ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΥΠΟΛΟΓΙΣΤΙΚΕΣ ΜΕΘΟΔΟΙ ΣΤΗ ΣΤΑΤΙΣΤΙΚΗ

Μήτσης Αντώνης | ge19079 | anmitsis@hotmail.com

Ιούνιος 2023

Προπτυχιακός Φοιτητής

Πίνακας περιεχομένων

1 Άσκηση 1

α. Θέλουμε να εκτιμήσουμε, με την απλή μέθοδο Monte Carlo τον όγκο της μοναδιαίας σφαίρας σε d διαστάσεις με $B_d = \{(x_1, \ldots, x_d) : x_1^2 + \cdots + x_d^2 < 1\}$. Για να κατανοήσουμε το πρόβλημα μπορούμε να το ανάγουμε σε 2 μόνο διαστάσεις, όπου θέλουμε να υπολογίσουμε το εμβαδόν του μοναδιαίου κύκλου. Θεωρούμε επίσης το τετράγωνο με κέντρο το 0, που περιέχει όλα τα σημεία με συντεταγμένες στο [-1,1]. Το τετράγωνο αυτό περιέχει μέσα του τον μοναδιαίο κύκλοκαι έχει την ιδιότητα ότι είναι εύκολο να κάνουμε δειγματοληψία μέσα σε αυτό. Επιλέγοντας λοιπόν ένα τυχαίο σημείο μέσα στο τετράγωνο, το σημείο αυτό θα βρίσκεται μέσα στον κύκλο με πιθανότητα p και θα βρίσκεται εκτός με πιθανότητα 1-p. Η πιθανότητα p είναι ο λόγος του εμβαδού του κύκλου προς το εμβαδό του τετραγώνου, οπότε δημιουργώντας αρκετά σημεία και μετρώντας πόσα βρίσκονται εντός του κύκλου μπορούμε να εκτιμήσουμε την πιθανότητα και άρα τον λόγο των εμβαδών των δύο σχημάτων. Τελικά η εκτίμηση του του εμβαδού του κύκλου είναι 2 φορές επί την πιθανότητα p αφού το εμβαδόν του τετραγώνου είναι 2. Επιστρέφοντας στο αρχικό πρόβλημα αρκεί να δημιουργήσουμε αρκετά σημεία το καθένα με d-συντεταγμένες στο [-1,1], δηλαδή μέσα στον κύβο d-διαστάσεων και να ελέγξουμε πόσα από αυτά τα σημεία βρίσκονται εντός της μοναδιαίας σφαίρας σε d διαστάσεις.

$$\frac{n_{inside\_sphere}}{n_{total}} \approx \frac{V_{sphere}}{V_{cube}} \tag{1.1}$$

Στην αριστερή πλαυρά της σχέσης έχουμε το πλήθος των σημείων που βρίσκονται μέσα στη σφαίρα προς το συνολικό πλήθος σημείων και δεξιά έχουμε τους όγκους. Η σχέση τελικά γίνεται:

$$\hat{V}_{sphere} = 2 \cdot \frac{n_{inside\_sphere}}{n_{total}} \tag{1.2}$$

```
ask1a<- function(max_d,n=10^5) {

  real_vol_calc<-function(d){
    if (d == 1) {
      return(2)
      } else if (d == 2) {
      return(pi)
      } else {
      return((2 * pi / (d-2 + 2)) * real_vol_calc(d - 2))
      }
  }

  for(d in 2:max_d){

    real_volume=real_vol_calc(d)
    count_inside = 0

    for(i in 1:n) {
      point = runif(d, min = -1, max = 1)
      if(sum(point^2) <= 1) {
        count_inside = count_inside + 1
      }
```

```
    }

    if(count_inside==0){
      break
    }
  }

  estimated_volume <- 2^d * (count_inside / n)
  error=abs((real_volume-estimated_volume)/real_volume)
  print(paste("D: ",d))
  print(paste("Real Volume:", real_volume))
  print(paste("Estimated Volume:", estimated_volume))
  print(paste("The percentage of error is:", round(error,5)*
    ↪ 100,"%",sep = ""))

}
```

