



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

**Διπλωματική εργασία**

**Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για  
την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή,  
Στρωματοποιημένη και Μπλοκ)**

Θεόδωρος Μπρότσης



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

## Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και Μπλοκ)

Θεόδωρος Μπρότσης

Axel Kowald

Humboldt Universitaet zu

Berlin

Ηλίας Ζιντζαράς

Πανεπιστήμιο Θεσσαλίας

Γεώργιος Ραχιώτης

Πανεπιστήμιο Θεσσαλίας

**Περίληψη:** Η παρούσα Διπλωματική Εργασία στοχεύει στην ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης όπως είναι η απλή τυχαιοποίηση, η στρωματοποιημένη τυχαιοποίηση και η τετραγωνισμένη τυχαιοποίηση (block randomization, permuted blocks).

**Λέξεις-κλειδιά:** τυχαιοποίηση, απλή τυχαιοποίηση, τυχαιοποίηση κατά μπλοκ, στρωματοποιημένη τυχαιοποίηση, θεραπευτικές αγωγές.

**Περιεχόμενο:** Κείμενο, εικόνες, πρόγραμμα σε γλώσσα Python, πρωτότυπο εφαρμογής.



## Περιεχόμενα

1. Περίληψη .....	5
2. Εισαγωγή .....	5
2.1. Στόχοι .....	5
2.2. Μέθοδοι .....	6
2.3. Συμπέρασμα .....	6
3. Μέθοδοι .....	6
3.1. Απλή τυχαιοποίηση (Simple randomization) .....	6
3.2. Τετραγωνισμένη τυχαιοποίηση (Permuted Blocks) .....	8
3.3. Στρωματοποιημένη τυχαιοποίηση (Stratified randomization) .....	8
4. Λογισμικό .....	9
4.1. Προδιαγραφές .....	9
4.2. Κλάσεις .....	10
4.3. Παρουσίαση εφαρμογής .....	11
5. Αποτελέσματα .....	14
6. Συμπέρασμα .....	18
7. Αναφορές .....	19
8. Appendix – Κώδικας Python .....	20
8.1. Κλάση guiRandomization .....	20
8.2. Κλάση randomization .....	26
8.3. Κλάση simpleRandomization .....	28
8.4. Κλάση permutedBlocks .....	30
8.5. Κλάση stratifiedRandomization .....	35



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

8.6.	Κλάση blockRandomization .....	39
8.7.	Module setup .....	43



## 1. Περίληψη

Η παρούσα Διπλωματική Εργασία στοχεύει στην ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python [5] για την εκτέλεση διαφόρων τύπων τυχαιοποίησης όπως είναι η απλή τυχαιοποίηση, η στρωματοποιημένη τυχαιοποίηση και η τυχαιοποίηση ανά μπλοκ (block randomization, permuted blocks).

## 2. Εισαγωγή

Με τον όρο τυχαιοποίηση (Randomization – random allocation) εννοούμε την διαδικασία κατά την οποία γίνεται τυχαία ανάθεση θεραπευτικών αγωγών (εμβόλια, νέες θεραπείες) σε ασθενείς. Η τυχαιοποίηση κατανέμει τους ασθενείς σε διαφορετικές ομάδες μίας μελέτης με παρόμοιο και δίκαιο τρόπο. Έτσι συγκρίνοντας παρόμοιες ομάδες ασθενών, οι ερευνητές είναι βέβαιοι ότι η μελέτη τους ελέγχει την διαφορά μεταξύ των θεραπευτικών αγωγών και όχι την διαφορά μεταξύ των ασθενών που συμμετέχουν στην μελέτη. Τα βασικά πλεονεκτήματα της τυχαιοποίησης είναι η εξάλειψη της μεροληψίας<sup>1</sup> της επιλογής (selection bias) των ασθενών. Δεν εγγυάται όμως ότι τα άτομα σε κάθε ομάδα θα έχουν για παράδειγμα την ίδια ηλικιακή κατανομή.

Σε γενικές γραμμές μία τυχαιοποιημένη κλινική δοκιμή αποτελεί ένα σημαντικό εργαλείο για τον έλεγχο της αποτελεσματικότητας μία θεραπείας.

### 2.1. Στόχοι

Η υλοποίηση της συγκεκριμένης διπλωματικής εργασίας περιλαμβάνει τους ακόλουθους βασικούς στόχους:

---

<sup>1</sup> Μεροληψία είναι η προκατάληψη η οποία μπορεί να οδηγήσει σε λάθος συμπεράσματα για τις επιπτώσεις μιας θεραπείας.



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

- Πραγματοποίηση εκτεταμένης μελέτης της σχετικής βιβλιογραφίας όπου θα διερευνηθούν και θα παρουσιαστούν οι μέθοδοι τυχαιοποίησης.
- Σχεδιασμός και υλοποίηση αλγορίθμων για την εκτέλεση των διαφόρων τύπων τυχαιοποίησης.
- Ανάπτυξη ενός πρωτοτύπου για την εφαρμογή.

Στην διπλωματική εργασία θα υλοποιηθούν διάφοροι τύποι τυχαιοποίησης όπως είναι η απλή τυχαιοποίηση (simple randomization), η στρωματοποιημένη τυχαιοποίηση (stratified randomization) και η τυχαιοποίηση κατά μπλοκ (block randomization, permuted blocks) σε περιβάλλον προγραμματισμού Python.

## 2.2. Μέθοδοι

Στην παρούσα διπλωματική εργασία μελετήθηκαν οι εξής μέθοδοι τυχαιοποίησης: απλή τυχαιοποίηση (simple randomization), στρωματοποιημένη τυχαιοποίηση (stratified randomization), τυχαιοποίηση κατά μπλοκ (block randomization, random permuted blocks).

## 2.3. Συμπέρασμα

Σ’ αυτή τη διπλωματική εργασία υλοποιήθηκε μία εφαρμογή για την εκτέλεση διαφόρων τύπων τυχαιοποίησης όπως είναι η απλή τυχαιοποίηση, η στρωματοποίηση κατά μπλοκ (block randomization, permuted blocks) και η στρωματοποιημένη τυχαιοποίηση.

## 3. Μέθοδοι

### 3.1. Απλή τυχαιοποίηση (Simple randomization)

Η τυχαιοποίηση η οποία βασίζεται σε μία απλή ακολουθία από τυχαίες αναθέσεις είναι γνωστή ως **απλή τυχαιοποίηση**. Η πιο απλή και κοινή μέθοδος της **απλής τυχαιοποίησης**

---

*Πανεπιστήμιο Θεσσαλίας: Διπλωματική Εργασία*



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

είναι η ρήψη ενός νομίσματος. Έτσι αν έχουμε για παράδειγμα δύο θεραπευτικές ομάδες (control vs. treatment) η πλευρά του νομίσματος (π. χ. κορώνα – control, γράμματα – treatment) καθορίζει την ανάθεση του κάθε ατόμου. Πλέον όμως χρησιμοποιούνται πίνακες τυχαίων αριθμών (Σχήμα 1) οι οποίοι κατασκευάζονται έτσι, ώστε οι αριθμοί να έχουν την ίδια πιθανότητα εμφάνισης ή προγράμματα ηλεκτρονικών υπολογιστών.

Ένα παράδειγμα υλοποίησης είναι το εξής: εφαρμόζουμε την θεραπευτική αγωγή Α για τα ψηφία 0 – 4 ενώ την θεραπευτική αγωγή Β τα ψηφία 5 – 9. Η εφαρμόζουμε την θεραπευτική αγωγή Α για μονό ψηφίο και την θεραπευτική αγωγή Β για ζυγό ψηφίο. Με τον τρόπο αυτό υπάρχει όμως η πιθανότητα να έχουμε άνισο αριθμό ασθενών. Η ανισότητα αυτή τείνει να μην υπάρχει όταν ο αριθμός των ασθενών είναι μεγάλος.

