps, mult_tests, alpha)
import mage_plots
mage_plots.plot_data('results.txt')
```

mage_gui.py

```
from dummy_test import *  # import mage.py file which contains all M.A.GE functions

import time

t0 = time.time()



def run_mage(file_list, index_file, boot_msg, num_of_reps, mult_tests, alpha, plots):
    num_of_reps = int(num_of_reps)

    alpha = float(alpha)

    def plot_data(meta_analysis_df):

        import numpy as np

        import scipy.stats as stats

        import matplotlib.pyplot as plt

        # QQ plot
        plt.figure("QQ plot ")
        measurements = meta_analysis_df["Effect size (Hedge's g)"]
        tau = meta_analysis_df['Tau_Squared']
        q = meta_analysis_df['Q']
        i2 = meta_analysis_df['I_Squared']
        stats.probplot(measurements, dist="norm", plot=plt)

        # Tau2 plot
        plt.figure("Tau - Squared DL Histogram ")
        plt.hist(tau, color='r', density=True, edgecolor='black', linewidth=1.2)
        plt.title("Tau - Squared DL Histogram")
        plt.xlabel('Values')
        plt.ylabel('Frequency')

        # Q plot
        plt.figure("Q Histogram ")
        plt.hist(q, color='g', density=True, edgecolor='black', linewidth=1.2)
        plt.title("Q Histogram")
        plt.xlabel('Values')
        plt.ylabel('Frequency')
```

```
        # I2 plot
        plt.figure("I - Squared Histogram ")
        plt.hist(i2, color='b', density=True, edgecolor='black', linewidth=1.2)
        plt.title("I - Squared DL Histogram ")
        plt.xlabel('Values')
        plt.ylabel('Frequency')

        # Volc Plot
        df = meta_analysis_df[meta_analysis_df.p_value != 0]
        p_values = np.array(df['p_value'].to_numpy().tolist(), dtype=float)

        smd = df["Effect size (Hedge's g)"]
        log_p = -(np.log10(p_values))

        color = []
        y1 = []
        y2 = []

        for i in range(len(p_values)):
            if p_values[i] < alpha:
                color.append(2)


            #     if p_values[i]==0.05:
            #         color.append(2)

            else:
                #         p_values[i]>0.05:
                color.append(1)

        plt.figure("Volcano plot")
        plt.scatter(smd, log_p, c=color, cmap='RdYlGn')
        plt.title("Volcano Plot")
        plt.xlabel("Effect Size Hedge's g")
        plt.ylabel("- log10 p_values")

        plt.show()

    ## load_data: Load data contains the file list and the index file
    # #contains the indices positions of cases and controls

    expressions_team1, expressions_team2, means1_table, means2_table = load_mage_data(file_list, index_file=index_file)

    # Conduct the meta-analysis of the genes (WARNING !!! Expressions 2 is for cases and Expressions 1 is for Controls)

    meta_analysis_df = meta_analysis(expressions_team2, expressions_team1)

    # # This function conducts the Bootstrap trial
    if boot_msg == 'YES':
        boot_df = bootstrap_analysis(expressions_team2, expressions_team1, means1_table, means2_table, n=num_of_reps)
        step_down = get_step_down_methods(boot_df, alpha)
        step_up = get_step_up_methods(boot_df, alpha)
        one_step = get_one_step_methods(boot_df, alpha)

        # #   all multiple tests  (bonferroni, sidak, holm, holland, hochberg and simes methods)
        # #   0 --> No statistical significance difference
        # #   1 --> Statistical significance difference

        all_tests = all_multiple_tests(boot_df, alpha)

        # if output =='YES':
        if mult_tests == "one_step":
            tests = one_step
        elif mult_tests == "step_up":
            tests = step_up

        elif mult_tests == "step_down":
            tests = step_down

        else:
            tests = all_tests

        meta_an = pd.concat([boot_df, all_tests], axis=1)
        print(meta_an)
        meta_an.to_csv("results_bootstrap.txt", sep='\t', mode='w')
        print('file created')

        if plots == 'YES':
            plot_data(boot_df)


    else:

        # Each function of these functions, conducts the multiple test functions  and returns a dataframe.
        step_down = get_step_down_methods(meta_analysis_df, alpha)
        step_up = get_step_up_methods(meta_analysis_df, alpha)
        one_step = get_one_step_methods(meta_analysis_df, alpha)

        # #   all multiple tests  (bonferroni, sidak, holm, holland, hochberg and simes methods)
        # #   0 --> No statistical significance difference
        # #   1 --> Statistical significance difference

        all_tests = all_multiple_tests(meta_analysis_df, alpha)

        # print(boot_df)
        t1 = time.time()
        total = t1 - t0

        if mult_tests == "one_step":
            tests = one_step
        elif mult_tests == "step_up":
```