Φτιάχνουμε μια συνάρτηση που δέχεται τη διάσταση που θέλουμε να υπολογίσουμε καθώς και το πλήθος των σημείων που θα προσομοιώσουμε. Αρχικά φτιάχνουμε μια συνάρτηση που θα υπολογίζει τον πραγματικό όγκο της σφαίρας αφού γνωρίζουμε πως $V_1 = 2, V_2 = \pi, V_{d+2} = \frac{2\pi}{d+2} V_d$. Επειδή για μεγάλες διαστάσεις ο όγκος είναι πολύ μικρός αριθμός επιλέγουμε στη συνάρτηση μας να ξεκινάει από διάσταση 2 και να ανεβαίνει διάσταση εως ότου φτάσει στην τελική διάσταση, δηλαδή αυτή που δόθηκε σαν όρισμα, ή μέχρι την στιγμή πο για πρώτη φορά κανένα σημείο δεν θα βρεθεί εντός της σφαίρας. Οπότε φτιάχνουμε το for loop που θα τρέξει απο διάσταση 2 έως max_d και μέσα δημιουργούμε n σημεία d διαστάσεων από την ομοιόμορφη κατανομή στο [-1,1]. Έπειτα ελέγχουμε αν το κάθε σημείο βρίσκεται εντός της σφαίρας, δηλαδή το άθροισμα των τετραγώνων των συντεταγμένων του να είναι μικρότερο ή ίσο του 1 και ανάλογα μεγαλώνουμε κατά ένα την τιμή count_inside. Επίσης ελέγχουμε μετα από τα n σημεία αν το count_inside είναι 0 για να τερματίσουμε τη διαδικασία. Τελικά υπολογίζουμε την εκτίμηση του όγκου σύμφωνα με την εξίσωση (1.2) και τυπώνουμε την διάσταση του όγκου, τον πραγματικό όγκο, την εκτίμηση καθώς και το ποσοστό σφάλματος. Καλούμε τη συνάρτηση για διάσταση 6 και πλήθος σημείων 10^5 και παίρνουμε:

```
> ask1a(6,10^5)
[1] "D:  6"
[1] "Real Volume: 5.16771278004997"
[1] "Estimated Volume: 5.17696"
[1] "The percentage of error is:0.179%"
```

Αν τώρα επιλέξουμε 17 διαστάσεις παίρνουμε το παρακάτω αποτέλεσμα:

```
> ask1a(17,10^5)
[1] "D:  16"
[1] "Real Volume: 0.235330630358893"
[1] "Estimated Volume: 0"
[1] "The percentage of error is:100%"
```

Στις 16 διαστάσεις η διαδικασία σταμάτησε αφού κανένα σημείο δεν βρέθηκε εντός της σφαίρας.

β. Στο προηγούμενο ερώτημα παρατηρήσαμε ότι για μεγάλο αριθμό διαστάσεων, όπου ο όγκος είναι πολύ μικρός η μέθοδος αποτυγχάνει. Μια λύση σε αυτό το πρόβλημα είναι να δημιουργήσουμε μια μαρκοβιανή αλυσίδα η οποία, αντί να επιλέγει τυχαία σημεία, το κάθε σημείο εξαρτάται απο το προηγούμενο και η αναλλοίωτη κατανομή της αλυσίδας θα είναι παρόμοια με αυτή που αναζητούμε. Στην περίπτωσή μας μπορούμε να θεωρήσουμε για χώρο καταστάσεων της αλυσίδας τον κύλινδρο $C_d = B_{d-1} \times [-1, 1]$ και αναλλοίωτη κατανομή την ομοιόμορφη στον C_d. Συνεπώς η αλυσίδα αυτή θα μας βοηθήσει να επιλέγουμε σημεία στον χώρο του κυλίνδρου προσομοιώνοντας την ομοιόμορφη κατανομή. Για προτεινόμενη κατανομή θα πάρουμε εκείνη που αρχικά από τις πρώτες d-1 διαστάσεις επιλέγει μια τυχαία και τη μεταβάλλει κατά μια ομοιόμορφη τυχαία μεταβλητή στο (-ε,ε), ενώ για την τελευταία διάσταση την επιλέγει ομοιόμορφα στο (-1,1). Τελικά η αλυσίδα μας θα λειτουργεί με τον εξής τρόπο, θα επιλέγει τυχαίες αρχικές τιμές για κάθε διάσταση μέσα στον κύλινδρο, στο επόμενο βήμα μέσω της προτεινόμενης κατανομής θα έχουμε έναν νέο συνδυασμό συντεταγμένων για την αλυσίδα όπου στη συνέχεια θα ελέγχουμε αν πληρεί τις προυποθέσεις, δηλαδή αν βρίσκεται μέσα στην σφαίρα d- διαστάσεων. Σε περίπτωση που ισχύει αυτό θα μετράμε το πλήθος των επιτυχημένων θέσεων. Μετά από αρκετές επαναλήψεις θα έχουμε μια εκτίμηση του λόγου των όγκων του κυλίνδρου προς τη σφαίρα. Αυτό από το εργοδικό θεώρημα όπου για h(x) θεωρούμε την δείκτρια συνάρτηση της σφαίρας.