0 5	2 7	8 4	3 7	4 1	6 8	3 8	5 1	5 6	9 6	8 1	8 0	4 7	8 8	7 4
5 9	7 2	4 0	2 3	6 3	1 8	5 0	2 6	0 9	9 6	9 2	1 8	8 5	0 3	7 9
2 5	9 8	4 3	8 9	5 2	8 4	6 4	4 2	7 5	4 4	9 2	8 1	8 6	9 3	2 2
3 0	3 6	6 5	1 3	1 8	7 7	6 0	0 9	4 3	6 8	0 5	9 5	1 7	5 2	4 2
6 8	1 5	6 7	7 5	7 5	3 4	8 8	0 8	8 8	6 5	2 1	2 8	1 2	8 9	5 5
7 3	6 5	8 3	7 8	1 0	7 9	7 5	5 9	9 9	7 3	8 9	8 3	8 4	5 8	3 2
7 0	1 0	0 7	5 4	1 3	0 1	9 6	6 9	9 5	5 3	2 5	6 4	4 6	1 6	5 6
2 8	1 5	3 3	1 2	8 3	9 0	2 0	2 5	1 5	5 7	0 0	0 4	4 7	4 7	4 4
1 5	6 6	7 2	9 3	0 9	2 4	6 1	1 7	9 1	4 0	6 0	2 5	1 5	2 8	3 3
0 0	2 1	8 2	3 2	0 0	6 8	3 6	5 9	7 3	4 5	3 4	5 0	0 5	6 6	9 6
9 9	6 7	3 8	3 2	5 4	8 3	2 5	3 3	9 5	6 3	5 2	2 1	4 9	5 0	3 4
3 6	8 0	5 3	3 8	1 5	2 1	2 1	4 8	3 0	5 3	7 8	9 1	3 4	7 7	2 7
2 9	1 9	2 9	1 8	8 4	0 9	9 0	8 7	0 4	9 3	3 9	0 3	9 0	4 1	7 9
5 4	6 3	6 0	3 1	6 5	3 5	9 5	4 5	1 4	0 8	7 9	6 8	0 4	9 9	9 9
0 2	2 7	2 4	6 0	5 7	0 5	1 6	4 7	5 0	2 2	0 1	2 6	4 9	5 4	9 3
7 3	2 5	6 8	3 2	4 4	6 1	2 5	9 7	9 6	2 2	9 9	9 1	4 2	5 0	4 3
9 0	2 8	6 7	2 6	9 7	6 3	0 4	2 7	5 6	2 4	8 0	1 1	0 1	6 3	3 4
6 1	3 3	9 3	6 2	4 0	6 0	6 8	9 9	1 9	0 2	0 3	3 2	8 7	7 4	6 4
7 0	7 7	1 1	5 7	2 1	3 4	4 6	1 1	2 4	7 2	4 2	4 4	9 2	1 6	5 9
8 3	6 4	0 7	2 2	9 3	3 2	5 5	9 1	5 3	6 3	7 8	0 7	7 5	8 7	1 2
7 3	3 2	5 8	8 6	2 5	0 4	6 6	0 0	5 8	3 9	0 1	5 8	3 0	9 3	0 6
7 6	6 3	0 0	8 8	7 4	2 3	9 5	8 1	2 6	7 1	4 8	6 9	7 1	9 6	9 9
4 3	2 8	9 0	4 5	7 9	0 5	2 3	5 8	7 9	1 6	4 2	7 5	6 2	4 3	8 4
0 5	6 6	0 1	5 1	7 2	8 7	3 5	1 6	8 9	9 6	2 4	5 5	0 9	2 5	6 6
0 6	1 4	3 7	7 4	1 2	8 9	1 3	2 7	7 5	2 0	4 0	4 8	3 6	3 9	2 7
6 9	7 7	8 5	1 4	9 6	1 8	4 0	6 3	4 3	8 3	5 2	2 5	9 5	5 7	6 8
2 1	4 8	6 3	9 7	7 9	2 0	3 1	2 1	5 6	0 3	1 9	2 3	4 3	8 4	6 2
4 7	6 9	5 6	1 7	5 8	6 8	2 9	2 5	9 5	3 1	0 9	8 9	6 8	6 0	5 9
2 5	5 7	7 5	8 3	3 1	9 1	4 3	7 9	7 7	0 3	5 5	8 8	7 5	6 9	2 0
5 6	5 2	8 8	7 4	5 1	6 3	2 4	0 1	6 5	6 9	2 0	7 3	8 6	7 3	2 1
4 8	3 6	7 3	7 5	3 9	5 9	9 8	2 5	7 2	5 8	9 4	1 3	1 7	2 7	5 5

Σχήμα 1: Πίνακας τυχαίων αριθμών

Έτσι αν εφαρμόσουμε αλγόριθμο για αριθμό ασθενών  $n=24$  και χρήση θεραπευτικών αγωγών Α και Β έχουμε:

A (0-4), B (5-9)

(2) A (3) A (8) B (4) A (0) A (9) B (5) B (3) A (0) A (2) A (1) A (7) B (1) A (9) B (1) A (9) B (6) B (6) B (7) B (9) B (7) B (7) B (1) A (6) B

$n_A = 11$ ,  $n_B = 13$



### 3.2. Τετραγωνισμένη τυχαιοποίηση (Permuted Blocks)

Η **τετραγωνισμένη τυχαιοποίηση** έχει σχεδιαστεί έτσι, ώστε να τυχαιοποιεί άτομα σε ομάδες (groups) οι οποίες έχουν ως αποτέλεσμα ίσου μεγέθους δείγμα.

Έστω για παράδειγμα δύο θεραπευτικές αγωγές A και B. Για ομάδες (block) μεγέθους δύο έχουμε δύο τρόπους κατανομής μίας θεραπείας: AB (για τα ψηφία 0 – 4) και BA (για τα ψηφία 5 – 9). Οπότε για 12 ασθενείς θα μπορούσαμε να έχουμε την ακόλουθη τυχαιοποίηση:

Τυχαιά ψηφία:	5	3	0	9	6	8
Ομάδες:	BA	AB	AB	BA	BA	BA

Ασθενής	1	2	3	4	5	6	7	8	9	10	11	12
Φάρμακο	B	A	A	B	A	B	B	A	B	A	B	A

Για δύο θεραπευτικές αγωγές A και B, ομάδες (block) μεγέθους τέσσερα έχουμε έξι τρόπους κατανομής μία θεραπείας: AABB, ABAB, ABBA, BAAB, BABA, BBAA. Η τυχαιοποίηση θα μπορούσε να γίνει σύμφωνα με την παρακάτω ανάθεση:

Τυχαιό ψηφίο	Μπλοκ
1	AABB
2	ABAB
3	ABBA
4	BAAB
5	BABA
6	BBAA
0, 7-9	αγνοούνται

### 3.3. Στρωματοποιημένη τυχαιοποίηση (Stratified randomization)





Στην **στρωματοποιημένη τυχαιοποίηση** μας ενδιαφέρει ο έλεγχος των συγχυτικών παραγόντων όπως είναι για παράδειγμα το φύλο ή η ηλικία. Επίσης, σε πολυκεντρικές μελέτες οι ασθενείς των κέντρων τυχαιοποιούνται χωριστά με αποτέλεσμα το κέντρο να αποτελεί και αυτό μία μεταβλητή διαστρωμάτωσης. Συνεπώς στον έλεγχο μίας μελέτης θα πρέπει να φροντίζουμε να εξαλείφουμε τους συγχυτικούς παράγοντες, ώστε οι ομάδες θεραπείας να μην διαφέρουν ως προς αυτούς. Στην συνέχεια χωρίζουμε τους ασθενείς σε ομάδες ανάλογα με τους συγχυτικούς παράγοντες και εφαρμόζουμε την τυχαιοποίηση σε κάθε ομάδα χωριστά. Είναι σημαντικό στην **στρωματοποιημένη τυχαιοποίηση** σε κάθε ομάδα (block) να γίνεται χρήση της τετραγωνισμένης τυχαιοποίησης (permuted blocks) και όχι της απλής τυχαιοποίησης (simple randomization) καθώς έτσι διασφαλίζεται η ισορροπία μεταξύ των θεραπευτικών αγωγών. Όπως αναφέρθηκε και παραπάνω, χρήση της **στρωματοποιημένης τυχαιοποίησης** γίνεται σε πολυκεντρικές μελέτες.

Η **στρωματοποιημένη τυχαιοποίηση** μπορεί να επεκταθεί σε δύο οι περισσότερες μεταβλητές στρωματοποίησης. Για παράδειγμα, θα μπορούσαμε να επεκτείνουμε την διαστρωμάτωση σε μία μελέτη του καρκίνου του μαστού, ως προς το μέγεθος του όγκου και στον αριθμό των θετικών λεμφαδένων. Έτσι αν είχαμε δύο μεγέθη όγκων (π. χ.  $size \leq 4cm$  και  $size > 4cm$ ) και τρεις ομάδες λεμφαδένων (0, 1-4, >4) καθώς και την κατάσταση της εμμηνόπαυσης (πριν και μετά) θα είχαμε συνολικά  $2 \times 3 \times 2 = 12$  στρώματα. Ένα σημαντικό όμως πρόβλημα στην χρήση πολλαπλών στρωμάτων είναι ότι μερικοί συνδυασμοί των κατηγοριών μπορεί να είναι σπάνιοι με αποτέλεσμα να μην επιτυγχάνεται η ισορροπία (balance) μεταξύ των θεραπευτικών αγωγών [2].

## 4. Λογισμικό

### 4.1. Προδιαγραφές

Η εφαρμογή έχει αναπτυχθεί εξ ολοκλήρου σε γλώσσα Python [5], έκδοση 2.7. Το εργαλείο που χρησιμοποιήθηκε είναι το *Eclipse*, έκδοση Luna Release (4.4.0) [7].



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

Οι λόγοι που χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python είναι:

1. Διδάσκεται στο μεταπτυχιακό πρόγραμμα σπουδών.
2. Αποτελεί μαζί με την Java μία από τις δημοφιλέστερες γλώσσες προγραμματισμού.
3. Έχει αναπτυχθεί ένας μεγάλος αριθμός τυποποιημένων βιβλιοθηκών (libraries) σε Python [8] για τη βιοπληροφορική.
4. Μπορεί να χρησιμοποιηθεί για την δημιουργία αυτόνομων εφαρμογών και εφαρμογών Web.