```
            tests = step_up

        elif mult_tests == "step down":
            tests = step_down

        else:
            tests = all_tests

        meta_an = pd.concat([meta_analysis_df, tests], axis=1)
        meta_an.to_csv("results.txt", sep='\t', mode='w')
        print(meta_an)
        print('file created')

        t1 = time.time()
        total = t1 - t0
        print(str(total / 3600) + "\t in hours" + str(total / 60) + "\t in mins    |" + '\t' + str(
            total) + '\t' + ' in seconds')

        if plots == 'YES':
            plot_data(meta_analysis_df)


from tkinter import filedialog

import tkinter as tk
import tkinter.ttk as ttk


class MageGuiFormApp:
    def __init__(self, master=None):
        # build ui
        frame_1 = ttk.Frame(master)
        self.Run = ttk.Button(frame_1, command=Run_button)
        self.Run.config(text='Run')
        self.Run.place(anchor='nw', relheight='0.06', relwidth='0.23', relx='0.72', rely='0.0', y='450')
        self.entry_3 = ttk.Entry(frame_1)
        self.entry_3.config(validate='none')
        self.entry_3.place(anchor='nw', relx='0.28', rely='0.26', x='0', y='0')
        self.entry_4 = ttk.Entry(frame_1)

        self.entry_4.place(anchor='nw', relx='0.28', rely='0.34', x='0', y='0')
        self.entry_5 = ttk.Entry(frame_1)
        self.entry_5.place(anchor='nw', relx='0.28', rely='0.43', x='0', y='0')
        label_1 = ttk.Label(frame_1)
        label_1.config(font='{Arial} 12 {}', text='Files')
        label_1.place(anchor='nw', relx='0.11', rely='0.1', x='0', y='0')
        label_2 = ttk.Label(frame_1)
        label_2.config(font='{Arial} 12 {}', text='Annotation')
        label_2.place(anchor='nw', relx='0.04', rely='0.18', x='0', y='0')
        label_3 = ttk.Label(frame_1)
        label_3.config(font='{arial} 12 {}', text='Bootstrap')
        label_3.place(anchor='nw', relx='0.04', rely='0.26', x='0', y='0')
        label_4 = ttk.Label(frame_1)
        label_4.config(font='{Arial} 11 {}', text='    Num of reps \n(if bootstrap = YES)')
        label_4.place(anchor='nw', relx='0.01', rely='0.34', x='0', y='0')
        label_5 = ttk.Label(frame_1)

        label_5.config(font='{Arial} 11 {}', text='Multiple Tests', textvariable="")
        label_5.place(anchor='nw', relx='0.03', rely='0.43', x='0', y='0')
        label_6 = ttk.Label(frame_1)
        label_6.config(font='{ARIAL} 24 {bold}', text='MAGE')
        label_6.place(anchor='nw', relx='0.7', x='0', y='0')
        self.button_1 = ttk.Button(frame_1, command=Import_Files)
        self.button_1.config(text=' Select studies')
        self.button_1.place(anchor='nw', relx='0.28', rely='0.1', x='0', y='0')
        self.button_2 = ttk.Button(frame_1, command=Annotation_File)
        self.button_2.config(text='Index File')
        self.button_2.place(anchor='nw', relheight='0.045', relwidth='0.16', relx='0.28', rely='0.18', x='0', y='0')
        self.label_1_2 = ttk.Label(frame_1)
        self.label_1_2.config(font='{Arial} 12 {}', text='Alpha')
        self.label_1_2.place(anchor='nw', relx='0.07', rely='0.50', x='0', y='0')
        self.entry_1 = ttk.Entry(frame_1)
        self.entry_1.place(anchor='nw', relx='0.28', rely='0.5', x='0', y='0')
        self.label_3_4 = ttk.Label(frame_1)
        self.label_3_4.config(anchor='center', font='{Arial} 12 {}', text='Plots')
        self.label_3_4.place(anchor='nw', relx='0.07', rely='0.59', x='0', y='0')
        self.entry_2 = ttk.Entry(frame_1)
        self.entry_2.place(anchor='nw', relx='0.28', rely='0.58', x='0', y='0')
        frame_1.config(height='500', width='500')
        frame_1.pack(side='top')