$$\frac{1}{T}\sum_{t=1}^{T} h(X^{(t)}) \to E_f[h(X)] \tag{1.3}$$

Τελικά έχουμε μια εκτίμηση για τον λόγο

$$\frac{V_d}{2 \times V_{d-1}} \tag{1.4}$$

Μποούμε έτσι επαγωγικά να εκτιμήσουμε τον όγκο V_d για $d = 3, 4, \ldots, d_{max}$ όπου d_{max} η διάσταση στην οποία το σφάλμα ξεπερνάει το 10%

```
ask1b <- function(d, nreps = 10000, c) {
  states <- matrix(0, nrow = nreps, ncol = d)
  states[1,] <- runif(d, -1, 1)
  accepted = 0

  dims <- 1:(d-1)

  for (i in 2:nreps) {
    prop_state <- states[i - 1, ]

    dim <- sample(dims, 1)
    prop_state[dim] <- prop_state[dim] + runif(1, -c, c)

    while (sum(prop_state[1:(d-1)]^2) > 1) {
      prop_state[dim] <- prop_state[dim] + runif(1, -c, c)
    }

    prop_state[d] <- runif(1, -1, 1)


```

```
    if (sum(prop_state^2) <= 1) {
      accepted <- accepted + 1
    }
    states[i, ] <- prop_state
  }

  list(states = states, acceptance_rate = accepted / nreps)
}
```

Η συνάρτηση που δημιουργήσαμε δέχεται ως ορίσματα τη διάσταση, τον αριθμό βημάτων που θα πραγματοποιήσει καθώς και την σταθερά ε, ή c στον κώδικα για την μεταβολή των καταστάσεων. Αρχικοποιούμε το διάνυσμα που θα αποθηκεύει τις καταστάσεις από τις οποίες περνάει η αλυσίδα και επιλέγει τυχαία τις αρχικές τιμές των συντεταγμένων στο [-1,1]. Προς το παρόν έχουμε 0 επιτυχημένες καταστάσεις και βάζουμε σε ένα διάνυσμα όλες τις διαστάσεις εκτός της τελευταίας. Ξεκινάμε τη διαδικασία με ένα for loop όσος και ο αριθμός βημάτων και αρχικά η προτεινόμενη κατάσταση είναι η προηγούμενη κατάσταση. Έπειτα μεταβάλλουμε μια τυχαία από τις πρώτες d-1 διαστάσεις με τον τρόπο που εξηγήσαμε και πρίν, φροντίζοντας ο χώρος καταστάσεων να είναι ο κύλινδρος. Δηλαδή κάθε κατάσταση που προτείνεται και είναι εκτός της σφαίρας διαστάσεων d-1 ξεχνιέται και προτείνεται νέα συντεταγμένη χωρίς να αποθηκευτεί η προηγούμενη. Για την τελευταία διάσταση επιλέγουμε ομοιόμορφα στο (-1,1). Στη συνέχεια ελέγχουμε αν η κατάσταση μετά τις μεταβολές βρίσκεται εντός της σφαίρας με d- διαστάσεις και αυξάνουμε κατα 1 τον μετρητή. Η κατάσταση αποθηκεύεται και συνεχίζουμε με το επόμενο βήμα. Τελικά επιστρέφουμε το σύνολο των καταστάσεων της μαρκοβιανής αλυσίδας καθώς και το σύνολο των σημείων που βρέθηκαν εντός της σφαίρας προς τα συνολικά σημεία που δημιουργήθηκαν.

Έχουμε λοιπόν την εκτίμηση για τον λόγο (1.4) και θα υπολογίσουμε επαγωγικά τον όγκο V_d με χρήση της παρακάτω συνάρτησης:

```
compute_vd <- function(d, nreps=10000, c=0.1){
  # Initial values
  v <- c(2, pi)

  # Compute v(d) for d >= 3
  for(i in 3:d){
    x <- ask1b_trace(i, nreps, c)$acceptance_rate
    v[i] <- 2 * x * v[i-1]

    # Calculate the percentage error
    real_vd = real_vol_calc(i)
    percentage_error = abs((v[i] - real_vd) / real_vd) * 100

    # If the percentage error is larger than 10%, stop the
      ↪ calculation
    if (percentage_error > 10){
      print(paste("Stopping at dimension", i, "due to high
        ↪ error percentage:", percentage_error))
      break
    }
  }
```

```
  result <- list(v = tail(v, n=1),real_vd=real_vd, percentage_error = percentage_error)
  class(result) <- "vd"
  return(result)
}

print.vd <- function(x) {
  print(paste("V = ", x$v,",Real volume = ",x$real_vd, ", Error = ", x$percentage_error, "%"))
}
```

Η παραπάνω επαναληπτική συνάρτηση ξεκινάει από 3 διαστάσεις και υπολογίζει με τον επαγωγικο τρόπο την εκτίμηση του όγκου καθώς και τον πραγματικό όγκο με τη συνάρτηση *real_vol_calc* του προηγούμενου ερωτήματος. Σε κάθε βήμα της η συνάρτηση ελέγχει το σφαλμα και την πρώτη φορά που υπερβαίνει το 10% τερματίζει. Στη συνέχεια θα πειραματιστούμε με διαφορετικές τιμές των n και ε, για να δούμε τι προτιμάμε και τι συμπεράσματα αντλούμε από τη μέθοδο.