#### 4.2. Κλάσεις

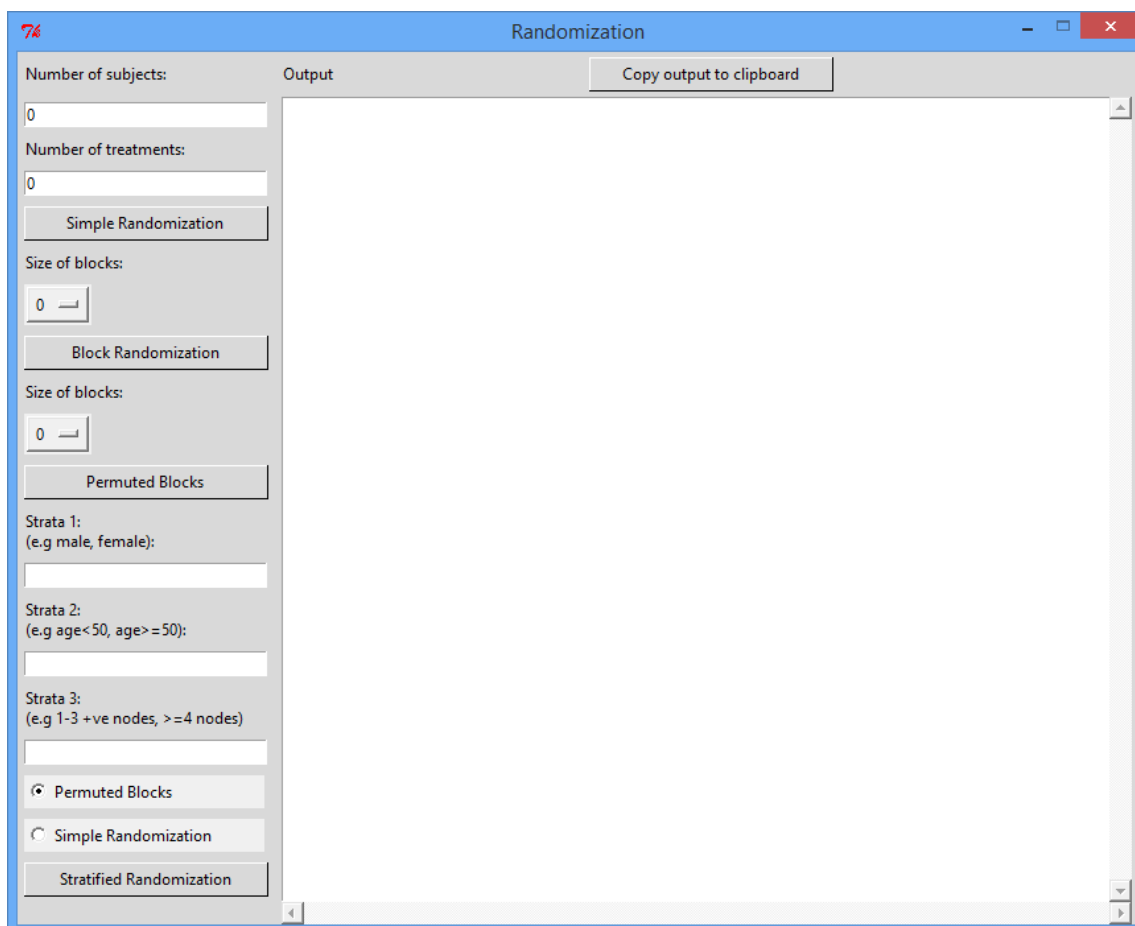
Η εφαρμογή αποτελείται από τις εξής κλάσεις:

1. Κλάση guiRandomization
2. Κλάση randomization
3. Κλάση simpleRandomization
4. Κλάση blockRandomization
5. Κλάση stratifiedRandomization
6. Κλάση permutedBlocks

Η κλάση **guiRandomization** είναι υπεύθυνη για την δημιουργία του παραθυρικού περιβάλλοντος της εφαρμογής όπως φαίνεται στο Σχήμα 2.



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’



Σχήμα 2: Το περιβάλλον της εφαρμογής

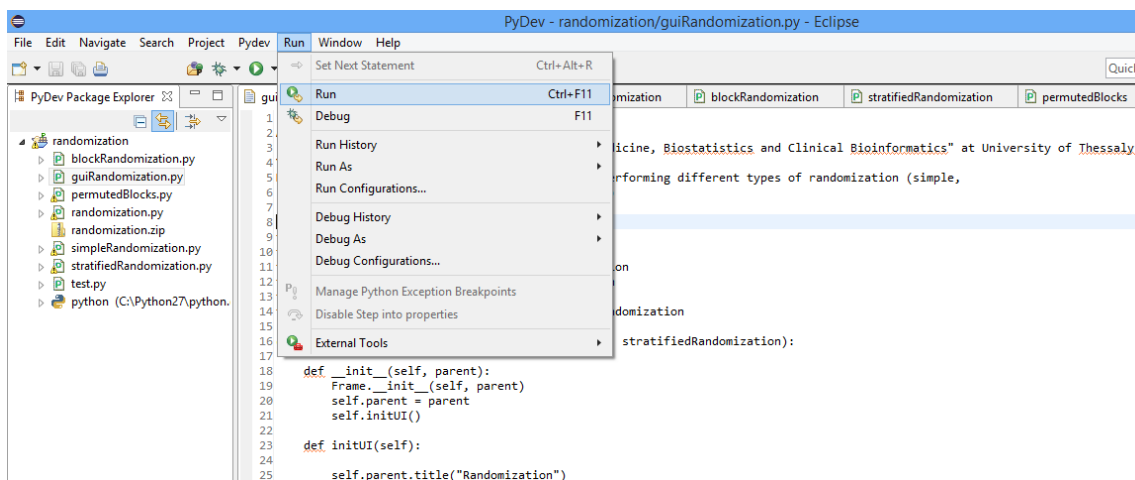
Η κλάση **randomization** είναι η κύρια κλάση η οποία κληρονομείται από τις υπόλοιπες και περιέχει μεθόδους που μπορούν να χρησιμοποιηθούν από τις υπόλοιπες. Η κλάση **simpleRandomization** υλοποιεί την λειτουργία της απλής τυχαιοποίησης, η κλάση **blockRandomization** και **permutedBlocks** την λειτουργία της τυχαιοποίησης κατά ομάδες και τέλος η κλάση **stratifiedRandomization** την λειτουργία της στρωματοποιημένης τυχαιοποίησης.

#### 4.3. Παρουσίαση εφαρμογής



## Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

Η εκκίνηση της εφαρμογής μπορεί να γίνει από το περιβάλλον του Eclipse (*Run*) από το μενού *Run* (Σχήμα 3)



**Σχήμα 3:** Εκκίνηση της εφαρμογής μέσα από το περιβάλλον του Eclipse

Μόλις γίνει η εκκίνηση της εφαρμογής εμφανίζεται το παράθυρο της επόμενης εικόνας



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

Σχήμα 4: Η εφαρμογή Randomization

Η διεπαφή αλληλεπίδρασης με τον χρήστη περιλαμβάνει:

- **Number of Subjects:** Ο αριθμός των ασθενών που θα συμπεριληφθεί στην τυχαιοποίηση.
- **Number of treatments:** Ο αριθμός των θεραπευτικών αγωγών (treatments) που θα συμπεριληφθούν στην τυχαιοποίηση.
- **Simple Randomization:** Εκτελεί την λειτουργία της **απλής τυχαιοποίησης**. Χρησιμοποιεί μόνο τα πεδία **Number of Subjects** και **Number of treatments**.
- **Size of blocks:** Ο αριθμός των ομάδων (blocks) κατά την τυχαιοποίηση κατά ομάδες (block randomization)



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

- **Block Randomization:** Εκτελεί την λειτουργία της **τυχαιοποίησης κατά ομάδες**. Χρησιμοποιεί τα πεδία **Number of Subjects**, **Number of treatments** και **Size of blocks**.
- **Size of blocks:** Ο αριθμός των ομάδων που θα χρησιμοποιηθούν στην τετραγωνισμένη τυχαιοποίηση (permuted blocks). Οι δυνατές τιμές είναι: 2, 4, 6, 8 για δύο θεραπευτικές αγωγές, 3, 6, 9 για τρεις θεραπευτικές αγωγές, 4, 8 για τέσσερις θεραπευτικές αγωγές και 5 για πέντε θεραπευτικές αγωγές. Για την τιμή π. χ. των τεσσάρων ομάδων γίνεται χρήση των εξής έξι ομάδων: [ 'AABB', 'ABAB', 'ABBA', 'BAAB', 'BABA', 'BBAA' ].
- **Permuted Blocks:** Εκτελεί την λειτουργία της **τετραγωνισμένης τυχαιοποίησης**. Χρησιμοποιεί τα πεδία **Number of Subjects** και **Number of treatments** και **Size of blocks**.
- **Strata 1, Strata 2 και Strata 3:** Τα στρώματα που θα δημιουργηθούν για την **στρωματοποιημένη τυχαιοποίηση**.
- **Permuted Blocks, Simple Randomization:** Ο τύπος της τυχαιοποίησης που θα χρησιμοποιηθεί στην **στρωματοποιημένη τυχαιοποίηση**.
- **Stratified Randomization:** Εκτελεί την λειτουργία της **στρωματοποιημένης τυχαιοποίησης**.
- **Output:** Στο πεδίο αυτό εμφανίζεται το αποτέλεσμα της τυχαιοποίησης που εκτελέστηκε.
- **Copy output to clipboard:** Αντιγράφει το αποτέλεσμα του output στην μνήμη για χρήση σε άλλη εφαρμογή.

## 5. Αποτελέσματα

Στις παρακάτω εικόνες παρουσιάζονται τα αποτελέσματα από την εκτέλεση της εφαρμογής.



