



**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ
ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ**

**Ένα ευέλικτο πακέτο λογισμικού για το συνδυασμό
εξαρτημένων ή ανεξάρτητων p-values**

Ευγενία Νικολίτσα

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Επιβλέπων:
Καθηγητής
Παντελής Μπάγκος**

Λαμία, Ιούνιος 2024



**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ
ΒΙΟΙΑΤΡΙΚΗ**

**Ένα ευέλικτο πακέτο λογισμικού για το συνδυασμό
εξαρτημένων ή ανεξάρτητων p-values**

Νικολίτσα Ευγενία

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Επιβλέπων:
Καθηγητής
Παντελής Μπάγκος**

Λαμία, Ιούνιος 2024

Πίνακας περιεχομένων

Εισαγωγή.....	7
1.1 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΒΙΟΣΤΑΤΙΣΤΙΚΗΣ.....	10
1.1.1 Δείγματα.....	10
1.1.2 Εξαρτημένες και ανεξάρτητες μεταβλητές.....	10
1.1.3 Παράμετροι και Στατιστικά στοιχεία.....	11
1.1.4 Βασικές πιθανότητες.....	11
1.2 Τυχαίες μεταβλητές και Κατανομές Πιθανοτήτων.....	11
1.2.1 Η Κανονική κατανομή.....	12
1.3 Στατιστική Συμπερασματολογία – Έλεγχος Υποθέσεων.....	15
1.3.1 Εισαγωγή στον έλεγχο υποθέσεων.....	15
1.3.2 Επίπεδο σημαντικότητας ενός τεστ (μέγεθος ενός τεστ).....	17
1.3.3 Ισχύς μιας δοκιμής(test).....	17
1.3.4 Student's t-test για κανονικά καταμεμημένα δεδομένα.....	18
1.3.5 Λόγος πιθανοτήτων (Odds ratio(OR)).....	19
1.3.6 Συντελεστής συσχέτισης(Correlation Coefficient).....	20
1.5 Λογιστική Παλινδρόμηση (Logistic Regression).....	20
2. Μέθοδοι.....	22
2.1 Logitp.....	22
2.2 Meanp.....	23
2.3 Fisher's method.....	23
2.4 Stouffer's method.....	23
2.5 Inverse chi-square method.....	24
2.6 Binomial test.....	24
2.7 Bonferroni method.....	25
2.8 Bonferroni's correction.....	25
2.9 Tippett's method(MinP).....	26
2.10 Effective number of tests.....	26
2.11 Cauchy Distribution (CCT).....	27
2.12 MinP-CCT-MinP CCT-MinP-CCT.....	27
2.13 Harmonic Mean P-value Method (HMP).....	27
2.14 Empirical Brown's Method (EBM) και Kost's Method.....	28
2.15 Brown's Method by Yang.....	28
3. Επεξήγηση Κώδικα στην PYTHON.....	31
3.1 Logitp().....	31
3.2 Meanp().....	31

3.3 Fisher()	31
3.4 Stouffer()	32
3.5 Inverse_chi2()	32
3.5 Binomial_test()	33
3.6 Cauchy()	33
3.7 MinP()	34
3.8 HMP()	34
3.9 effective_number_of_tests_cheverud_nyholt()	34
3.10 h_function()	35
3.11 effective_number_of_tests_li_ji()	35
3.12 effective_number_of_tests_gao()	35
3.13 effective_number_of_tests_galwey()	36
3.14 bonferroni_method_with_effective_tests()	36
3.15 Bonferronis_correction_g	36
3.16 EmpiricalBrownsMethod	37
3.16 Brown's Method by Yang	39
4. Διαχείριση αρχείου	42
4.1 Διάβασμα αρχείου για τις μεθόδους 1-11	42
4.2 Διάβασμα αρχείου για Empirical Brown's Method(EBM) και Brown's Method by Yang και Kost's Method	43
4.3 Διάβασμα αρχείου για Bonferroni with effectives tests	43
5. Επεξήγηση Κώδικα στο STATA	45
5.1 Logitp	45
5.2 Meanp	45
5.3 Fisherp	46
5.4 Stoufferp	46
5.5 Invchi2p	46
5.6 Binomialp	47
5.7 Cauchyp	47
5.8 Minp	48
5.9 Bonferronip	48
5.10 Bonferronig_p	48
5.11 cheverud_nyholtp	49
5.12 Li_jip	49
5.13 Gaop	50
5.14 Galweyp	51

5.15	EBM.....	52
5.16	KostsMethodEBM.....	53
5.17	BrownbyYang.....	54
6.	Αποτελέσματα	56
7.	Συμπεράσματα	62
8.	Αναφορές.....	66
ΠΑΡΑΡΤΗΜΑ Α:ΚΩΔΙΚΑΣ ΣΤΗΝ PYTHON.....		68
ΠΑΡΑΡΤΗΜΑ Β: ΚΩΔΙΚΑΣ ΣΤΟ STATA		80

Εισαγωγή

Ο συνδυασμός των p-values είναι μια σημαντική τεχνική σε διάφορους τομείς. Για παράδειγμα, στις μελέτες συσχέτισης γονιδιώματος (GWAS), διαφορετικοί πολυμορφισμοί ενός νουκλεοτιδίου (SNPs) εντός ενός γονιδίου συνδυάζονται στις λεγόμενες gene-based μεθόδους ελέγχου για την ενσωμάτωση δεδομένων από σπάνιες παραλλαγές. Άλλες παρόμοιες προσεγγίσεις συναντώνται στην ανάλυση πολλαπλών χαρακτηριστικών (traits analysis) ή σε πιο γενικές μεθόδους multi-omics. Στην παρούσα εργασία, παρουσιάζεται ένα λογισμικό που υλοποιεί πληθώρα μεθόδων για το συνδυασμό τιμών p-values, τόσο ανεξάρτητων όσο και εξαρτημένων.

Η εργασία υλοποιεί τυποποιημένες μεθόδους που γενικά εφαρμόζονται για το συνδυασμό ανεξάρτητων p-values, όπως οι μέθοδοι των Fisher, Stouffer, Edgington, η μέθοδος του Inverse chi-squared, οι μέθοδοι logit και meanp και η διωνυμική (binomial) δοκιμή. Όσον αφορά τις εξαρτημένες p-values εφαρμόζεται η δοκιμή συνδυασμού Cauchy (CCT), τη μέθοδο MinP, οι συνδυασμένες μέθοδοι MinP-CCTMinP (MCM) και CCT-MinP-CCT (CMC), η μέθοδος Brown μαζί με τις τροποποιήσεις της, και η δοκιμή Harmonic Mean P-value (HMP). Εφαρμόζουμε επίσης διάφορες γνωστές μεθόδους που προσαρμόζονται για τον αριθμό των αποτελεσματικών δοκιμών (effective number of tests) στις μεθόδους (Gao, Galwey, Li_Ji, Cheverud_Nyholt) που υποθέτουν ανεξαρτησία. Το λογισμικό υλοποιείται στις γλώσσες **Python** και **STATA**.

Το πρόγραμμα λαμβάνει ως είσοδο ένα αρχείο κειμένου n γραμμών και m στηλών, καθώς και τις επικεφαλίδες, που περιέχουν τις τιμές p-values ή τα z-scores που πρέπει να συνδυαστούν. Διαφορετικές γραμμές μπορούν να αντιστοιχούν σε διαφορετικά SNPs ή γονίδια από ένα GWAS, ή ανιχνευτές/γονίδια από μελέτες μεταγραφώματος. Οι διαφορετικές στήλες μπορεί να αντιστοιχούν σε διαφορετικές στατιστικές δοκιμές, διαφορετικά χαρακτηριστικά, διαφορετικά SNPs εντός ενός γονιδίου κ.ο.κ., ανάλογα με τη ρύθμιση. Σε περίπτωση που το ίδιο SNP/γονίδιο εμφανίζεται σε πολλαπλές γραμμές, μπορεί να πραγματοποιηθεί μετα-ανάλυση πριν από το συνδυασμό των διαφορετικών τιμών p-values από διαφορετικές στήλες.

Το εργαλείο που παρουσιάζεται στην παρούσα εργασία είναι πολύ γρήγορο και εύκολο στη χρήση, απαιτώντας μόνο ελάχιστες εισαγωγές δεδομένων. Προσφέρει μια χρήσιμη πηγή για το συνδυασμό τόσο ανεξάρτητων όσο και εξαρτημένων p-values, ανταποκρινόμενο σε ποικίλες ανάγκες στην ανάλυση για ερευνητές σε διάφορους τομείς. Ανάλογα με τα δεδομένα εισόδου μπορεί να χρησιμοποιηθεί για δοκιμές με βάση τα γονίδια ή για την ανάλυση πολλαπλών χαρακτηριστικών σε GWAS, ή για το συνδυασμό ποικίλων δεδομένων multi-omics, όπως αυτά ενός TWAS, ενός colocalization ή μιας RNAseq. Σε σύγκριση με άλλα παρόμοια πακέτα (όπως το roor ή το metap) προσφέρει τη μεγαλύτερη δυνατή συλλογή μεθόδων ανάλυσης. Ο πηγαίος κώδικας θα είναι διαθέσιμος στο www.github.com/pbagos

1.1 ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΒΙΟΣΤΑΤΙΣΤΙΚΗΣ

1.1.1 Δείγματα

Τα δείγματα είναι παρατηρούμενα σύνολα μετρήσεων που αποτελούν υποσύνολα ενός αντίστοιχου πληθυσμού. Τα δείγματα χρησιμοποιούνται για την περιγραφή και την εξαγωγή συμπερασμάτων σχετικά με τους πληθυσμούς από τους οποίους προέρχονται. Οι στατιστικές μέθοδοι βασίζονται σε αυτά τα δείγματα που έχουν ληφθεί τυχαία από τον πληθυσμό. Ωστόσο, στην πράξη, αυτό σπάνια συμβαίνει. Θα υποθέτουμε πάντα ότι το δείγμα είναι αντιπροσωπευτικό του πληθυσμού που μας ενδιαφέρει. Τα παραδείγματα περιλαμβάνουν:

- SA1 Αριθμός CD4 100 ασθενών με AIDS την 1η Ιανουαρίου 1996.
- SA2 Ποσότητα δραστικού φαρμάκου σε 2000 κάψουλες Prozac των 20 mg που παρασκευάστηκαν τον Ιούνιο του 1996.
- SA3 Κατάσταση προηγούμενου εμφράγματος του μυοκαρδίου (ναι ή όχι) μεταξύ 150 ανδρών ηλικίας 45 έως 64 ετών.
- SA4 Βιοδιαθεσιμότητες μιας δόσης από το στόμα (σε σχέση με την ενδοφλέβια δόση) σε 24 υγιείς εθελοντές.
- SA5 Παρουσία ή απουσία εμφράγματος του μυοκαρδίου σε καθορισμένο χρονικό διάστημα για 310 υπέρταστικούς.
- ασθενείς που λάμβαναν αναστολείς διαύλων ασβεστίου.
- SA6 Αποτελέσματα τεστ (θετικά ή αρνητικά) μεταξύ 50 εγκύων γυναικών που έκαναν τεστ εγκυμοσύνης στο σπίτι.

1.1.2 Εξαρτημένες και ανεξάρτητες μεταβλητές

Οι εφαρμογές της στατιστικής βασίζονται συχνά στη σύγκριση αποτελεσμάτων μεταξύ ομάδων υποκειμένων. Δηλαδή, θα θέλαμε να συγκρίνουμε αποτελέσματα μεταξύ διαφορετικών πληθυσμών. Η μεταβλητή (ή οι μεταβλητές) που μετράμε ως το αποτέλεσμα που μας ενδιαφέρει είναι η εξαρτημένη μεταβλητή ή η απόκριση. Η μεταβλητή που καθορίζει το πληθυσμό από τον οποίο προκύπτει η μέτρηση είναι η ανεξάρτητη μεταβλητή (ή προγνωστικός παράγοντας). Για παράδειγμα, αν θέλουμε να συγκρίνουμε τις βιοδιαθεσιμότητες διαφόρων δοσολογικών μορφών, η εξαρτημένη μεταβλητή θα είναι AUC (περιοχή κάτω από την καμπύλη συγκέντρωσης-χρόνου), και η ανεξάρτητη μεταβλητή θα ήταν η δοσολογία μορφή. Θα επεκτείνουμε το εύρος των δυνατοτήτων των ανεξάρτητων μεταβλητών αργότερα στο κείμενο. Ο χαρακτηρισμός των μεταβλητών ως εξαρτημένες και ανεξάρτητες έχει την ιδιότητα να υποδηλώνει τη σχέση ότι η ανεξάρτητη μεταβλητή «οδηγεί» στο επίπεδο της εξαρτημένης μεταβλητής. Ωστόσο, έχουν κάποιες ατυχείς συνέπειες επίσης. Σε όλο το κείμενο, θα αναφερόμαστε στην εξαρτημένη μεταβλητή ως την απόκριση και την/τις ανεξάρτητη/ες μεταβλητή/ές ως επεξηγηματικές μεταβλητές.

1.1.3 Παράμετροι και Στατιστικά στοιχεία

Τα στατιστικά στοιχεία είναι αριθμητικά περιγραφικά μέτρα που αντιστοιχούν σε δείγματα. Θα χρησιμοποιήσουμε το γενικό συμβολισμό $\hat{\theta}$ για την αναπαράσταση των στατιστικών. Δεδομένου ότι τα δείγματα είναι «τυχαία υποσύνολα» του πληθυσμού, τα στατιστικά είναι τυχαίες μεταβλητές με την έννοια ότι διαφορετικά δείγματα θα δώσουν διαφορετικές τιμές του στατιστικού. Στην περίπτωση των αριθμητικών μετρήσεων, ας υποθέσουμε ότι έχουμε n μετρήσεις στο δείγμα μας και ότι τις ονομάζουμε $y_1, y_2, \dots, y_n$. Στη συνέχεια, υπολογίζουμε τη μέση τιμή του δείγματος, τη **διακύμανση**, την **τυπική απόκλιση** και το **συντελεστή διακύμανσης** ως εξής:

$$\begin{aligned}\hat{\mu} = \bar{y} &= \frac{\sum_{i=1}^n y_i}{n} = \frac{y_1 + y_2 + \dots + y_n}{n} \\ s^2 &= \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n - 1} = \frac{(y_1 - \bar{y})^2 + (y_2 - \bar{y})^2 + \dots + (y_n - \bar{y})^2}{n - 1} \quad s = \sqrt{s^2} \\ CV &= \left(\frac{s}{\bar{y}} \right) 100\%\end{aligned}$$

Τα στατιστικά που βασίζονται σε δείγματα θα χρησιμοποιηθούν για την εκτίμηση των παραμέτρων που αντιστοιχούν σε πληθυσμούς, όπως καθώς και για τον έλεγχο υποθέσεων σχετικά με τις πραγματικές τιμές των παραμέτρων.

1.1.4 Βασικές πιθανότητες

Η πιθανότητα χρησιμοποιείται για τη μέτρηση της «**πιθανοφάνειας**» ή των «**πιθανοτήτων**» ορισμένων γεγονότων (προκαθορισμένων αποτελεσμάτων) ενός πειράματος. Ορισμένοι κανόνες πιθανοτήτων θα χρησιμοποιηθούν σε αυτό το κείμενο και περιγράφονται εδώ. Αρχικά θα ορίσουμε 2 γεγονότα A και B , με πιθανότητες $P(A)$ και $P(B)$, αντίστοιχα. Η **τομή** των γεγονότων A και B είναι το γεγονός ότι συμβαίνουν **και το A και το B** , με τον συμβολισμό AB (μερικές φορές γράφεται $A \cap B$). Η **ένωση** των γεγονότων A και B είναι το γεγονός ότι συμβαίνει **είτε το A είτε το B** (γράφεται $A \cup B$). Το **συμπλήρωμα** του γεγονότος A είναι το γεγονός ότι **το A δεν συμβαίνει**, και γράφεται $\bar{A}$. Μερικοί χρήσιμοι κανόνες για την απόκτηση αυτών και άλλων πιθανοτήτων περιλαμβάνουν:

- $P(A \cup B) = P(A) + P(B) - P(AB)$
- $P(A|B) = P(A \text{ occurs given } B \text{ has occurred}) = \frac{P(AB)}{P(B)}$ (assuming $P(B) > 0$)
- $P(AB) = P(A)P(B|A) = P(B)P(A|B)$
- $P(\bar{A}) = 1 - P(A)$

Μια ορισμένη κατάσταση συμβαίνει όταν τα γεγονότα A και B λέγεται ότι είναι **ανεξάρτητα**. Αυτό συμβαίνει όταν $P(A|B) = P(A)$, ή ισοδύναμα $P(B|A) = P(B)$, στην περίπτωση αυτή, $P(AB) = P(A)P(B)$.

1.2 Τυχαίες μεταβλητές και Κατανομές Πιθανοτήτων

Προηγουμένως παρουσιάσαμε τις έννοιες των πληθυσμών, των δειγμάτων, των μεταβλητών και της στατιστικής. Υπενθυμίζουμε ότι παρατηρούμε ένα δείγμα από κάποιο πληθυσμό, μετράμε μια μεταβλητή έκβασης (κατηγορική ή αριθμητική) σε κάθε στοιχείο του δείγματος και υπολογίζουμε στατιστικά στοιχεία για να περιγράψουμε το δείγμα (όπως $\bar{Y}$ ή $\hat{\pi}$). Οι μεταβλητές που παρατηρούνται στο δείγμα, καθώς και τα στατιστικά που χρησιμοποιούνται για τον υπολογισμό τους, είναι τυχαίες μεταβλητές. Η ιδέα είναι ότι υπάρχει ένας πληθυσμός τέτοιων αποτελεσμάτων και παρατηρούμε ένα τυχαίο υποσύνολο αυτών στο δείγμα μας. Η συλλογή όλων των πιθανών αποτελεσμάτων στον πληθυσμό, και των αντίστοιχων σχετικών συχνοτήτων τους ονομάζεται κατανομή πιθανοτήτων. Οι κατανομές πιθανοτήτων μπορούν να ταξινομηθούν ως συνεχείς ή διακριτές. Σε κάθε περίπτωση, υπάρχουν παράμετροι που σχετίζονται με την κατανομή πιθανοτήτων- θα εστιάσουμε την προσοχή μας στην εξαγωγή συμπερασμάτων σχετικά με τον μέσο όρο του πληθυσμού (μ), τη διάμεσο και το ποσοστό που έχει κάποιο χαρακτηριστικό (π).

1.2.1 Η Κανονική κατανομή

Πολλές συνεχείς μεταβλητές, καθώς και οι στατιστικές του δείγματος, έχουν κατανομές πιθανότητας που μπορούν να θεωρηθούν ως καμπανοειδείς. Δηλαδή, οι περισσότερες από τις μετρήσεις στον πληθυσμό τείνουν να πέφτουν γύρω από κάποιο κεντρικό σημείο (το μέσο όρο, μ), και καθώς η απόσταση από το μ αυξάνεται, η σχετική συχνότητα των μετρήσεων μειώνεται. Μεταβλητές (και στατιστικές) που έχουν κατανομές πιθανότητας που μπορούν να χαρακτηριστούν με αυτόν τον τρόπο λέγεται ότι είναι κανονικά κατανεμημένες. Κανονικές κατανομές είναι συμμετρικές κατανομές που ταξινομούνται με βάση τη μέση τιμή τους (μ), και την τυπική τους απόκλιση (σ).

Οι τυχαίες μεταβλητές που είναι προσεγγιστικά κανονικά κατανεμημένες έχουν τις ακόλουθες ιδιότητες:

1. Περίπου οι μισές (50%) των μετρήσεων πέφτουν πάνω (και επομένως, οι μισές πέφτουν κάτω) από την μέση τιμή.
2. Περίπου το 68% των μετρήσεων πέφτουν εντός μιας τυπικής απόκλισης από τη μέση τιμή (σε εύρος $(\mu - \sigma, \mu + \sigma)$).
3. Περίπου το 95% των μετρήσεων εμπίπτουν εντός δύο τυπικών αποκλίσεων του μέσου όρου (στο εύρος $(\mu - 2\sigma, \mu + 2\sigma)$).
4. Σχεδόν όλες οι μετρήσεις βρίσκονται εντός τριών τυπικών αποκλίσεων από τη μέση τιμή.

Εάν μια τυχαία μεταβλητή, Y , είναι κανονικά κατανεμημένη, με μέση τιμή μ και τυπική απόκλιση σ , έχουμε

θα χρησιμοποιήσουμε τον συμβολισμό $Y \sim N(\mu, \sigma)$. Αν $Y \sim N(\mu, \sigma)$, μπορούμε να γράψουμε τις παραπάνω προτάσεις ως προς δηλώσεις πιθανότητας:

$$P(Y \geq \mu) = 0,50 \quad P(\mu - \sigma \leq Y \leq \mu + \sigma) = 0,68 \quad P(\mu - 2\sigma \leq Y \leq \mu + 2\sigma) = 0,95 \quad P(\mu - 3\sigma \leq Y \leq \mu + 3\sigma) \approx 1$$

Παράδειγμα 1.2.1 Τα ύψη (ή μήκη) των ενήλικων ζώων τείνουν να έχουν κατανομές που περιγράφονται καλά από κανονικές κατανομές. Τα σχήματα 2.1 και 2.2 δίνουν κατανομές σχετικών συχνοτήτων (ιστογράμματα) του ύψους των γυναικών και των ανδρών ηλικίας 25-34 ετών, αντίστοιχα (U.S. Bureau of Census, 1992). Σημειώστε ότι και τα δύο ιστογράμματα τείνουν να έχουν σχήμα καμπάνας, με τους περισσότερους ανθρώπους να πέφτουν σχετικά κοντά σε κάποιο γενικό μέσο όρο, με λιγότερα άτομα να εμπίπτουν σε περιοχές αυξανόμενης απόστασης από το μέσο όρο. Στο Σχήμα 2.3 δίνονται οι «εξομαλυμένες» κανονικές κατανομές για τις γυναίκες και τους άνδρες. Για

τις γυναίκες, ο μέσος όρος είναι 63,5 ίντσες με τυπική απόκλιση 2,5. Μεταξύ των ανδρών, ο μέσος όρος είναι 68,5 ίντσες με τυπική απόκλιση 2,7. Εάν συμβολίσουμε ένα τυχαία επιλεγμένο ύψος γυναίκας ως Y_F , και ένα τυχαία επιλεγμένο ύψος άνδρα ως Y_M , θα μπορούσαμε να γράψουμε: $Y_F \sim N(63.5, 2.5)$ και $Y_M \sim N(68.5, 2.7)$.

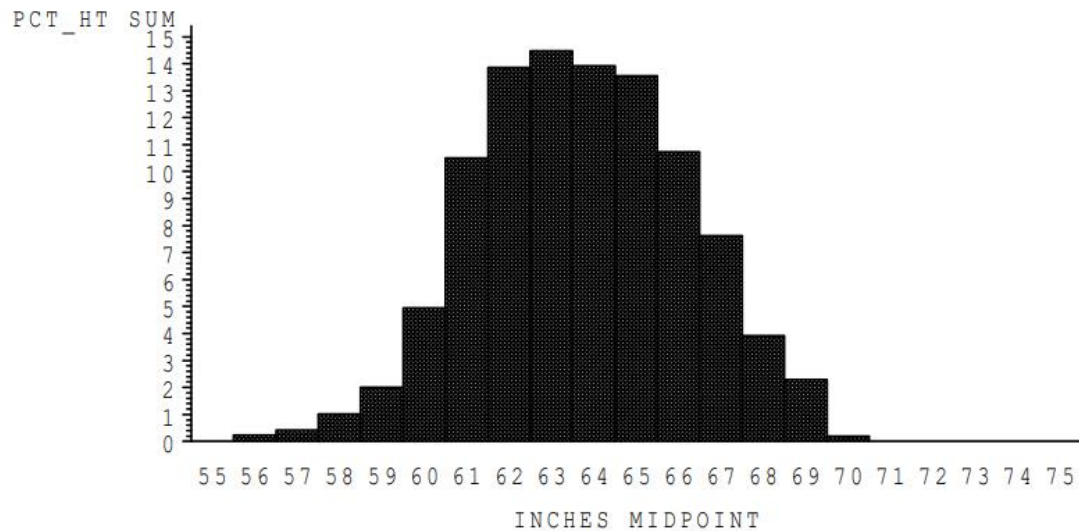


Figure 2.1: Histogram of the population of U.S. female heights (age 25–34)

Ενώ υπάρχει άπειρος αριθμός συνδυασμών των μ και σ , και επομένως άπειρος αριθμός πιθανών κανονικών κατανομών, όλες έχουν το ίδιο φάσμα μετρήσεων που κυμαίνονται σε ένα σταθερό αριθμό τυπικών αποκλίσεων από τη μέση τιμή. Μπορούμε να τυποποιήσουμε μια κανονική τυχαία μεταβλητή με βάση τον ακόλουθο μετασχηματισμό:

$$Z = \frac{Y - \mu}{\sigma}$$

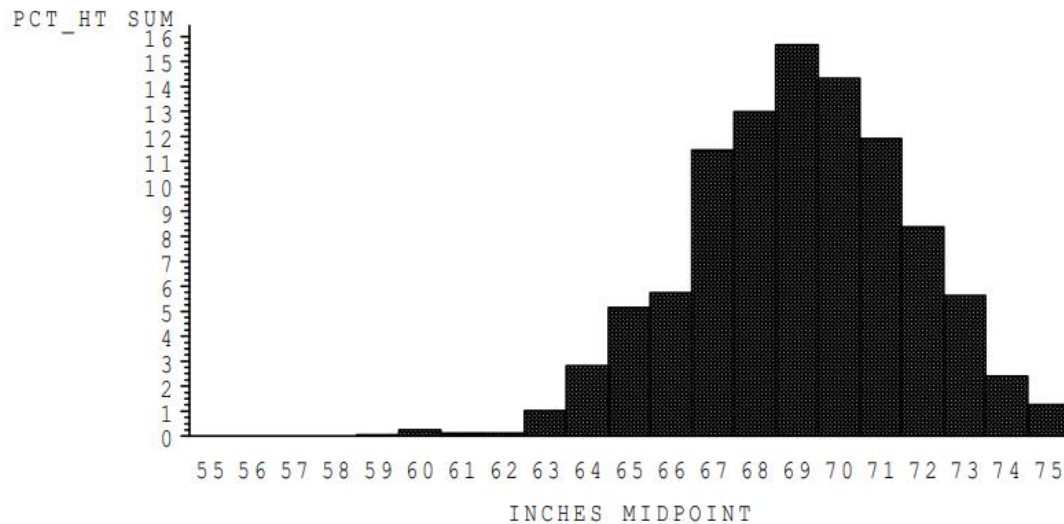


Figure 2.2: Histogram of the population of U.S. male heights (age 25-34)

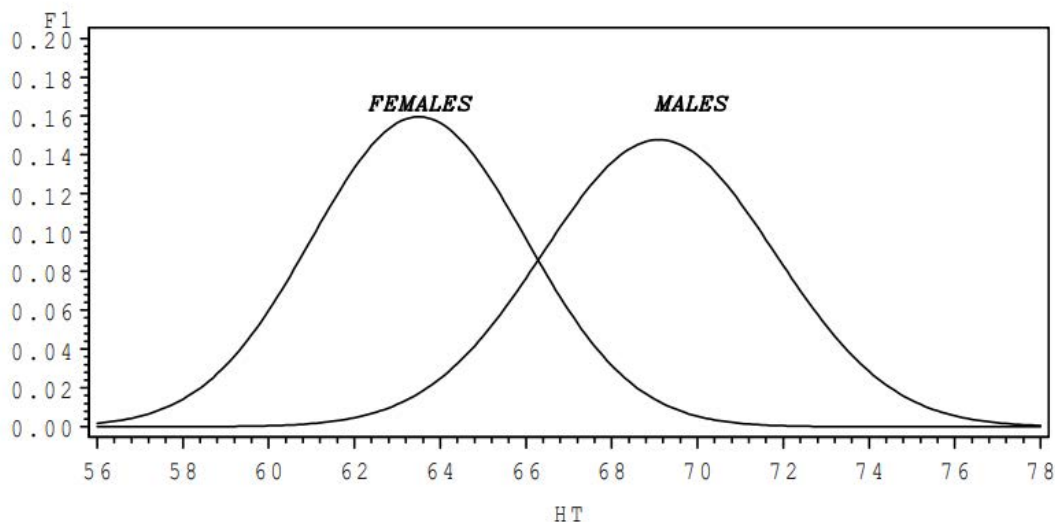


Figure 2.3: Normal distributions used to approximate the distributions of heights of males and females (age 25-34)

Σημειώστε ότι το Z δίνει ένα μέτρο της «σχετικής θέσης» για το Y - είναι ο αριθμός των τυπικών αποκλίσεων πάνω (αν είναι θετική) ή κάτω (αν είναι αρνητική) από τη μέση τιμή που έχει η Y . Για παράδειγμα, εάν μια τυχαία επιλεγμένη γυναίκα από τον πληθυσμό που περιγράφηκε στην προηγούμενη ενότητα παρατηρείται και το ύψος της, Y , βρίσκεται να είναι 68 ίντσες, μπορούμε να τυποποιήσουμε το ύψος της ως εξής:

$$Z = \frac{Y - \mu}{\sigma} = \frac{68 - 63.5}{2.5} = 1.80$$

Έτσι, μια γυναίκα με ύψος 1.80 m είναι 1.80 τυπικές αποκλίσεις πάνω από το μέσο ύψος των γυναικών αυτής της ηλικιακής ομάδας. Η τυχαία μεταβλητή, Z , ονομάζεται τυπική κανονική τυχαία μεταβλητή και γράφεται ως $Z \sim N(0, 1)$.

Πίνακες που δίνουν τις πιθανότητες των περιοχών κάτω από την τυπική κανονική κατανομή δίνονται στα περισσότερα βιβλία στατιστικής. Θα χρησιμοποιήσουμε τον συμβολισμό ότι z_α είναι η τιμή που είναι τέτοια ώστε η πιθανότητα ότι z τυπική

κανονική τυχαία μεταβλητή είναι μεγαλύτερη από z_α είναι α . Το Σχήμα 2.4 απεικονίζει αυτή την ιδέα.