        # Main widget
        self.mainwindow = frame_1

    def run(self):
        self.mainwindow.mainloop()


def Import_Files(event=None):
    global filename_studies

    filename = filedialog.askopenfilenames()
    new_files = []
    for i in filename:
        new_files.append(i)
    filename_studies = new_files

    return filename_studies


def Annotation_File(event=None):
    global filename_annot
```

```
    filename = filedialog.askopenfilenames()

    for i in filename:
        filename_annot = i
    return filename_annot


def Run_button():
    file_names = filename_studies
    index_files = filename_annot
    boot_msg = app.entry_3.get()
    num_of_reps = app.entry_4.get()
    mult_tests = app.entry_5.get()
    alpha = app.entry_1.get()
    plots = app.entry_2.get()

    print(file_names, index_files, boot_msg, num_of_reps, mult_tests, alpha, plots)
    run_mage(file_names, index_files, boot_msg, num_of_reps, mult_tests, alpha, plots)


if __name__ == '__main__':
    root = tk.Tk()
    root.iconbitmap('uth.ico')
    root.title("MAGE - Meta-Analysis of Genetic Expression @ DIB UTH ")
    global app
    app = MageGuiFormApp(root)
    app.run()
```

Κώδικας STATA

```
program define myDL5, rclass
version 10.1
syntax , x(varlist numeric max=1) Type(varlist numeric max=1)
Study(varlist numeric max=1) [ Effect(string ) ]
preserve
tempvar last
qui bysort `study': gen `last'=_n==_N
qui sum `study'
local max=r(max)
tempvar x0 x1 sd0 sd1 n0 n1 d sed m gmfm gmfm_1 J g seg
qui gen double `x0'=.
qui gen double `x1'=.
qui gen double `sd0'=.
qui gen double `sd1'=.
qui gen double `n0'=.
qui gen double `n1'=.
qui gen double `d'=.
qui gen double `sed'=.
qui gen double `m'=.
qui gen double `gmfm'=.
qui gen double `gmfm_1'=.
qui gen double `J'=.
qui gen double `g'=.
qui gen double `seg'=.
forvalues i=1(1)`max' {
qui sum `x' if `study'==`i' & `type'==0
qui replace `x0'=r(mean) if `study'==`i'
qui replace `sd0'=r(sd) if `study'==`i'
qui replace `n0'=r(N) if `study'==`i'
qui sum `x' if `study'==`i' & `type'==1
qui replace `x1'=r(mean) if `study'==`i'
qui replace `sd1'=r(sd) if `study'==`i'
qui replace `n1'=r(N) if `study'==`i'
qui replace `m'= (`n1'+ `n0'-2) if `study'==`i'
qui replace `d'=(`x1' - `x0')/(sqrt( ( (`n1'- 1)*(`sd1')^2 +(`n0'-
1)*(`sd0')^2)/( `n1' + `n0' -2 ))) if `study'==`i'
qui replace `sed'= sqrt( (`n1' + `n0')/(`n1'*`n0')
+(`d')^2/(2*(`n1'+`n0'))) if `study'==`i'
qui replace `gmfm' = exp((lngamma(`m'/2))) if `study'==`i'
qui replace `gmfm_1'= exp((lngamma((`m'-1)/2))) if `study'==`i'
qui replace `J' =`gmfm'/((sqrt(`m'/2))*`gmfm_1') if `study'==`i'
qui replace `g' =`J'*`d' if `study'==`i'
qui replace `seg'=(`J')^2*`sed' if `study'==`i'
** var(g)=J*var(d)
}
qui keep if `last'
qui keep if `x' !=.
metan `g' `seg' , `effect' nograph
**metan `n1' `x1' `sd1' `n0' `x0' `sd0', `effect' nograph
**metan `n1' `x1' `sd1' `n0' `x0' `sd0',fixed nograph hedges
```

```
**metan `d' `sed' , random nograph
**list
Restore
return scalar g=r(ES)
return scalar seg=r(seES)
end
```

Κώδικας Stata για πραγματοποίηση μονομεταβλητής μετα-ανάλυσης

```
set more off
file open meta using resultmyDL5.txt, write append
file write meta "gene"
file write meta ","
file write meta "g"
file write meta ","
file write meta "seg" _n
foreach var of varlist a1bg - vtrna3_1p {
di ""
di ""
di in ye "Performing meta-analysis for `var' gene"
di ""
myDL5,x(`var') type(type) study(study) effect(random)
file write meta "`var'"
file write meta ","
file write meta "`r(g)'"
file write meta ","
file write meta "`r(seg)'"_n
}
file close meta
```

Κώδικας Stata για πραγματοποίηση μονομεταβλητής μετα-ανάλυσης με bootstrap

```
set more off
file open meta using resultsbootstrap5.txt, write append
file write meta "gene"
file write meta ","
file write meta "g"
file write meta ","
file write meta "se" _n
foreach var of varlist a1bg-vtrna3_1p{
di ""
di ""
di in ye "Performing meta-analysis for `var' gene"
di ""
bootstrap g=r(g), nowarn nohead reps(200) strata(study type):
myDL5,x(`var') type(type) study(study) effect(random)
mat g=e(b)
local g=g[1,1]
mat se=e(se)
local se=se[1,1]
file write meta "`var'"
file write meta ","
file write meta "`g'"
file write meta ","
file write meta "`se'"_n
}
file close meta
```