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

Randomization

Copy output to clipboard

Number of subjects: 10

Number of treatments: 2

Simple Randomization

Size of blocks: 0

Block Randomization

Size of blocks: 0

Permuted Blocks

Strata 1:  
(e.g male, female):

Strata 2:  
(e.g age<50, age>=50):

Strata 3:  
(e.g 1-3 +ve nodes, >=4 nodes)

• Permuted Blocks

○ Simple Randomization

Stratified Randomization

Output

Simple Randomization

Number of subjects: 10  
Number of treatments: 2  
Group of digits: [[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]]

Subject 1 is allocated to treatment A (Random number: 4)  
Subject 2 is allocated to treatment A (Random number: 2)  
Subject 3 is allocated to treatment A (Random number: 4)  
Subject 4 is allocated to treatment A (Random number: 4)  
Subject 5 is allocated to treatment B (Random number: 5)  
Subject 6 is allocated to treatment B (Random number: 7)  
Subject 7 is allocated to treatment B (Random number: 6)  
Subject 8 is allocated to treatment A (Random number: 3)  
Subject 9 is allocated to treatment B (Random number: 9)  
Subject 10 is allocated to treatment A (Random number: 0)

Imbalance

A / B  
6 / 4

Σχήμα 5: Απλή τυχαιοποίηση (Simple randomization)



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

The screenshot shows a software window titled 'Randomization'. On the left, there are input fields and buttons for different randomization methods. The 'Block Randomization' method is selected. The input fields are: 'Number of subjects: 10', 'Number of treatments: 2', 'Size of blocks: 2'. The output area on the right shows the results of the randomization process.

Number of subjects: 10  
Number of treatments: 2  
Simple Randomization  
Size of blocks: 2  
Block Randomization  
Size of blocks: 0  
Permuted Blocks  
Strata 1: (e.g male, female):  
Strata 2: (e.g age<50, age>=50):  
Strata 3: (e.g 1-3 +ve nodes, >=4 nodes)  
• Permuted Blocks  
• Simple Randomization  
Stratified Randomization

Output  
Copy output to clipboard

Block Randomization  
Number of Subjects: 10  
Number of Treatments: 2  
Group of digits: [[0], [1]]  
Block 1: Subject 1 is allocated to treatment B (Random number: 1)  
Subject 2 is allocated to treatment A (Random number: 0)  
Block 2: Subject 3 is allocated to treatment B (Random number: 1)  
Subject 4 is allocated to treatment A (Random number: 0)  
Block 3: Subject 5 is allocated to treatment A (Random number: 0)  
Subject 6 is allocated to treatment B (Random number: 1)  
Block 4: Subject 7 is allocated to treatment B (Random number: 1)  
Subject 8 is allocated to treatment A (Random number: 0)  
Block 5: Subject 9 is allocated to treatment B (Random number: 1)  
Subject 10 is allocated to treatment A (Random number: 0)

Imbalance  
A / B  
5 / 5

Σχήμα 6: Τυχαιοποίηση κατά ομάδες (Block randomization)





Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

**Randomization**

Number of subjects: 10  
Number of treatments: 2  
Simple Randomization  
Size of blocks: 2  
Block Randomization  
Size of blocks: 4  
Permuted Blocks  
Strata 1: (e.g male, female):  
Strata 2: (e.g age<50, age>=50):  
Strata 3: (e.g 1-3 +ve nodes, >=4 nodes)  
• Permuted Blocks  
○ Simple Randomization  
Stratified Randomization

**Output** Copy output to clipboard

Permuted Blocks

Number of Subjects: 10  
Number of Treatments: 2  
Group of digits: [[1], [2], [3], [4], [5], [6]]  
Block Size: 4 (['AABB', 'ABAB', 'ABBA', 'BAAB', 'BABA', 'BBAA'])

Subject 1 is allocated to treatment B (Random number: 5)  
Subject 2 is allocated to treatment A  
Subject 3 is allocated to treatment B  
Subject 4 is allocated to treatment A  
Subject 5 is allocated to treatment B (Random number: 4)  
Subject 6 is allocated to treatment A  
Subject 7 is allocated to treatment A  
Subject 8 is allocated to treatment B  
Subject 9 is allocated to treatment B (Random number: 6)  
Subject 10 is allocated to treatment B  
Subject 11 is allocated to treatment A  
Subject 12 is allocated to treatment A

Imbalance

A	/	B
6	/	6

Σχήμα 7: Τετραγωνισμένη τυχαιοποίηση (Permuted blocks)



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

The screenshot shows a software window titled 'Randomization'. On the left, there are input fields and buttons for different randomization methods. The 'Stratified Randomization' method is selected. The input fields are set to: Number of subjects: 10, Number of treatments: 2, Size of blocks: 4, and Strata: male, female. The output area on the right displays the results of the stratified randomization, including a table of subject assignments and a summary of the imbalance.

Subject	male	female
1	B	B
2	A	B
3	B	A
4	A	A
5	B	A
6	A	B
7	B	A
8	A	B
9	A	B
10	A	B
11	B	A
12	B	A

Imbalance  
male {A/B (6/6)}  
female {A/B (6/6)}

Σχήμα 8: Στρωματοποιημένη τυχαιοποίηση (Stratified Randomization)

## 6. Συμπέρασμα

Σ' αυτή τη διπλωματική εργασία υλοποιήθηκε μία εφαρμογή για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (απλή τυχαιοποίηση, τετραγωνισμένη τυχαιοποίηση, στρωματοποίηση κατά ομάδες και στρωματοποιημένη τυχαιοποίηση). Η χρήση της γλώσσας προγραμματισμού PYTHON σημαίνει ότι το σύστημα μπορεί να χρησιμοποιηθεί τόσο σε περιβάλλον Windows όσο και σε περιβάλλον UNIX. Ακόμα και αν επιτεύχθηκε ο σκοπός της τυχαιοποίησης υπάρχει ακόμα χώρος για μελλοντικές



Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

προεκτάσεις όπως για παράδειγμα να συμπεριληφθούν και άλλοι τύποι τυχαιοποίησης όπως covariate adaptive randomization, response-adaptive randomization.

## 7. Αναφορές

- [1] Frane JW. A method of biased coin randomization, its implementation and validation. Drug Inf J. 1998;32:423–32.
- [2] Altaman DG, Bland JM. How to use randomize. BMJ. 1999;319:703–4.
- [3] Altaman DG, Bland JM. Statistics notes. Treatment allocation in controlled trials: Why randomize? BMJ. 1999;318:1209.
- [4] An overview of randomization techniques: An unbiased assessment of outcome in clinical research <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3136079/>
- [5] Python <https://www.python.org/>
- [6] The Python tutorial <https://docs.python.org/2/tutorial/>
- [7] Eclipse <https://eclipse.org/>
- [8] Biopython [http://biopython.org/wiki/Main\\_Page](http://biopython.org/wiki/Main_Page)
- [9] py2exe <http://www.py2exe.org>



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

## 8. Appendix – Κώδικας Python

### 8.1. Κλάση guiRandomization

```
"""
Author:      Theodoros Mprotsis
              (MSc Research Methodology in Biomedicine, Biostatistics and
              Clinical Bioinformatics" at University of Thessaly)
Year:        2014/2015
Description: Develop a software in Python for performing different types of
              randomization (simple, stratified and block randomization)
"""

from Tkinter import *
from ttk import Frame, Button, Label, Style
from simpleRandomization import simpleRandomization
from permutedBlocks import permutedBlocks
from blockRandomization import blockRandomization
from stratifiedRandomization import stratifiedRandomization

class guiRandomization(Frame):

    # Initialization
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.randomizations={'simple': simpleRandomization(), \
                              'block': blockRandomization(), \
                              'permuted': permutedBlocks(), \
                              'stratified': stratifiedRandomization()}

        self.initUI()

    # Initialize the GUI
    def initUI(self):

        self.parent.title("Randomization")
        self.style = Style()
        self.style.theme_use("default")
        self.pack(fill=BOTH, expand=TRUE)

        maxRows=21

        self.columnconfigure(1, weight=1)
        self.columnconfigure(3, pad=7)
        self.rowconfigure(maxRows, weight=1)
        self.rowconfigure(maxRows, pad=7)

        rowNumber=0
```

---

Πανεπιστήμιο Θεσσαλίας: Διπλωματική Εργασία



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
# Options for permuted blocks
self.PERMUTED_LIST={'2': {2, 4, 6, 8}, '3': {3, 6, 9}, '4': {4, 8},
'5': {5}}

# Modes for stratified randomization
MODES = [
    ("Permuted Blocks\n(preferred for balance)", "PERMUTED"),
    ("Simple Randomization", "SIMPLE"),
]

lblOutput = Label(self, text="Output")
lblOutput.grid(row=0, column=3, sticky=W, pady=4, padx=5)

btncopyToClipboard=Button(self, text="Copy output to clipboard",
command=self.copytoclipboard, width=29)
btncopyToClipboard.grid(row=0, column=3, sticky=E, pady=4, padx=5)

self.output = Text(self, wrap="none", height=20)

self.output.grid(row=1, column=3, columnspan=2, rowspan=maxRows,
padx=5, sticky=E+W+S+N)
hscrl = Scrollbar(self, orient=HORIZONTAL, command=self.output.xview)
vscrl = Scrollbar(self, command=self.output.yview)

vscrl.grid(row=rowNumber+1, column=3, columnspan=2, rowspan=maxRows,
sticky='ens', padx=5)
hscrl.grid(row=maxRows+1, column=3, columnspan=2, sticky='ew',
padx=5)

self.output["xscrollcommand"]=hscrl.set
self.output["yscrollcommand"]=vscrl.set

lblsimpleNumberOfSubjects=Label(self, text="Number of subjects:")
lblsimpleNumberOfSubjects.grid(row=rowNumber, column=0, sticky=W,
pady=4, padx=5)
rowNumber=rowNumber+1