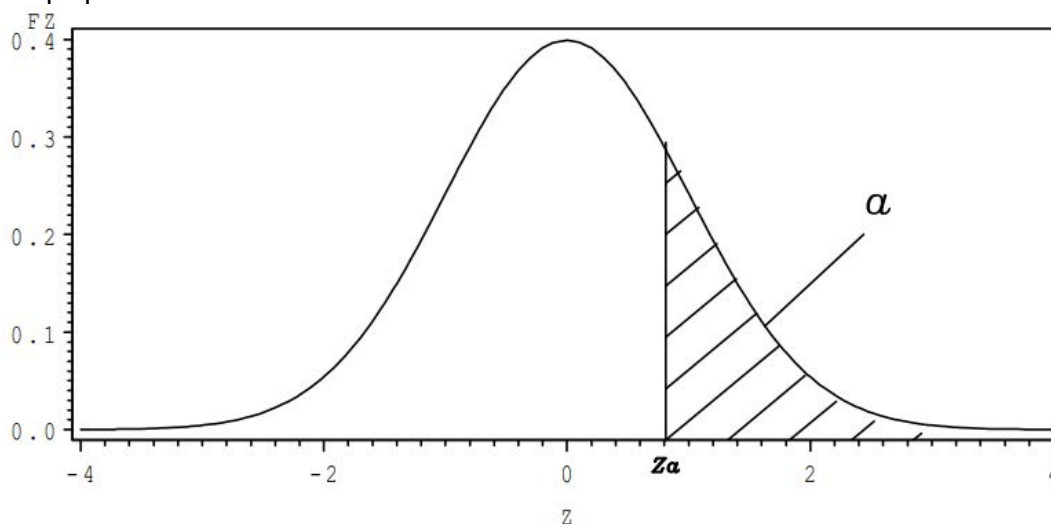


Figure 2.4: Standard Normal Distribution depicting z_α and the probability Z exceeds it

1.3 Στατιστική Συμπερασματολογία – Έλεγχος Υποθέσεων

1.3.1 Εισαγωγή στον έλεγχο υποθέσεων

Ο έλεγχος υποθέσεων περιλαμβάνει την επιλογή μεταξύ δύο προτάσεων σχετικά με μια άγνωστη τιμή της παραμέτρου. Στις περιπτώσεις αυτού του κεφαλαίου, οι προτάσεις θα είναι ότι οι μέσοι όροι είναι ίσοι (δεν υπάρχει επίδραση του φαρμάκου), ή ότι οι μέσοι όροι διαφέρουν (επίδραση του φαρμάκου). Εργαζόμαστε με την υπόθεση ότι δεν υπάρχει επίδραση του φαρμάκου, και θα απορρίψουμε τον ισχυρισμό αυτό μόνο εάν τα δεδομένα του δείγματος μας δώσουν επαρκείς αποδείξεις εναντίον του, υπέρ του ισχυρισμού ότι το φάρμακο είναι αποτελεσματικό. Η διαδικασία ελέγχου περιλαμβάνει τη δημιουργία δύο αντιφατικών δηλώσεων σχετικά με την αληθινή τιμή της παραμέτρου, γνωστές ως **μηδενική υπόθεση** και **εναλλακτική υπόθεση**, αντίστοιχα. Υποθέτουμε ότι η μηδενική υπόθεση είναι αληθής και συνήθως (αλλά όχι πάντα) επιθυμούμε να δείξουμε ότι η εναλλακτική είναι πράγματι αληθής. Αφού συλλέξουμε δειγματικά δεδομένα, υπολογίζουμε μια στατιστική δοκιμασίας η οποία χρησιμοποιείται ως απόδειξη υπέρ ή κατά της μηδενικής υπόθεσης (την οποία υποθέτουμε ότι είναι αληθής κατά τον υπολογισμό του στατιστικού ελέγχου). Το σύνολο των τιμών του στατιστικού τεστ που θεωρούμε ότι παρέχουν επαρκείς αποδείξεις για την απόρριψη της μηδενικής υπόθεσης υπέρ της εναλλακτικής υπόθεσης ονομάζεται **περιοχή απόρριψης**. Η πιθανότητα που θα μπορούσαμε να αποκτήσουμε εξίσου ισχυρές ή ισχυρότερες αποδείξεις κατά της μηδενικής υπόθεσης (όταν αυτή είναι αληθής) από αυτές που παρατηρήσαμε από τα δεδομένα του δείγματός μας ονομάζεται παρατηρούμενο **επίπεδο σημαντικότητας** ή **p-value**.

Μπορούμε να αναλύσουμε την πραγματικότητα και την απόφασή μας στον Πίνακα 3.1. Θα θέλαμε να

		Decision	
		H_0 True	H_0 False
Actual State	H_0 True	Correct Decision	Type I Error
	H_0 False	Type II Error	Correct Decision

Table 3.1: Possible outcomes of a hypothesis test

ορίσουμε την περιοχή απόρριψης ώστε να διατηρείται η πιθανότητα σφάλματος τύπου I (α) και η πιθανότητα σφάλματος τύπου II (β) όσο το δυνατόν μικρότερη. Δυστυχώς για ένα σταθερό μέγεθος δείγματος, αν προσπαθήσουμε να μειώσουμε το α , αυξάνουμε αυτομάτως το β και το αντίστροφο. Θα ορίσουμε περιοχές απόρριψης για τον έλεγχο του α , και δεν θα ασχοληθούμε με το β . Εδώ α είναι η πιθανότητα που απορρίπτουμε τη μηδενική υπόθεση όταν είναι αληθής.

Μπορούμε να γράψουμε τη γενική μορφή ενός ελέγχου υποθέσεων μεγάλου δείγματος με τα ακόλουθα βήματα, όπου $\hat{\theta}$ είναι μια παράμετρος του πληθυσμού που έχει έναν εκτιμητή ($\hat{\theta}$) που είναι κατά προσέγγιση κανονικός.

1. $H_0 : \theta = \theta_0$
2. $H_A : \theta \neq \theta_0$ or $H_A : \theta > \theta_0$ or $H_A : \theta < \theta_0$ (πρέπει να είναι σαφές ποια εναλλακτική λύση είναι κατάλληλη)
3. T.S. : $z_{obs} = \frac{\hat{\theta} - \theta_0}{\hat{\sigma}_{\hat{\theta}}}$
4. R.R.: $|z_{obs}| > z_{\alpha/2}$ or $z_{obs} > z_{\alpha}$ or $z_{obs} < -z_{\alpha}$ (η οποία R.R. εξαρτάται από την εναλλακτική υπόθεση που χρησιμοποιείται)

5. p-value: $2P(Z > |z_{obs}|)$ ή $P(Z > z_{obs})$ ή $P(Z < z_{obs})$ (και πάλι, ανάλογα με το ποια εναλλακτική χρησιμοποιείται).

Σε όλες τις περιπτώσεις, μια τιμή p-value μικρότερη από α αντιστοιχεί σε ένα στατιστικό ελέγχου που βρίσκεται στην περιοχή απόρριψης (απόρριψη της H_0), και μια τιμή p-value μεγαλύτερη από α αντιστοιχεί σε ένα στατιστικό ελέγχου που δεν βρίσκεται στην περιοχή απόρριψης (αποτυχία απόρριψης της H_0). Θα παρουσιάσουμε αυτή την ιδέα σε ένα παράδειγμα παρακάτω.

	Sample 1	Sample 2
Mean	$\bar{y}_1$	$\bar{y}_2$
Std Dev	s_1	s_2
Sample Size	n_1	n_2

Table 3.2: Sample statistics needed for a large-sample test of $\mu_1 - \mu_2$

$H_0 : \mu_1 - \mu_2 = 0$	Test Statistic: $z_{obs} = \frac{(\bar{y}_1 - \bar{y}_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$	
Alternative Hypothesis	Rejection Region	P-Value
$H_A : \mu_1 - \mu_2 > 0$	RR: $z_{obs} > z_{\alpha}$	$p - val = P(Z \geq z_{obs})$
$H_A : \mu_1 - \mu_2 < 0$	RR: $z_{obs} < -z_{\alpha}$	$p - val = P(Z \leq z_{obs})$
$H_A : \mu_1 - \mu_2 \neq 0$	RR: $ z_{obs} > z_{\alpha/2}$	$p - val = 2P(Z \geq z_{obs})$

Table 3.3: Large-sample test of $\mu_1 - \mu_2 = 0$ vs various alternatives

1.3.2 Επίπεδο σημαντικότητας ενός τεστ (μέγεθος ενός τεστ)

Είδαμε στο προηγούμενο κεφάλαιο ότι για μεγάλα δείγματα, ο μέσος όρος του δείγματος, $\bar{Y}$, έχει μια δειγματοληπτική κατανομή που είναι κατά προσέγγιση κανονική, με μέσο όρο μ και τυπικό σφάλμα $\sigma_{\bar{Y}} = \sigma/\sqrt{n}$. Αυτό μπορεί να επεκταθεί στην περίπτωση όπου έχουμε δύο δειγματικούς μέσους από δύο πληθυσμούς (με ανεξάρτητα δείγματα). Στην περίπτωση αυτή, όταν τα n_1 και n_2 είναι μεγάλα, έχουμε:

$$\bar{Y}_1 - \bar{Y}_2 \sim N(\mu_1 - \mu_2, \sigma_{\bar{Y}_1 - \bar{Y}_2}) \quad \sigma_{\bar{Y}_1 - \bar{Y}_2} = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$$

Έτσι, όταν $\mu_1 = \mu_2$ (το φάρμακο είναι αναποτελεσματικό), το

$$Z = \frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

είναι κανονικό. Συνεπώς, αν ελέγχουμε $H_0 : \mu_1 - \mu_2 = 0$ έναντι $H_A : \mu_1 - \mu_2 > 0$, θα έχουμε την ακόλουθη πιθανότητα

$$P\left(\frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \geq z_\alpha\right) = P(Z \geq z_\alpha) = \alpha$$

Αυτό δίνει μια φυσική περιοχή απόρριψης για το στατιστικό ελέγχου μας για τον έλεγχο του α , την πιθανότητα να απορρίψουμε τη μηδενική υπόθεση όταν είναι αληθής (ισχυρισμός ότι ένα αναποτελεσματικό φάρμακο είναι αποτελεσματικό). Ομοίως, η τιμή p-value είναι η πιθανότητα να πάρουμε μεγαλύτερες τιμές του z_{obs} (και επομένως $\bar{Y}_1 - \bar{Y}_2$) από αυτές που παρατηρήσαμε. Αυτή είναι η περιοχή στα δεξιά του στατιστικού ελέγχου μας σύμφωνα με την τυπική κανονική κατανομή. Ο πίνακας 3.5 δίνει τιμές αποκοπής για διάφορες τιμές του α για καθεμία από τις τρεις εναλλακτικές υποθέσεις. Η τιμή α για ένα τεστ ονομάζεται επίπεδο σημαντικότητας ή μέγεθος του.

α	Alternative Hypothesis (H_A)		
	$H_A : \mu_1 - \mu_2 > 0$	$H_A : \mu_1 - \mu_2 < 0$	$H_A : \mu_1 - \mu_2 \neq 0$
.10	$z_{obs} \geq 1.282$	$z_{obs} \leq -1.282$	$ z_{obs} \geq 1.645$
.05	$z_{obs} \geq 1.645$	$z_{obs} \leq -1.645$	$ z_{obs} \geq 1.96$
.01	$z_{obs} \geq 2.326$	$z_{obs} \leq -2.326$	$ z_{obs} \geq 2.576$

Table 3.5: Rejection Regions for tests of size α

1.3.3 Ισχύς μιας δοκιμής(test)

Η ισχύς μιας δοκιμής αντιστοιχεί στην πιθανότητα απόρριψης της μηδενικής υπόθεσης όταν αυτή είναι ψευδής. Δηλαδή, όσον αφορά μια δοκιμή αποτελεσματικότητας ενός φαρμάκου, η πιθανότητα να συμπεράνουμε σωστά ότι το φάρμακο είναι αποτελεσματικό, όταν στην πραγματικότητα είναι. Υπάρχουν πολλές τιμές παραμέτρων που αντιστοιχούν στην εναλλακτική υπόθεση, και η ισχύς εξαρτάται από την πραγματική τιμή των $\mu_1 - \mu_2$ (ή οποιασδήποτε άλλης παραμέτρου).

Ένας ερευνητής ενδιαφέρεται να ελέγξει την $H_0 : \mu_1 - \mu_2 = 0$ έναντι $H_0 : \mu_1 - \mu_2 > 0$ με $\alpha = 0,05$ επίπεδο σημαντικότητας. Υποθέτουμε, ότι οι διακυμάνσεις κάθε πληθυσμού είναι γνωστές, και $\sigma_1^2 = \sigma_2^2 = 25.0$. Ο ερευνητής παίρνει δείγματα μεγέθους $n_1 = n_2 = 25$ από κάθε πληθυσμό. Η περιοχή απόρριψης είναι ορισμένη κάτω από την H_0 όπως (δες Πίνακα 3.5):

$$z_{obs} = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \geq 1.645 \quad \Rightarrow \quad \bar{y}_1 - \bar{y}_2 \geq 1.645 \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} = 1.645 \sqrt{2.0} = 2.326$$

Δηλαδή, θα απορρίψουμε την H_0 εάν $\bar{y}_1 - \bar{y}_2 \geq 2,326$. Υπό τη μηδενική υπόθεση ($\mu_1 = \mu_2$), η πιθανότητα ότι $\bar{y}_1 - \bar{y}_2$ είναι μεγαλύτερη από 2,326 είναι 0,05 (πολύ απίθανο).

Τώρα η ισχύς του ελέγχου αντιπροσωπεύει την πιθανότητα να απορρίψουμε σωστά τη μηδενική υπόθεση όταν αυτή είναι ψευδής (η εναλλακτική είναι αληθής, $\mu_1 > \mu_2$). Μας ενδιαφέρει να βρούμε την πιθανότητα ότι $\bar{Y}_1 - \bar{Y}_2 \geq 2,326$ όταν η H_A είναι αληθής. Αυτή η πιθανότητα εξαρτάται από την πραγματική τιμή του $\mu_1 - \mu_2$, δεδομένου ότι

$$\bar{Y}_1 - \bar{Y}_2 \sim N\left(\mu_1 - \mu_2, \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}\right)$$

Αν υποθέσουμε ότι $\mu_1 - \mu_2 = 3,0$, τότε έχουμε το ακόλουθο αποτέλεσμα, το οποίο χρησιμοποιείται για να ληφθεί η ισχύς της δοκιμής:

$$P(\bar{Y}_1 - \bar{Y}_2 \geq 2.326) = P\left(\frac{(\bar{Y}_1 - \bar{Y}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \geq \frac{2.326 - 3.0}{\sqrt{2.0}}\right) = P(Z \geq -0.48) = .6844$$

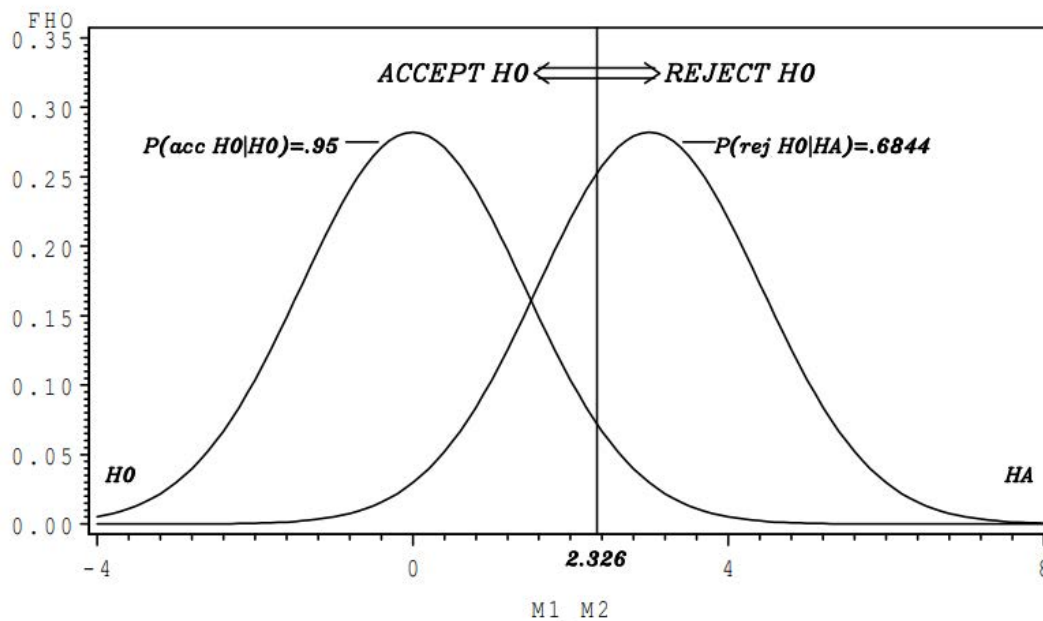


Figure 3.1: Significance level and power of test under H_0 ($\mu_1 - \mu_2 = 0$) and H_A ($\mu_1 - \mu_2 = 3$)

1.3.4 Student's t-test για κανονικά κατανομημένα δεδομένα

Αυτή η διαδικασία είναι πανομοιότυπη με το τεστ-μεγάλου δείγματος, με τη διαφορά ότι οι κρίσιμες τιμές για την απόρριψη βασίζονται στην κατανομή t με $\nu = n_1 + n_2 - 2$ βαθμούς ελευθερίας. Θα υποθέσουμε ότι οι δύο πληθυσμιακές αποκλίσεις είναι ίσες στο t -τεστ 2 δειγμάτων. Εάν δεν είναι, απλές προσαρμογές μπορούν να γίνουν για να προκύψει το κατάλληλο τεστ. Στη συνέχεια «συγκεντρώνουμε» τις 2 δειγματικές αποκλίσεις για να πάρουμε μία εκτίμηση της κοινής διακύμανσης $\sigma^2 = \sigma_1^2 = \sigma_2^2$. Αυτή η εκτίμηση, την οποία θα γράφουμε s_p^2 υπολογίζεται ως εξής:

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}.$$

Ο έλεγχος της υπόθεσης που αφορά $\mu_1 - \mu_2$ εκτελείται ως εξής:

1. $H_0: \mu_1 - \mu_2 = 0$
2. $H_A: \mu_1 - \mu_2 \neq 0$ ή $H_A: \mu_1 - \mu_2 > 0$ ή $H_A: \mu_1 - \mu_2 < 0$
3. T.S.: $t_{obs} = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{s_p^2 (\frac{1}{n_1} + \frac{1}{n_2})}}$
4. R.R.: $|t_{obs}| > t_{a/2, n_1+n_2-2}$ ή $t_{obs} > t_{a/2, n_1+n_2-2}$ ή $t_{obs} < t_{a/2, n_1+n_2-2}$ (η οποία R.R. εξαρτάται από την εναλλακτική υπόθεση που χρησιμοποιείται)
5. p-value: $2P(T > |t_{obs}|)$ or $P(T > t_{obs})$ or $P(T < t_{obs})$ (πάλι εξαρτάται από την εναλλακτική υπόθεση που χρησιμοποιείται)

1.3.5 Λόγος πιθανοτήτων (Odds ratio(OR))

Για τις αναδρομικές μελέτες (case-control) τα υποκείμενα προσδιορίζονται ως περιπτώσεις (D) ή έλεγχοι ($\bar{D}$), και παρατηρείται αν τα υποκείμενα είχαν εκτεθεί στον παράγοντα κινδύνου (E) ή όχι ($\bar{E}$). Δεδομένου ότι δεν κάνουμε δειγματοληψία από τους πληθυσμούς των εκτεθειμένων και των μη εκτεθειμένων, και παρατηρούμε εάν ή όχι εμφανιστεί η ασθένεια (όπως κάνουμε στις προοπτικές μελέτες), δεν μπορούμε να εκτιμήσουμε την $P(D|E)$ ή την $P(D|\bar{E})$. Πρώτα ορίζουμε τις πιθανότητες εμφάνισης ενός συμβάντος. Αν π είναι η πιθανότητα να συμβεί ένα γεγονός, η πιθανότητα ο να συμβεί είναι $\pi = \pi / (1 - \pi)$. Οι πιθανότητες μπορούν να ερμηνευθούν ως ο αριθμός των φορών που θα συμβεί το γεγονός για κάθε φορά που δεν θα συμβεί αν η διαδικασία επαναληφθεί πολλές φορές.

		Disease State		Total
		D (Present)	$\bar{D}$ (Absent)	
Exposure State	E (Present)	n_{11}	n_{12}	$n_{1.}$
	$\bar{E}$ (Absent)	n_{21}	n_{22}	$n_{2.}$
Total		$n_{.1}$	$n_{.2}$	n

Table 5.2: A 2×2 Contingency Table

Ενώ δεν μπορούμε να υπολογίσουμε το $P(D|E)$ ή το $P(D|\bar{E})$ για αναδρομικές μελέτες, μπορούμε να υπολογίσουμε τις πιθανότητες ότι ένα άτομο εκτέθηκε δεδομένου ότι έχει την ασθένεια, και τις πιθανότητες ότι ένα άτομο εκτέθηκε δεδομένου ότι δεν έχει την ασθένεια. Ο λόγος αυτών των δύο πιθανοτήτων ονομάζεται λόγος πιθανοτήτων **odds ratio (OR)**. Οι λόγος πιθανοτήτων είναι παρόμοιος με τον σχετικό κίνδυνο, και είναι ουσιαστικά ισοδύναμος με αυτόν όταν ο επιπολασμός της της νόσου ($P(D)$) είναι χαμηλός. Ο λόγος των πιθανοτήτων υπολογίζεται ως εξής:

$$OR = \frac{\text{odds of disease given exposed}}{\text{odds of disease given unexposed}} = \frac{\text{odds of exposure given diseased}}{\text{odds of exposure given not diseased}} = \frac{n_{11}/n_{21}}{n_{12}/n_{22}} = \frac{n_{11}n_{22}}{n_{12}n_{21}}$$

Ο λόγος των πιθανοτήτων είναι παρόμοιος με τον σχετικό κίνδυνο υπό την έννοια ότι είναι μια πληθυσμιακή παράμετρος που πρέπει να εκτιμηθεί, καθώς και οι ερμηνείες που σχετίζονται με αυτήν από την άποψη του κατά πόσον η τιμή της είναι πάνω από, κάτω ή ίση με 1.0. Αυτό σημαίνει ότι:

- Εάν ο λόγος των πιθανοτήτων είναι μεγαλύτερος από 1.0, οι πιθανότητες (και συνεπώς η πιθανότητα) εμφάνισης της νόσου είναι υψηλότερες μεταξύ των εκτεθειμένων από ό,τι των μη εκτεθειμένων.
- Εάν ο λόγος των πιθανοτήτων είναι μικρότερος από 1.0, οι πιθανότητες (και συνεπώς η πιθανότητα) της νόσου είναι μικρότερες μεταξύ των εκτεθειμένων από ό,τι των μη εκτεθειμένων.

- Εάν ο λόγος των πιθανοτήτων είναι 1.0, οι πιθανότητες (και συνεπώς η πιθανότητα) της νόσου είναι ίδιες και για τις δύο ομάδες (δεν υπάρχει συσχέτιση μεταξύ της έκθεσης στον παράγοντα κινδύνου και της κατάστασης της νόσου).

$$s^2 = MSE = \frac{SSE}{n-2} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n-2} = \frac{S_{yy} - \frac{(S_{xy})^2}{S_{xx}}}{n-2}.$$

1.3.6 Συντελεστής συσχέτισης (Correlation Coefficient)

Σε πολλές περιπτώσεις, θα θέλαμε να λάβουμε ένα μέτρο της ισχύος της γραμμικής συσχέτισης μεταξύ των μεταβλητών y και x . Ένα μέτρο αυτής της συσχέτισης είναι ο συντελεστής συσχέτισης του *Pearson*. Αυτό το μέτρο, που συμβολίζεται με r , είναι ένας αριθμός που μπορεί να κυμαίνεται από -1 έως +1. Μια τιμή του r κοντά στο 0 υποδηλώνει ότι υπάρχει πολύ μικρή συσχέτιση μεταξύ των δύο μεταβλητών (το y τείνει να μην αυξάνεται ή να μην μειώνεται καθώς αυξάνεται το x). Μια θετική τιμή του r σημαίνει ότι υπάρχει θετική συσχέτιση μεταξύ y και x (το y τείνει να αυξάνεται καθώς αυξάνεται το x). Ομοίως, μια αρνητική τιμή σημαίνει ότι υπάρχει αρνητική συσχέτιση (y τείνει να μειώνεται καθώς αυξάνεται το x). Εάν το r είναι είτε +1 είτε -1, σημαίνει ότι τα δεδομένα εμπίπτουν σε ευθεία γραμμή ($SSE = 0$) που έχει είτε θετική είτε αρνητική κλίση, ανάλογα με το πρόσημο του r . Ο τύπος για τον υπολογισμό του r είναι

$$r = \frac{S_{xy}}{\sqrt{S_{xx}S_{yy}}}.$$

Σημειώστε ότι το πρόσημο του r είναι πάντα το ίδιο με το πρόσημο του β_1 . Επίσης, μπορεί να ελεγχθεί αν ο συντελεστής συσχέτισης πληθυσμού είναι 0 (δεν υπάρχει γραμμική συσχέτιση μεταξύ y και x) και είναι αλγεβρικά ισοδύναμο με τον έλεγχο $H_0 : \beta_1 = 0$.

Ένα άλλο μέτρο συσχέτισης που έχει σαφέστερη φυσική ερμηνεία από το r είναι το r^2 , ο συντελεστής προσδιορισμού. Αυτό το μέτρο κυμαίνεται πάντα μεταξύ 0 και 1, οπότε δεν αντικατοπτρίζει αν τα y και x συνδέονται θετικά ή αρνητικά, και αντιπροσωπεύει το ποσοστό της συνολικής μεταβολής στη μεταβλητή απόκρισης που «εξηγείται» από την προσαρμογή της παλινδρόμησης στο x . Ο τύπος για το r^2 είναι

$$r^2 = (r)^2 = \frac{S_{yy} - SSE}{S_{yy}}.$$

Σημειώστε ότι το $S_{yy} = \sum_{i=1}^n (y_i - \bar{y})^2$ αντιπροσωπεύει τη συνολική διακύμανση της μεταβλητής απόκρισης, ενώ $SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ αντιπροσωπεύει τη διακύμανση των παρατηρούμενων αποκρίσεων σχετικά με την προσαρμοσμένη εξίσωση (λαμβάνοντας υπόψη το x). Αυτός είναι ο λόγος για τον οποίο μερικές φορές λέμε ότι r^2 είναι «το ποσοστό της διακύμανσης στο y που «εξηγείται» από το x ».

Όταν τα δεδομένα (ή τα σφάλματα) δεν είναι σαφώς κανονικά κατανομημένα (μεγάλες ακραίες τιμές γενικά εμφανίζονται στα διαγράμματα), ένα μη παραμετρικό μέτρο συσχέτισης είναι ο συντελεστής συσχέτισης του *Spearman*. Ο συντελεστής συσχέτισης του *Spearman* περιλαμβάνει την κατάταξη των τιμών x από το 1 έως το n και των τιμών y από το 1 έως n , και στη συνέχεια υπολογίζει το r όπως στο συντελεστή συσχέτισης *Pearson*, αλλά αντικαθιστά τα ακατέργαστα δεδομένα με τα ranks.

1.5 Λογιστική Παλινδρόμηση (Logistic Regression)

Η λογιστική παλινδρόμηση είναι μια μέθοδος που είναι χρήσιμη όταν η μεταβλητή απόκρισης είναι διχοτομική (έχει δύο επίπεδα) και τουλάχιστον μία από τις επεξηγηματικές μεταβλητές είναι συνεχής. Σε αυτή την περίπτωση, μοντελοποιούμε την πιθανότητα η μεταβλητή απόκρισης να πάρει το επίπεδο που μας ενδιαφέρει (επιτυχία) ως συνάρτηση της (των) επεξηγηματικής(-ών) μεταβλητής(-ών). Για την περίπτωση που έχουμε μία ανεξάρτητη μεταβλητή, θα προσαρμόσουμε το ακόλουθο μοντέλο:

$$\pi(x) = \frac{e^{\alpha+\beta x}}{1 + e^{\alpha+\beta x}}$$

Εδώ $\pi(x)$ είναι η πιθανότητα η μεταβλητή απόκρισης να λάβει το χαρακτηριστικό που μας ενδιαφέρει (επιτυχία) και x είναι το επίπεδο της αριθμητικής επεξηγηματικής μεταβλητής. Ενδιαφέρον παρουσιάζει το αν $\beta = 0$ ή όχι. Αν $\beta = 0$, τότε η πιθανότητα επιτυχίας είναι ανεξάρτητη από το επίπεδο του x . Εάν $\beta > 0$, τότε η πιθανότητα της επιτυχίας αυξάνεται καθώς αυξάνεται το x , αντίστροφα, εάν $\beta < 0$, τότε η πιθανότητα επιτυχίας μειώνεται καθώς αυξάνεται το x . Για να ελέγξουμε αυτή την υπόθεση, διεξάγουμε τον ακόλουθο έλεγχο, με βάση τις εκτιμήσεις που λαμβάνονται από ένα στατιστικό πακέτο υπολογιστών:

1. $H_0 : \beta = 0$
2. $H_A : \beta \neq 0$
3. T.S.: $X_{obs}^2 = \left(\frac{\hat{\beta}}{\hat{\sigma}_{\hat{\beta}}} \right)^2$
4. R.R.: $X_{obs}^2 \geq \chi_{\alpha,1}^2$
5. p-value: $P(\chi_1^2 \geq X_{obs}^2)$

Στη λογιστική παλινδρόμηση, $e^{\hat{\beta}}$ είναι η μεταβολή του λόγου των πιθανοτήτων μιας επιτυχίας στα επίπεδα της επεξηγηματικής μεταβλητής με διαφορά μίας μονάδας. Υπενθυμίζεται ότι οι πιθανότητες να συμβεί ένα γεγονός είναι:

$$o = \frac{\pi}{1 - \pi} \implies o(x) = \frac{\pi(x)}{1 - \pi(x)}.$$

Τότε ο λόγος των πιθανοτήτων στο $x + 1$ προς τις πιθανότητες στο x (ο λόγος των πιθανοτήτων) μπορεί να γραφεί (ανεξάρτητη του x) ως εξής:

$$OR(x+1, x) = \frac{o(x+1)}{o(x)} = \frac{e^{\alpha+\beta(x+1)}}{e^{\alpha+\beta x}} = e^{\beta}$$

Αν ο λόγος πιθανοτήτων είναι μεγαλύτερος από 1 σημαίνει ότι η πιθανότητα επιτυχίας αυξάνεται όσο αυξάνεται το x , και αν ο λόγος πιθανοτήτων είναι μικρότερος από 1 σημαίνει ότι η πιθανότητα επιτυχίας μειώνεται καθώς αυξάνεται το x . Συχνά, στις μελέτες αναφέρεται ο λόγος των πιθανοτήτων και όχι το $\hat{\beta}$.

2. Μέθοδοι

Η εργασία υλοποιεί τυποποιημένες μεθόδους που γενικά εφαρμόζονται για το συνδυασμό ανεξάρτητων p-values, όπως οι μέθοδοι των Fisher, Stouffer, Edgington, η μέθοδος του Inverse chi-squared, οι μέθοδοι logit και meanp και η διωνυμική (binomial) δοκιμή. Όσον αφορά τις εξαρτημένες p-values εφαρμόζεται η δοκιμή συνδυασμού Cauchy (CCT), τη μέθοδο MinP, οι συνδυασμένες μέθοδοι MinP-CCTMinP (MCM) και CCT-MinP-CCT (CMC), η μέθοδος Brown μαζί με τις τροποποιήσεις της, και η δοκιμή Harmonic Mean P-value (HMP). Εφαρμόζουμε επίσης διάφορες γνωστές μεθόδους που προσαρμόζονται για τον αριθμό των αποτελεσματικών δοκιμών (effective number of tests) στις μεθόδους (Gao, Galwey, Li_Ji, Cheverud_Nyholt) που υποθέτουν ανεξαρτησία. Το λογισμικό υλοποιείται στις γλώσσες **Python** και **STATA**.

2.1 Logitp

Η συνάρτηση logitp συνδυάζει τις τιμές των p-values χρησιμοποιώντας τη μεταβλητή

$$t = - \frac{\sum \log \left(\frac{p}{1-p} \right)}{C}$$

που αποτελεί την τιμή του Student's t, όπου

$$C = \sqrt{k\pi^2(5k+2)/3(5k+4)}$$

και k το πλήθος των μελετών. Τέλος εφαρμόζεται η μεταβλητή t εφαρμόζεται στον τύπο

$$p_c = 1 - T(|t|, df)$$

, όπου $T(|t|, df)$ είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution function (cdf)) της κατανομής T με $df = 2 \cdot k$ βαθμούς ελευθερίας.

Η συνάρτηση logit είναι ένα στατιστικό μοντέλο που μοντελοποιεί τις λογαριθμικές πιθανότητες ενός γεγονότος ως γραμμικό συνδυασμό μιας ή περισσότερων ανεξάρτητων μεταβλητών και ορίζεται ως η αντίστροφη $g = \sigma^{-1}$ της συνάρτησης logistic και ισοδύναμα,

$$\log \left(\frac{p(x)}{1-p(x)} \right) = \text{logit}(p(x)) = g(p(x)) = \beta_0 + \beta_1 x$$

Μετά την εκθετικοποίηση και στις δύο πλευρές έχουμε:

$$\frac{p(x)}{1-p(x)} = e^{\beta_0 + \beta_1 x}$$

, όπου $\frac{p}{1-p} = \text{odds ratio (OR)}$ ο λόγος σχετικών πιθανοτήτων.

Αν υπάρχουν πολλαπλές μεταβλητές, η παραπάνω έκφραση $\beta_0 + \beta_1 x_1$ γίνεται $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_m x_m$. Στη συνέχεια, όταν αυτή η σχέση χρησιμοποιείται στην

εξίσωση που συνδέει τις λογαριθμικές πιθανότητες επιτυχίας με τις τιμές των εκτιμητών, η συνάρτηση logit ισοδυναμεί με

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_m x_m$$

και

$$p = \frac{1}{1 + b^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_m x_m)}}$$

, όπου συνήθως $b = e$.

η γραμμική παλινδρόμηση μετατρέπεται σε μια πολλαπλή παλινδρόμηση με m εκτιμητές και τις παραμέτρους β_j με $j = 1, 2, \dots, m$.

2.2 Meanp

Η μέθοδος meanp υπολογίζει μια συνδυασμένη τιμή p υπολογίζοντας το μέσο όρο των p -value εισόδου, μετατρέποντας το αποτέλεσμα σε z -score και στη συνέχεια μετατρέποντας το z -score ξανά σε p -value. Το z -score υπολογίζεται ως:

$$z = (0.5 - \bar{p})\sqrt{12k}$$

, όπου $\bar{p}$ είναι ο μέσος όρος των p -values, και k είναι το πλήθος των p -values. Τα combined p -values υπολογίζονται με την εξίσωση:

$$p_c = 1 - N(z)$$

, όπου N η αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (cumulative distribution function of normal distribution).

2.3 Fisher's method

Η μέθοδος του Fisher (Fisher 1932) είναι μια από τις πιο συχνά χρησιμοποιούμενες τεχνικές για το συνδυασμό ενός συνόλου p -values. Υποθέτοντας ότι $p_i \sim \text{Uniform}(0, 1)$, μπορεί να αποδειχθεί ότι $-2\ln(p_i)$ είναι κατανομημένο σύμφωνα με την κατανομή X^2 με 2 βαθμούς ελευθερίας (df). Στη συνέχεια μπορεί να υπολογιστεί ένα συνδυασμένο στατιστικό ελέγχου

$$X^2 = -2 \sum_{i=1}^k \ln(p_i)$$

Δεδομένου ότι το άθροισμα ανεξάρτητων τυχαίων μεταβλητών κατανομημένων με κατανομή X^2 είναι επίσης κατανομημένο με κατανομή X^2 (με βαθμούς ελευθερίας ίσους με το άθροισμα των βαθμών ελευθερίας των αθροισμάτων), η X^2 ακολουθεί κατανομή X^2 με $df = 2k$ υπό την κοινή μηδενική υπόθεση. Επομένως, το combined p -value υπολογίζεται με την εξίσωση $p_c = 1 - F(X^2, 2k)$, όπου $F(\cdot, f)$ δηλώνει την αθροιστική συνάρτηση κατανομής της κατανομής X^2 (cumulative distribution function of X^2 distribution) με $df = f$.

2.4 Stouffer's method

Η μέθοδος Stouffer (Stouffer et al. 1949) είναι μια άλλη γνωστή μέθοδος για το συνδυασμό ενός συνόλου p -values. Έστω η $\Phi(\cdot)$ δηλώνει την αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (cumulative distribution function of normal

distribution) και $\Phi^{-1}(\cdot)$ η αντίστροφη της. Δεδομένου ότι τα p-values είναι κανονικά καταναμημένα υπό την μηδενική υπόθεση, το $z_i = \Phi^{-1}(1 - p_i)$ ακολουθεί κανονική κατανομή και επομένως το

$$z = \sum_{i=1}^k z_i / \sqrt{k}$$

είναι επίσης κανονικά καταναμημένο υπό την μηδενική υπόθεση. Πρέπει να σημειωθεί ότι χρησιμοποιείται $1 - p_i$ για τον υπολογισμό του z_i ώστε να δίνει μια μεγάλη θετική τιμή όταν το p_i είναι μικρό. Το combined p-value υπολογίζεται με την εξίσωση $p_c = 1 - \Phi(z)$.

2.5 Inverse chi-square method

Αντίστοιχα με τη μέθοδο “inverse-normal” του Stouffer, στη μέθοδο inverse chi-square (που δεν πρέπει να μπερδεύεται με τη μέθοδο του Fisher) είναι δυνατό τα p-values να συνδυαστούν με την αθροιστική συνάρτηση κατανομής της κατανομής X^2 (cumulative distribution function of X^2 distribution) με έναν βαθμό ελευθερίας ($df = 1$) η οποία ορίζεται ως $F^{-1}(\cdot, 1)$. Συνεπώς η

$$X^2 = \sum_{i=1}^k F^{-1}(1 - p_i, 1)$$

ακολουθεί κατανομή X^2 με $df = k$ και το combined p-value υπολογίζεται με την εξίσωση $p_c = 1 - F(X^2, k)$.

2.6 Binomial test

Το διωνυμικό τεστ (Wilkinson 1951) προκύπτει με φυσικό τρόπο αφού σημειωθεί ότι η απόρριψη έναντι της μη απόρριψης μιας αληθινής μηδενικής υπόθεσης μπορεί να θεωρηθεί ως τυχαία μεταβλητή καταναμημένη σύμφωνα με την κατανομή Bernoulli. με την πιθανότητα απόρριψης να συμβολίζεται με a . Συνεπώς, όταν και οι k μηδενικές υποθέσεις είναι αληθείς, ο αριθμός των δοκιμών που οδηγούν σε απόρριψη, ας πούμε r , ακολουθεί μια διωνυμική κατανομή, δηλαδή $r \sim \text{Binomial}(k, a)$. Επομένως, μπορούμε να υπολογίσουμε το combined p-value με την εξίσωση

$$p_c = \sum_{x=r}^k \binom{k}{x} a^x (1 - a)^{k-x}$$

η οποία δίνει την πιθανότητα να έχουμε r ή περισσότερες απορρίψεις («επιτυχίες») υπό την κοινή μηδενική υπόθεση. Συνεπώς, εάν $p_c \leq a_c$, απορρίπτουμε την κοινή μηδενική υπόθεση. Μπορούμε επομένως να θεωρήσουμε το διωνυμικό τεστ ως μια μέθοδο εξέτασης του κατά πόσον υπάρχει «περίσσεια σημαντικών αποτελεσμάτων» μεταξύ του συνόλου των k δοκιμών (tests) όταν υποθέτουμε ότι η κοινή μηδενική υπόθεση είναι αληθής.

Πρέπει να σημειωθεί ότι το a (δηλ. το κατώφλι για τον χαρακτηρισμό μιας μεμονωμένης δοκιμής ως σημαντικής ή μη) μπορεί να είναι διαφορετικό από το a_c (το επιθυμητό ποσοστό Type I error για τη συνδυασμένη δοκιμή (test)). Στην πραγματικότητα, λόγω της διακριτής φύσης του p_c όπως υπολογίζεται με την παραπάνω εξίσωση, το ποσοστό Type I error της διωνυμικής δοκιμής περιορίζεται από το a_c από πάνω, δηλαδή το πραγματικό ποσοστό Type I error μπορεί να είναι

χαμηλότερο από το a_c (δηλαδή, $\Pr(p_c \leq a_c) \leq a_c$ για το διωνυμικό τεστ, με αυστηρή ισότητα που ισχύει για τις μεθόδους του Fisher, του Stouffer, και του Inverse chi-square).

2.7 Bonferroni method

Η διόρθωση Bonferroni (Bonferroni correction) (π.χ. Goeman and Solari 2014), αναμφισβήτητα μία από τις πιο συχνά εφαρμοζόμενες τεχνικές διόρθωσης πολλαπλών δοκιμών στην έρευνα. Το ποσοστό Type I error ανά υπόθεση (δηλαδή το α) διαιρείται με τον αριθμό των ταυτόχρονων δοκιμών (δηλ. k). Στη συνέχεια, το α/k χρησιμοποιείται ως όριο σημαντικότητας για τις επιμέρους δοκιμές, το οποίο μπορεί να αποδεικνύεται ότι ελέγχει το συνολικό ποσοστό Type I error (δηλαδή η πιθανότητα ύπαρξης τουλάχιστον ενός Type I error σε όλες τις k δοκιμές είναι τότε το πολύ α). Ισοδύναμα, μπορούμε να πολλαπλασιάσουμε το p-value κάθε δοκιμής επί k και να χρησιμοποιήσουμε το α ως κατώφλι σημαντικότητας (δηλαδή, οι δοκιμές για τις οποίες $p_i \times k \leq \alpha$ είναι σημαντικές).

Παρόλο που η μέθοδος Bonferroni διατυπώνεται συνήθως ως διόρθωση πολλαπλών δοκιμών, είναι επίσης δυνατό να χρησιμοποιηθεί ως μέθοδος συνδυασμού p-values. Το κίνητρο είναι το γεγονός ότι η κοινή μηδενική υπόθεση θα πρέπει να απορρίπτεται εάν τουλάχιστον μία από τις επιμέρους μηδενικές υποθέσεις απορρίπτεται. Για το σκοπό αυτό, χρειάζεται να λάβουμε υπόψη μόνο το μικρότερο τι p-value των δοκιμών των οποίων τα p-values πρόκειται να συνδυαστούν. Επομένως, το combined p-value υπολογίζεται με τη σχέση

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times k)$$

2.8 Bonferroni's correction

Ο πολλαπλός έλεγχος και η επίδρασή του στα ποσοστά σφάλματος τύπου I και II συζητούνται συχνά στη στατιστική. Προτείνεται μια νέα προσαρμογή της παραδοσιακής προσέγγισης Bonferroni που λαμβάνει υπόψη συσχετιζόμενα δεδομένα. Ένας απλός διορθωτικός παράγοντας που βασίζεται στην ενδοταξική συσχέτιση εφαρμόζεται στην τυπική μέθοδο Bonferroni για να ξεπεραστούν οι ελλείψεις της τυπικής προσαρμογής Bonferroni, αλλά ταυτοχρόνως να διατηρηθούν τα πλεονεκτήματά της.

Μια προσαρμοσμένη τιμή για το g , δηλαδή το g , που ενσωματώνει τη συσχέτιση των g ελέγχων υποθέσεων είναι επιθυμητή.

$$g^* = (g + 1) - [1 + (g - 1)\widehat{ICC}]$$

, όπου g είναι ο αριθμός των εξεταζόμενων τελικών σημείων και $\widehat{ICC}$ είναι η εκτίμηση της ενδοταξικής συσχέτισης με βάση το δείγμα. Το διορθωμένο προσαρμοσμένο p-value ισούται με τη μη προσαρμοσμένη $p_value \times g^*$. Σημειώστε ότι όταν $\widehat{ICC} = 0$, ο συντελεστής διόρθωσης ICC (g^*) ανάγεται στην τυπική προσαρμογή Bonferroni, δηλαδή,

$$g^* = (g + 1) - [1 + (g - 1) * 0] = 0$$

Ομοίως, όταν $\widehat{ICC} = 1$,

$$g^* = (g + 1) - [1 + (g - 1) * 1] = 1.$$

2.9 Tippett's method(MinP)

Η μέθοδος του Tippett (επίσης γνωστή ως διόρθωση Dunn-Šidák, με βάση τις μεθόδους Dunn 1958 και Šidák 1967) είναι άλλη μία διόρθωση πολλαπλών δοκιμών (Tippett 1931). Το ποσοστό σφάλματος ανά υπόθεση ορίζεται σε $1 - (1 - \alpha)^{1/k}$, το οποίο ελέγχει το συνολικό ποσοστό Type I error στο α , αλλά εγγυάται ότι το κάνει μόνο υπό συνθήκες ανεξαρτησίας. Η κοινή μηδενική υπόθεση επομένως θα απορρίπτεται εάν $\min(p_i) \leq 1 - (1 - \alpha)^{1/k}$. Αντίστοιχα, το combined p-value με βάση τη μέθοδο Tippett δίνεται ως εξής

$$p_c = 1 - (1 - \min(p_1, p_2, \dots, p_k))^k.$$

2.10 Effective number of tests

Κατά τη διεξαγωγή k δοκιμών που είναι εξαρτημένες μπορούμε να σκεφτούμε αυτό ως ένα σενάριο όπου ουσιαστικά διεξάγουμε έναν μικρότερο αριθμό ανεξάρτητων δοκιμών. Για παράδειγμα, ας υποθέσουμε ότι τρεις SNPs ελέγχονται για τη συσχέτισή τους με κάποιον φαινότυπο που μας ενδιαφέρει. Εάν τα SNPs βρίσκονται σε τέλεια ισορροπία σύνδεσης (Linkage Disequilibrium) (και συνεπώς δεν υπάρχει συσχέτιση μεταξύ των γονοτύπων στις τρεις θέσεις), διεξάγουμε τρεις ανεξάρτητες δοκιμές. Από την άλλη πλευρά, εάν τα SNPs βρίσκονται σε τέλεια LD, τότε στην πραγματικότητα διεξάγουμε την ίδια δοκιμή τρεις φορές, οδηγώντας σε τρία πανομοιότυπα p-values, και επομένως ο πραγματικός αριθμός δοκιμών είναι ένας. Ανάλογα με τον βαθμό LD μεταξύ των SNPs, ο πραγματικός αριθμός των δοκιμών θα είναι επομένως κάπου μεταξύ αυτών των δύο άκρων. Έχουν προταθεί διάφορες μέθοδοι με βάση αυτή την ιδέα για την ποσοτικοποίηση του βαθμού εξάρτησης μεταξύ πολλαπλών δοκιμών. Όλες αυτές οι μέθοδοι ξεκινούν με την εφαρμογή PCA στον R_x , τον πίνακα με τις συσχετίσεις μεταξύ των στοιχείων (Cheverud 2001 - Galwey 2009 - Gao et al. 2008 - Li and Ji 2005, Nyholt 2004). Έστω λ_i η i -οστή ιδιοτιμή, έτσι ώστε $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$. Τότε οι Cheverud (2001) και Nyholt (2004) προτείνουν την εκτίμηση του *effective number of tests* με

$$m_{CN} = 1 + (k - 1) \left(1 - \frac{\text{Var}(\lambda)}{k}\right)$$

, όπου $\text{Var}(\lambda)$ είναι η παρατηρούμενη δειγματική διακύμανση μεταξύ των k ιδιοτιμών. Εναλλακτικά, οι Li και Ji (2005) προτείνουν τον υπολογισμό του *effective number of tests* με

$$m_{LJ} = \sum_{i=1}^k h(|\lambda_i|)$$

, όπου $h(x) = I(x \geq 1) + (x - \lfloor x \rfloor)$ και $\lfloor \cdot \rfloor$ είναι η συνάρτηση floor.

Μια άλλη προσέγγιση, που περιγράφεται από τους Gao et al. (2008), εκτιμά τον *effective number of tests* με

$$m_{GAO} = \underset{x \in \{1, \dots, k\}}{\text{argmin}} \left(\frac{\sum_{i=1}^x \lambda_i}{\sum_{i=1}^k \lambda_i} > C \right)$$

, όπου C είναι μια παράμετρος που ορίζεται από τον χρήστη και η οποία συνήθως ορίζεται σε 0,995 (δηλαδή, m_{GAO} είναι ο αριθμός των κύριων συνιστωσών που απαιτούνται ώστε να λαμβάνεται υπόψη το $C \times 100\%$ της συνολικής διακύμανσης). Τέλος, σύμφωνα με τον Galwey (2009), ο *effective number of tests* πρέπει να υπολογίζεται με

$$m_{GAL} = \frac{(\sum_{i=1}^k \sqrt{\lambda_i'})^2}{\sum_{i=1}^k \lambda_i}$$

, όπου $\lambda_i' = \max [0, \lambda_i]$.

Συνεπώς, για τη μέθοδο του Bonferroni, απλά αντικαθιστούμε το k με το m και ως εκ τούτου το combined p-value υπολογίζεται με

$$p_c = \min (1, \min(p_1, p_2, \dots, p_k) \times m).$$

2.11 Cauchy Distribution (CCT)

Ας υποθέσουμε ότι θέλουμε να συνδυάσουμε k p-values $P_1, P_2, \dots, P_k$ μέσω του ελέγχου της συνολικής μηδενικής υπόθεσης $H_1 = \cap_i H_{i,0}$ έναντι της συνολικής εναλλακτικής υπόθεσης $H_1 = \cup_i H_{i,1}$, όπου $H_{i,0}$ και $H_{i,1}$ είναι οι επιμέρους μηδενικές και εναλλακτικές υποθέσεις αντίστοιχα, για τη μελέτη i ($i = 1, 2, \dots, k$). Υποθέτουμε επίσης ότι υπό την H_0 , κάθε $P_i \sim U(0, 1)$, μια ομοιόμορφη κατανομή μεταξύ 0 και 1. Χρησιμοποιώντας την τυπική κατανομή Cauchy, $C(0, 1)$ μπορούμε να μετατρέψουμε πρώτα τα p-values σε $T = w_i \sum_{i=1}^k \tan \{(0.5 - p_i)\pi\}$ για $i = 1, 2, \dots, k$.

Το CCT κατασκευάζεται χρησιμοποιώντας το ακόλουθο στατιστικό: $T = w_i \sum_{i=1}^k T_i$ όπου $w_i \geq 0$ είναι τα βάρη που ικανοποιούν τη σχέση $\sum_{i=1}^k w_i = 1$ και το combined p-value υπολογίζεται ως $p_{CCT} = P[C(0, 1) \geq t]$, όπου $t = \sum_{i=1}^k w_i t_i$ είναι το παρατηρούμενο στατιστικό T .

Το CCT δεν προτιμάται για το συνδυασμό ανεξάρτητων p-values.

2.12 MinP-CCT-MinP CCT-MinP-CCT

Για να ενσωματωθούν οι καλές ιδιότητες τόσο από το CCT όσο και από το MinP, προτείνουμε τις ακόλουθες δύο δοκιμές. Η πρώτη ονομάζεται MinP-CCT-MinP (MCM), της οποίας το p-value υπολογίζεται ως εξής: $p_{MCM} = 2\min \{p_{CCT}, p_{MinP}, 0.5\}$ όπου το p_{MinP} προκύπτει με την εφαρμογή του MinP στα αρχικά p-values που πρόκειται να συνδυαστούν. Η δεύτερη ονομάζεται CCT-MinP-CCT-(CMC), της οποίας το p-value υπολογίζεται ως εξής: $p_{CMC} = CCT\{p_{CCT}, p_{MinP}\}$.

2.13 Harmonic Mean P-value Method (HMP)

Η μέθοδος Harmonic Mean p-Value ελέγχει το FWER ενώ βελτιώνει σημαντικά τη στατιστική ισχύ συνδυάζοντας εξαρτημένες δοκιμές (dependent tests) με τη χρήση του γενικευμένου κεντρικού οριακού θεωρήματος (generalized central limit theorem). Το familywise error rate (FWER) είναι η πιθανότητα λανθασμένης απόρριψης μιας μηδενικής υπέρ μιας εναλλακτικής υπόθεσης σε μία ή περισσότερες από όλες τις δοκιμών που πραγματοποιήθηκαν. Στην ουσία, η μέθοδος χρησιμοποιεί τον αρμονικό μέσο όρο των p-values,

$$\bar{p} = \frac{L}{\sum_{i=0}^L 1/p_i}$$

με βάση τον οποίο υπολογίζεται μια ασυμπτωτικά ακριβής συνδυασμένη τιμή p-value χρησιμοποιώντας την κατανομή Landau.

$$p_c = \int_{1/p_c}^{\infty} f_{Landau}(x | \log L + 0.874, \frac{\pi}{2}) dx.$$

Σύμφωνα με τις υποθέσεις του γενικευμένου κεντρικού οριακού θεωρήματος (generalized central limit theorem), αυτό το μετασχηματισμένο p-value γίνεται πιο ακριβές όσο αυξάνεται ο αριθμός των δοκιμών (tests) L . Για τον υπολογισμό του συνδυασμένου p-value χρησιμοποιείται η κατανομή Landau, της οποίας η συνάρτηση πυκνότητας (density function) γράφεται

$$f_{Landau}(x|\mu, \sigma) = \frac{1}{\pi\sigma} \int_0^\infty e^{-t\frac{(x-\mu)}{\sigma} - \frac{2}{\pi}t \log t} \sin(2t) dt.$$

Η μέθοδος είναι ανθεκτική στη θετική εξάρτηση μεταξύ των p-values και δεν απαιτεί περαιτέρω δεδομένα εκτός από τις p-values που πρέπει να συνδυαστούν.

2.14 Empirical Brown's Method (EBM) και Kost's Method

Έστω k τιμές p-values, συμβολιζόμενες με P_i , που προκύπτουν από k στατιστικούς ελέγχους (tests) που βασίζονται σε k κανονικά κατανομημένες τυχαίες μεταβλητές, οι οποίες συμβολίζονται με X_i . Ο Fisher έδειξε ότι για ανεξάρτητες τιμές p-values, το στατιστικό $\Psi = -2 \sum_{i=1}^k \log P_i$ ακολουθεί κατανομή χ^2 με $2k$ βαθμούς ελευθερίας, $\Psi \sim \chi_{2k}^2$. Ο Brown επέκτεινε τη γενικευμένη μέθοδο του Fisher για την περίπτωση εξαρτημένων p-values χρησιμοποιώντας μία ανα-κλιμακούμενη κατανομή χ^2 $\Psi \sim c\chi_{2f}^2$.

Οι σταθερές αντιπροσωπεύουν έναν ανα-κλιμακούμενο αριθμό βαθμών ελευθερίας f και έναν συντελεστή κλίμακας c (που είναι ο λόγος των βαθμών ελευθερίας των μεθόδων του Fisher και του Brown. Ο Brown υπολόγισε αυτές τις σταθερές εξισώνοντας τα δύο πρώτα στιγμιότυπα των Ψ και $c\chi_{2f}^2$ καταλήγοντας σε

$$f = \frac{E[\Psi]^2}{\text{var}[\Psi]} \quad \text{και} \quad c = \frac{\text{var}[\Psi]}{2E[\Psi]} = \frac{k}{f}.$$

Επιπλέον, ο Brown έδειξε ότι η αναμενόμενη τιμή και η διακύμανση του Ψ μπορούν να υπολογιστούν απευθείας μέσω αριθμητικής ολοκλήρωσης για να βρεθεί η συνδιακύμανση, αντίστοιχα

$$E[\Psi] = 2k \quad \text{και} \quad \text{var}[\Psi] = 4k + 2 \sum_{i < j} \text{cov}(-2\log P_i, -2\log P_j).$$

Ωστόσο, η αριθμητική ολοκλήρωση δεν είναι εφικτή για μεγάλα σύνολα δεδομένων λόγω υπολογιστικής πολυπλοκότητας. Οι Kost και McDermott προσάρμοσαν ένα πολυώνυμο τρίτης τάξης για να προσεγγίσουν αυτή τη συνδιακύμανση

$$\text{cov}(-2\log P_i, -2\log P_j) \approx 3.263\rho_{ij} + 0.710\rho_{ij}^2 + 0.027\rho_{ij}^3$$

όπου ρ_{ij} είναι η συσχέτιση μεταξύ των τυχαίων μεταβλητών P_i και P_j . Το combined p-value δίνεται τότε από τη σχέση

$$p_c = 1.0 - \Phi_{2f}(\psi/c)$$

όπου $\psi = -2 \sum_{i=1}^k \log P_i$ και Φ_{2f} είναι η αθροιστική συνάρτηση κατανομής της χ_{2f}^2 .

2.15 Brown's Method by Yang

Για συσχετιζόμενα φαινοτυπικά χαρακτηριστικά, το T είναι το άθροισμα των εξαρτημένων στατιστικών χ^2 . Οι Brown και Yang έδειξαν ότι, υπό τη μηδενική υπόθεση της μη ύπαρξης συσχέτισης μεταξύ ενός SNP και πολυπαραγοντικών φαινοτυπικών χαρακτηριστικών, η κατανομή του στατιστικού T ακολουθεί μια κατανομή χ^2 με scale ($\gamma\chi_\nu^2$) ή, ισοδύναμα, μια κατανομή gamma με shape parameter $\nu/2$ και scale parameter 2γ . Επομένως, για να υπολογίσουμε τη συνολική τιμή p-value

του στατιστικού T , χρειάζεται μόνο να εκτιμήσουμε τις παραμέτρους ν και γ . Ας υποθέσουμε ότι η μέση τιμή της T είναι μ και η διακύμανση της T είναι σ^2 . Χρησιμοποιώντας την πρώτη και τη δεύτερη ροπή της T , οι τιμές των ν και γ μπορούν να υπολογιστούν ως $\nu = 2(\mu^2/\sigma^2)$ και $\gamma = \sigma^2/\mu$. Ακολουθούν τεχνικές λεπτομέρειες για την εξαγωγή της μέσης τιμής και της διακύμανσης της στατιστικής T όταν οι οριακές p-values βασίζονται σε αμφίπλευρους ελέγχους (two-sided tests): Υποθέτουμε ότι η στατιστική δοκιμής συσχέτισης για το jοστό φαινοτυπικό χαρακτηριστικό είναι z_j , όπου $j = 1, 2, \dots, j_m$. Το αντίστοιχο two-sided p-value ορίζεται ως $p_j = 2\Phi(-|z_j|)$, όπου Φ είναι η τυπική συνάρτηση κατανομής Gauss. Υπό τη μηδενική υπόθεση της μη ύπαρξης συσχέτισης μεταξύ ενός SNP και πολυπαραγοντικών φαινοτυπικών χαρακτηριστικών, η κατανομή του στατιστικού T προσεγγίζεται από μια κατανομή Gauss με τη μέση τιμή του T ως εξής

$$\mu = E[T] = 2m$$

και τη διακύμανση του T ως

$$\begin{aligned}\sigma^2 = Var[T] &= Var\left\{\sum_{i=1}^m -2\log(p_j)\right\} \\ &= \sum_{i=1}^m Var\{-2\log(p_j)\} \\ &+ \sum_{j \neq k} cov\{-2\log(p_j), -2\log(p_k)\} \\ &= 4m + \sum_{j \neq k} cov\{-2\log(p_j), -2\log(p_k)\}.\end{aligned}$$

Έστω

$$\delta_{jk} = cov\{-2\log(p_j), -2\log(p_k)\}$$

Μπορούμε να προσεγγίσουμε τη σχέση μεταξύ δ_{jk} και ρ_{jk} , όπου ρ_{jk} ο συντελεστής συσχέτισης μεταξύ των p_k και p_k , χρησιμοποιώντας ένα πολώνυμο δέκατης τάξης: $\delta_{jk} = 3.9081\rho_{j,k}^2 + 0.0313\rho_{j,k}^4 + 0.1022\rho_{j,k}^6 + (-0.1378)\rho_{j,k}^8 + 0.0941\rho_{j,k}^{10}$ (2)

Για τον υπολογισμό του δ_{jk} με ακρίβεια, ακολουθούνται δύο βήματα για την αφαίρεση των μεροληψιών. Πρώτον, εφόσον η δειγματική συσχέτιση $\hat{\rho}_{jk}$ δεν είναι μη-μεροληπτικός εκτιμητής του ρ_{jk} , το ρ_{jk} μπορεί να εκτιμηθεί από την διορθωμένη ως προς την μεροληψία δειγματική συσχέτιση $\hat{r}_{jk}$:

$$r_{j,k} = \rho_{j,k} \left(1 + \frac{1-\rho_{j,k}^2}{2(n-3)}\right) \quad (3)$$

όπου n το μέγεθος του δείγματος που χρησιμοποιήθηκε για τον υπολογισμό του $\hat{\rho}_{jk}$.