self.simpleNumberOfSubjects=IntVar()
self.simpleNumberOfSubjects.trace('w', self.update_all_options)
txtsimpleNumberOfSubjects=Entry(self, width=30,
textvariable=self.simpleNumberOfSubjects)
txtsimpleNumberOfSubjects.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

lblsimpleNumberOfTreatments=Label(self, text="Number of treatments:")
lblsimpleNumberOfTreatments.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
self.simpleNumberOfTreatments=IntVar()
self.simpleNumberOfTreatments.trace('w', self.update_all_options)
txtsimpleNumberOfTreatments=Entry(self, width=30,
textvariable=self.simpleNumberOfTreatments)
txtsimpleNumberOfTreatments.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

btnsimpleRandomization=Button(self, text="Simple Randomization",
command=self.simpleRandomization, width=29)
btnsimpleRandomization.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

lblblockSizeOfBlocks=Label(self, text="Size of blocks:")
lblblockSizeOfBlocks.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

self.blockSizeOfBlocks=IntVar()
self.optblockSizeOfBlocks=OptionMenu(self, self.blockSizeOfBlocks,
'')
self.optblockSizeOfBlocks.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

btnblockRandomization=Button(self, text="Block Randomization",
command=self.blockRandomization, width=29)
btnblockRandomization.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

lblsizeOfBlocks=Label(self, text="Size of blocks:")
lblsizeOfBlocks.grid(row=rowNumber, column=0, sticky=W+N, pady=4,
padx=5)
rowNumber=rowNumber+1

self.SizeOfBlocks=IntVar()
self.optSizeOfBlocks=OptionMenu(self, self.SizeOfBlocks, '')
self.optSizeOfBlocks.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

btnpermutedBlocks=Button(self, text="Permuted Blocks",
command=self.permutedBlocks, width=29)
btnpermutedBlocks.grid(row=rowNumber, column=0, sticky=W+N, pady=4,
padx=5)
rowNumber=rowNumber+1

lblstrata1=Label(self, text="Strata 1:\n(e.g. male, female):")
lblstrata1.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
rowNumber=rowNumber+1

self.strata1=StringVar()
txtstrata1=Entry(self, width=30, textvariable=self.strata1)
txtstrata1.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
rowNumber=rowNumber+1

lblstrata2=Label(self, text="Strata 2: \n(e.g. age<50, age>=50):")
lblstrata2.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
rowNumber=rowNumber+1

self.strata2=StringVar()
txtstrata2=Entry(self, width=30, textvariable=self.strata2)
txtstrata2.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
rowNumber=rowNumber+1

lblstrata3=Label(self, text="Strata 3: \n(e.g. 1-3 +ve nodes, >=4
nodes)")
lblstrata3.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
rowNumber=rowNumber+1

self.strata3=StringVar()
txtstrata3=Entry(self, width=30, textvariable=self.strata3)
txtstrata3.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
rowNumber=rowNumber+1

self.stratifiedOptions=StringVar()
self.stratifiedOptions.set("PERMUTED") # initialize
for text, mode in MODES:
    b=Radiobutton(self, text=text, variable=self.stratifiedOptions,
value=mode, justify=LEFT, width=22, anchor=W)
    b.grid(row=rowNumber, column=0, sticky=W+N, pady=4, padx=5)
    rowNumber=rowNumber+1

btnstratifiedRandomization=Button(self, text="Stratified
Randomization", command=self.stratifiedRandomization, width=29)
btnstratifiedRandomization.grid(row=rowNumber, column=0, sticky=W+N,
pady=4, padx=5)
rowNumber=rowNumber+1

def simpleRandomization(self):
    self.output.delete(1.0, END)
    try:

self.randomizations['simple'].setNumberOfSubjects(self.simpleNumberOfSubjects
.get())

self.randomizations['simple'].setNumberOfTreatments(self.simpleNumberOfTreatm
ents.get())
        self.randomizations['simple'].allocateSubjects()
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
        self.output.insert(INSERT,
self.randomizations['simple'].consoleOutput())
        except ValueError as v:
            self.output.insert(INSERT, "Please insert a valid
number.\n\nError message:%s"%v)
        except Exception as e:
            self.output.insert(INSERT, e)

def permutedBlocks(self):
    self.output.delete(1.0, END)
    try:

self.randomizations['permuted'].setNumberOfTreatments(self.simpleNumberOfTrea
tments.get())

self.randomizations['permuted'].setNumberOfSubjects(self.simpleNumberOfSubjec
ts.get())

self.randomizations['permuted'].setSizeOfBlocks(self.SizeOfBlocks.get())
        self.randomizations['permuted'].allocateSubjects()
        self.output.insert(INSERT,
self.randomizations['permuted'].consoleOutput())
        except ValueError as v:
            self.output.insert(INSERT, "Please insert a valid
number.\n\nError message: %s"%v)
        except Exception as e:
            self.output.insert(INSERT, e)

def blockRandomization(self):
    self.output.delete(1.0, END)
    try:

self.randomizations['block'].setNumberOfSubjects(self.simpleNumberOfSubjects.
get())

self.randomizations['block'].setNumberOfTreatments(self.simpleNumberOfTreatme
nts.get())

self.randomizations['block'].setSizeOfBlocks(self.blockSizeOfBlocks.get())
        self.randomizations['block'].allocateSubjects()
        self.output.insert(INSERT,
self.randomizations['block'].consoleOutput())
        except ValueError as v:
            self.output.insert(INSERT, "Please insert a valid
number.\n\nError message: %s"%v)
        except Exception as e:
            self.output.insert(INSERT, e)

def stratifiedRandomization(self):
    self.output.delete(1.0, END)
    try:
```





Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
self.randomizations['stratified'].setNumberOfTreatments(self.simpleNumberOfTreatments.get())

self.randomizations['stratified'].setNumberOfSubjects(self.simpleNumberOfSubjects.get())
    if self.stratifiedOptions.get()=='PERMUTED':

self.randomizations['stratified'].setSizeOfBlocks(self.SizeOfBlocks.get())

self.randomizations['stratified'].setRandomization(self.stratifiedOptions.get())
    self.randomizations['stratified'].setStrata(self.strata1.get())
    self.randomizations['stratified'].setStrata(self.strata2.get())
    self.randomizations['stratified'].setStrata(self.strata3.get())
    self.randomizations['stratified'].allocateSubjects()
    self.output.insert(INSERT,
self.randomizations['stratified'].consoleOutput())
    except ValueError as v:
        self.output.insert(INSERT, "Please insert a valid
number.\n\nError message: %s"%v)
    except Exception as e:
        self.output.insert(INSERT, e)

# Copies the content of the output into the clipboard
def copytoclipboard(self):
    output_value=self.output.get("1.0", 'end-1c') # Get the output
    value, but remove line return at end
    self.parent.clipboard_clear() # Clear clipboard
    contents
    self.parent.clipboard_append(output_value) # Append new value to
    clipbaord

# Updates the option menus "Size of blocks" for Block randomization and
permuted blocks
def update_all_options(self, *args):
    try:
        self.update_option_block_randomization(self, *args)
        self.update_option_permuted_blocks(self, *args)
    except Exception:
        pass

# Updates the option menu "Size of blocks" for Block randomization
def update_option_block_randomization(self, *args):
    try:
        x=blockRandomization()
        x.setNumberOfSubjects(self.simpleNumberOfSubjects.get())
        x.setNumberOfTreatments(self.simpleNumberOfTreatments.get())
        menu=self.optblockSizeOfBlocks['menu']
        menu.delete(0, 'end')
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
        for subjectsInBlock in
sorted(x.getNumbersOfSubjectsInBlock(self.simpleNumberOfSubjects.get())):
            menu.add_command(label=subjectsInBlock, command=lambda
sib=subjectsInBlock: self.blockSizeOfBlocks.set(sib))
        except Exception:
            pass

# Updates the option menu "Size of blocks" for permuted blocks
def update_option_permuted_blocks(self, *args):
    try:
        menu=self.optSizeOfBlocks['menu']
        menu.delete(0, 'end')
        for blockNumber in
sorted(self.PERMUTED_LIST[str(self.simpleNumberOfTreatments.get())]):
            menu.add_command(label=blockNumber, command=lambda
blkNr=blockNumber: self.SizeOfBlocks.set(blkNr))
        except Exception:
            pass

def main():

    root = Tk()
    root.geometry("+300+50")      # width x height + x_offset + y_offset
    root.resizable(0,0)          # Not resizable
    guiRandomization(root)
    root.mainloop()

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

## 8.2. Κλάση randomization

```
"""
Author:      Theodoros Mprotsis
              (MSc Research Methodology in Biomedicine, Biostatistics and
Clinical Bioinformatics at University of Thessaly)
Year:        2014/2015
Description: Develop a software in Python for performing different types of
randomization (simple,
               stratified and block randomization)
"""

"""
Randomization
-----
Main class.
Inherited by the other classes (simpleRandomization, permutedBlocks,
stratifiedRandomization,
```

---

Πανεπιστήμιο Θεσσαλίας: Διπλωματική Εργασία



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
blockRandomization.
"""

class randomization(object):

    # Error Messages
    _ERROR_MSG_NUMBER_OF_TREATMENTS= 'Number of treatments must be between 1
and 10'
    _ERROR_MSG_NUMBER_OF_SUBJECTS= 'Number of subjects must be greater than 0'
    _ERROR_MSG_NUMBER_OF_BLOCKS= 'Number of blocks must be: '
    _ERROR_MSG_STRATA_FORMAT= 'Strata must be in the format: group1, group2
for instance: male, female'
    _ERROR_MSG_STRATA_GROUPS_NOT_DEFINED= 'Please define the strata groups
first'
    _ERROR_MSG_STRATA_SUB_RANDOMIZATION= 'Possible values are SIMPLE and
PERMUTED'
    _ERROR_MSG_NO_STRATA_DEFINED= 'No strata where defined'

    # Constant variables
    _MAX_NUMBER_OF_TREATMENTS=10
    _NUMBER_OF_DIGITS=10

    def __init__(self):
        self.numberOfTreatments=2
        self.numberOfSubjects=10
        self.allocation=[]
        self.randomNumber=[]

    # Shows the the possible imbalance in randomization with the treatments
    def possibleImbalance(self):
        returnValue1=''
        returnValue2=''
        returnValue1=returnValue1+ '\nImbalance\n\n'
        for i in range(0, self.numberOfTreatments):
            returnValue1 = returnValue1+ '{0:3s}'.format(chr(i+65))+ ' / '
            returnValue2 =
        returnValue2+ '{0:3s}'.format(str(self.allocation.count(chr(i+65))))+ ' / '
        return returnValue1[:-3]+ '\n'+returnValue2[:-3]

    # Output in the console
    def consoleOutput(self):
        output=''
        for i in range(0, len(self.allocation)):
            output=output+ 'Subject {0:3d} is allocated to treatment {1}
(Random number: {2})\n'.format(i+1, self.allocation[i], self.randomNumber[i])
        output=output+self.possibleImbalance()
        return output

    # Creating groups
    def createGroups(self):
        self.groupOfDigis=[[]]
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
numbersInEachGroup=self._NUMBER_OF_DIGITS / self.numberOfTreatments
currentGroup=1
numberInCurrentGroup=1
lbound=0
excludeDigits = self._NUMBER_OF_DIGITS % self.numberOfTreatments
if (excludeDigits > 0):
    lbound = excludeDigits
for i in range(lbound, self._NUMBER_OF_DIGITS):
    self.groupOfDigis[currentGroup-1].append(i)
    if numberInCurrentGroup==numbersInEachGroup:
        numberInCurrentGroup=1
        self.numberOfGroups = currentGroup
        currentGroup=currentGroup+1
        if i<self._NUMBER_OF_DIGITS-1:
            self.groupOfDigis.append([])
    else:
        numberInCurrentGroup=numberInCurrentGroup+1
return

# Setters
# Sets the number of subjects
def setNumberOfSubjects(self, numberOfSubjects):
    if (numberOfSubjects<=0):
        raise Exception(self._ERROR_MSG_NUMBER_OF_SUBJECTS)
    else:
        self.numberOfSubjects=numberOfSubjects

# Sets the number of treatments for allocation
def setNumberOfTreatments(self, numberOfTreatments):
    if (numberOfTreatments>self._MAX_NUMBER_OF_TREATMENTS or
        numberOfTreatments<=0):
        raise Exception(self._ERROR_MSG_NUMBER_OF_TREATMENTS)
    else:
        self.numberOfTreatments=numberOfTreatments

# Getters
# Gets the allocation
def getAllocation(self):
    return self.allocation

# Gets the number of treatments
def getNumberOfTreatments(self):
    return self.numberOfTreatments
```

### 8.3. Κλάση simpleRandomization

"""  
Author: [Theodoros Mprotsis](#)

---

Πανεπιστήμιο Θεσσαλίας: Διπλωματική Εργασία



Θεόδωρος Μπρότσας, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

(MSc Research Methodology in Biomedicine, Biostatistics and Clinical Bioinformatics" at University of Thessaly)

Year: 2014/2015

Description: Develop a software in Python for performing different types of randomization (simple, stratified and block randomization)

"""

```
from randomization import randomization
```

```
import random
```

"""

*Simple Randomization*

-----

Randomization based on a single sequence of random assignments is known as simple randomization. This technique maintains complete randomness of the assignment of a subject to a particular group. The most common and basic method of simple randomization is flipping a coin (source: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3136079>).

```
class simpleRandomization(randomization):
```

```
# Constructor
```

```
def __init__(self):  
    randomization.__init__(self)  
    self.groupOfDigis=[]  
    self.numberOfGroups=0
```

```
# Allocate subjects (patients) to groups (treatments)
```

```
def allocateSubjects(self):  
    self.allocation=[]  
    self.randomNumber=[]  
    # Creating the groups  
    self.createGroups()  
    nOfGroups=self.numberOfTreatments  
    lbound=self._NUMBER_OF_DIGITS % self.numberOfTreatments  
    ubound=self._NUMBER_OF_DIGITS-1  
    currentGroup=0  
    for i in range(lbound, self.numberOfSubjects+lbound):  
        randomNumber=random.randint(lbound, ubound)  
        for j in range(currentGroup, nOfGroups):  
            if randomNumber in self.groupOfDigis[j]:  
                self.allocation.append(chr(j+65))  
                self.randomNumber.append(randomNumber)
```

```
# Override
```

---

Πανεπιστήμιο Θεσσαλίας: Διπλωματική Εργασία



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
# Output in the console
def consoleOutput(self):
    output=self.getOutputHeader()
    output=output+randomization.consoleOutput(self)
    return output

# Gets the header of the output
def getOutputHeader(self):
    output=self.getName()+'\n\n'
    output=output+'Number of subjects: %d\n'%(self.numberOfSubjects)
    output=output+'Number of treatments: %d\n'%(self.numberOfTreatments)
    output=output+'Group of digits: %s\n\n'%(str(self.groupOfDigits))
    return output

# Getters
# Gets the Number of Treatments for allocation
def getNumberOfTreatments(self):
    return self.numberOfTreatments

# Gets the Number of Groups
def getNumberOfGroups(self):
    return self.numberOfGroups

# Gets the number of subjects allocated in each group (treatment)
def getNrOfSubjectsInGroup(self, treatment):
    return self.allocation.count(treatment)

# Gets the name of the randomization
def getName(self):
    return 'Simple Randomization'

def getInternal(self):
    return 'SIMPLE'

if __name__ == "__main__":
    try:
        x=simpleRandomization()
        x.setNumberOfTreatments(3)
        x.setNumberOfSubjects(10)
        x.allocateSubjects()
        print x.consoleOutput()
    except Exception as e:
        print e
```

#### 8.4. Κλάση permutedBlocks

```
"""
Author: Theodoros Mprotsis
```



Θεόδωρος Μπρότσας, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
(MSc Research Methodology in Biomedicine, Biostatistics and
Clinical Bioinformatics" at University of Thessaly)
Year: 2014/2015
Description: Develop a software in Python for performing different types of
randomization (simple,
stratified and block randomization)
"""

from randomization import randomization

import random
import itertools

"""
    Permuted Blocks
    -----
    An alternative to table of random numbers procedure is to adopt a random
    permutation procedure
    (a random ordering of numbers). In this procedure, we consider all
    possible ordering ways in defining
    blocks of the same.
    """

class permutedBlocks(randomization):

    # Error Messages
    _ERROR_MSG_NUMBER_OF_TREATMENTS= 'Number of treatments must be 2, 3, 4 or
5'

    # Constant variables
    _MAX_NUMBER_OF_TREATMENTS=5

    # Constructor
    def __init__(self):

        randomization.__init__(self)
        # Default number of blocks
        self.numberOfBlocks=2
        self.newNumberOfBlocks=self.numberOfBlocks
        # The group of the digits for instance [[0, 1, 2, 3, 4], [5, 6, 7, 8,
9]] when two group block
        # randomization is used
        self.groupOfDigis=[[[]]]
        # The treatment groups for 1:1 ratio for instance ['AABB', 'ABAB',
'ABBA', 'BAAB', 'BABA', 'BBAA']
        # when four group block randomization is used
        self.treatmentGroups=[]
        # Options for permuted blocks
        self.PERMUTED_LIST={'2': {2, 4, 6, 8}, '3': {3, 6, 9}, '4': {4, 8},
'5': {5}}
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
# Creates all the possible combinations of the treatments e.g.
# for treatments A and B and two blocks the possible combinations are
# ['AB', 'BA']
# for treatments A and B and four blocks the possible combinations are
# ['AABB', 'ABAB', 'ABBA', 'BAAB', 'BABA', 'BBAA']
def createCombinationsOfTreatments(self):

    # Create all the possible combinations of the two treatments
    product=itertools.product(self.getASCIITreatments(),
repeat=self.numberOfBlocks)

    # Number of unique combinations
    uniqueBlocks=0
    for s in product:
        # The ratio is 1:1
        if self.checkTreatmentRatio(s):
            uniqueBlocks=uniqueBlocks+1
            self.treatmentGroups.append(''.join(s))

    # When the new number of blocks is greater then 10 (0..9)
    # expand the number of digits e. g. (0..19)
    if uniqueBlocks > self._NUMBER_OF_DIGITS:
        self._NUMBER_OF_DIGITS=len(self.treatmentGroups)
        uniqueBlocks=self._NUMBER_OF_DIGITS

    return uniqueBlocks

# Checks the treatment ratio in the permutation block (1:1)
def checkTreatmentRatio(self, arrayOfTreatments):
    equal=True
    i=1
    while i<len(self.getASCIITreatments()) and equal:
        if
arrayOfTreatments.count(chr(65+i))<>arrayOfTreatments.count(chr(65+i-1)):
            equal=False
        i=i+1
    return equal

# Defines the numbers in each group
def defineNumbersInEachGroup(self):
    # First we assume that we have two blocks
    numbersInEachGroup=5
    # If the number of blocks are not two e.g. 4,6,8...
    if self.numberOfBlocks<>2:
        # ...the numbers in each group will be one
        numbersInEachGroup=1
    return numbersInEachGroup

# Creates the groups for allocation and returns the number of 1:1 ratio
groups
def createGroups(self):
```