Τώρα, ας ορίσουμε τη δεξιά πλευρά της εξίσωσης (2) ως εξής

$$f(r) = c_1 r^2 + c_2 r^4 + c_3 r^6 + c_4 r^8 + c_5 r^{10}.$$

Η εκτίμηση $f(\hat{r}_{jk})$ εξακολουθεί να είναι ένας μεροληπτικός εκτιμητής του δ_{jk} . Έτσι, προτείνουμε ένα δεύτερο βήμα για την αφαίρεση της μεροληψίας. Χρησιμοποιώντας το ανάπτυγμα σειράς Taylor, μπορούμε να εκτιμήσουμε τη μεροληψία ως εξής

$$\frac{c_1}{n} (1 - r_{j,k}^2)^2 \quad (4)$$

Επομένως, ο προτεινόμενος μη μεροληπτικός εκτιμητής του δ_{jk} είναι

$$f(r_{j,k}) - \frac{c_1}{n} (1 - r_{j,k}^2)^2$$

Συνεπώς, με βάση τις εξισώσεις (3) και (4), η διακύμανση του T μπορεί να εκτιμηθεί ως

$$\sigma^2 = var[T] = 4m + \sum_{i \neq j} \left(f(r_{j,k}) - \frac{c_1}{n} (1 - r_{j,k}^2)^2\right)$$

Δεδομένων των προτεινόμενων εκτιμητών των μ και σ^2 , το συνολικό p-value του στατιστικού T μπορεί να υπολογιστεί αποτελεσματικά χρησιμοποιώντας τη συνάρτηση κατανομής gamma ως εξής:

$$globalp - value = 1 - \Gamma(T, v/2, 2\gamma)$$

, όπου $\Gamma(v/2, 2\gamma)$ η αθροιστική συνάρτηση κατανομής της κατανομής gamma με παράμετρο shape $v/2$ και παράμετρο scale 2γ .

3. Επεξήγηση Κώδικα στην PYTHON

3.1 Logitp()

Η συνάρτηση `logit`, αρχικά μετατρέπει το διάνυσμα `p-values` που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζει ότι όλα τα `p-values` βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `logit()` και ότι $0 \leq p\text{-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται ο συντελεστής $C = \frac{\sqrt{k\pi^2(5k+2)/3(5k+4)}}{C}$ όπου

$$C = \sqrt{k\pi^2(5k+2)/3(5k+4)}$$

, και τέλος υπολογίζεται το `t_value = -np.sum(np.log((p_values) / (1 - p_values))) / C` όπου,

$$t = - \frac{\sum \log\left(\frac{p}{1-p}\right)}{C}$$

το οποίο εφαρμόζεται στη συνάρτηση `logit` για τον υπολογισμό του combined p-value `combined_p = 2 * (1 - t.cdf(np.abs(t_value), df))`, όπου

$$pc = 1 - T(|t|, df)$$

`t.cdf` είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution function(cdf)) της κατανομής T με `df = 2*k` βαθμούς ελευθερίας.

.

3.2 Meanp()

Η συνάρτηση `meanp`, αρχικά μετατρέπει το διάνυσμα `p-values` που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-09, 1 - 1e-09)` διασφαλίζει ότι όλα τα `p-values` βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `meanp()` και ότι $0 \leq p\text{-values} \leq 1$, αφού πρόκειται για πιθανότητες. τέλος υπολογίζεται το `z_value = (0.5 - np.mean(p_values)) * math.sqrt(12 * k)` όπου,

$$z = (0.5 - \bar{p})\sqrt{12k}$$

το οποίο εφαρμόζεται στη συνάρτηση `meanp` για τα combined p-values `combined_p = 1-norm.cdf(z_value)`.

3.3 Fisher()

Η συνάρτηση `fisher`, αρχικά μετατρέπει το διάνυσμα `p-values` που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζει ότι όλα τα `p-values` βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `fisher()` και ότι $0 \leq p\text{-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται το άθροισμα `chi_squared = -2 * np.sum(np.log(p_values))` όπου

$$X^2 = -2 \sum_{i=1}^k \log(p_i)$$

όπου k το πλήθος των p-values στο διάνυσμα. Τέλος, υπολογίζεται η συνάρτηση `fisher_p` για τα combined p-values `fisher_p = 1 - chi2.cdf(chi_squared, df)` όπου `chi2.cdf` είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution function (cdf)) της κατανομής X^2 με $2k$ βαθμούς ελευθερίας, που βρίσκονται υλοποιημένες στο πακέτο `scipy` και

$$p_c = 1 - F(X^2, 2k).$$

3.4 Stouffer()

Η συνάρτηση `stouffer`, αρχικά μετατρέπει το διάνυσμα p-values που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζει ότι όλα τα p-values βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `stouffer()` και ότι $0 \leq p\text{-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζονται τα `z_scores = norm.ppf(1 - p_values)`, όπου `norm.ppf` η αντίστροφη αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (inverse cumulative distribution function of normal distribution). Τα `combined_z =`

`np.sum(z_scores/math.sqrt(len(p_values)))` όπου

$$z = \sum_{i=1}^k z_i / \sqrt{k}$$

, χρησιμοποιούνται κατόπιν για τον τελικό υπολογισμό του combined p-value, όπου `combined_p = 1 - norm.cdf(combined_z)` όπου `norm.cdf` η αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (cumulative distribution function of normal distribution) και

$$p_c = 1 - \Phi(z).$$

3.5 Inverse_chi2()

Η συνάρτηση `inverse_chi2`, αρχικά μετατρέπει το διάνυσμα p-values που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζεται ότι όλα τα p-values βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `inverse_chi2()` και ότι $0 \leq p\text{-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται το `chi_squared = np.sum(chi2.ppf((1 - p_values), 1))` όπου `chi2.ppf` είναι η αντίστροφη της αθροιστικής συνάρτησης κατανομής (inverse cumulative distribution function) της κατανομής X^2 που βρίσκονται υλοποιημένες στο πακέτο `scipy` με έναν βαθμό ελευθερίας (`df = 1`) και

$$X^2 = \sum_{i=1}^k F^{-1}(1 - p_i, 1).$$

Τέλος, για τον υπολογισμό του combined p-value χρησιμοποιείται η συνάρτηση `inv_chi2_combined_p = 1 - chi2.cdf(chi_squared, df)` όπου

`chi2.cdf` είναι η αθροιστική συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής X^2 με $df = k$ βαθμούς ελευθερίας και $p_c = 1 - F(X^2, k)$.

3.5 Binomial_test()

Η συνάρτηση `binomial_test`, αρχικά μετατρέπει το διάνυσμα `p-values` που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Επίσης, υπολογίζεται ο αριθμός των στατιστικά σημαντικών `p-values` στο διάνυσμα των `p-values`(number of successes)ή με άλλα λόγια ο αριθμός των test που οδηγούν σε απόρριψη $r = \text{sum}(p < \alpha \text{ for } p \text{ in } p_values)$, όπου $\alpha = 0.05$ (για $p_c < \alpha$ απορρίπτουμε την κοινή μηδενική υπόθεση H_0). Για τον υπολογισμό του combined p-value χρησιμοποιείται η συνάρτηση `combined_p` += $\frac{\text{math.factorial}(k)}{\text{math.factorial}(x) * \text{math.factorial}(k - x)} * (\alpha ** x) * ((1 - \alpha) ** (k - x))$, όπου k το πλήθος των μηδενικών υποθέσεων H_0 , $a = 0.5$ η πιθανότητα απόρριψης της μηδενικής υπόθεσης H_0 και $x=r$ και

$$p_c = \sum_{x=r}^k \binom{k}{x} a^x (1-a)^{k-x}.$$

Το combined p-value αποτελεί την διωνυμική πιθανότητα να παρατηρηθούν τουλάχιστον r successes.

3.6 Cauchy()

Η συνάρτηση `cauchy`, αρχικά μετατρέπει το διάνυσμα `p-values` που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζεται ότι όλα τα `p-values` βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `cauchy ( )` και ότι $0 \leq p_values \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται και το $T = \text{np.tan}((0.5 - p_values) * \text{np.pi})$, όπου T το στατιστικό τον συνδυασμό τιμών της κατανομής Cauchy και $t = \text{np.sum}(T) / k$ μια μεταβλητή που εφαρμόζει τα βάρη $w_i = 1/k$, όπου $k = \text{len}(p_values)$, στο άθροισμα των T στατιστικών για τα p_i pvalues. Έτσι ικανοποιείται η συνάρτηση

$$T = w_i \sum_{i=1}^k \tan \{(0.5 - p_i)\pi\}$$

Τέλος , για τον υπολογισμό του combined p-value χρησιμοποιείται η συνάρτηση `combined_p = 1 - cauchy_cdf(t)`, όπου `cauchy_cdf ( )` η αθροιστική συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής Cauchy .

3.7 MinP()

Η συνάρτηση `minp`, η οποία αποτελεί την υλοποίηση της μεθόδου του Tippett για τον υπολογισμό του combined p-value, αρχικά μετατρέπει το διάνυσμα p-values που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζεται ότι όλα τα p-values βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `minp()` και ότι $0 \leq \text{p-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται το combined p-value ως `combined_p = 1 - (1 - np.min(p_values)) ** k` όπου η συνάρτηση `np.min(p_values)` επιστρέφει το μικρότερο p-value από το διάνυσμα των p-values και `k` το πλήθος των p-values στο διάνυσμα και

$$p_c = 1 - (1 - \min(p_1, p_2, \dots, p_k))^k.$$

3.8 HMP()

Η συνάρτηση `HMP`, αρχικά μετατρέπει το διάνυσμα p-values που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης `numpy` για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζεται ότι όλα τα p-values βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `HMP()` και ότι $0 \leq \text{p-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται ο αρμονικός μέσος όρος των p-values `harmonic_mean = L / np.sum(1 / p_values)` και το combined p-value ως `combined_p = moyal.cdf(-2 * np.log(harmonic_mean), 1, L)` όπου και

$$p_c = \int_{1/p_c}^{\infty} f_{\text{Landau}}(x | \log L + 0.874, \frac{\pi}{2}) dx.$$

Effective number of tests

Η διόρθωση Bonferroni αλλά το ποσοστό σφάλματος τύπου I (Type I error rate) μπορεί να είναι αρκετά συντηρητικό όταν τα test είναι εξαρτημένα. Επομένως, μπορούμε να εφαρμόσουμε ορισμένες προσαρμογές που μπορούν να βοηθήσουν να φέρουν τα ποσοστά σφάλματος τύπου I πιο κοντά σε μη σημαντικό επίπεδο ακόμη και όταν τα test είναι εξαρτημένα.

3.9 effective_number_of_tests_cheverud_nyholt()

Η συνάρτηση `effective_number_of_tests_cheverud_nyholt()` υπολογίζει το πλήθος των ιδιοτιμών στο διάνυσμα `eigenvalues` ως `k = len(eigenvalues)` και τη δειγματική διακύμανση `Var(λ)` των ιδιοτιμών `sample_variance = np.var(eigenvalues)`. Τέλος, υπολογίζει το `effective_number_of_tests = 1 + (k - 1) * (1 - sample_variance / k)` χρησιμοποιώντας τον τύπο που προτάθηκε από τους Cheverud και Nyholt όπου

$$m_{CN} = 1 + (k - 1) \left(1 - \frac{\text{Var}(\lambda)}{k}\right).$$

3.10 h_function()

Η συνάρτηση *h_function()*, που χρησιμοποιείται για την υλοποίηση του τύπου που προτάθηκε από τους Li και Ji για τον υπολογισμό των effective number of tests, ισούται με $h = 1 + (x - \text{math.floor}(x))$ για $x \geq 1$ και με $h = x - \text{math.floor}(x)$ για $x < 1$, και

$$h(x) = I(x \geq 1) + (x - [x]).$$

3.11 effective_number_of_tests_li_ji()

Η συνάρτηση *effective_number_of_tests_li_ji()* υπολογίζει τις απόλυτες τιμές των ιδιοτιμών στο διάνυσμα *eigenvalues* ως *abs_eigenvalues = np.abs(eigenvalues)* και το *effective_number_of_tests = np.sum([h_function(abs_lambda) for abs_lambda in abs_eigenvalues])* ως το άθροισμα των αποτελεσμάτων που προκύπτουν από την εφαρμογή των ιδιοτιμών στη συνάρτηση *h_function()* η οποία αναλύεται παραπάνω, χρησιμοποιώντας τον τύπο που προτάθηκε από τους Li και Ji

$$m_{LJ} = \sum_{i=1}^k h(|\lambda_i|).$$

3.12 effective_number_of_tests_gao()

Η συνάρτηση *effective_number_of_tests_gao()* υπολογίζει το πλήθος των ιδιοτιμών στο διάνυσμα *eigenvalues* ως $k = \text{len}(\text{eigenvalues})$, το άθροισμα των ιδιοτιμών *total_sum = np.sum(eigenvalues)*, το άθροισμα των ιδιοτιμών έως την τιμή *x* ως $x_sum = \text{np.sum}(\text{eigenvalues}[:x+1])$ for x in $\text{range}(k)$. Τέλος, εάν $x_sum / \text{total_sum} > C$ όπου $C = 0.995$ υπολογίζει το *effective_number_of_tests = x + 1* χρησιμοποιώντας τον τύπο που προτάθηκε από τον Gao

$$m_{GAO} = \underset{x \in \{1, \dots, k\}}{\text{argmin}} \left(\frac{\sum_{i=1}^x \lambda_i}{\sum_{i=1}^k \lambda_i} > C \right).$$

3.13 effective_number_of_tests_galway()

Η συνάρτηση `effective_number_of_tests_galway()` υπολογίζει το πλήθος των ιδιοτιμών στο διάνυσμα `eigenvalues` ως `k = len(eigenvalues)`, τις προσαρμοσμένες ιδιοτιμές λ_i' ως `lambda_prime = np.maximum(0, eigenvalues)` ώστε οι αρνητικές ιδιοτιμές να αντικατασταθούν με μηδενικά, καθώς και το τετραγωνικό άθροισμα των $\sqrt{\lambda_i'}$ `sum_squared_lambda_prime = (np.sum(np.sqrt(lambda_prime))) ** 2`. Τέλος, υπολογίζει το `effective_number_of_tests = sum_squared_lambda_prime / np.sum(lambda_prime)` χρησιμοποιώντας τον τύπο που προτάθηκε από τον Galway

$$m_{GAL} = \frac{(\sum_{i=1}^k \sqrt{\lambda_i'})^2}{\sum_{i=1}^k \lambda_i}.$$

3.14 bonferroni_method_with_effective_tests()

Η συνάρτηση `bonferroni_method_with_effective_tests()`, αρχικά μετατρέπει το διάνυσμα `p-values` που έχει ως όρισμα σε έναν πίνακα της βιβλιοθήκης *numpy* για την υλοποίηση αριθμητικών πράξεων. Στη συνέχεια, με την συνάρτηση `np.clip(p_values, 1e-16, 1 - 1e-16)` διασφαλίζεται ότι όλα τα `p-values` βρίσκονται εντός των αριθμητικών ορίων που μπορεί να υποστηρίξει η συνάρτηση `bonferroni_method_with_effective_tests()` και ότι $0 \leq p\text{-values} \leq 1$, αφού πρόκειται για πιθανότητες. Έπειτα, υπολογίζεται το combined p-value ως `combined_p = min(1, min_p_value * effective_number_of_tests)` όπου η συνάρτηση `min_p_value = min(p_values)` επιστρέφει το μικρότερο p-value από το διάνυσμα των `p-values` και `k` το πλήθος των `p-values` στο διάνυσμα και

$$pc = \min(1, \min(p_1, p_2, \dots, p_k) \times \text{effective_number_of_tests}).$$

3.15 Bonferronis_correction_g

Η συνάρτηση `Bonferronis_correction` αρχικά υπολογίζει τον μέγιστο συντελεστή συσχέτισης με τη συνάρτηση `_`, `ICC = CalculateKostCovarianceEBM(data_matrix)` και στη συνέχεια το προσαρμοσμένο πλήθος `p-values` `g_adjusted = (g + 1) - (1 + (g - 1) * ICC)`,

$$g^* = (g + 1) - (1 + (g - 1) * \widehat{ICC})$$

όπου `g = len(data_matrix)` το πλήθος των `p-values` του διανύσματος.

3.16 EmpiricalBrownsMethod

```
def EmpiricalBrownsMethod(data_matrix, p_values,
                           extra_info=False):
```

Η συνάρτηση `EmpiricalBrownsMethod()` με ορίσματα έναν $m \times n$ πίνακα(`data_matrix`) με m γραμμές που απεικονίζουν τις μεταβλητές(SNPs) και n στήλες που απεικονίζουν τα δείγματα, ένα διάνυσμα που αποτελείται από m p-values και τη boolean μεταβλητή `extra_info` η οποία όταν είναι **True** επιστρέφει, εκτός από το combined p-value που προκύπτει από το Empirical Brown's Method (EBM), και το p-value από τη μέθοδο του Fisher, τον scale factor c , και τους νέους βαθμούς ελευθερίας(degrees of freedom) από τη μέθοδο του Brown. Υπολογίζεται ο πίνακας συνδιακύμανσης από τη συνάρτηση `CalculateCovariances()`
`covar_matrix = CalculateCovariances(data_matrix)` και επιστρέφεται το combined p-value που προκύπτει από το Empirical Brown's Method (EBM) `return CombinePValuesEBM(covar_matrix, p_values, extra_info)`.

```
def TransformData(data_vector):
```

Η συνάρτηση `TransformData()` με όρισμα ένα raw διάνυσμα(`data_vector`) μίας μεταβλητής(SNP) (μπορεί να είναι λίστα ή πίνακας), υπολογίζει τη μέση τιμή m ($m = np.mean(data_vector)$) και την τυπική απόκλιση $sd = np.std(data_vector)$ του διανύσματος των p-values, και την κανονικοποιημένη τιμή κάθε p-value του διανύσματος $s = [(d - m) / sd \text{ for } d \text{ in } data_vector]$. Η συνάρτηση $W(x)$ χρησιμοποιεί μια συνάρτηση `lambda`, η οποία υπολογίζει τον μετασχηματισμό του Fisher που περιλαμβάνει την εμπειρική αθροιστική συνάρτηση κατανομής (ECDF) του τυποποιημένου διανύσματος δεδομένων $W = \lambda x: -2 * np.log(ECDF(s)(x))$. Εφαρμόζεται ο μετασχηματισμός $W(x)$ σε κάθε στοιχείο x του διανύσματος κανονικοποιημένων τιμών(s) του διανύσματος. Επιστρέφει το μετασχηματισμένο διάνυσμα `return np.array([W(x) for x in s])`.

```
def CalculateCovariances(data_matrix):
```

Η συνάρτηση `CalculateCovariances()` με όρισμα έναν πίνακα δεδομένων $m \times n$ (`data_matrix`), όπου κάθε m γραμμή αντιπροσωπεύει μια μεταβλητή και κάθε n στήλη αντιπροσωπεύει ένα δείγμα. Υπολογίζει τον κανονικοποιημένο πίνακα κάθε διανύσματος `transformed_data_matrix = np.array([TransformData(f) for f in data_matrix])` του `data_matrix` και επιστρέφει έναν πίνακα $m \times m$ με τις συνδιακυμάνσεις ανά ζεύγη των κανονικοποιημένων διανυσμάτων των raw δεδομένων `covar_matrix = np.cov(transformed_data_matrix)`.

```
def CombinePValuesEBM(covar_matrix, p_values,
                       extra_info=False):
```

Η συνάρτηση `CombinePValuesEBM()` με ορίσματα έναν πίνακα $m \times m$ με συνδιακυμάνσεις μεταξύ κανονικοποιημένων διανυσμάτων δεδομένων

`covar_matrix` και ένα διάνυσμα `m` p-values προς συνδυασμό. Στη συνέχεια υπολογίζονται το πλήθος των γραμμών `m` του πίνακα συνδιακύμανσης `m = int(covar_matrix.shape[0])`, οι βαθμοί ελευθερίας για τη μέθοδο του Fisher `df_fisher = 2.0 * m` και την αναμενόμενη μέση τιμή `Expected = 2.0 * m` και το άθροισμα των συνδιακυμάνσεων `cov_sum += covar_matrix[i, j]` βάση της εξίσωσης: $\sum_{i < j} cov(-2\log P_i, -2\log P_j)$ και τη διακύμανση `Var = 4.0 * m + 2 * cov_sum`

$$var[\Psi] = 4k + 2 \sum_{i < j} cov(-2\log P_i, -2\log P_j)$$

καθώς και τον scale factor `c = Var / (2.0 * Expected)`. Επιπλέον υπολογίζονται οι βαθμοί ελευθερίας για τη μέθοδο του Brown `df_brown = 2.0 * Expected ** 2 / Var`. Εάν οι `df_brown > df_fisher` τότε οι βαθμοί ελευθερίας για τη μέθοδο του Brown ορίζονται ίσοι με τους βαθμούς ελευθερίας για τη μέθοδο του Fisher `df_brown = df_fisher` και ο scale factor ορίζεται `c = 1.0`.

υπολογίζεται το άθροισμα `x = 2.0 * sum([-np.log(np.clip(p, 1e-15, 1 - 1e-15)) for p in p_values])`

$$X^2 = -2 \sum_{i=1}^k \log(p_i)$$

, το οποίο ακολουθεί κατανομή X^2 με $2k$ βαθμούς ελευθερίας, όπου k το πλήθος των p-values στο διάνυσμα

Τέλος, υπολογίζεται τα combined p-values όπου `p_brown = chi2_cdf(df_brown, 1.0 * x / c)` και `p_fisher = chi2_cdf(df_fisher, 1.0 * x)` με βάση την εξίσωση του Fisher :

$$p_c = 1 - F(X^2, 2k)$$

, όπου `chi2_cdf` είναι η αθροιστική συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής X^2 .

`def KostPolyFitEBM(cor):`

Η συνάρτηση `KostPolyFitEBM()` με όρισμα τον συντελεστή συσχέτισης `cor` μεταξύ δύο $n \times n$ διανυσμάτων p-values επιστρέφει το πολώνυμο `a1 * cor + a2 * cor ** 2 + a3 * cor ** 3` με συντελεστές `a1, a2, a3 = 3.263, 0.710, 0.027`

$$cov(-2\log P_i, -2\log P_j) \approx 3.263\rho_{ij} + 0.710\rho_{ij}^2 + 0.027\rho_{ij}^3$$

, το οποίο αποτελεί την προσέγγιση του Kost για τη συνδιακύμανση μεταξύ των αθροιστικών κατανομών $-\log$.

```
def CalculateKostCovarianceEBM(data_matrix):
```

Η συνάρτηση `CalculateKostCovarianceEBM()` με όρισμα ένα πίνακα $m \times n$ `data_matrix` με κάθε m γραμμή να αντιπροσωπεύει μια μεταβλητή(SNP) και κάθε n στήλη να αντιπροσωπεύει ένα δείγμα, αρχικά ορίζει δύο μηδενικοί πίνακες διαστάσεων $n \times n$ `covar_matrix = np.zeros((n, n))` `corr_matrix = np.zeros((n, n))`, όπου $n = \text{data_matrix.shape}[1]$ το πλήθος των στηλών του πίνακα `data_matrix`. Στη συνέχεια, υπολογίζεται ο πίνακας συσχέτισης με τη χρήση της συνάρτησης συσχέτισης του Pearson για τη συσχέτιση μεταξύ των στηλών του πίνακα, δηλαδή μεταξύ των μετρήσεων για κάθε δείγμα(π.χ. οι μετρήσεις για το p_1 για καθεμία από τις μεταβλητές(SNPs) συσχετίζονται με τις μετρήσεις για το p_2 κλπ). `cor, p_val = pearsonr(data_matrix[i, :], data_matrix[j, :])` `corr_matrix[i, j] = cor`. Επίσης, υπολογίζεται ο πίνακας συνδιακύμανσης `covar_matrix[i, j] = covar`; `covar_matrix[j, i] = covar` με τη χρήση της συνάρτησης `KostPolyFitEBM` (πολυωνυμικής προσέγγιση του Kost για την συνδιακύμανση) `covar = KostPolyFitEBM(corr)`. Και ορίζεται ως `max_cor = max(max_cor, cor)` ο μέγιστος όρος(συντελεστής συσχέτισης) του πίνακα συσχέτισης που υπολογίστηκε παραπάνω. Επιστρέφεται ο $n \times n$ πίνακας συνδιακύμανσης `covar_matrix` και ο μέγιστος συντελεστής συσχέτισης `max_cor`.

3.16 Brown's Method by Yang

```
def BrownsMethodByYang(delta_matrix, p_values, extra_info=False):
```

Η συνάρτηση `EmpiricalBrownsMethod()` με ορίσματα έναν $m \times n$ πίνακα(`data_matrix`), με m γραμμές που απεικονίζουν τις μεταβλητές(SNPs) και n στήλες που απεικονίζουν τα δείγματα, ένα διάνυσμα που αποτελείται από m p-values και τη boolean μεταβλητή `extra_info` η οποία όταν είναι `True` επιστρέφει, εκτός από το combined p-value που προκύπτει από το Brown's Method By Yang(EBM), και το p-value από τη μέθοδο του Fisher, τον *scale factor* C , και τους νέους βαθμούς ελευθερίας(*degrees of freedom*) από τη μέθοδο του Brown. Υπολογίζεται ο πίνακας συνδιακύμανσης `delta_matrix = CalculateCovariances(data_matrix)` και επιστρέφεται το combined p-value που προκύπτει από το Brown's Method by Yang `return CombinePValuesYang(delta_matrix, p_values, extra_info)`.

```
def CombinePValuesYang(delta_matrix, p_values, extra_info=False):
```

Η συνάρτηση `CombinePValuesEBM()` με ορίσματα έναν πίνακα $m \times m$ με συνδιακυμάνσεις μεταξύ κανονικοποιημένων διανυσμάτων δεδομένων `delta_matrix` και ένα διάνυσμα m p-values προς συνδυασμό. Στη συνέχεια υπολογίζονται το πλήθος των γραμμών m του πίνακα συνδιακύμανσης $m =$

`int(delta_matrix.shape[0])`, οι βαθμοί ελευθερίας για τη μέθοδο του Fisher
`df_fisher = 2.0 * m` και την αναμενόμενη μέση τιμή `mean = 2.0 * m` και
το άθροισμα των `delta_sum += delta_matrix[i, j]` όπου
`delta_matrix[i, j]`:

$$\delta_{jk} = cov\{-2\log(p_j), -2\log(p_k)\}$$

όπου δ_{jk} είναι η συνάρτηση συσχέτισης μεταξύ του p_j και του p_k ,
και τη διακύμανση `Var = 4.0 * m + delta_sum`,

$$\sigma^2 = var[T] = 4m + \sum_{i \neq j} (f(r_{j,k}) - \frac{c_1}{n} (1 - r_{j,k}^2)^2).$$

Επιπλέον υπολογίζονται οι βαθμοί ελευθερίας για τη μέθοδο του Fisher
`df_fisher = 2.0 * Expected ** 2 / Var` και υπολογίζεται το στατιστικό
`T = 2.0 * sum([-np.log(np.clip(p, 1e-15, 1 - 1e-15)) for p
in p_values])` για τη μέθοδο του Fisher

$$X^2 = -2 \sum_{i=1}^k \log(pi)$$

, το οποίο ακολουθεί κατανομή X^2 με $2k$ βαθμούς ελευθερίας, όπου k το πλήθος των
p-values στο διάστημα

Τέλος, υπολογίζεται τα combined p-values όπου `p_yang = 1 - gamma.cdf(T,
v/2, scale=2 * gamma_value)`, όπου

$$globalp = 1 - \Gamma(T, v/2, 2\gamma)$$

, όπου `gamma.cdf` η αθροιστική συνάρτηση κατανομής της κατανομής Γ με
παράμετρο *shape* $v/2$, όπου $v = mean ** 2 / Var$,

$$v = 2(\mu^2 / \sigma^2)$$

και παράμετρο *scale* 2γ , όπου `gamma_value = Var / mean`,

$$\gamma = \sigma^2 / \mu$$

και `p_fisher = chi2_cdf(df_fisher, 1.0 * x)` με βάση την εξίσωση
του Fisher :

$$p_c = 1 - F(X^2, 2k)$$

, όπου `chi2_cdf` είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution
function(cdf)) της κατανομής X^2 και $2k$ οι βαθμοί ελευθερίας .

```
def PolyFitYang(cor):
```

Η συνάρτηση `KostPolyFitYang()` με όρισμα τον συντελεστή συσχέτισης `cor` μεταξύ δύο $n \times n$ διανυσμάτων p -values επιστρέφει το πολυώνυμο δέκατης τάξης $c1 * cor^{**2} + c2 * cor^{**4} + c3 * cor^{**6} + c4 * cor^{**8} + c5 * cor^{**10}$ με συντελεστές $c1, c2, c3, c4, c5 = 3.9081, 0.0313, 0.1022, -0.1378, 0.0941$ όπου,

$$\delta_{jk} = 3.9081\rho_{j,k}^2 + 0.0313\rho_{j,k}^4 + 0.1022\rho_{j,k}^6 + (-0.1378)\rho_{j,k}^8 + 0.0941\rho_{j,k}^{10}$$

, το οποίο προσεγγίζει τη σχέση μεταξύ της συνδιακύμανσης δ_{jk} και του συντελεστή συσχέτισης $\rho_{j,k}$.

```
def CalculateKostCovarianceYang(data_matrix):
```

Η συνάρτηση `CalculateKostCovarianceYang()` με όρισμα ένα πίνακα $m \times n$ `data_matrix` με κάθε m γραμμή να αντιπροσωπεύει μια μεταβλητή (SNP) και κάθε n στήλη να αντιπροσωπεύει ένα δείγμα, αρχικά ορίζει δύο μηδενικοί πίνακες διαστάσεων $n \times n$ `delta_matrix = np.zeros((n, n))` `corr_matrix = np.zeros((n, n))`, όπου $n = data_matrix.shape[1]$ το πλήθος των στηλών του πίνακα `data_matrix`. Στη συνέχεια, υπολογίζεται ο πίνακας συσχέτισης με τη χρήση της συνάρτησης συσχέτισης του Pearson για τη συσχέτιση μεταξύ των στηλών του πίνακα, δηλαδή μεταξύ των μετρήσεων για κάθε δείγμα (π.χ. οι μετρήσεις για το $p1$ για καθεμία από τις μεταβλητές (SNPs) συσχετίζονται με τις μετρήσεις για το $p2$ κλπ). `cor, p_val = pearsonr(data_matrix[:, i], data_matrix[:, j])` και υπολογίζεται ο συντελεστής συνδιακύμανσης δ_{jk} , `covar = KostPolyFitEBM(cor)` (πολυωνυμική προσέγγιση για την συνδιακύμανση). Επίσης, ορίζεται ο πίνακας συνδιακύμανσης `covar_matrix[i, j] = covar`, ο οποίος γεμίζει με τις τιμές του συντελεστή `covar` σε κάθε θέση `[i, j]` του πίνακα.