Θεόδωρος Μπρότσης, ‘Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)’

```
self.treatmentGroups = []
self.groupOfDigis=[]
lbound=0
currentGroup=1
numbersInEachGroup=self.defineNumbersInEachGroup()

# Creates all possible combinations of the treatments and returns the
# number of blocks in 1:1 ratio
uniqueBlocks=self.createCombinationsOfTreatments()

excludeDigits=self._NUMBER_OF_DIGITS % uniqueBlocks

ubound=self._NUMBER_OF_DIGITS-excludeDigits
if (excludeDigits>0):
    lbound=1
    ubound=ubound+1

numberInCurrentGroup=1
for i in range(lbound, ubound):
    self.groupOfDigis[currentGroup-1].append(i)
    if numberInCurrentGroup==numbersInEachGroup:
        numberInCurrentGroup=1
        self.numberOfGroups=currentGroup
        currentGroup=currentGroup+1
        if i<ubound-1:
            self.groupOfDigis.append([])
    else:
        numberInCurrentGroup=numberInCurrentGroup+1

return uniqueBlocks

# Allocate subjects to groups
def allocateSubjects(self):

    self.allocation=[]
    self.randomNumber=[]
    # Creating the groups
    noOfGroups=self.createGroups()

    lbound=self.groupOfDigis[0][0]
    ubound=self.groupOfDigis[len(self.groupOfDigis)-
1][len(self.groupOfDigis[len(self.groupOfDigis)-1])-1]
    currentSubject=0
    while currentSubject<self.numberOfSubjects:
        randomNumber=random.randint(lbound, ubound)
        currentGroup=0
        while (currentGroup<noOfGroups):
            if randomNumber in self.groupOfDigis[currentGroup]:
                y=0
                self.randomNumber.append(randomNumber)
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
while (y<self.newNumberOfBlocks):

self.allocation.append(self.treatmentGroups[currentGroup][y])
    y=y+1
    currentSubject=currentSubject+1
    currentGroup=currentGroup+1

# Override: prints the block size too
def consoleOutput(self):
    # The name of the randomization
    output=self.getOutputHeader()
    # Allocation
    for i in range(0, len(self.allocation)):
        if i % self.newNumberOfBlocks==0:
            output=output+'Subject {0:3d} is allocated to treatment {1}
(Random number: {2})\n'.format(i+1, self.allocation[i],
self.randomNumber[i/self.newNumberOfBlocks])
        else:
            output=output+'Subject {0:3d} is allocated to treatment
{1}\n'.format(i+1, self.allocation[i])

    output=output+self.possibleImbalance()
    return output

def getOutputHeader(self):
    # The name of the randomization
    output=self.getName()+'\n\n'
    # The Number of Subjects
    output=output+'Number of Subjects: %d\n'%(self.numberOfSubjects)
    output=output+'Number of Treatments: %d\n'%(self.numberOfTreatments)
    output=output+'Group of digits: %s\n\n'%(str(self.groupOfDigits))
    # The block size used e.g. 2,4,6,8
    output=output+'Block Size: %d (%s)%(self.numberOfBlocks,
str(self.treatmentGroups))+'\n\n'
    return output

# Returns the letters of the treatments based on the number of treatments
def getASCIITreatments(self):
    ascii=''
    for i in range (0, self.numberOfTreatments):
        ascii=ascii+chr(65+i)
    return ascii

# Setters
# Sets the size of the blocks
def setSizeOfBlocks(self, numberOfBlocks):
    if (numberOfBlocks in
self.PERMUTED_LIST[str(self.numberOfTreatments)]):
        self.numberOfBlocks=numberOfBlocks
        self.newNumberOfBlocks=self.numberOfBlocks
    else:
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
        raise
Exception(self._ERROR_MSG_NUMBER_OF_BLOCKS+str(sorted(self.PERMUTED_LIST[str(
self.numberOfTreatments)])))

# Sets the number of the treatments
def setNumberOfTreatments(self, numberOfTreatments):
    if numberOfTreatments<2 or
numberOfTreatments>self._MAX_NUMBER_OF_TREATMENTS:
        raise Exception(self._ERROR_MSG_NUMBER_OF_TREATMENTS)
    else:
        self.numberOfTreatments=numberOfTreatments

# Getters
# Gets the name of the randomization
def getName(self):
    return 'Permuted Blocks'

def getInternal(self):
    return 'PERMUTED'

def getSizeOfBlocks(self):
    return self.newNumberOfBlocks

def getTreatmentGroups(self):
    return self.treatmentGroups

if __name__ == "__main__":
    try:
        x=permutedBlocks()
        x.setNumberOfTreatments(5)
        x.setSizeOfBlocks(5)
        x.setNumberOfSubjects(12)
        x.allocateSubjects()
        print x.consoleOutput()
    except Exception as e:
        print e
```

## 8.5. Κλάση stratifiedRandomization

```
"""
Author:      Theodoros Mprotsis
             (MSc Research Methodology in Biomedicine, Biostatistics and
             Clinical Bioinformatics" at University of Thessaly)
Year:       2014/2015
Description: Develop a software in Python for performing different types of
             randomization (simple,
             stratified and block randomization)
"""
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
from simpleRandomization import simpleRandomization
from permutedBlocks import permutedBlocks

import itertools

"""
    Stratified Randomization
    -----
    The stratified randomization method addresses the need to control and
    balance the
    influence of covariates. This method can be used to achieve
    balance among groups in terms of subjects baseline characteristics
    (covariates).
    (source: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3136079)

    Restriction: The stratified randomization class works for two, three,
    four and five treatments
    """

class stratifiedRandomization(simpleRandomization, permutedBlocks):

    # Error Messages
    _ERROR_MSG_NUMBER_OF_TREATMENTS= 'Number of treatments must be 2, 3, 4 or
5'

    # Constant variables
    _MAX_NUMBER_OF_TREATMENTS=5
    _LEFT_OUTPUT_SPACES=16

    # Constructor
    def __init__(self):
        simpleRandomization.__init__(self)
        permutedBlocks.__init__(self)
        self.strata=[]
        self.noOfSubjectsInEachGroup=0

    # Allocate subjects to groups
    def allocateSubjects(self):

        self.allocation=[]

        if self.strata:
            self.numberOfStrata=self.getNumberOfStrata()
            self.noOfSubjectsInEachGroup=self.numberOfSubjects
            for i in range(0, self.getNumberOfStrata()):

self.strataRandomization.setNumberOfTreatments(self.numberOfTreatments)
                if self.strataRandomization.getInternal()=='PERMUTED':

self.strataRandomization.setSizeOfBlocks(self.getSizeOfBlocks())
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
self.strataRandomization.setNumberOfSubjects(self.noOfSubjectsInEachGroup)
self.strataRandomization.allocateSubjects()

self.allocation.append(self.strataRandomization.getAllocation())

self.strataRandomization.setNumberOfSubjects(self.numberOfSubjects)
else:
    raise Exception(self._ERROR_MSG_NO_STRATA_DEFINED)

# Override
# Shows the the possible imbalance in randomization with the treatments
def possibleImbalance(self):
    returnvalue='Imbalance\n\n'
    for i in range(0, self.numberOfStrata):
        returnValue1=''
        returnValue2=''
        for j in range(0, self.numberOfTreatments):
            returnValue1 = returnValue1 + chr(j+65) + '/'
            returnValue2 =
returnValue2+str(self.allocation[i].count(chr(j+65)))+ '/'
            returnValue=returnvalue+'%s'%(self.combineGroupHeaders()[i])+
{''+returnValue1[:-1]+' ('+returnValue2[:-1]+')}\n'
        return returnvalue

# Combines all stratas
def combineStrata(self, arrays):
    out=[]
    for t in itertools.product(*arrays):
        out.append(t)
    return out

# Override: prints the block size too
def consoleOutput(self):

    maximumLengthOfCombinedGroups=self.getMaximumLengthOfCombinedGroups()

    # Main header
    output=self.getName()+'\n\n'
    # Header from the corresponding randomization of the groups
    output=output+'Randomization between groups:
%s'%(self.strataRandomization.getOutputHeader())+'\n'
    output=output+' '*self._LEFT_OUTPUT_SPACES
    for i in range(0, len(self.combineStrata(self.strata))):
        output=output+'{0:{1}s}'.format(self.combineGroupHeaders()[i],
maximumLengthOfCombinedGroups)

    output=output+'\n'
    output=output+' '*self._LEFT_OUTPUT_SPACES+'-
'*(len(self.combineStrata(self.strata))*maximumLengthOfCombinedGroups-1)
    output=output+'\n'
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
for i in range(0, len(self.allocation[0])):
    output=output+'Subject {0:3d}'.format(i+1)
    for j in range(0, len(self.allocation)):
        output=output+'{0:{1}s}'.format(self.allocation[j][i],
maximumLengthOfCombinedGroups)
    output=output+'\n'

output=output+'\n'+self.possibleImbalance()

return output

# Combines the group headers
def combineGroupHeaders(self):
    groupHeaders=[]
    for i in range(0, len(self.combineStrata(self.strata))):
        header=''
        for j in range(0, len(self.combineStrata(self.strata)[i])):
            header=header+self.combineStrata(self.strata)[i][j]+'/'
        # Remove the last /
        header=header[:-1]
        groupHeaders.append(header)
    return groupHeaders

# Setters
# Sets the Strata e.g. age < 50, age >= 50, 1-3 +ve nodes, >= 4 +ve nodes
def setStrata(self, strata):
    if strata.strip()<>'':
        strata=strata.replace(' ', ',').replace(' ', ',')
        self.strata.append(strata.split(','))
        maxgroups=1
        for i in range(0, len(self.strata)):
            maxgroups=maxgroups*len(self.strata[i])
        if (maxgroups==1):
            raise Exception(self._ERROR_MSG_STRATA_FORMAT)

# Sets the randomization between the groups
def setRandomization(self, strataRandomization):
    if strataRandomization=='SIMPLE':
        self.strataRandomization=simpleRandomization()
    elif strataRandomization=='PERMUTED':
        self.strataRandomization=permutedBlocks()
    else:
        raise Exception(self._ERROR_MSG_STRATA_SUB_RANDOMIZATION)

# Sets the number of treatments
def setNumberOfTreatments(self, numberOfTreatments):
    if numberOfTreatments<2 or
numberOfTreatments>self._MAX_NUMBER_OF_TREATMENTS:
        raise Exception(self._ERROR_MSG_NUMBER_OF_TREATMENTS)
    else:
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
self.strata=[]
self.numberOfTreatments = numberOfTreatments

# Getters
# Gets the number of strata
def getNumberOfStrata(self):
    n=1
    for i in range(0, len(self.strata)):
        n=n*len(self.strata[i])
    return n

# Gets the maximum length of the combined strata
def getMaximumLengthOfCombinedGroups(self):
    maxLength=0
    for i in range(0, len(self.combineStrata(self.strata))):
        groupName=''
        for j in range(0, len(self.combineStrata(self.strata)[i])):
            groupName=groupName+self.combineStrata(self.strata)[i][j]+'/'
        groupName=groupName[:-1]
        if len(groupName)>maxLength:
            maxLength=len(groupName)
    return maxLength+1

def getName(self):
    return 'Stratified Randomization'

def getInternal(self):
    return 'STRATIFIED'

if __name__ == "__main__":
    try:
        x=stratifiedRandomization()
        x.setNumberOfSubjects(12)
        x.setNumberOfTreatments(2)
        x.setSizeOfBlocks(2)
        x.setStrata('male, female')
        x.setStrata('age<50, age>=50')
        #x.setStrata('1-3 +ve nodes, >= 4 +ve nodes')
        x.setRandomization('PERMUTED')
        x.allocateSubjects()
        print x.consoleOutput()
    except Exception as e:
        print e
```