Στη συνέχεια, υπολογίζονται οι εξισώσεις για τον διορθωμένο ως προς τη μεροληψία συντελεστή συσχέτισης `biased_corrected_cor = cor * (1 + (1 - cor**2 / 2 * (n - 3)))`,

$$r_{j,k} = \rho_{j,k} \left(1 + \frac{1 - \rho_{j,k}^2}{2(n-3)}\right)$$

τον μεροληπτικό εκτιμητή του δ_{jk} , `f_r = PolyFitYang(biased_corrected_cor)`,

$$f(r) = c1r^2 + c2r^4 + c3r^6 + c4r^8 + c5r^{10}$$

τη μεροληψία `bias = (c1 / n) * (1 - biased_corrected_cor**2) **2`,

$$\frac{c1}{n} (1 - r_{j,k}^2)^2$$

και τελικά το μη μεροληπτικό συντελεστή δ_{jk} , `unbiased_delta = f_r - bias`,

$$f(r_{j,k}) - \frac{c_1}{n}(1 - r_{j,k}^2)^2.$$

Τέλος, ορίζεται ο πίνακας συνδιακύμανσης `delta_matrix[i, j] = unbiased_delta`, ο οποίος γεμίζει με τις τιμές του συντελεστή `unbiased_delta` σε κάθε θέση `[i, j]` του πίνακα.

4. Διαχείριση αρχείου

`transform_to_pvalues()`

Η συνάρτηση `transform_to_pvalues` μετατρέπει τα z-scores που λαμβάνει ως όρισμα σε two-sided p-values, σε περίπτωση που το αρχείο που δίνει ο χρήστης περιέχει z-scores αντί για p-values, με τη συνάρτηση `p_values = 2 * norm.cdf(-np.abs(scores))`, όπου `norm.cdf` είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution function(cdf)) της κανονικής κατανομής που βρίσκεται υλοποιημένη στο πακέτο `scipy`.

`read_from_file_tranform(file_path, data_type)`

Με τη συνάρτηση `open(file_path, 'r')` as file: ανοίγει το αρχείο για διάβασμα και η μέθοδος `file.readlines()` επιστρέφει μια λίστα που περιέχει κάθε γραμμή του αρχείου ως στοιχείο λίστας.

Στην εντολή `elements = line.strip().split()`, η `line.strip()` αφαιρεί τα κενά από την αρχή και το τέλος κάθε γραμμής του αρχείου καθώς και τους χαρακτήρες αλλαγής γραμμής \n και η `split()` διαχωρίζει κάθε γραμμή του αρχείου σε μια λίστα από τιμές p-values.

`SNP, values = elements[0], list(map(float, elements[1:]))`: Ο SNP αποτελεί το πρώτο στοιχείο `elements[0]` κάθε λίστας, και `values` είναι η υπόλοιπη λίστα με τις τιμές των p-values ανά γραμμή.

Εάν ο χρήστης πληκτρολογήσει 'z-scores' το οποίο καταχωρείται στη μεταβλητή `data_type` και `data_type.lower() == 'z-scores'`, τότε τα z-scores που βρίσκονται στη λίστα `values` μετατρέπονται σε p-values με τη συνάρτηση `values = transform_to_pvalues(values)`. Αλλιώς, εάν ο χρήστης πληκτρολογήσει 'p-values', τότε δε γίνεται κάποια αλλαγή στη λίστα με τα `values`.

`SNP_values.append((SNP, values))`: Οι SNPs και η λίστα `values` κάθε γραμμής του αρχείου προστίθενται στη λίστα `SNP_values = []`.

4.1 Διάβασμα αρχείου για τις μεθόδους 1-11

Για τις μεθόδους 1 έως 11, οι οποίες δεν απαιτούν τον υπολογισμό πίνακα συσχέτισης, τα περιεχόμενα της λίστας `SNP_values = []` χρησιμοποιούνται ως εξής:

Αν ο χρήστης έχει δώσει ως όρισμα ότι επιθυμεί να γίνει μετα-ανάλυση (`metanalysis_needed = args.meta_choice.lower() == 'yes'`) if `metanalysis_needed`:, τότε for `SNP, values in SNP_values`:

`SNP_dict.setdefault(SNP, []).append(values)` τα SNPs με τα αντίστοιχα `values` καταχωρούνται στο dictionary `SNP_dict`.

Για τα `values` κάθε SNP που έχει καταχωρηθεί παραπάνω από μία φορά (if `len(values) > 1`) στο dictionary `SNP_dict` οι τιμές για κάθε θέση της λίστας `values` ομαδοποιούνται σε `grouped_values = list(zip(*values))` (π.χ. [(p1 p1)], [(p2, p2)] για δύο SNPs με όνομα SNP3). Επιπλέον, δημιουργείται μια λίστα `combined_results = [combine_function(np.array(grouped_value)) for grouped_value in grouped_values]` που περιέχει τις συνδυασμένες τιμές των p-values των ομάδων `grouped_values` για τον αντίστοιχο SNP. Οι λίστες `combined_results` με τις συνδυασμένες τιμές των καταχωρούνται στο `SNP_dict` στον αντίστοιχο SNP και τελικά οι νέες τιμές του `SNP_dict` τοποθετούνται στη λίστα `p_values` με την εντολή `p_values.append({'SNP': SNP, 'values': SNP_dict[SNP]})`. Τέλος, για κάθε λίστα με τιμές p-values `p_value['values']`, συνδυάζονται οι τιμές με την επιλεγμένη από το χρήστη συνάρτηση `combine_function` και οι συνδυασμένες τιμές για κάθε SNP `p_value['SNP']` εκτυπώνεται στο `output.txt`.

Αν ο χρήστης έχει δώσει ως όρισμα ότι δεν επιθυμεί να γίνει μετα-ανάλυση (`metanalysis_needed = args.meta_choice.lower() == 'no'`), τότε τα SNPs με τα αντίστοιχα `values` καταχωρούνται στη λίστα `p_values` και για κάθε λίστα με τιμές p-values `p_value['values']`, συνδυάζονται οι τιμές με την επιλεγμένη από το χρήστη συνάρτηση `combine_function` και οι συνδυασμένες τιμές για κάθε SNP `p_value['SNP']` εκτυπώνεται στο `output.txt`.

4.2 Διάβασμα αρχείου για Empirical Brown's Method (EBM) και Brown's Method by Yang και Kost's Method

Οι μέθοδοι Empirical Brown's Method (EBM), Brown's Method by Yang και Bonferroni with effective tests διαχειρίζονται διαφορετικά το αρχείο με τα p-values (ή τα z-scores) καθώς απαιτούν τον υπολογισμό του πίνακα συσχέτισης μεταξύ των δειγμάτων p-values.

Για τις μεθόδους Empirical Brown's Method (EBM) και Brown's Method by Yang, αρχικά ορίζεται μία λίστα `data = []` στην οποία τοποθετείται το αποτέλεσμα `data = read_from_file_transform(file_path, data_type)`, δηλαδή η λίστα `SNP_values` και ορίζονται οι λίστες `dep_var = np.array([values for dep_var, _ in data])` με τα SNPs και `values = np.array([values for _, values in data])` με τις τιμές των p-values κάθε γραμμής του αρχείου. Τελικά, υπολογίζεται το combined p-value με τη συνάρτηση που έχει επιλέξει ο χρήστης `combine_function(values, pvals, extra_info=True)`.

4.3 Διάβασμα αρχείου για Bonferroni with effective tests

Για τη συνάρτηση Bonferroni with effective tests υπολογίζει τις ιδιοτιμές `eigenvalues, _ = np.linalg.eig(cleaned_corr_matrix)` και τα

ιδιοδιανύσματα (τα οποία αγνοούνται εδώ αναθέτοντάς τα στο `_`) του πίνακα συσχέτισης `cleaned_corr_matrix` χρησιμοποιώντας τη μονάδα γραμμικής άλγεβρας `np.linalg` της NumPy, τις οποίες και ταξινομεί κατά φθίνουσα σειρά με `sorted_eigenvalues = np.sort(eigenvalues)[::-1]`.

Στη συνέχεια, εφαρμόζει τις ταξινομημένες ιδιοτιμές του πίνακα συσχέτισης στις συναρτήσεις που υπολογίζουν τις προσαρμογές των Cheverud-Nyholt, `cn = effective_number_of_tests_cheverud_nyholt(sorted_eigenvalues)`, του Gao, `gao = effective_number_of_tests_gao(sorted_eigenvalues)`, του Galway, `gal = effective_number_of_tests_galway(sorted_eigenvalues)`, των Li-Ji, `li_ji = effective_number_of_tests_li_ji(sorted_eigenvalues)` και το προσαρμοσμένο `g_adjusted = Bonferronis_correction_g(values)`, για τον υπολογισμό των effective number of tests.

Οι προσαρμογές που υπολογίστηκαν εφαρμόζονται στη συνάρτηση επιλογής του χρήστη(στη συγκεκριμένη περίπτωση η `bonferroni_method_with_effective_tests`) για τον υπολογισμό των combined p-values ξεχωριστά για τις τέσσερις προτεινόμενες μεθόδους για τα effective number of tests `combined_p_value_cn = combine_function(values, cn, 0.05)`, `combined_p_value_gao = combine_function(values, gao, 0.05)`, `combined_p_value_gal = combine_function(values, gal, 0.05)`, `combined_p_value_li_ji = combine_function(values, li_ji, 0.05)`, `combined_p_value_bc = combine_function(values, g_adjusted)`, για κάθε SNP στη λίστα `SNP` και των αντίστοιχων p-values στη λίστα `values` (`for SNP, values in SNP_values`).

5. Επεξήγηση Κώδικα στο STATA

Η εντολή `program drop` χρειάζεται σε περίπτωση αλλαγής του προγράμματος και επανάκλησης του `do file`, διότι δεν είναι δυνατό να επαναπροσδιορισθεί ένα ήδη υπάρχον πρόγραμμα. Η εντολή `capture` χρειάζεται μόνο την πρώτη φορά, όταν δεν υπάρχει κάτι για να γίνει `drop`.

5.1 Logitp

Η εντολή `syntax` προσδιορίζει ότι το πρόγραμμα `logit` αναμένει ως όρισμα μια λίστα αριθμητικών μεταβλητών `varlist(numeric)`. Η εντολή `tempvar num_vars` δημιουργεί μια προσωρινή μεταβλητή με το όνομα `num_vars` για να αποθηκεύσει τον αριθμό των μη ελλিপών τιμών για κάθε `observation` της λίστας μεταβλητών `varlist`. Η `egen num_vars = rownonmiss(varlist)` χρησιμοποιεί τη συνάρτηση `egen` για τη δημιουργία της μεταβλητής `num_vars`, η οποία ισούται με τον αριθμό των μεταβλητών σε κάθε παρατήρηση (`observation`). Η μεταβλητή `local C = sqrt(num_vars * pi_value^2 * (5 * num_vars + 2) / (3 * (5 * num_vars + 4)))` υπολογίζει τη σταθερά C του τύπου της `logit`

$$C = \sqrt{k\pi^2(5k+2)/3(5k+4)}$$

Στη συνέχεια, με το βρόχο `foreach var of varlist varlist { gen t_value_`var' = ln(`var' / (1 - `var'))}` δημιουργείται η μεταβλητή `t_value` για κάθε παρατήρηση των μεταβλητών της `varlist` και υπολογίζεται το άθροισμα των `t_values` κάθε παρατήρησης (`observation`) του `dataset`. Τέλος, το combined p-value `gen logit_p = 2 * (1 - t(2 * num_vars, abs(-t_sum / C)))` σύμφωνα με τον μαθηματικό τύπο,

$$p_c = - \frac{\sum \log\left(\frac{p}{1-p}\right)}{C}.$$

5.2 Meanp

Η εντολή `egen z_mean = rowmean(`varlist')` υπολογίζει τη μέση τιμή κάθε παρατήρησης που περιλαμβάνει τις μεταβλητές της `varlist`. Στη συνέχεια, με την `gen z_value = (0.5 - z_mean) * sqrt(12 * `num_vars')` δημιουργείται η μεταβλητή `z_value` σύμφωνα με τον μαθηματικό τύπο,

$$z = (0.5 - \bar{p})\sqrt{12k}$$

το οποίο εφαρμόζεται στον τύπο της συνάρτησης `meanp` για τη δημιουργία των combined p-values `gen mean_p = 1 - normal(z_value)`, όπου `normal` η αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (`cumulative distribution function of normal distribution`).

5.3 Fisherp

Με το βρόχο `foreach var of varlist `varlist' { gen log_`var' = ln(`var') }` δημιουργείται η μεταβλητή `log` η οποία υπολογίζει το λογάριθμο για κάθε μεταβλητή της `varlist` και με την εντολή `egen chi_squared_sum = rowtotal(log_*)` υπολογίζεται το άθροισμα των `log` κάθε παρατήρησης(observation) του dataset, σύμφωνα με τον μαθηματικό τύπο,

$$X^2 = -2 \sum_{i=1}^k \log(p_i).$$

Τέλος, δημιουργούνται τα combined p-values για κάθε παρατήρηση `gen fisher_p = 1 - chi2(2 * `num_vars', -2 * chi_squared_sum)` όπου `chi2` είναι η αθροιστική συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής  $X^2$  με `2 * `num_vars'` βαθμούς ελευθερίας, όπου ``num_vars'` το πλήθος των p-values στο διάνυσμα σύμφωνα με τον μαθηματικό τύπο,

$$p_c = 1 - F(X^2, 2k).$$

5.4 Stoufferp

Με το βρόχο `foreach var of varlist `varlist' { gen z_score_`var' = invnorm(1 - `var') }` δημιουργείται η μεταβλητή `z_score`, όπου `invnorm` η αντίστροφη αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (inverse cumulative distribution function of normal distribution). Έπειτα με την εντολή `egen z_sum = rowtotal(z_score_*)` υπολογίζεται το άθροισμα `z_sum` των `z_scores` κάθε παρατήρησης το οποίο διαιρούμενο από `sqrt(`num_vars')` σύμφωνα με τον μαθηματικό τύπο,

$$z = \sum_{i=1}^k z_i / \sqrt{k}$$

, θέτοντας `k = `num_vars'`, χρησιμοποιείται κατόπιν για τον τελικό υπολογισμό των combined p-values κάθε παρατήρησης, με τη δημιουργία της μεταβλητής `gen stouffer_p = 1 - normal(z_sum / sqrt(`num_vars'))` όπου `normal` η αθροιστική συνάρτηση κατανομής της κανονικής κατανομής (cumulative distribution function of normal distribution) και

$$p_c = 1 - \Phi(z).$$

5.5 Invchi2p

Με το βρόχο `foreach var of varlist `varlist' { gen invchi_squared_`var' = invchi2(1, 1 - `var') }` δημιουργείται η μεταβλητή `invchi_squared` για κάθε μεταβλητή της `varlist` όπου `invchi2` είναι η αντίστροφη της αθροιστικής συνάρτησης κατανομής(inverse cumulative distribution function) της κατανομής  $X^2$  με έναν βαθμό ελευθερίας( $df = 1$ ) και με την εντολή `egen invchi_squared_sum = rowtotal(invchi_squared_*)` υπολογίζεται το άθροισμα των `invchi_squared` κάθε παρατήρησης(observation) του dataset σύμφωνα με το μαθηματικό τύπο,

$$X^2 = \sum_{i=1}^k F^{-1}(1 - p_i, 1).$$

Τέλος, υπολογίζεται το combined p-value κάθε παρατήρησης, με τη δημιουργία της μεταβλητής `gen inv_chi2_p = 1 - chi2(`num_vars', invchi_squared_sum)`, όπου *chi2* είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution function (cdf)) της κατανομής X^2 με $df = k$ βαθμούς ελευθερίας και

$$p_c = 1 - F(X^2, k).$$

5.6 Binomialp

Η τοπική μεταβλητή `local alpha = 0.05` ορίζει το επίπεδο σημαντικότητας ίσο με 0.05. Με το βρόχο `foreach var of varlist `varlist' {if `var' < alpha' {local num_of_successes = `num_of_successes' + 1}}` εάν η τιμή της παρατήρησης της τρέχουσας μεταβλητής της `varlist` είναι μικρότερη του 0.05 τότε αυξάνεται κατά 1 το πλήθος (count) του `num_of_successes`. Έπειτα, με το βρόχο `forvalues x = `num_of_successes' / `num_vars' {local pmf = `pmf' + binomial(`x', `num_vars', `a')}` υπολογίζεται η τοπική μεταβλητή `pmf` η οποία αποτελεί το άθροισμα των διωνυμικών κατανομών όπου  $k = \text{`num_vars'}$ το πλήθος των μηδενικών υποθέσεων H_0 , $a = 0.5$ η πιθανότητα απόρριψης της μηδενικής υπόθεσης H_0 και $x = r = \text{num_of_successes}$, σύμφωνα με τον μαθηματικό τύπο,

$$p_c = \sum_{x=r}^k \binom{k}{x} a^x (1-a)^{k-x}.$$

Τέλος, δημιουργείται το combined p-value `binomial_p = `pmf'` κάθε παρατήρησης (observation). Το combined p-value αποτελεί την διωνυμική πιθανότητα να παρατηρηθούν τουλάχιστον `r successes`.

5.7 Cauchy

Η εντολή `syntax` προσδιορίζει ότι το πρόγραμμα `logit` αναμένει ως όρισμα μια λίστα αριθμητικών μεταβλητών `varlist(numeric)`. Η εντολή `tempvar num_vars` δημιουργεί μια προσωρινή μεταβλητή με το όνομα `num_vars` για να αποθηκεύσει τον αριθμό των μη ελλিপών τιμών για κάθε observation της λίστας μεταβλητών `varlist`. Η `egen num_vars = rownonmiss(varlist)` χρησιμοποιεί τη συνάρτηση `egen` για τη δημιουργία της μεταβλητής `num_vars`, η οποία ισούται με τον αριθμό των μεταβλητών σε κάθε παρατήρηση (observation). Στη συνέχεια, με το βρόχο `foreach var of varlist `varlist' { gen T_`var' = tan((0.5 - `var') * `pi_val')}` δημιουργείται η μεταβλητή `T` για κάθε μεταβλητή της `varlist` και με την εντολή `egen T_sum = rowtotal(T_*)` υπολογίζεται το άθροισμα `T_sum` των `T` κάθε παρατήρησης (observation) του dataset, σύμφωνα με τον μαθηματικό τύπο,

$$T = w_i \sum_{i=1}^k \tan \{ (0.5 - p_i) \pi \}$$

Τέλος, η μεταβλητή `t = T_sum / `num_vars'` χρησιμοποιείται για τον υπολογισμό των combined p-values για κάθε παρατήρηση `gen cauchy_p = 1 - (0.5 + atan(t / `gamma_val') / `pi_val')` η οποία χρησιμοποιεί την αθροιστική

συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής Cauchy με $x = (t / \text{'gamma_val'})$.

$$F(x) = 0.5 + \frac{\arctan(x)}{\pi}$$

5.8 Minp

Με την `egen min_val = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και στη συνέχεια δημιουργείται η μεταβλητή `gen min_p = 1 - (1 - min_val) ^ `num_vars'`, η οποία υπολογίζει το combined p-value κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = 1 - (1 - \min(p_1, p_2, \dots, p_k))^k.$$

5.9 Bonferroni

Με την `egen min_val = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και στη συνέχεια δημιουργείται η μεταβλητή `gen bonferroni_p = min(1, min_val * `num_vars')`, η οποία υπολογίζει το combined p-value κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times k)$$

, όπου $k = \text{'num_vars'}$.

5.10 Bonferroni_g

Το πρόγραμμα παίρνει ως όρισμα μία λίστα μεταβλητών ``varlist'`. Οι μεταβλητές συσχετίζονται με την εντολή `corr `varlist'` και παράγεται ο πίνακας των συσχετίσεων, ο οποίος αποθηκεύεται ως `matrix corr_matrix = r(C)`. Με το βρόχο `forval i = 1/`nrows' {forval j = 1/`ncols' {local corr_ij = corr_matrix['i', 'j']` κάθε συντελεστής συσχέτισης του πίνακα `corr_matrix` ανατίθεται στη μεταβλητή `corr_ij` και εντοπίζεται ο μεγαλύτερος συντελεστής συσχέτισης `max_corr` με την εντολή `if `corr_ij' > max_corr {scalar max_corr = `corr_ij'}`. Με την `egen `min_val' = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και υπολογίζεται το προσαρμοσμένο πλήθος μεταβλητών `g_adjusted = (`num_vars' + 1) - (1 + (`num_vars' - 1) * max_corr)`. Στη συνέχεια, δημιουργείται η μεταβλητή `gen bonferroni_p = min(1, min_val * g_adjusted)`, η οποία υπολογίζει το combined p-value κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times g_adjusted).$$

5.11 cheverud_nyholtp

Αρχικά, με την εντολή `mkmat var1-var23, matrix(correlation_matrix)` κατασκευάζεται ένας πίνακας `correlation matrix` που περιέχει τις μεταβλητές της `varlist` που αποτελούν τον πίνακα συσχέτισης για των παρατηρήσεων(SNPs). Στη συνέχεια, με τον πίνακα `matrix eigenvalues lambda V = correlation_matrix` υπολογίζονται οι ιδιοτιμές και τα ιδιοδιανύσματα του μη συμμετρικού πίνακα `correlation matrix` που τοποθετούνται στους πίνακες `lambda` και `V` αντίστοιχα. Στη συνέχεια, χρησιμοποιείται η γλώσσα προγραμματισμού `mata`, η οποία είναι μια γλώσσα πινάκων. Με την εντολή `lambda = st_matrix(«lambda»)` ανακτάται ο πίνακας με το όνομα `lambda` από τη Stata στο χώρο εργασίας `mata` και με την `st_matrix("variance", diagonal(variance(lambda)))` υπολογίζεται η διακύμανση των ιδιοτιμών του πίνακα `lambda` η οποία αποθηκεύεται στον πίνακα Stata με όνομα `variance`. Μετά την έξοδο από το μπλοκ `Mata(end_mata)`, ο πίνακας `variance` ανατίθεται σε έναν νέο πίνακα Stata με όνομα `eig_variance` και η δειγματική διακύμανση που περιέχει ο πίνακας αποθηκεύεται στη μεταβλητή `scalar sample_variance = eig_variance[1,1]`. Τέλος, υπολογίζονται τα `effective_number_of tests` με τη συνάρτηση `gen effective_number_of_tests = 1 + (num_vars' - 1) * (1 - sample_variance / num_vars')` σύμφωνα με τον τύπο που προτάθηκε από τους Cheverud και Nyholt όπου

$$m_{CN} = 1 + (k - 1) \left(1 - \frac{\text{Var}(\lambda)}{k}\right).$$

Με την `egen min_val = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και στη συνέχεια δημιουργείται η μεταβλητή `gen bonferroni_p = min(1, min_val * `effective_number_of_tests')`, η οποία υπολογίζει το `combined p-value` κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times \text{effective_number_of_tests}).$$

5.12 Li_jip

Αρχικά, με την εντολή `mkmat var1-var23, matrix(correlation_matrix)` κατασκευάζεται ένας πίνακας `correlation matrix` που περιέχει τις μεταβλητές της `varlist` που αποτελούν τον πίνακα συσχέτισης για των παρατηρήσεων(SNPs). Στη συνέχεια, με τον πίνακα `matrix eigenvalues lambda V = correlation_matrix` υπολογίζονται οι ιδιοτιμές και τα ιδιοδιανύσματα του μη συμμετρικού πίνακα `correlation matrix` που τοποθετούνται στους πίνακες `lambda` και `V` αντίστοιχα. Στη συνέχεια, χρησιμοποιείται η γλώσσα προγραμματισμού `mata`, η οποία είναι μια γλώσσα πινάκων. Με την εντολή `lambda = st_matrix(«lambda»)` ανακτάται ο πίνακας με το όνομα `lambda` από τη Stata στο χώρο εργασίας `mata`, υπολογίζονται οι απόλυτες τιμές

του `lambda` και αποθηκεύονται στον πίνακα `abs_eig = abs(lambda)` και με την `st_matrix("abs_eig", abs_eig)` αποθηκεύεται ως πίνακα Stata με όνομα `abs_eig`. Μετά την έξοδο από το μπλοκ `Mata(end_mata)`, υπολογίζεται το πλήθος των γραμμών και των στηλών του πίνακα `abs_eig` και αποθηκεύονται στις τοπικές μεταβλητές `local rows = rowsof(abs_eig)` και `local cols = colsof(abs_eig)`. Με το βρόχο `forval i = 1/`rows' {forval j = 1/`cols' {}}` υπολογίζεται η μεταβλητή h , η οποία ισούται με $h = 1 + (\text{abs_eig}[i, j] - \text{floor}(\text{abs_eig}[i, j]))$ αν το τρέχον στοιχείο του πίνακα `abs_eig[i, j]` είναι ≥ 1 και με $h = \text{abs_eig}[i, j] - \text{floor}(\text{abs_eig}[i, j])$ αν το τρέχον στοιχείο του πίνακα `abs_eig[i, j]` είναι < 1 , σύμφωνα με τη συνάρτηση

$$h(x) = I(x \geq 1) + (x - \lfloor x \rfloor).$$

Τέλος, υπολογίζονται τα `effective_number_of_tests` με τη συνάρτηση `effective_number_of_tests = effective_number_of_tests + `h'`, κατά την οποία κάθε  $h$  που υπολογίζεται προστίθεται στο άθροισμα των `effective_number_of_tests`, σύμφωνα με τον τύπο που προτάθηκε από τους L_i και J_i

$$m_{LJ} = \sum_{i=1}^k h(|\lambda_i|).$$

Με την `egen min_val = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και στη συνέχεια δημιουργείται η μεταβλητή `gen bonferroni_p = min(1, min_val * `effective_number_of_tests')`, η οποία υπολογίζει το combined p-value κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times \text{effective_number_of_tests}).$$

5.13 Gaop

Αρχικά, με την εντολή `mkmat var1-var23, matrix(correlation_matrix)` κατασκευάζεται ένας πίνακας `correlation_matrix` που περιέχει τις μεταβλητές της `varlist` που αποτελούν τον πίνακα συσχέτισης για των παρατηρήσεων (SNPs). Στη συνέχεια, με τον πίνακα `matrix eigenvalues lambda V = correlation_matrix` υπολογίζονται οι ιδιοτιμές και τα ιδιοδιανύσματα του μη συμμετρικού πίνακα `correlation_matrix` που τοποθετούνται στους πίνακες `lambda` και `V` αντίστοιχα. Στη συνέχεια, χρησιμοποιείται η γλώσσα προγραμματισμού `mata`, η οποία είναι μια γλώσσα πινάκων. Με την εντολή `lambda = st_matrix(«lambda»)` ανακτάται ο πίνακας με το όνομα `lambda` από τη Stata στο χώρο εργασίας `mata`, υπολογίζεται το άθροισμα των ιδιοτιμών του `lambda` και αποθηκεύεται στον πίνακα `sum_eig = rowsum(lambda)` και με την `st_matrix("sum_eig", sum_eig)` τον αποθηκεύεται ως πίνακα Stata με όνομα `sum_eig`. Μετά την έξοδο από το μπλοκ `Mata(end_mata)`, το άθροισμα των

ιδιοτιμών που περιέχει ο πίνακας `sum_eig` αποθηκεύεται στη μεταβλητή `total_sum = sum_eig[1,1]`, και στη συνέχεια υπολογίζεται το πλήθος των γραμμών και των στηλών του πίνακα `lambda` και αποθηκεύονται στις τοπικές μεταβλητές `local rows = rowsof(lambda)` και `local cols = colsof(lambda)`. Με το βρόχο `forval i = 1/`rows'{ }` υπολογίζεται το άθροισμα των ιδιοτιμών έως την τιμή `x` ως `x_sum = `x_sum' + lambda[1, `x']`. Τέλος, υπολογίζονται τα `effective_number_of_tests` με τη συνάρτηση `effective_number_of_tests = `x'`, σύμφωνα με τον τύπο που προτάθηκε από τον Gao

$$m_{GAO} = \underset{x \in \{1, \dots, k\}}{\operatorname{argmin}} \left(\frac{\sum_{i=1}^x \lambda_i}{\sum_{i=1}^k \lambda_i} > C \right).$$

Με την `egen min_val = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και στη συνέχεια δημιουργείται η μεταβλητή `gen bonferroni_p = min(1, min_val * `effective_number_of_tests')`, η οποία υπολογίζει το combined p-value κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times \text{effective_number_of_tests}).$$

5.14 Galweyp

Αρχικά, με την εντολή `mkmat var1-var23, matrix(correlation_matrix)` κατασκευάζεται ένας πίνακας `correlation matrix` που περιέχει τις μεταβλητές της `varlist` που αποτελούν τον πίνακα συσχέτισης για των παρατηρήσεων (SNPs). Στη συνέχεια, με τον πίνακα `matrix eigenvalues lambda V = correlation_matrix` υπολογίζονται οι ιδιοτιμές και τα ιδιοδιανύσματα του μη συμμετρικού πίνακα `correlation matrix` που τοποθετούνται στους πίνακες `lambda` και `V` αντίστοιχα. Στη συνέχεια, χρησιμοποιείται η γλώσσα προγραμματισμού `mata`, η οποία είναι μια γλώσσα πινάκων. Με την εντολή `lambda = st_matrix(«lambda»)` ανακτάται ο πίνακας με το όνομα `lambda` από τη Stata στο χώρο εργασίας `mata`, και κατασκευάζεται ο μηδενικός πίνακας `lambda_prime = J(rows(lambda), cols(lambda), 0)`, με ίσο αριθμό γραμμών και στηλών με τον πίνακα `lambda`, ο οποίος στη συνέχεια γεμίζει με τη χρήση του βρόχου `for (i=1; i<=rows(lambda); i++) {for (j=1; j<=cols(lambda); j++) { if (lambda[i, j] > 0) {lambda_prime[i, j] = lambda[i, j]}}}`, με τις θετικές τιμές του πίνακα `lambda` και με την `st_matrix("lambda_prime", lambda_prime)` τον αποθηκεύεται ως πίνακα Stata με όνομα `lambda_prime`. Επιπλέον, υπολογίζεται ο πίνακας `sum_lambda_prime = rowsum(lambda_prime)` ο οποίος περιέχει το άθροισμα των θετικών ιδιοτιμών του πίνακα `lambda_prime` καθώς και ο πίνακας `sum_eig = rowsum(sqrt(lambda_prime))^2` που περιέχει το τετραγωνικό άθροισμα των υπόριζων τιμών του `lambda_prime`. Οι πίνακες `sum_lambda_prime` και `sum_eig` αποθηκεύονται ως ομώνυμοι πίνακες Stata `st_matrix("sum_eig", sum_eig)` και

`st_matrix("sum_lambda_prime", sum_lambda_prime)` και τα αθροίσματα που περιέχονται στους πίνακες αποθηκεύονται στις μεταβλητές `sum_lambda_prime = sum_lambda_prime[1,1]` και `sum_squared_lambda_prime = sum_eig[1,1]`. Τέλος, υπολογίζονται τα `effective_number_of_tests` με τη συνάρτηση `effective_number_of_tests = sum_squared_lambda_prime / sum_lambda_prime`, σύμφωνα με τον τύπο που προτάθηκε από τον Galway

$$m_{GAL} = \frac{(\sum_{i=1}^k \sqrt{\lambda_i'})^2}{\sum_{i=1}^k \lambda_i}.$$

Με την `egen min_val = rowmin(`varlist')` υπολογίζεται για κάθε παρατήρηση η μεταβλητή με την ελάχιστη τιμή και στη συνέχεια δημιουργείται η μεταβλητή `gen bonferroni_p = min(1, min_val * `effective_number_of_tests')`, η οποία υπολογίζει το combined p-value κάθε παρατήρησης, σύμφωνα με το μαθηματικό τύπο

$$p_c = \min(1, \min(p_1, p_2, \dots, p_k) \times \text{effective_number_of_tests}).$$

5.15 EBM

Το πρόγραμμα παίρνει ως όρισμα μία λίστα μεταβλητών `varlist`. Με την `egen m = rowmean(`varlist')` υπολογίζεται η μέση τιμή της `varlist` και με την `egen sd = rowstd(`varlist')` υπολογίζεται η τυπική απόκλιση (standard deviation) της `varlist`. Στη συνέχεια, οι μεταβλητές της λίστας `varlist` συσχετίζονται με την εντολή `corr `varlist', cov`, και παράγεται ο πίνακας των συνδιακυμάνσεων λόγω του ορίσματος `cov`, ο οποίος αποθηκεύεται ως `matrix covar_matrix = r(C)`. Με το βρόχο `forval r = 1/`m' { forval c = `r'/`m' { local cov_sum = `cov_sum' + covar_matrix[`r', `c'] }}`, υπολογίζεται το άθροισμα `cov_sum = `cov_sum' + covar_matrix[`r', `c']` των συνδιακυμάνσεων του `covar_matrix`, το οποίο και εφαρμόζεται στον τύπο για τον υπολογισμό της διακύμανσης `Var = 4 * `m' + 2 * `cov_sum'`, σύμφωνα με τον τύπο

$$\text{var}[\Psi] = 4k + 2 \sum_{i < j} \text{cov}(-2\log P_i, -2\log P_j)$$

. Στη συνέχεια, με το βρόχο `foreach var of varlist `varlist' { gen log_`var' = ln(`var') }` δημιουργείται η μεταβλητή `log` η οποία υπολογίζει το λογάριθμο για κάθε μεταβλητή της `varlist` και με την εντολή `egen chi_squared_sum = rowtotal(log_*)` υπολογίζεται το άθροισμα των `log` κάθε παρατήρησης (observation) του dataset, σύμφωνα με τον μαθηματικό τύπο

$$X^2 = -2 \sum_{i=1}^k \log(p_i).$$

Τέλος, δημιουργείται το combined p-value των παρατηρήσεων για τη μέθοδο του Fisher `gen fisher_p = 1 - chi2(df_fisher, -2 *`

`chi_squared_sum`) όπου *chi2* είναι η αθροιστική συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής X^2 με `df_fisher = 2 * `m'` βαθμούς ελευθερίας(όπου `m = rowsof(covar_matrix)` το πλήθος των γραμμών του πίνακα `covar_matrix`), σύμφωνα με τον μαθηματικό τύπο,

$$p_c = 1 - F(X^2, 2k)$$

, καθώς και `gen p_brown = chi2(df_brown, -2 * chi_squared_sum_brown)`, όπου `df_brown = 2 * Expected^2 / Var` βαθμούς ελευθερίας, `Expected = 2 * `m'` η αναμενόμενη(μέση) τιμή και `chi_squared_sum_brown = chi_squared_sum / `c'`.

5.16 KstsMethodEBM

Το πρόγραμμα παίρνει ως όρισμα μία λίστα μεταβλητών ``varlist'`. Οι μεταβλητές συσχετίζονται με την εντολή `corr `varlist'` και παράγεται ο πίνακας των συσχετίσεων, ο οποίος αποθηκεύεται ως `matrix corr_matrix = r(C)`. Με το βρόχο `forval i = 1/`nrows' {forval j = 1/`ncols' {local corr_ij = corr_matrix[`i', `j']` κάθε συντελεστής συσχέτισης του πίνακα `corr_matrix` ανατίθεται στη μεταβλητή `corr_ij`, η οποία χρησιμοποιείται για τον υπολογισμό του συντελεστή συνδιακύμανσης `covar = a1 * `corr_ij' + a2 * (`corr_ij'^ 2) + a3 * (`corr_ij'^ 3)` με την πολωνυμική προσέγγιση του Kost. Οι συντελεστές συνδιακύμανσης που υπολογίζονται για κάθε θέση τοποθετούνται στην αντίστοιχη θέση του `covar_matrix[`i', `j'] = covar`. Επιπλέον, υπολογίζεται το άθροισμα `cov_sum = `cov_sum' + covar_matrix[`r', `c']` των συνδιακυμάνσεων του `covar_matrix`, το οποίο και εφαρμόζεται στον τύπο για τον υπολογισμό της διακύμανσης `Var = 4 * `m' + 2 * `cov_sum'`, σύμφωνα με τον τύπο

$$var[\Psi] = 4k + 2 \sum_{i < j} cov(-2 \log P_i, -2 \log P_j)$$

. Στη συνέχεια, με το βρόχο `foreach var of varlist `varlist' { gen log_`var' = ln(`var')}` δημιουργείται η μεταβλητή `log` η οποία υπολογίζει το λογάριθμο για κάθε μεταβλητή της `varlist` και με την εντολή `egen chi_squared_sum = rowtotal(log_*)` υπολογίζεται το άθροισμα των `log` κάθε παρατήρησης(observation) του dataset, σύμφωνα με τον μαθηματικό τύπο

$$X^2 = -2 \sum_{i=1}^k \log(p_i).$$

Τέλος, δημιουργείται το combined p-value των παρατηρήσεων για τη μέθοδο του Fisher `gen fisher_p = 1 - chi2(df_fisher, -2 * chi_squared_sum)` όπου *chi2* είναι η αθροιστική συνάρτηση κατανομής(cumulative distribution function(cdf)) της κατανομής X^2 με `df_fisher = 2 * `m'` βαθμούς ελευθερίας(όπου `m = rowsof(covar_matrix)` το πλήθος των γραμμών του πίνακα `covar_matrix`), σύμφωνα με τον μαθηματικό τύπο,

$$p_c = 1 - F(X^2, 2k)$$

, καθώς και `gen p_brown = chi2(df_brown, -2 * chi_squared_sum_brown)`, όπου `df_brown = 2 * Expected^2 / Var` βαθμούς ελευθερίας, `Expected = 2 * `m'` η αναμενόμενη(μέση) τιμή και `chi_squared_sum_brown = chi_squared_sum / `c'`.

5.17 BrownbyYang

Το πρόγραμμα παίρνει ως όρισμα μία λίστα μεταβλητών ``varlist'`. Οι μεταβλητές συσχετίζονται με την εντολή `corr `varlist'` και παράγεται ο πίνακας των συσχετίσεων, ο οποίος αποθηκεύεται ως `matrix corr_matrix = r(C)`. Με το βρόχο `forval i = 1/`nrows' {forval j = 1/`ncols' {local corr_ij = corr_matrix[`i', `j']` κάθε συντελεστής συσχέτισης του πίνακα `corr_matrix` ανατίθεται στη μεταβλητή `corr_ij`, η οποία χρησιμοποιείται για τον υπολογισμό του διορθωμένου συντελεστή συσχέτισης `biased_corrected_cor = `corr_ij' * (1 + (1 - `corr_ij'^2 / 2 * (`ncols' - 3)))`. Στη συνέχεια, υπολογίζονται ο μεροληπτικός εκτιμητής της διακύμανσης  $\delta_{jk}$  `f_r = c1 * (`biased_corrected_cor' ^ 2) + c2 * (`biased_corrected_cor' ^ 4) + c3 * (`biased_corrected_cor' ^ 6) + c4 * (`biased_corrected_cor' ^ 8) + c5 * (`biased_corrected_cor' ^ 10)`, που ισούται με ένα πολυώνυμο 10^{ης} τάξης αποτελούμενο από το διορθωμένο συντελεστή συσχέτισης και τους συντελεστές `c1 = 3.9081`, `c2 = 0.0313`, `c3 = 0.1022`, `c4 = -0.1378`, `c5 = 0.0941`, και η μεροληψία `bias = (c1 / `ncols') * (1 - `biased_corrected_cor'^2)^2`. Οι μετρήσεις αυτές χρησιμοποιούνται για τον υπολογισμό του αμερόληπτου συντελεστή διακύμανσης `unbiased_delta = f_r - `bias'`, ο οποίος ανατίθεται σε κάθε θέση `delta_matrix[`i', `j']` του πίνακα `delta matrix`.

$$\delta_{jk} = cov\{-2\log(p_j), -2\log(p_k)\}$$

Επιπλέον, υπολογίζεται το άθροισμα `delta_sum = `delta_sum' + delta_matrix[`r', `c']` των συνδιακυμάνσεων του `delta_matrix`, το οποίο και εφαρμόζεται στον τύπο για τον υπολογισμό της διακύμανσης `Var = 4 * `m' + 2 * `delta_sum'`, σύμφωνα με τον τύπο

$$\sigma^2 = var[T] = 4m + \sum_{i \neq j} (f(r_{j,k}) - \frac{c_1}{n} (1 - r_{j,k}^2)^2).$$

Επιπλέον υπολογίζονται οι βαθμοί ελευθερίας για τη μέθοδο του Fisher `df_fisher = 2.0 * m`, όπου `m = rowsof(delta_matrix)`, και με το βρόχο `foreach var of varlist `varlist' { gen log_`var' = ln(`var')}` δημιουργείται η μεταβλητή `log` η οποία υπολογίζει το λογάριθμο για κάθε μεταβλητή της `varlist` και με την εντολή `egen chi_squared_sum = rowtotal(log_*)` υπολογίζεται το άθροισμα των `log` κάθε παρατήρησης(observation) του dataset, σύμφωνα με τον μαθηματικό τύπο

$$X^2 = -2 \sum_{i=1}^k \log(p_i)$$

Τέλος, υπολογίζονται τα combined p-values όπου $_p_yang = 1 - \text{gammaden}(-2 * \text{'chi_squared_sum'}, \text{'v'}/2, 2 * \text{gamma_value})$, όπου

$$globalp = 1 - \Gamma(T, v/2, 2\gamma)$$

, όπου gammaden η συνάρτηση πυκνότητας της κατανομής Γ με παράμετρο *shape* $v/2$, όπου $v = 2 * (\text{mean}^2 / \text{Var})$,

$$v = 2(\mu^2 / \sigma^2)$$

και παράμετρο *scale* 2γ , όπου $\text{gamma_value} = \text{Var} / \text{mean}$,

$$\gamma = \sigma^2 / \mu$$

και $_yfisher_p = 1 - \text{chi2}(\text{df_fisher}, -2 * \text{'chi_squared_sum'})$ με βάση την εξίσωση του Fisher :

$$p_c = 1 - F(X^2, 2k)$$

, όπου chi2 είναι η αθροιστική συνάρτηση κατανομής (cumulative distribution function(cdf)) της κατανομής X^2 και $2k$ οι βαθμοί ελευθερίας .

6. Αποτελέσματα

Το πρόγραμμα λαμβάνει ως είσοδο ένα αρχείο κειμένου n γραμμών και m στηλών, καθώς και τις επικεφαλίδες, που περιέχουν τις τιμές p -values ή τα z -scores που πρέπει να συνδυαστούν. Διαφορετικές γραμμές μπορούν να αντιστοιχούν σε διαφορετικά SNPs ή γονίδια από ένα GWAS, ή ανιχνευτές/γονίδια από μελέτες μεταγραφώματος. Το διαφορετικές στήλες μπορεί να αντιστοιχούν σε διαφορετικές στατιστικές δοκιμές, διαφορετικά χαρακτηριστικά, διαφορετικά SNPs εντός ενός γονιδίου κ.ο.κ., ανάλογα με τη ρύθμιση. Σε περίπτωση που το ίδιο SNP/γονίδιο εμφανίζεται σε πολλαπλές γραμμές, μπορεί να πραγματοποιηθεί μετα-ανάλυση πριν από το συνδυασμό των διαφορετικών τιμών p -values από διαφορετικές στήλες.

Για να αποδειχθεί η εγκυρότητα των αποτελεσμάτων των μεθόδων συνδυασμού γίνεται σύγκριση με τα αποτελέσματα των μεθόδων MIN2 και MAX του λογισμικού πακέτου GVAR(Dimou, et. al.,) (Πίνακας 1). Με τη χρήση των μεθόδων `cauchy`, `minp`, `CMC`, `MCM` γίνεται συνδυασμός των p -values(`_p1`, `_p2`, `_p3`), που προκύπτουν από το μετασχηματισμό των z _scores, `_z1`, `_z2`, `_z3`, τα οποία προκύπτουν με την εκτέλεση της εντολής `gvar s0 s1 s2 r0 r1 r2, method(max)`. Στη συνέχεια, χρησιμοποιούνται οι εντολές `metacp _p1-_p3, method(cauchy)`, `metacp _p1-_p3, method(minp)`, `metacp _cauchy_p _min_p, method(CMC)`, `metacp _p1-_p3, method(MCM)`. Όπως ήταν αναμενόμενο τα αποτελέσματα των μεθόδων του πακέτου `metacp` προσεγγίζουν αυτά του GVAR(Πίνακας 2). Παρακάτω παρατίθενται οι εντολές που χρησιμοποιήθηκαν για τον υπολογισμό των `_pmax` και `_z_max` του GVAR και για τον συνδυασμό των `_p1`, `_p2`, `_p3` για τον υπολογισμό των `_cauchy_p`, `_min_p`, `_CMC_p`, `_MCM_p`.

```
. use "C:\Users\USER\Stata\gvar.dta", clear

. gvar s0 s1 s2 r0 r1 r2, method(max)

. gen _p1 = 2 * (1 - normal(abs(_z1)))

. gen _p2 = 2 * (1 - normal(abs(_z2)))

. gen _p3 = 2 * (1 - normal(abs(_z3)))

. metacp _p1-_p3, method(cauchy)

. metacp _p1-_p3, method(minp)

. metacp _cauchy_p _min_p, method(CMC)

. metacp _p1-_p3, method(MCM)
```

Πίνακας 1. SNPs που σχετίζονται με τον T2DM και εντοπίστηκαν με τη χρήση Human1 BeadArrays. Είκοσι οκτώ SNPs του σταδίου 1 πέρασαν το όριο επιλογής ($\lambda_{\text{mbdcorrected}} \text{pMAX} < 1 \times 10^{-4}$) για τη συστοιχία Human1. Τα ονόματα γονιδίων αναφέρονται στο γονίδιο στο οποίο εντοπίζεται το SNP ή στο πλησιέστερο γονίδιο. Τα r0, r1 και r2 είναι οι αριθμοί γονότυπων για τις περιπτώσεις, όπου r2 είναι η καταμέτρηση για τους ομοζυγώτες που φέρουν το αλληλόμορφο κινδύνου. s0, s1 και s2 είναι οι αντίστοιχες καταμετρήσεις για τους ελέγχους.

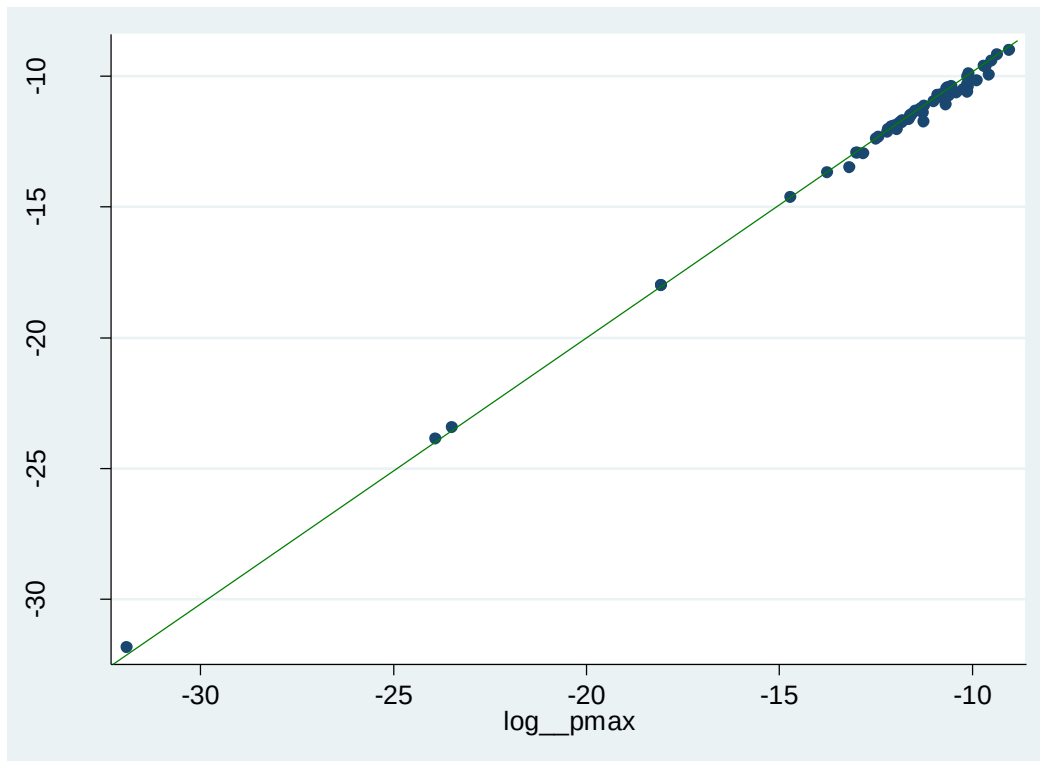
	snp	chr	r0	r1	r2	s0	s1	s2	closest_gene
1	rs7900150	10	129	326	229	198	325	143	TCF7L2
2	rs7100927	10	129	328	229	198	326	143	TCF7L2
3	rs1193179	1	340	288	58	423	202	44	CAMTA1
4	rs932206	2	134	285	267	158	333	178	CXCR4
5	rs1978717	19	300	308	75	364	260	36	ZNF615
6	rs11084127	19	300	311	75	363	266	36	ZNF615
7	rs1111875	10	77	298	310	122	316	231	HHEX
8	rs11084128	19	299	302	76	363	264	36	ZNF615
9	rs282705	4	24	239	423	60	264	345	LOC644419
10	rs1836002	19	300	311	75	364	268	37	ZNF615
11	rs3740878	11	25	273	386	65	249	353	EXT2
12	rs11037909	11	25	274	387	65	251	353	EXT2
13	rs8101509	19	303	297	80	344	285	33	ZNF649
14	rs2499953	11	646	39	1	660	9	0	MMP26
15	rs6670163	1	34	204	448	45	266	358	RYR2
16	rs945384	9	614	69	3	640	28	1	FAM69B
17	rs1113132	11	25	271	390	63	251	355	EXT2
18	rs2278419	19	319	294	69	368	270	27	ZNF350
19	rs7651936	3	156	326	204	186	351	131	LOC131149
20	rs10211998	22	26	189	466	37	249	380	FLJ90680
21	rs5756371	22	28	194	462	41	251	376	FLJ90680
22	rs13064991	3	15	177	494	27	233	409	SLC6A20
23	rs1256517	14	471	184	17	527	116	15	LOC646279

	snp	chr	r0	r1	r2	s0	s1	s2	closest_gene
24	rs6541240	1	83	253	350	101	303	265	TTC13
25	rs6413504	19	126	340	215	185	313	163	LDLR
26	rs2050831	9	36	184	466	29	258	381	VPS13A
27	rs873492	22	270	321	95	337	256	76	FLJ27365
28	rs11078674	17	297	304	83	354	267	46	NLGN2
29	rs7903146	10	197	348	149	335	254	65	TCF7L2
30	rs12255372	10	221	342	131	332	267	55	TCF7L2
31	rs10885409	10	121	324	248	191	325	138	TCF7L2
32	rs7904519	10	118	329	247	189	325	140	TCF7L2
33	rs932206	2	135	282	272	156	328	170	CXCR4
34	rs35666	12	494	188	12	539	108	7	BTG1
35	rs7950175	11	30	216	448	45	274	335	KIRREL3
36	rs4918789	10	158	338	191	220	309	125	TCF7L2
37	rs7923837	10	66	300	328	116	296	242	HHEX
38	rs1037386	3	60	342	292	113	278	263	CNTN6
39	rs1193179	1	350	283	59	414	201	39	CAMTA1
40	rs1256526	14	202	358	132	270	283	101	LOC646279
41	rs6894954	5	313	275	104	308	301	45	KCTD16
42	rs290483	10	72	271	339	111	284	244	TCF7L2
43	rs7712842	5	315	275	104	310	299	45	KCTD16
44	rs2317948	1	48	327	311	97	293	264	TMEM61
45	rs859101	1	205	359	130	271	278	105	SLC44A3
46	rs2327112	6	146	325	223	157	360	136	LOC389365
47	rs1111875	10	77	302	315	119	308	227	HHEX
48	rs2589001	16	196	334	163	249	305	99	LOC654106
49	rs9290240	3	310	313	69	374	238	42	SI
50	rs282705	4	23	244	427	58	258	338	LOC644419
51	rs2866016	4	37	270	387	70	289	295	TSPAN5
52	rs7949067	11	138	356	188	201	295	154	ALX4
53	rs1978717	19	304	315	75	357	259	37	ZNF615
54	rs7480010	11	301	327	66	363	246	45	LOC387761
55	rs2288887	19	304	314	75	357	259	37	ZNF615
55	rs2288887	19	304	314	75	357	259	37	ZNF615
56	rs729287	11	26	275	393	64	244	346	ALX4
57	rs12629276	3	18	195	478	34	242	377	RAFTLIN
58	rs1005316	17	13	224	457	44	211	399	LOC124685
59	rs1888533	21	136	377	181	196	294	164	LOC728117
60	rs375694	21	415	238	41	465	166	22	SLC37A1
61	rs1293143	20	112	315	267	131	347	176	PFDN4
62	rs13266634	8	54	229	411	53	293	307	SLC30A8
63	rs1293144	20	134	323	236	157	346	151	PFDN4
64	rs11249433	1	88	352	253	141	305	208	LOC653464
65	rs2876711	13	99	322	272	121	351	182	KCTD12
66	rs231461	17	529	153	12	559	89	6	PYY

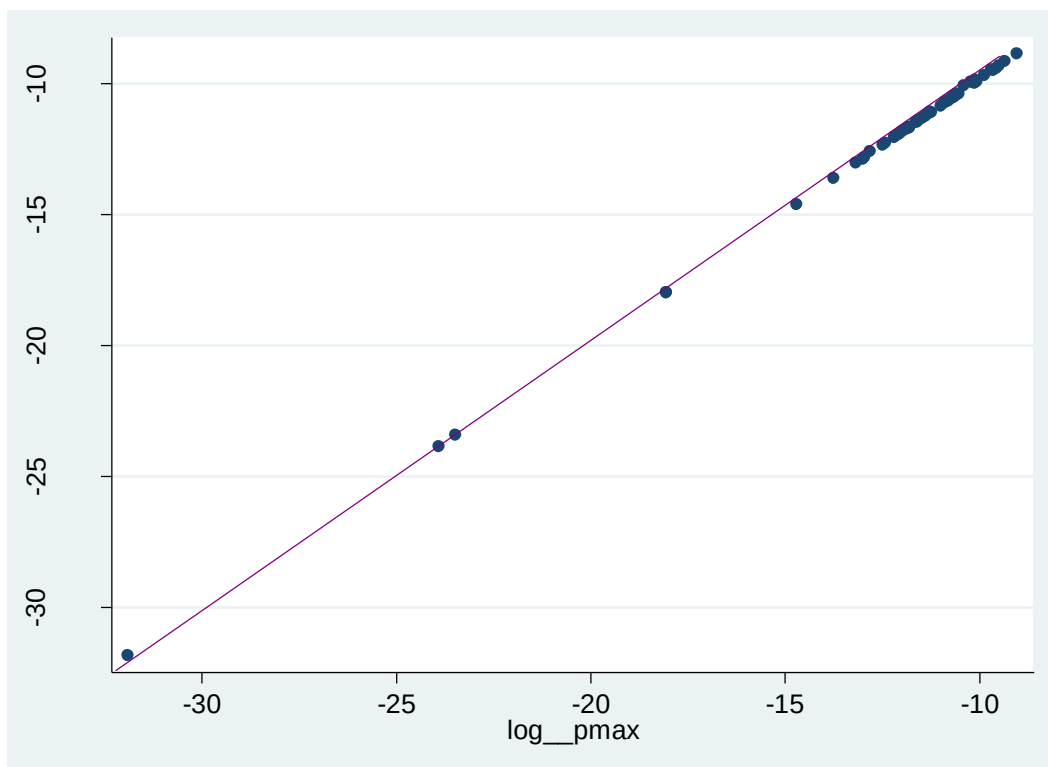
Πίνακας 2. Τα z-scores $_z1$, $_z2$, $_z3$, καθώς και τα $_pmax$, $_zmax$ προκύπτουν από το πακέτο GWAR. Τα $_z1$, $_z2$, $_z3$ στη συνέχεια μετατρέπονται σε p-values, $_p1$, $_p2$, $_p3$, αντίστοιχα. Τα $p1$, $p2$, $p3$ συνδυάζονται με τις μεθόδους cauchy, min, MCM, CMC του πακέτου metacp. Το $_pMAX_corr$ αναφέρεται στη μικρότερη τιμή p-value που προέκυψε από οποιοδήποτε από τα τρία γενετικά μοντέλα, ενώ το $_pMAX_perm$ που προέκυψε από τις μεταθέσεις εκτιμήθηκε δημιουργώντας τη μηδενική κατανομή του στατιστικού MAX. Και οι δύο p-τιμές έχουν διορθωθεί για τον πληθωρισμό της διακύμανσης(variance inflation).