## 8.6. Κλάση blockRandomization

```
"""
Author:      Theodoros Mprotsis
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

(MSc Research Methodology in Biomedicine, Biostatistics and Clinical Bioinformatics" at University of Thessaly)

Year: 2014/2015

Description: Develop a software in Python for performing different types of randomization (simple, stratified and block randomization)

"""

```
from randomization import randomization
```

```
import random
```

"""

*Block Randomization*

-----

*Block randomization restricts randomization to ensure approximately equal numbers*

*of patients on each treatment. In block randomization, we consider the patients*

*in blocks of a specific size and we assign the treatments to blocks randomly.*

"""

```
class blockRandomization(randomization):
```

```
    # Constructor
```

```
    def __init__(self):
```

```
        randomization.__init__(self)
```

```
        self.numberOfSubjectsInEachBlock=0
```

```
        self.treatmentGroups=[]
```

```
        self.groupOfDigis=[[ ]]
```

```
        self.permutedDigits=[0,1,2,3,4,5,6,7,8,9]
```

```
        self.numberOfBlocks=0
```

```
    # Creating groups
```

```
    def createGroups(self):
```

```
        self.groupOfDigis=[[ ]]
```

```
        # Calculating the possible numbers of subjects in each block and get the maximum of them
```

```
        self.numberOfSubjectsInEachBlock=self.numberOfBlocks
```

```
        # Calculating the numbers in each sub group e.g [1,2],[3,4],[5,6] when there are three
```

```
        # treatments
```

```
        numbersInEachTreatmentGroup=self.numberOfSubjectsInEachBlock/self.numberOfTreatments
```

```
        currentGroup=1
```

---

Πανεπιστήμιο Θεσσαλίας: Διπλωματική Εργασία





Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
numberInCurrentGroup=1
lbound=0

excludeDigits = self._NUMBER_OF_DIGITS % self.numberOfTreatments
ubound=self.numberOfSubjectsInEachBlock+excludeDigits

if (excludeDigits > 0):
    lbound=excludeDigits

for i in range(lbound, ubound):
    self.groupOfDigis[currentGroup-1].append(i)
    if numberInCurrentGroup==numbersInEachTreatmentGroup:
        numberInCurrentGroup=1
        self.numberOfGroups=currentGroup
        currentGroup=currentGroup+1
        if i<ubound-1:
            self.groupOfDigis.append([])
    else:
        numberInCurrentGroup=numberInCurrentGroup+1

return

# Allocate subjects (patients) to groups (treatments)
def allocateSubjects(self):

    self.randomNumber=[]
    self.allocation=[]
    # Creating the groups
    self.createGroups()

    currentTreatmentGroup=0
    nOfTreatmentGroups=self.numberOfTreatments

    nOfpermutedBlocks=self.numberOfSubjects/self.numberOfSubjectsInEachBlock

    for i in range(0, nOfpermutedBlocks):
        random.shuffle(self.permutedDigits)
        for j in self.permutedDigits:
            for y in range(currentTreatmentGroup, nOfTreatmentGroups):
                if j in self.groupOfDigis[y]:
                    self.randomNumber.append(j)
                    self.allocation.append(chr(y+65))

# Output in the console
def consoleOutput(self):
    output=self.getName()+'\n\n'
    output=output+'Number of Subjects: %d\n'%(self.numberOfSubjects)
    output=output+'Number of Treatments: %d\n'%(self.numberOfTreatments)
    output=output+'Group of digits: %s\n\n'%(str(self.groupOfDigis))
    blockNumber=0
    for i in range(0, len(self.allocation)):
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
        if i % self.numberOfSubjectsInEachBlock==0:
            blockNumber=blockNumber+1
            output=output+'Block {0:2d}: Subject {1:3d} is allocated to
treatment {2} (Random number: {3})\n'.format(blockNumber, i+1,
self.allocation[i], self.randomNumber[i])
        else:
            output=output+'                Subject {0:3d} is allocated to
treatment {1} (Random number: {2})\n'.format(i+1, self.allocation[i],
self.randomNumber[i])
            output=output+self.possibleImbalance()
        return output

# Getters
def getName(self):
    return 'Block Randomization'

def getInternal(self):
    return 'BLOCK'

def getAllocation(self):
    return self.allocation

# Gets the maximum number of blocks
# return: array
def getNumbersOfSubjectsInBlock(self, numberOfSubjects):
    subjectsInEachBlock=[]
    permutedDigits=[0,1,2,3,4,5,6,7,8,9]
    for i in [10,9,8,7,6,5,4,3,2,1]:
        if numberOfSubjects % i==0:
            if i % self.numberOfTreatments==0:
                self.numberOfSubjects=numberOfSubjects
                subjectsInEachBlock.append(i)
            else:
                permutedDigits.pop() # removes the last entry from the array
    permutedDigits
    if subjectsInEachBlock:
        return subjectsInEachBlock
    else:
        return self.getNumbersOfSubjectsInBlock(numberOfSubjects+1)

# Gets the maximum number of subjects in each block
def getMaxNumberOfSubjectsInBlock(self, numberOfSubjects):
    return self.getNumbersOfSubjectsInBlock(numberOfSubjects)[0]

# Setters
# Sets the maximum number of subjects in each block
def setSizeOfBlocks(self, numberOfBlocks):
    if (numberOfBlocks in
self.getNumbersOfSubjectsInBlock(self.numberOfSubjects)):
        self.numberOfBlocks=numberOfBlocks
    else:
```



Θεόδωρος Μπρότσης, 'Ανάπτυξη ενός λογισμικού σε γλώσσα προγραμματισμού Python για την εκτέλεση διαφόρων τύπων τυχαιοποίησης (Απλή, Στρωματοποιημένη και τετραγωνισμένη τυχαιοποίηση)'

```
        raise
Exception(self._ERROR_MSG_NUMBER_OF_BLOCKS+str(sorted(self.getNumbersOfSubjectsInBlock(self.numberOfSubjects))))
```

```
if __name__ == "__main__":
    try:
        x=blockRandomization()
        x.setNumberOfSubjects(10)
        x.setNumberOfTreatments(2)
        x.setSizeOfBlocks(2)
        x.allocateSubjects()
        print x.consoleOutput()
    except Exception as e:
        print e
```

## 8.7. Module setup

Αν υπάρχουν δικαιώματα διανομής του αρχείου msvcrl10.dll, τότε μπορεί να εκτελεστεί το παρακάτω module για την δημιουργία εκτελέσιμου αρχείου. Η δημιουργία του εκτελέσιμου αρχείου (python setup.py py2exe) γίνεται μέσω της γραμμής εντολών (command line) στον φάκελο που βρίσκονται τα αρχεία του κώδικα της Python [9].

```
from distutils.core import setup
from glob import glob
import py2exe
data_files = [("Microsoft.VC11.CRT", glob(r'C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\redist\x86\Microsoft.VC110.CRT\*..*'))]

setup(console=['guiRandomization.py'],
      data_files=data_files)
```