$_z1$	$_z2$	$_z3$	$_pmax$	$_z_max$	$_p1$	$_p2$	$_p3$	$_cauchy_p$	$_min_p$	$_CMC_p$	$_MCM_p$	$_pMAX_corr$	$_pMAX_perm$
4.9370102	5.8467402	4.6607212	1.44e-08	5.668154	7.93e-07	5.01e-09	3.15e-06	1.49e-08	1.50e-08	1.50e-08	2.98e-08	5.1e-08	2.1e-08
4.9189044	5.845574	4.6738512	1.45e-08	5.666929	8.70e-07	5.05e-09	2.96e-06	1.50e-08	1.51e-08	1.51e-08	3.01e-08	5.2e-08	2.1e-08
1.3098616	4.5207878	5.0706497	1.07e-06	4.878254	.1902427	6.16e-06	3.96e-07	1.12e-06	1.19e-06	1.15e-06	2.23e-06	1.2e-06	6.3e-07
4.8256693	4.1408433	1.8278999	3.84e-06	4.62006	1.40e-06	.0000346	.0675646	4.02e-06	4.19e-06	4.10e-06	8.05e-06	4.6e-06	2.8e-06
3.6770322	4.8083079	4.114246	4.10e-06	4.606256	.000236	1.52e-06	.0000388	4.37e-06	4.57e-06	4.46e-06	8.73e-06	7.5e-06	4.9e-06
3.693354	4.7223442	3.989945	6.26e-06	4.517524	.0002213	2.33e-06	.0000661	6.69e-06	6.99e-06	6.84e-06	.0000134	.000011	7.4e-06
4.0287914	4.6924596	3.6345976	7.35e-06	4.483241	.0000561	2.70e-06	.0002784	7.66e-06	8.10e-06	7.87e-06	.0000153	.000012	8.6e-06
3.8329542	4.6872991	3.8750655	7.42e-06	4.481356	.0001266	2.77e-06	.0001066	7.93e-06	8.31e-06	8.11e-06	.0000159	.000013	8.8e-06
3.7483082	4.6834651	4.1746978	7.44e-06	4.480587	.000178	2.82e-06	.0000298	7.62e-06	8.46e-06	8.02e-06	.0000152	.000013	9.0e-06
3.6105972	4.6402146	3.9309996	9.31e-06	4.432698	.0003055	3.48e-06	.0000846	9.92e-06	.0000104	.0000102	.0000198	.000015	.000011
1.2954782	2.8509633	4.4881598	.0000189	4.27806	.1905153	.0043587	7.18e-06	.0000215	.0000216	.0000215	.000043	.000018	.000013
1.3487339	2.892756	4.4873202	.0000189	4.277194	.1774225	.0038188	7.21e-06	.0000216	.0000216	.0000216	.0000432	.000018	.000013
4.471592	4.0673072	2.7141616	.0000206	4.258153	7.76e-06	.0000476	.0066444	.00002	.0000233	.0000215	.00004	.000022	.000016
.48139636	4.2872719	4.3481025	.0000309	4.16671	.6302348	.0000181	.0000137	.0000234	.0000412	.0000299	.0000468	.000023	.000017
4.4211039	4.1302207	1.3903546	.0000255	4.210418	9.82e-06	.0000362	.1644212	.0000232	.0000295	.0000259	.0000464	.000027	.00002
.97642677	4.2661009	4.3169034	.000037	4.125718	.328853	.0000199	.0000158	.0000264	.0000475	.000034	.0000529	.000036	.000029
1.400881	2.8553208	4.3111787	.0000422	4.095171	.1612497	.0042993	.0000162	.0000485	.0000487	.0000486	.0000971	.000037	.000029
4.320072	4.3027789	3.1435535	.0000408	4.102922	.0000156	.0000169	.0016691	.0000242	.0000468	.0000319	.0000484	.000041	.000033
4.3174139	3.9626441	2.1609704	.0000426	4.092917	.0000158	.0000741	.0306976	.000039	.0000474	.0000428	.0000781	.000041	.000033
4.3173449	4.1441631	1.510086	.0000404	4.10509	.0000158	.0000341	.1310215	.0000324	.0000474	.0000385	.0000648	.000041	.000033
4.2628682	4.1487943	1.70758	.0000517	4.047924	.0000202	.0000334	.0877143	.0000377	.0000605	.0000465	.0000755	.000051	.000042
4.2450852	4.3215896	1.963775	.0000389	4.113665	.0000219	.0000155	.0495562	.0000272	.0000465	.0000343	.0000544	.000055	.000046
.29762611	3.7620074	4.2137774	.0000623	4.003848	.7659885	.0001686	.0000251	.0000656	.0000753	.0000701	.0001312	.000055	.000046
4.2173607	3.7936099	1.6106235	.0000648	3.994498	.0000247	.0001485	.1072618	.0000636	.0000742	.0000684	.0001271	.000062	.000052
2.8141386	4.2013995	4.1172776	.0000708	3.973634	.0048908	.0000265	.0000383	.0000469	.0000796	.000059	.0000938	.000082	.000071
4.141001	3.1511136	-.78008585	.0000878	3.922141	.0000346	.0016265	.4353403	.0001016	.0001037	.0001026	.0002032	.000084	.000073
1.3789506	3.6224512	4.0765026	.0001192	3.847868	.16791	.0002918	.0000457	.0001186	.0001371	.0001272	.0002371	.000096	.000085
3.275289	4.1823088	3.5498902	.0000755	3.95835	.0010555	.0000289	.0003854	.0000785	.0000866	.0000824	.0001571	.000098	.000087
5.7897985	8.9385502	8.5734008	-2.22e-16	.	7.05e-09	0	0	0	0	0	0	3.2e-17	3.3e-10
5.5686123	7.8313599	7.0581665	1.41e-14	7.695449	2.57e-08	4.88e-15	1.69e-12	1.47e-14	1.47e-14	1.47e-14	2.93e-14	1.4e-13	3.3e-10
5.9576258	6.7545264	5.1065683	4.21e-11	6.596525	2.56e-09	1.43e-11	3.28e-07	4.27e-11	4.30e-11	4.29e-11	8.55e-11	1.8e-10	3.3e-10
5.75303	6.6925448	5.2050785	6.44e-11	6.533219	8.77e-09	2.19e-11	1.94e-07	6.56e-11	6.58e-11	6.57e-11	1.31e-10	2.7e-10	3.3e-10
5.2561707	4.4511022	1.8938721	4.12e-07	5.0631	1.47e-07	8.54e-06	.058242	4.34e-07	4.41e-07	4.37e-07	8.68e-07	6.3e-07	3.9e-07

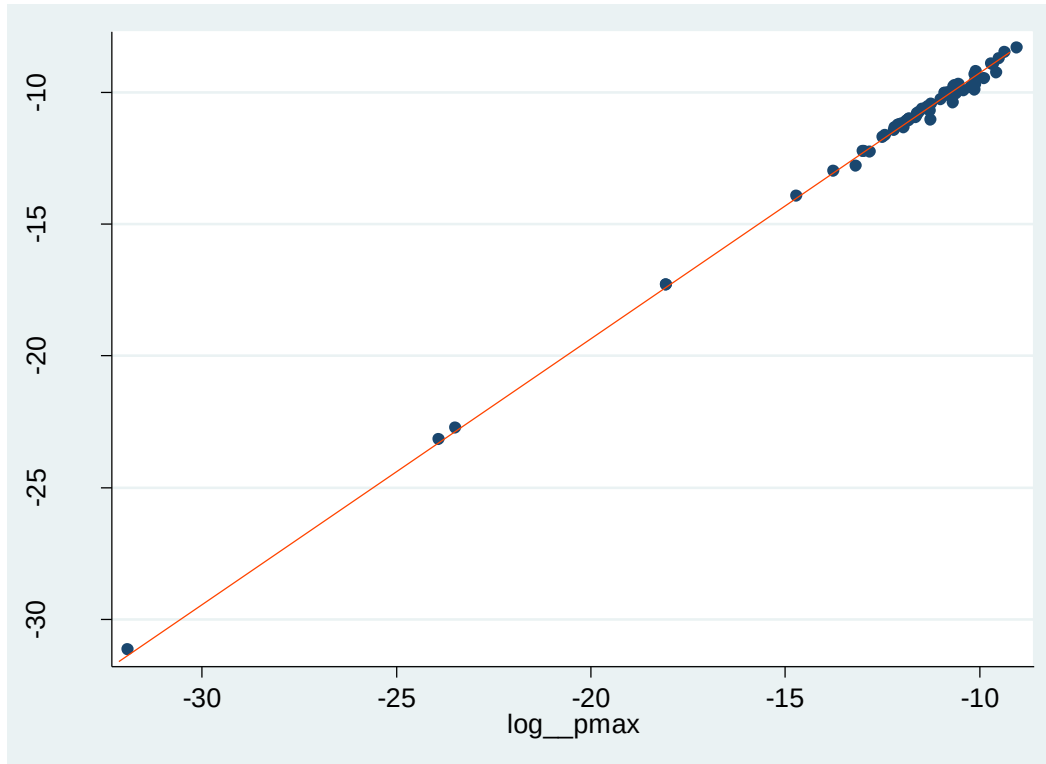
_z1	_z2	_z3	_pmax	_z_max	_p1	_p2	_p3	_cauchy_p	_min_p	_CMC_p	_MCM_p	_pMAX_corr	_pMAX_perm
1.0254352	4.7112115	4.8714238	2.74e-06	4.68924	.3051579	2.46e-06	1.11e-06	2.29e-06	3.32e-06	2.71e-06	4.58e-06	2.1e-06	1.5e-06
4.9572516	4.8546415	2.0476877	1.90e-06	4.763772	7.15e-07	1.21e-06	.0405906	1.35e-06	2.14e-06	1.65e-06	2.69e-06	2.6e-06	1.9e-06
3.7474889	4.9284075	4.3290109	2.31e-06	4.724637	.0001786	8.29e-07	.000015	2.35e-06	2.49e-06	2.41e-06	4.69e-06	3.3e-06	2.4e-06
3.8106529	4.9217025	4.4173894	2.36e-06	4.720218	.0001386	8.58e-07	9.99e-06	2.36e-06	2.57e-06	2.46e-06	4.71e-06	3.4e-06	2.5e-06
.69381603	2.8397901	4.736101	5.94e-06	4.528396	.4877976	.0045143	2.18e-06	6.53e-06	6.54e-06	6.53e-06	.0000131	4.0e-06	3.0e-06
1.8086036	4.4568841	4.7098213	6.58e-06	4.50687	.0705126	8.32e-06	2.48e-06	5.73e-06	7.44e-06	6.47e-06	.0000115	4.6e-06	3.4
1.7601562	4.1098829	4.6472412	9.20e-06	4.435276	.0783813	.0000396	3.36e-06	9.30e-06	.0000101	9.68e-06	.0000186	6.1e-06	4.7e-06
4.7619431	2.7394261	.68548728	5.20e-06	4.556727	1.92e-06	.0061547	.4930365	5.75e-06	5.75e-06	5.75e-06	.0000115	6.4e-06	5.0e-06
4.2146596	4.766536	3.5824556	5.09e-06	4.561114	.000025	1.87e-06	.0003404	5.20e-06	5.62e-06	5.41e-06	.0000104	6.8e-06	5.3e-06
4.7430333	2.7687829	.74018104	5.70e-06	4.537301	2.11e-06	.0056266	.4591902	6.31e-06	6.32e-06	6.32e-06	.0000126	7.0e-06	5.4e-06
1.8367367	3.5546467	4.6148633	.0000106	4.403869	.0662488	.0003785	3.93e-06	.0000117	.0000118	.0000117	.0000234	7.1e-06	5.5e-06
1.2947058	3.7995091	4.568152	.0000134	4.353781	.1954217	.000145	4.92e-06	.0000143	.0000148	.0000145	.0000286	8.8e-06	7.0e-06
4.6901693	2.7509663	1.3202404	7.54e-06	4.477794	7.73e-06	.0001762	.1867548	8.06e-06	8.19e-06	8.13e-06	.0000161	8.8e-06	7.0e-06
3.9966868	4.7013117	3.6961996	7.04e-06	4.492436	.0000642	2.58e-06	.0002189	7.37e-06	7.75e-06	7.56e-06	.0000147	9.1e-06	7.3e-06
3.8716255	4.6900666	3.8387498	7.49e-06	4.479404	.0001081	2.73e-06	.0001237	7.82e-06	8.19e-06	8.00e-06	.0000156	9.5e-06	7.7e-06
2.3658394	4.5674142	4.5440032	.0000131	4.357983	.0179892	4.94e-06	5.52e-06	7.82e-06	.0000148	.0000102	.0000156	9.8e-06	7.9e-06
3.6465555	4.6447862	4.2886279	8.96e-06	4.440902	.0002658	3.40e-06	.000018	8.50e-06	.0000102	9.28e-06	.000017	.000012	.000095
3.9111562	4.6309815	3.64643	9.70e-06	4.423748	.0000919	3.64e-06	.0002659	.0000104	.0000109	.0000106	.0000207	.000012	1.0e-05
1.6176832	3.715377	4.4765368	.0000206	4.258099	.1057309	.0002029	7.59e-06	.0000219	.0000228	.0000223	.0000439	.000013	.000011
3.4151122	4.5980852	3.9870864	.0000114	4.389177	.0006376	4.26e-06	.0000669	.000012	.0000128	.0000124	.0000239	.000014	.000012
1.7551572	4.2478964	4.4531113	.0000224	4.239426	.0792324	.0000216	8.46e-06	.0000182	.0000254	.0000212	.0000365	.000015	.000012
3.4231184	4.5817974	3.9624561	.0000123	4.372358	.0006191	4.61e-06	.0000742	.0000129	.0000138	.0000134	.0000259	.000015	.000013
1.3727163	2.8938945	4.439719	.0000236	4.227986	.1698406	.003805	9.01e-06	.000027	.000027	.000027	.0000539	.000016	.000013
4.3575605	4.5638982	2.4719472	.0000129	4.361877	.0000132	5.02e-06	.0134379	.0000109	.0000151	.0000126	.0000218	.000016	.000014
1.845128	3.1160824	4.4265883	.0000245	4.219065	.0650189	.0018327	9.57e-06	.0000286	.0000287	.0000286	.0000571	.000016	.000014
.4222704	2.9460946	4.4177069	.0000271	4.196792	.6728277	.0032181	9.98e-06	.0000298	.0000299	.0000299	.0000597	.000017	.000015
2.2053692	4.4395591	4.3980911	.0000232	4.231718	.0274282	9.01e-06	.0000109	.0000148	.000027	.0000191	.0000296	.000019	.000016
4.5164203	4.0638457	1.857934	.0000171	4.299389	6.29e-06	.0000483	.0631784	.0000167	.0000189	.0000177	.0000334	.000019	.000016
4.4882945	3.6076558	.22750363	.0000189	4.277021	7.18e-06	.000309	.8200321	.0000211	.0000215	.0000213	.0000421	.000021	.000018
4.4453129	4.0632169	2.0814603	.0000238	4.225434	8.78e-06	.0000484	.0373918	.0000223	.0000263	.0000241	.0000446	.000025	.000023
1.0613865	2.558915	4.2193758	.0000271	4.197315	.0589811	.00347	.0000151	.0000434	.0000453	.0000443	.0000868	.000025	.000023
1.0613865	2.558915	4.2193758	.0000271	4.197315	.0589811	.00347	.0000151	.0000434	.0000453	.0000443	.0000868	.000025	.000023
1.2975792	4.2307734	4.3014774	.000016	4.100058	.194432	.0000233	.000017	.0000294	.0000509	.0000373	.0000589	.000028	.000025



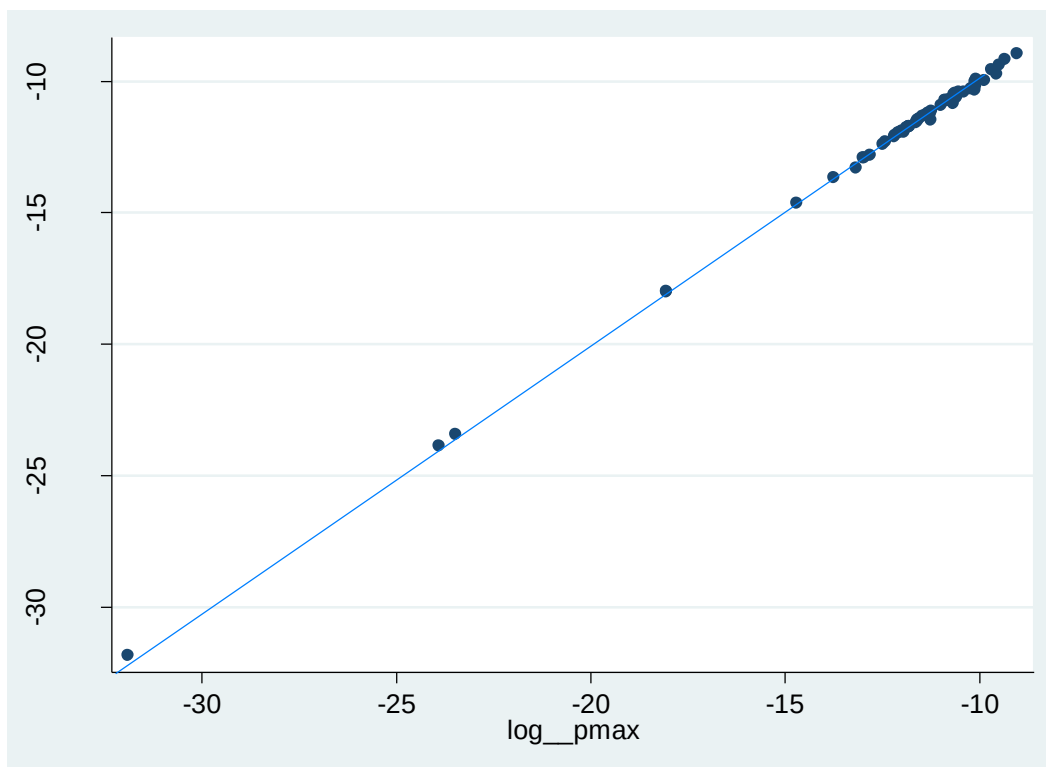
```
. twoway (scatter log__cauchy_p log__pmax, sort)
```



```
. twoway (scatter log__min_p log__pmax, sort)
```



```
. twoway (scatter log__MCM_p log__pmax, sort)
```



```
. twoway (scatter log__CMC_p log__pmax, sort)
```

Τα παραπάνω διαγράμματα αποδεικνύουν τη γραμμική σχέση των συνδυασμένων p-values `_cauchy_p`, `_min_p`, `_CMC_p`, `_MCM_p` με το `_pmax` που υπολογίστηκε από το `gwar`.

7. Συμπεράσματα

Το εργαλείο που παρουσιάζεται στην παρούσα εργασία είναι πολύ γρήγορο και εύκολο στη χρήση, απαιτώντας μόνο ελάχιστες εισαγωγές δεδομένων. Προσφέρει μια χρήσιμη πηγή για το συνδυασμό τόσο ανεξάρτητων όσο και εξαρτημένων p-values, ανταποκρινόμενο σε ποικίλες ανάγκες στην ανάλυση για ερευνητές σε διάφορους τομείς. Ανάλογα με τα δεδομένα εισόδου μπορεί να χρησιμοποιηθεί για δοκιμές με βάση τα γονίδια ή για την ανάλυση πολλαπλών χαρακτηριστικών σε GWAS, ή για το συνδυασμό ποικίλων δεδομένων multi-omics, όπως αυτά ενός TWAS, ενός colocalization ή μιας RNAseq. Σε σύγκριση με άλλα παρόμοια πακέτα (όπως το `poolr` ή το `metap`) προσφέρει τη μεγαλύτερη δυνατή συλλογή μεθόδων ανάλυσης. Ο πηγαίος κώδικας θα είναι διαθέσιμος στο www.github.com/pbagos

Πίνακας 3. Σύγκριση των μεθόδων συνδυασμού που προσφέρει το πακέτο `metacp` σε σχέση με άλλα λογισμικά πακέτα για το συνδυασμό p-values.

	metac p	M AT LA B	meta p	Robust P- value- combin- ation- tests	poolr	SciPy scipy. stats.co mbine_p values	Julia	https://bioconductor.org/packages/release/bioc/html/EmpiricalBrownMethod	Statsmodels (statsmodels. stats.meta_a nalysis.comb ine_pvalues)	Survcomp (combine. test)
Logitp	✓	-	✓	-	-	-	-	-	-	-
Meanp	✓	-	✓	-	-	-	-	-	-	-
Fisher's method	✓		✓	✓	✓	✓	✓	-	✓	✓
Stouffer's method	✓		✓	✓	✓	✓	✓	-	✓	✓
Inverse Chi2	✓	-	-	-	✓	-	-	-	-	+
Binomial test	✓	-	-	-	✓	-	-	-	-	-
Bonferroni 's method	✓	-	-	-	✓		-	-	-	-
MinP	✓			-	✓	✓	✓	-	-	✓
Cauchy test (CCT)	✓		-	✓	-	-	-	-	-	-
MinP- Cauchy- MinP (MCM)	✓		-	✓	-	-	-	-	-	-
Cauchy- MinP- Cauchy (CMC)	-	-	-	✓						
Harmonic Mean P-value (HMP)	✓		-	-	✓	-	-	-	-	-
Empirical Brown's Method	✓		-	-	-	-	-	✓	-	-

(EBM)										
Kost's Method	✓	-	-	-	+	-	✓	-	-	
Brown's method By Yang	✓	-	-	-	-	-	✓	-	-	
Bonferroni's with Cheverud-Nyholt Effective number of tests	✓	-	-	✓	-	-	-	-	-	
Bonferroni's with Gao's Effective number of tests	✓	-	-	✓	-	-	-	-	-	
Bonferroni's with Li & Ji Effective number of tests	✓	-	-	✓	-	-	-	-	-	
Bonferroni's with Galwey's Effective number of tests	✓	-	-	✓	-	-	-	-	-	

8. Αναφορές

1. Poole William, Gibbs David L., Shmulevich Ilya, Bernard B., Knijnenburg Theo A. (2016) “Combining dependent P-values with an empirical adaptation of Brown’s method”, *Bioinformatics*, 32, i430–i436, doi: 10.1093/bioinformatics/btw438
2. Cinar O., Viechtbauer W. (2022), “The poolr Package for Combining Independent and Dependent p Values”, *Journal of Statistical Software*, Volume 101, Issue 1., doi: 10.18637/jss.v101.i01
3. Wilsona Daniel J. (2019), “The harmonic mean p-value for combining dependent tests”, *PNAS*, 116 (4), 1195–1200, <https://doi.org/10.1073/pnas.1814092116>
4. Chen Zhongxue, (2022), “Robust tests for combining p-values under arbitrary dependency structures”, www.nature.com/scientificreports/, 2:3158, <https://doi.org/10.1038/s41598-022-07094-7>
5. Shi Q., Pavey E.S. and Carter R.E. (2012), “Bonferroni-based correction factor for multiple, correlated endpoints”, *Pharmaceutical Statistics*, 11, 300–309, (wileyonlinelibrary.com) DOI: 10.1002/pst.1514
6. Liu Y., Xie J. (2020), “Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures”, *HHS Public Access*, 115(529), 393–402, doi:10.1080/01621459.2018.1554485
7. Dewey M. (2016), “Meta-Analysis of Significance Values”, *metap-package*
8. Winner L. (2004), Introduction to Biostatistics, Department of Statistics University of Florida
9. Yang J.J., Li J., Williams L.K., Buu A. (2016), “An efficient genome-wide association test for multivariate phenotypes based on the Fisher combination function”, *BMC Bioinformatics*, 17: 19, doi:10.1186/s12859-015-0868-6
10. Dimou N., et. Al. (2017), GVAR: Robust analysis and meta-analysis of GWAS, *Bioinformatics*, <https://doi.org/10.1093/bioinformatics/btx008>
11. <https://github.com/IlyaLab/CombiningDependentPvaluesUsingEBM/tree/master/Data>
12. <https://www.stata.com/manuals13/psyntax.pdf>
13. <https://docs.scipy.org/doc/scipy/tutorial/stats/continuous.html>
14. <https://www.stata.com/manuals13/m-4statistical.pdf>
15. <https://docs.scipy.org/doc/scipy/tutorial/stats.html>
16. <https://www.stata.com/manuals13/rcorrelate.pdf>
17. <https://www.stata.com/manuals/pmatriceigenvalues.pdf>
18. <https://www.stata.com/statalist/archive/2002-06/msg00049.html>

ΠΑΡΑΡΤΗΜΑ Α:ΚΩΔΙΚΑΣ ΣΤΗΝ PYTHON

```
import math
import numpy as np
from scipy.stats import t
from scipy.stats import chi2
from scipy.stats import norm
from scipy.stats import gamma
from scipy.stats import pearsonr
from statsmodels.distributions.empirical_distribution import ECDF
from scipy.special import chdtrc as chi2_cdf
from scipy.stats import moyal
import argparse

def logit(p_values):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all probabilities are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-16, 1 - 1e-16)

    k = len(p_values)
    C = np.sqrt(k * np.pi ** 2 * (5 * k + 2) / (3 * (5 * k + 4)))
    # Compute the logit transform
    t_value = -np.sum(np.log((p_values) / (1 - p_values))) / C
    df = 2 * k
    combined_p = 2 * (1 - t.cdf(np.abs(t_value), df))

    return combined_p

def meanp(p_values):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-09, 1 - 1e-09)
    k = len(p_values)
    # Compute the mean of the p-values
    z_value = (0.5 - np.mean(p_values)) * math.sqrt(12 * k)
    combined_p = 1 - norm.cdf(z_value) # right-side test
    return combined_p

def fisher_method(p_values):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-16, 1 - 1e-16)

    # Compute the combined test statistic using Fisher's method
    chi_squared = -2 * np.sum(np.log(p_values))

    # Calculate the degrees of freedom
    df = 2 * len(p_values)

    # Calculate the combined p-value using the chi-squared
    distribution
```

```

    fisher_p = 1 - chi2.cdf(chi_squared, df)

    return fisher_p

def stouffer(p_values, one_tailed=True):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-16, 1 - 1e-16)
    if one_tailed:
        z_scores = norm.ppf(1 - p_values) # norm.ppf =inverse
cumulative distribution function of normal distribution
    else:
        z_scores = norm.ppf(1 - p_values / 2)

    # z_scores = norm.ppf(1 - p_values) # norm.ppf =inverse
cumulative distribution function of normal distribution
    combined_z = np.sum(z_scores / math.sqrt(len(p_values)))
    combined_p = 1 - norm.cdf(combined_z) # norm.cdf = cumulative
distribution function of normal distribution

    return combined_p

def inverse_chi2(p_values):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-16, 1 - 1e-16)

    df = len(p_values)

    chi_squared = np.sum(chi2.ppf((1 - p_values), 1)) # ppf =
inverse of cdf
    inv_chi2_combined_p = 1 - chi2.cdf(chi_squared, df) # chi2.cdf =
cumulative distribution function of chi2

    return inv_chi2_combined_p

def binomial_test(p_values, alpha=0.05):
    p_values = np.array(p_values)

    k = len(p_values)

    # Count the number of significant p-values
    r = sum((p < alpha) for p in p_values)
    # Calculate the binomial probability of observing at most
num successes successes
    combined_p = 0
    for x in range(r, k + 1):
        combined_p += math.factorial(k) / (math.factorial(x) *
math.factorial(k - x)) * (alpha ** x) * (
(1 - alpha) ** (k - x))

    return combined_p

```

```

def cauchy_cdf(x, x0=0, gamma=1):
    """Cumulative distribution function (CDF) of the Cauchy
    distribution."""

    return 0.5 + (np.arctan((x - x0) / gamma) / np.pi)

def cauchy_method(p_values, alternative='two-sided'):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-15, 1 - 1e-15)

    k = len(p_values)

    T = np.tan((0.5 - p_values) * np.pi)

    t = np.sum(T) / k

    # Calculate the combined p-value using the Cauchy distribution
    combined_p = 1 - cauchy_cdf(t)

    return combined_p

def minP(p_values):
    # Convert input to numpy array for numerical operations
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    p_values = np.clip(p_values, 1e-15, 1 - 1e-15)

    k = len(p_values)

    # Calculate the combined p-value as the minimum p-value
    combined_p = 1 - (1 - np.min(p_values)) ** k

    return combined_p

def CMC(p_values):
    p_value_cauchy = cauchy_method(p_values)
    p_value_minp = minP(p_values)
    combined_p = cauchy_method((p_value_cauchy, p_value_minp)) #
    pCMC = CCT{pCCT, pMinP}
    return combined_p

def MCM(p_values):
    p_value_cauchy = cauchy_method(p_values)
    p_value_minp = minP(p_values)
    combined_p = 2 * min(p_value_cauchy, p_value_minp, 0.5) # pMCM =
2 min{pCCT, pMinP, 0.5}

    return combined_p

def HMP(p_values):
    p_values = np.array(p_values)
    L = len(p_values)

```

```

    harmonic_mean = L / np.sum(1 / p_values)
    combined_p = moyal.cdf(-2 * np.log(harmonic_mean), 1, L)
    return combined_p

def EmpiricalBrownsMethod(data_matrix, extra_info=False):
    covar_matrix = CalculateCovariances(data_matrix)
    return CombinePValuesEBM(covar_matrix, data_matrix, extra_info)

def KostsMethod(data_matrix, extra_info=False):
    covar_matrix, _ = CalculateKostCovarianceEBM(data_matrix)
    return CombinePValuesEBM(covar_matrix, data_matrix, extra_info)

def BrownsMethodbyYang(data_matrix, extra_info=False):
    delta_matrix = CalculateCovarianceYang(data_matrix)
    return CombinePValuesYang(delta_matrix, data_matrix, extra_info)

# Input: raw data vector (of one variable) with no missing samples.
# May be a list or an array.
# Output Transformed data vector w.
def TransformData(data_vector):
    m = np.mean(data_vector)
    sd = np.std(data_vector)
    s = [(d - m) / sd for d in data_vector]
    W = lambda x: -2 * np.log(ECDF(s)(x))
    return np.array([W(x) for x in s])

# Input: An m x n data matrix with each of m rows representing a
# variable and each of n columns representing a sample.
# Should be of type numpy.array.
# Note: Method does not deal with missing values within the data.
# Output: An m x m matrix of pairwise covariances between transformed
# raw data vectors
def CalculateCovariances(data_matrix):
    transformed_data_matrix = np.array([TransformData(f) for f in
data_matrix])
    covar_matrix = np.cov(transformed_data_matrix)

    return covar_matrix

# Input: A m x m numpy array of covariances between transformed data
# vectors and a vector of m p-values to combine.
# Output: A combined P-value.
# If extra_info == True: also returns the p-value from Fisher's
# method, the scale factor c, and the new degrees of freedom from
# Brown's Method
def CombinePValuesEBM(covar_matrix, p_values, extra_info=False):
    m = covar_matrix.shape[0]
    # print "m", m
    df_fisher = 2.0 * m
    Expected = 2.0 * m
    cov_sum = 0
    for i in range(m):
        for j in range(i + 1, m):
            cov_sum += covar_matrix[i, j]

```

```

print("cov sum", cov_sum)
Var = 4.0 * m + 2 * cov_sum
c = Var / (2.0 * Expected)
print(c)
df_brown = 2.0 * Expected ** 2 / Var
if df_brown > df_fisher:
    df_brown = df_fisher
    c = 1.0

x = 2.0 * sum([-np.log(np.clip(p, 1e-15, 1 - 1e-15)) for p in
p_values])
# print "x", x
p_brown = chi2_cdf(df_brown, 1.0 * x / c)
p_fisher = chi2_cdf(df_fisher, 1.0 * x)

if extra_info:
    return p_brown, p_fisher, c, df_brown
else:
    return p_brown

def CombinePValuesYang(delta_matrix, p_values, extra_info=False):
    m = int(delta_matrix.shape[0])
    mean = 2.0 * m
    df_fisher = 2.0 * m
    delta_sum = 0
    for i in range(m):
        for j in range(i + 1, m):
            delta_sum += delta_matrix[i, j]

    Var = 4.0 * m + delta_sum
    v = 2 * (mean ** 2 / Var)
    gamma_value = Var / mean
    T = 2.0 * sum([-np.log(np.clip(p, 1e-15, 1 - 1e-15)) for p in
p_values])
    p_fisher = chi2_cdf(df_fisher, 1.0 * T)
    p_yang = 1 - gamma.cdf(T, v / 2, scale=2 * gamma_value)

    if extra_info:
        return p_yang, p_fisher
    else:
        return p_yang

# Input: An m x n data matrix with each of m rows representing a
variable and each of n columns representing a sample. Should be of
type numpy.array
#       A vector of m P-values to combine. May be a list or of type
numpy.array.
# Output: A combined P-value using Kost's Method.
#       If extra_info == True: also returns the p-value from
Fisher's method, the scale factor c, and the new degrees of freedom
from Brown's Method

def KostiMethodYang(data_matrix):
    delta_matrix = CalculateCovarianceYang(data_matrix)
    return CombinePValuesYang(delta_matrix, data_matrix)

# Input correlation between two n x n data vectors.
# Output: Kost's approximation of the covariance between the -log

```

```

cumulative
# distributions. This is calculated with a cubic polynomial fit.
def KostPolyFitEBM(cor):
    a1, a2, a3 = 3.263, 0.710, .027 # Kost cubic coefficients
    return a1 * cor + a2 * cor ** 2 + a3 * cor ** 3

def PolyFitYang(cor):
    c1, c2, c3, c4, c5 = 3.9081, 0.0313, 0.1022, -0.1378, 0.0941
    return c1 * cor ** 2 + c2 * cor ** 4 + c3 * cor ** 6 + c4 * cor
** 8 + c5 * cor ** 10

# Input: An m x n data matrix with each of m rows representing a
variable and each of n columns representing a sample. Should be of
type numpy.array.
# Note: Method does not deal with missing values within the data.
# Output: An m x m matrix of pairwise covariances between the data
vectors calculated using Kost's polynomial fit and numpy's pearson
correlation function.
def CalculateKostCovarianceEBM(data_matrix):
    n = data_matrix.shape[1] # Number of columns
    covar_matrix = np.zeros((n, n))
    max_cor = -np.inf
    for i in range(n):
        for j in range(i + 1, n):
            cor, p_val = pearsonr(data_matrix[:, i], data_matrix[:,
j]) # Correlate columns
            covar = KostPolyFitEBM(cor)
            covar_matrix[i, j] = covar
            covar_matrix[j, i] = covar

            max_cor = max(max_cor, cor)

    return covar_matrix, max_cor

def CalculateCovarianceYang(data_matrix):
    c1 = 3.9081
    n = data_matrix.shape[1] # Number of columns
    delta_matrix = np.zeros((n, n))
    corr_matrix = np.zeros((n, n))
    max_cor = -np.inf
    for i in range(n):
        for j in range(i + 1, n):
            cor, p_val = pearsonr(data_matrix[:, i], data_matrix[:,
j]) # Correlate columns
            corr_matrix[i, j] = cor
            biased_corrected_cor = cor * (1 + (1 - cor ** 2 / 2 * (n
- 3)))
            # print(cor)

            f_r = PolyFitYang(biased_corrected_cor)
            bias = (c1 / n) * (1 - biased_corrected_cor ** 2) ** 2

            unbiased_delta = f_r - bias
            delta_matrix[i, j] = unbiased_delta
            delta_matrix[j, i] = unbiased_delta

    return delta_matrix

```

```

def Bonferronis_correction_g(p_values):
    g = len(p_values)
    _, ICC = CalculateKostCovarianceEBM(p_values)
    print(ICC)
    g_adjusted = (g + 1) - (1 + (g - 1) * ICC)

    return g_adjusted

def effective_number_of_tests_cheverud_nyholt(eigenvalues):
    k = len(eigenvalues)
    sample_variance = np.var(eigenvalues)
    print(sample_variance)
    effective_number_of_tests = 1 + (k - 1) * (1 - sample_variance /
k)
    return effective_number_of_tests

def h_function(x):
    if x >= 1:
        h = 1 + (x - math.floor(x))
    else:
        h = x - math.floor(x)
    return h

def effective_number_of_tests_li_ji(eigenvalues):
    abs_eigenvalues = np.abs(eigenvalues)
    effective_number_of_tests = np.sum([h_function(abs_lambda) for
abs_lambda in abs_eigenvalues])

    return effective_number_of_tests

def effective_number_of_tests_gao(eigenvalues, C=0.995):
    k = len(eigenvalues)
    total_sum = np.sum(eigenvalues)

    for x in range(k):
        x_sum = np.sum(eigenvalues[:x + 1]) # Sum of eigenvalues up
to index x
        if x_sum / total_sum > C:
            effective_number_of_tests = x + 1 # Add 1 to convert
from index to count
            break

    return effective_number_of_tests

def effective_number_of_tests_galwey(eigenvalues):
    k = len(eigenvalues)
    lambda_prime = np.maximum(0, eigenvalues)

    # Step 3: Compute sum of squares of  $\lambda'_i$ 
    sum_squared_lambda_prime = (np.sum(np.sqrt(lambda_prime))) ** 2

    # Step 4: Divide by the sum of all  $\lambda'_i$ 
    effective_number_of_tests = sum_squared_lambda_prime /
np.sum(lambda_prime)

```

```

    return effective_number_of_tests

def bonferroni_method_with_effective_tests(p_values,
effective_number_of_tests):
    p_values = np.array(p_values)

    # Ensure that all p-values are within the valid range (0, 1)
    min_p_value = min(p_values)
    combined_p = min(1, min_p_value * effective_number_of_tests)

    return combined_p

def default():
    print("Invalid choice")

def transform_to_pvalues(z_scores):
    # Transform z-scores to two-sided p-values
    p_values = [2 * (1 - norm.cdf(abs(z))) for z in z_scores]
    return p_values

def read_from_file_transform(file_path, data_type):
    SNP_values = []

    with open(file_path, 'r') as file:
        content = file.readlines()

    for line in content:
        elements = line.strip().split()
        SNP, values = elements[0], list(map(float, elements[1:]))
        if data_type == 'z-scores':
            values = transform_to_pvalues(values)
        SNP_values.append((SNP, values))

    return SNP_values

def read_matrix_from_file(file_path):
    matrix = []
    with open(file_path, 'r') as file:
        for line in file:
            values = line.strip().split()
            row = [float(value) for value in values]
            matrix.append(row)
    return matrix

def combine_pvalues(combine_function, SNP_values, metanalysis_needed,
file):
    p_values = []

    if metanalysis_needed:
        SNP_dict = {}
        for SNP, values in SNP_values:
            SNP_dict.setdefault(SNP, []).append(values)

```

```

        for SNP, values in SNP_dict.items():
            if len(values) > 1:
                grouped_values = list(zip(*values))
                combined_results =
[combine_function(np.array(grouped_value)) for grouped_value in
grouped_values]
                SNP_dict[SNP] = np.float_(combined_results)
            else:
                # Directly use the single set of values
                SNP_dict[SNP] = np.float_(values[0])

        p_values.append({'SNP': SNP, 'values': SNP_dict[SNP]})

    for p_value in p_values:
        print(f"Combining p-values for SNP {p_value['SNP']}:
{p_value['values']}") # Debug print
        comb_p = combine_function(p_value['values'])
        file.write(f"Combined p-values for SNP {p_value['SNP']}:
{comb_p}\n")

    else:
        for SNP, values in SNP_values:
            values = np.float_(values) # Ensure values are converted
to float
            p_values.append({'SNP': SNP, 'values': values})

        for p_value in p_values:
            print(f"Combining p-values for SNP {p_value['SNP']}:
{p_value['values']}") # Debug print
            comb_p = combine_function(np.array(p_value['values']))
            file.write(f"Combined p-values for SNP {p_value['SNP']}:
{comb_p}\n")

    return p_values

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Process text files
and data type.')
    parser.add_argument('file_path', type=str, help='Path to the
input file')
    parser.add_argument('data_type', type=str, choices=['p-values',
'z-scores'], help="Type of data in the file ('p-values' or 'z-
scores')")
    parser.add_argument('meta_choice', type=str, choices=['Yes',
'No'], help="Do you want the program to perform meta-analysis? ('Yes'
or 'No')")
    parser.add_argument('correlation_matrix', type=str,
choices=['Yes', 'No'], help="Do you have a text file with the
correlation matrix of the SNPs? ('Yes' or 'No')")
    args = parser.parse_args()

    SNP_values = read_from_file_transform(args.file_path,
args.data_type.lower())
    metanalysis_needed = args.meta_choice.lower() == 'yes'
    cor_mat_needed = args.correlation_matrix.lower() == 'yes'

    method_functions = {
        '1': logit,
        '2': meanp,
        '3': fisher_method,

```

```

'4': stouffer,
'5': inverse_chi2,
'6': binomial_test,
'7': cauchy_method,
'8': minP,
'9': CMC,
'10': MCM,
'11': HMP,
'12': EmpiricalBrownsMethod,
'13': KostsMethod,
'14': BrownsMethodbyYang,
'15': bonferroni_method_with_effective_tests,
}

method_names = {
    '1': "Logitp Method",
    '2': "Meanp Method",
    '3': "Fisher's Method",
    '4': "Stouffer's Method",
    '5': "Inverse Chi2 Method",
    '6': "Binomial Test",
    '7': "Cauchy Method (CCT)",
    '8': "MinP Method (Tippett's Method)",
    '9': "CMC (CCT-MinP-CCT)",
    '10': "MCM (MinP-CCT-MinP)",
    '11': "Harmonic Mean P-value (HMP)",
    '12': "Empirical Brown's Method (EBM)",
    '13': "Empirical Brown's Method (EBM) by Kost",
    '14': "Brown's Method by Yang",
    '15': "Bonferroni with effective number of tests",
    '16': "Bonferroni's correction"
}

print("\n***** Combining methods for dependent and independent
P-values *****\n")
for key, value in method_names.items():
    print(f"{key}: *** {value} ***")

methods = input("\nEnter your choices separated by commas (e.g.,
'1,2,3'): ").split(',')

with open("output.txt", "w") as file:
    for method in methods:
        if method in method_functions:
            file.write(f"\nSelected Method: ***
{method_names[method]} ***\n")
            combine_function = method_functions[method]

            if method in ['12', '13', '14', '15']:
                # Extracting values from the tuples
                values = np.array([values for _, values in
SNP_values])

                SNP = np.array([values for SNP, _ in SNP_values])
                g_adjusted = Bonferronis_correction_g(values)

                if method in ['12', '13', '14']:
                    # Get combined values for each SNP
                    combined_values =
file.write(str(combine_function(values, extra_info=False)))

            print(g_adjusted)

```

```

elif method in ['15']:
    if cor_mat_needed:
        matrix_path = input("\nEnter the path to
the .txt file with the correlation matrix for your data(e.g
filename.txt): ")
        corr_matrix =
read_matrix_from_file(matrix_path)
        cleaned_corr_matrix = [row for row in
corr_matrix if row]

        eigenvalues, _ =
np.linalg.eig(cleaned_corr_matrix)
        sorted_eigenvalues =
np.sort(eigenvalues)[::-1]

        cn =
effective_number_of_tests_cheverud_nyholt(sorted_eigenvalues)
        gao =
effective_number_of_tests_gao(sorted_eigenvalues)
        gal =
effective_number_of_tests_galwey(sorted_eigenvalues)
        li_ji =
effective_number_of_tests_li_ji(sorted_eigenvalues)

        file.write(f"Effective Number of Tests
(Cheverud-Nyholt): {cn}\n")

        for SNP, values in SNP_values:
            combined_p_value_cn =
combine_function(values, cn)
            file.write(f"Combined p-value for
{SNP}: {combined_p_value_cn}\n")

        file.write(f"Effective Number of Tests
(Gao): {gao}\n")

        for SNP, values in SNP_values:
            combined_p_value_gao =
combine_function(values, gao)
            file.write(f"Combined p-value for
{SNP}: {combined_p_value_gao}\n")

        file.write(f"Effective Number of Tests
(Galwey): {gal}\n")

        for SNP, values in SNP_values:
            combined_p_value_gal =
combine_function(values, gal)
            file.write(f"Combined p-value for
{SNP}: {combined_p_value_gal}\n")

        file.write(f"Effective Number of Tests
(Li-Ji): {li_ji}\n")

        for SNP, values in SNP_values:
            combined_p_value_li_ji =
combine_function(values, li_ji)
            file.write(f"Combined p-value for
{SNP}: {combined_p_value_li_ji}\n")

        file.write(f"Effective Number of Tests

```

```

Bonferroni's Correction(from Pharmaceutical Statistics):
{g_adjusted}\n")
        for SNP, values in SNP_values:
            combined_p_value_bc =
combine_function(values, g_adjusted)
            file.write(f"Combined p-value for
{SNP}: {combined_p_value_bc}\n")

        else:
            p_values = combine_pvalues(combine_function,
SNP_values, metanalysis_needed, file)
        else:
            file.write(f"Invalid method choice: {method}\n")

```

ΠΑΡΑΡΤΗΜΑ Β: ΚΩΔΙΚΑΣ ΣΤΟ STATA

```
capture program drop metacp
program define metacp
    version 13.0
    syntax varlist(numeric),method(string) [ corr_matrix(varlist) ]
    tempvar num_vars t_sum z_mean z_value chi_squared_sum z_sum invchi_squared_sum T_sum t min_val C_sum c h C
    egen `num_vars' = rownonmiss(`varlist')

    if "`method'" == "logitp" {

        * Define the numerical value of pi
        local pi_value = 3.141592653589793

        * Calculate the constant C
        local C = sqrt(`num_vars' * `pi_value'^2 * (5 * `num_vars' + 2) / (3 * (5 * `num_vars' + 4)))

        * Loop through each variable
        foreach var of varlist `varlist' {
            * Calculate the t-value for each observation
            gen t_value_`var' = ln(`var' / (1 - `var'))
        }

        * Sum the t-values for each observation
        egen `t_sum' = rowtotal(t_value_*)

        * Calculate the combined p-value
        gen _logitp = 2 * (1 - t(2 * `num_vars', abs(-`t_sum' / `C'))))

        * Display the combined p-value
        di "Combined p-value: " _logitp
    }

    else if "`method'" == "meanp" {

        * Define the numerical value of pi
        local pi_value = 3.141592653589793

        egen `z_mean' = rowmean(`varlist')
        gen `z_value' = (0.5 - `z_mean') * sqrt(12 * `num_vars')
```

```

    * Calculate combined p-value
    gen _meanp = 1 - normal(`z_value')
    di "Combined p-value:", _meanp
}
else if "`method'" == "fisherp" {

    foreach var of varlist `varlist' {
        * Calculate the t-value for each observation
        gen log_`var' = ln(`var')
    }

    egen `chi_squared_sum' = rowtotal(log_*)

    * Calculate combined p-value
    gen _fisher_p = 1 - chi2(2 * `num_vars', -2 * `chi_squared_sum')
    di "Combined p-value:", _fisher_p
}
else if "`method'" == "stoufferp" {
    * Define the numerical value of pi
    local pi_value = 3.141592653589793
    foreach var of varlist `varlist' {
        * Calculate the t-value for each observation
        gen z_score_`var' = invnorm(1 - `var')
    }

    egen `z_sum' = rowtotal(z_score_*)

    * Calculate combined p-value
    gen _stouffer_p = 1 - normal(`z_sum' / sqrt(`num_vars'))
    di "Combined p-value:", _stouffer_p
}
else if "`method'" == "invchi2p" {
    * Loop through each variable in the varlist
    foreach var of varlist `varlist' {
        * Calculate the inverse chi-squared statistic for each observation
        gen invchi_squared_`var' = invchi2(1, 1 - `var')
    }
}

```

```

    * Display combined p-value
    di "Combined p-value:", _inv_chi2_p
}

else if "`method'" == "binomialp" {

    local alpha = 0.05
    local a = 0.5
    tempvar num_vars num_of_successes

    local `num_of_successes' = 0

    * Loop through each variable in the varlist
    foreach var of varlist `varlist' {
        if `var' < `alpha' {
            * Increment count of successes
            local `num_of_successes' = `num_of_successes' + 1
        }
    }

    * Initialize combined p-value
    local pmf = 0

    * Calculate combined p-value
    forvalues x = `num_of_successes' / `num_vars' {
        * Calculate probability mass function for each possible success count
        local pmf = `pmf' + binomial(`x', `num_vars', `a')
    }

    * Generate combined p-value
    gen _binomial_p = `pmf'

    * Display combined p-value
    di "Combined p-value:", _binomial_p
}

else if "`method'" == "cauchy" {

    local pi_val = 3.141592653589793

```

```

local gamma_val = 1

* Loop through each variable in the varlist
foreach var of varlist `varlist' {
    * Calculate the inverse chi-squared statistic for each observation
    gen T_`var' = tan((0.5 - `var') * `pi_val')
}

* Calculate the sum of inverse chi-squared statistics across variables
egen `T_sum' = rowtotal(T_*)

gen `t' = `T_sum' / `num_vars'
* Calculate combined p-value calculated as 1 - cauchy.cdf(t)
gen _cauchy_p = 1 - (0.5 + atan(`t' / `gamma_val') / `pi_val')

* Display combined p-value
di "Combined p-value:", _cauchy_p
}

else if "`method'" == "minp" {

    * Find the minimum value across variables in each observation
    egen `min_val' = rowmin(`varlist')

    * Calculate combined p-value
    gen _min_p = 1 - (1 - `min_val') ^ `num_vars'

    * Display combined p-value
    di "Combined p-value:", _min_p
}

else if "`method'" == "MCM" {

    gen _MCM_p = 2 * min(_cauchy_p, _min_p, 0.5)
    * Display combined p-value
    di "Combined p-value by MCM(MinP-Cauchy-MinP):", _MCM_p
}

```

```

*For the CMC method the user should first calculate the '_cauchy_p' and the '_min_p'
*The CMC method should be run like this 'metacp __cauchy_p _min_p,method(CMC)'
else if "`method'" == "CMC" {
    local pi_val = 3.141592653589793
    local gamma_val = 1

    * Loop through each variable in the varlist
    foreach var of varlist `varlist' {
        gen C_`var' = tan((0.5 - `var') * `pi_val')
    }

    * Calculate the sum of inverse chi-squared statistics across variables
    egen `C_sum' = rowtotal(C_*)

    * Calculate combined p-value
    gen `c' = `C_sum' / `num_vars'

    * Calculate combined p-value by CMC method (Cauchy-MinP-Cauchy)
    gen _CMC_p = 1 - (0.5 + atan(`c' / `gamma_val') / `pi_val')

    * Display combined p-value
    di "Combined p-value by CMC(Cauchy-MinP-Cauchy):", _CMC_p
}

else if "`method'" == "bonferronip" {
    egen `min_val' = rowmin(`varlist')

    * Calculate combined p-value
    gen _bonferroni_p = min(1, `min_val' * `num_vars')

    * Display combined p-value
    di "Combined p-value:", _bonferroni_p
}

else if "`method'" == "cheverud_nyholtp" {
    local end_mata end
    mkmat `corr_matrix', matrix(correlation_matrix)
    matrix eigenvalues lambda V = correlation_matrix
    mat list lambda
    mata:
        lambda = st_matrix("lambda")

        st_matrix("variance", diagonal(variance(lambda)))

```

```

    `end_mata'
    mat eig_variance = variance

    * Extract the value of the standard deviation
    scalar sample_variance = eig_variance[1,1]
    di "Sample variance of Eigenvalues: " sample_variance
    gen effective_number_of_tests = 1 + (`num_vars' - 1) * (1 - sample_variance / `num_vars')
    egen min_val = rowmin(`varlist')
    gen _cheverud_nyholt_p = min(1, min_val * effective_number_of_tests)
    di "CN: " _cheverud_nyholt_p
}

else if "`method'" == "li_jip" {
    mkmat `corr_matrix', matrix(correlation_matrix)
    matrix eigenvalues lambda V = correlation_matrix
    mat list lambda
    mata:
        lambda = st_matrix("lambda")
        abs_eig = abs(lambda)
        st_matrix("abs_eig", abs_eig)
    `end_mata'
    mat list abs_eig
    scalar effective_number_of_tests = 0
    local rows = rowsof(abs_eig)
    local cols = colsof(abs_eig)
    forval i = 1/`rows' {
        local h = 0
        forval j = 1/`cols' {
            if abs_eig[`i', `j'] >= 1 {
                local h = 1 + (abs_eig[`i', `j'] - floor(abs_eig[`i', `j'])))
            }
            else {
                local h = abs_eig[`i', `j'] - floor(abs_eig[`i', `j'])
            }
            scalar effective_number_of_tests = effective_number_of_tests + `h'
        }
    }
    di "Effective number of tests: " effective_number_of_tests
    egen min_val = rowmin(`varlist')
    gen _li_ji_p = min(1, min_val * effective_number_of_tests)
    di "Li_Ji: " _li_ji_p
}

```

```

}

else if "`method'" == "gaop" {
    mkmat `corr_matrix', matrix(correlation_matrix)
    matrix eigenvalues lambda V = correlation_matrix
    mat list lambda
    mata:
        lambda = st_matrix("lambda")
        sum_eig = rowsum(lambda)
        st_matrix("sum_eig", sum_eig)
    `end_mata'
    mat list sum_eig
    scalar total_sum = sum_eig[1,1]
    local rows = rowsof(lambda)
    local cols = colsof(lambda)
    local C = 0.995
    scalar effective_number_of_tests = 0
    local x_sum = 0
    forval x = 1/`cols' {
        local x_sum = `x_sum' + lambda[1, `x']
        if (`x_sum' / total_sum) > `C' {
            scalar effective_number_of_tests = `x'
        }
    }
    di "Effective number of tests: " effective_number_of_tests
    egen min_val = rowmin(`varlist')
    gen _gao_p = min(1, min_val * effective_number_of_tests)
    di "GAO: " _gao_p
}

else if "`method'" == "galweyp" {
    mkmat `corr_matrix', matrix(correlation_matrix)
    matrix eigenvalues lambda V = correlation_matrix
    mat list lambda
    mata:
        lambda = st_matrix("lambda")
        lambda_prime = J(rows(lambda), cols(lambda), 0)
        for (i=1; i<=rows(lambda); i++) {
            for (j=1; j<=cols(lambda); j++) {
                if (lambda[i, j] > 0) {
                    lambda_prime[i, j] = lambda[i, j]
                }
            }
        }
    `end_mata'
}

```

```

    }
    st_matrix("lambda_prime", lambda_prime)
    sum_lambda_prime = rowsum(lambda_prime)
    st_matrix("sum_lambda_prime", sum_lambda_prime)
    sum_eig = rowsum(sqrt(lambda_prime))^2
    st_matrix("sum_eig", sum_eig)
`end_mata'
mat list lambda_prime
mat list sum_lambda_prime
mat list sum_eig
scalar sum_squared_lambda_prime = sum_eig[1,1]
scalar sum_lambda_prime = sum_lambda_prime[1,1]
scalar effective_number_of_tests = sum_squared_lambda_prime / sum_lambda_prime
di "Effective number of tests: " effective_number_of_tests
egen min_val = rowmin(`varlist')
gen _galwey_p = min(1, min_val * effective_number_of_tests)
di "Galwey: " _galwey_p
}
else if "`method'" == "EBM" {
    corr `varlist', cov
    matrix covar_matrix = r(C)
    local m = rowsof(covar_matrix)

    *matrix accum R = var2-varl2, nocons dev
    *matrix R = corr(R)
    *mat list R

    scalar df_fisher = 2 * `m'
    scalar Expected = 2 * `m'

    local cov_sum = 0

    forval r = 1/`m' {
        forval c = `r'/'m' {
            local cov_sum = `cov_sum' + covar_matrix[`r', `c']
        }
    }
    di `cov_sum'
    scalar Var = 4 * `m' + 2 * `cov_sum'
    local c = Var / (2 * Expected)
    scalar df_brown = 2 * Expected^2 / Var
    di `c'
}

```

```

    if df_brown > df_fisher {
        scalar df_brown = df_fisher
        local c = 1
    }

    foreach var of varlist `varlist' {
        * Calculate the t-value for each observation
        gen _blog_`var' = ln(`var')
    }

    egen `chi_squared_sum' = rowtotal(_blog_*)

    gen _bfisher_p = 1 - chi2(df_fisher, -2 * `chi_squared_sum')
    di "Fisher p-value:", _bfisher_p

    gen chi_squared_sum_brown = `chi_squared_sum' / `c'
    gen _p_brown = 1 - chi2(df_brown, -2 * chi_squared_sum_brown)
    di "Brown p-value:", _p_brown
}

else if "`method'" == "KostsMethodEBM" {
    scalar a1 = 3.263
    scalar a2 = 0.710
    scalar a3 = 0.027
    corr `varlist'
    matrix corr_matrix = r(C)
    local m = rowsof(corr_matrix)

    *matrix accum R = _all, nocons dev
    *matrix R = corr(R)
    *mat list R

    local nrows = rowsof(corr_matrix)
    local ncols = colsof(corr_matrix)

    * Initialize covar_matrix with missing values
    matrix covar_matrix = J(`nrows', `ncols', .)

```

```

* Loop to compute covariance
forval i = 1/`nrows' {
    forval j = 1/`ncols' {
        local corr_ij = corr_matrix[`i', `j']

        * Calculate covariance (assuming Kost's polynomial fit)
        scalar covar = a1 * `corr_ij' + a2 * (`corr_ij'^ 2) + a3 * (`corr_ij'^ 3)

        * Assign covariance values to the matrix
        matrix covar_matrix[`i', `j'] = covar
        matrix covar_matrix[`j', `i'] = covar
    }
}

mat list covar_matrix
local m = rowsof(covar_matrix)
scalar df_fisher = 2 * `m'
scalar Expected = 2 * `m'

local cov_sum = 0

forval r = 1/`m' {
    forval c = `r'/`m' {
        local cov_sum = `cov_sum' + covar_matrix[`r', `c']
    }
}
di `cov_sum'
scalar Var = 4 * `m' + 2 * `cov_sum'
local c = Var / (2 * Expected)
scalar df_brown = 2 * Expected^2 / Var
di `c'
if df_brown > df_fisher {
    scalar df_brown = df_fisher
    local c = 1
}

foreach var of varlist `varlist'{
    * Calculate the t-value for each observation
    gen _klog_`var' = ln(`var')
}

egen `chi_squared_sum' = rowtotal(_klog_*)

```

```

* Calculate combined p-value
gen _kfisher_p = 1 - chi2(df_fisher, -2 * `chi_squared_sum')
di "Fisher p-value:", _kfisher_p

gen chi_squared_sum_kost_brown = `chi_squared_sum' / `c'
gen _p_kost_brown = 1 - chi2(df_brown, -2 * chi_squared_sum_kost_brown)
di "Brown p-value:", _p_kost_brown
}
else if "`combining_method'" == "BrownbyYang" {
  scalar c1 = 3.9081
  scalar c2 = 0.0313
  scalar c3 = 0.1022
  scalar c4 = -0.1378
  scalar c5 = 0.0941

  corr `varlist'
  matrix corr_matrix = r(C)
  local m = rowsof(corr_matrix)

  *matrix accum R = _all, nocons dev
  *matrix R = corr(R)
  *mat list R

  local nrows = rowsof(corr_matrix)
  local ncols = colsof(corr_matrix)

  * Initialize covar_matrix with missing values
  matrix delta_matrix = J(`nrows', `ncols', .)

  * Loop to compute covariance
  forval i = 1/`nrows' {
    forval j = 1/`ncols' {
      local corr_ij = corr_matrix[`i', `j']
      local biased_corrected_cor = `corr_ij' * (1 + (1 - `corr_ij'^2 / 2 * (`ncols' - 3)))

      scalar f_r = c1 * (`biased_corrected_cor' ^ 2) + c2 * (`biased_corrected_cor' ^ 4) + c3 * (`biased_corrected_cor' ^ 6 )
      + c4 * (`biased_corrected_cor' ^ 8) + c5 * (`biased_corrected_cor' ^ 10)

      local bias = (c1 / `ncols') * (1 - `biased_corrected_cor'^2)^2
    }
  }
}

```

```

        scalar unbiased_delta = f_r - `bias'
        matrix delta_matrix[`i', `j'] = unbiased_delta
        matrix delta_matrix[`j', `i'] = unbiased_delta
    }
}

mat list delta_matrix
local m = rowsof(delta_matrix)
scalar df_fisher = 2 * `m'
scalar Expected = 2 * `m'

scalar mean = 2 * `m'

local delta_sum = 0

forval r = 1/`m' {
    forval c = `r'/`m' {
        local delta_sum = `delta_sum' + delta_matrix[`r', `c']
    }
}
di `delta_sum'
scalar Var = 4 * `m' + 2 * `delta_sum'
local v = 2*(mean^2 / Var)
scalar gamma_value = Var / mean
scalar scale = `v' / 2
di `v'

foreach var of varlist `varlist'{
    * Calculate the t-value for each observation
    gen _ylog_`var' = ln(`var')
}

egen `chi_squared_sum' = rowtotal(_ylog_*)

* Calculate combined p-value
gen _yfisher_p = 1 - chi2(df_fisher, -2 * `chi_squared_sum')
di "Fisher p-value:", _yfisher_p

gen _p_yang = 1 - gammaden(-2 * `chi_squared_sum', scale, 2 * gamma_value)
di "Brown p-value by Yang:", _p_yang
}

```