Αρχικά θα μελετήσουμε την επίδραση του πλήθους βημάτων. Επιλέγουμε $d = 4$ και ε=0.1 και τρέχουμε για πλήθος βημάτων $10^2, 10^3, 10^4, 10^5$.

```
print(compute_vd(4,10^2,0.1))
[1] "Stopping at dimension 3 due to high error percentage: 49 "
[1] "V = 2.13628300444106 ,Real volume = 4.18879020478639 , Error = 49 %"
print(compute_vd(4,10^3,0.1))
[1] "Stopping at dimension 4 due to high error percentage: 12.3002787502778"
[1] "V = 4.32780777410284 ,Real volume = 4.93480220054468 , Error = 12.3002787502778 %"
print(compute_vd(4,10^4,0.1))
[1] "V = 4.65900741500601 ,Real volume = 4.93480220054468 , Error = 5.58877082263251 %"
print(compute_vd(4,10^5,0.1))
[1] "V = 4.88229785958542 ,Real volume = 4.93480220054468 , Error = 1.06396039447058 %"
```

Παρατηρούμε πως αυξάνοντας τον αριθμο των βημάτων η ακρίβεια αυξάνεται καθώς και η σταθερότητα, αφού για μικρά n οι τιμες της εκτίμησης έχουν τεράστιες διακυμάνσεις. Αυτό οφείλεται στο burn-in, συγκεκριμένα η μέθοδος αυτή χρειάζεται ένα συγκεκριμένο αριθμό βηματων ώστε να φτάσει στην αναλλοίωτη κατανομή και να έχει την εκτιμητική ικανότητα που χρειαζομαστε. Συνεπώς χρειαζόμαστε μεγάλο αριθμό βημάτων και τα 10^5 είναι μια καλή ισορροπία σε απόδοση και υπολογιστικό κοστος. Στη συνέχεια θα αλλάξουμε τις τιμές του ε.

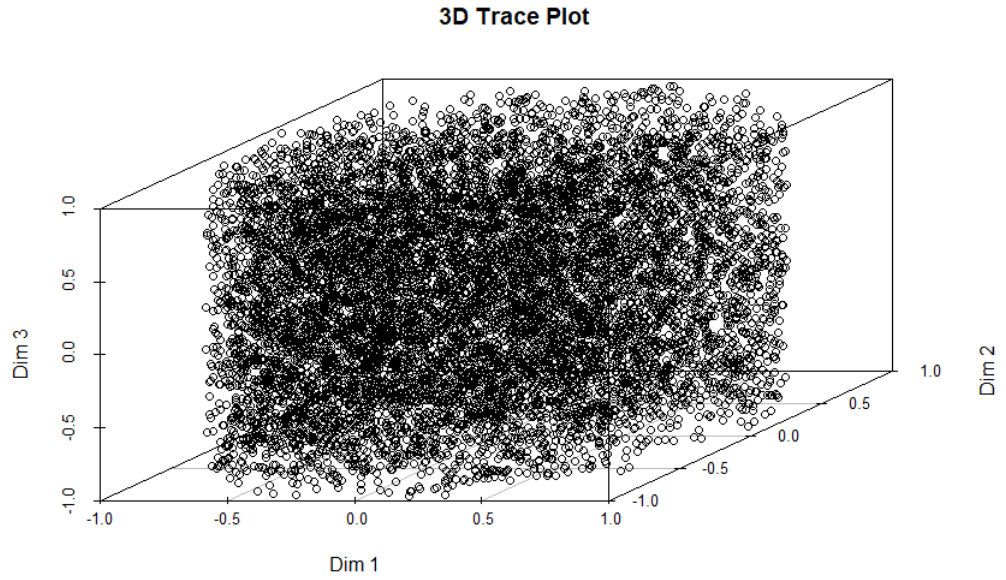
```
> print(compute_vd(4,10^5,0.001))
[1] "Stopping at dimension 3 due to high error percentage: 17.8"
```

```
[1] "V = 3.44318554833441 , Real volume = 4.18879020478639 , Error = 17.8 %"
> print(compute_vd(4,10^5,0.1))
[1] "V = 4.88229785958542 , Real volume = 4.93480220054468 , Error = 1.06396039447058 %"
> print(compute_vd(4,10^4,0.3))
[1] "V = 4.92568929855576 , Real volume = 4.93480220054468 , Error = 0.184666003186762 %"
> print(compute_vd(4,10^4,0.7))
[1] "V = 4.9180380124798 , Real volume = 4.93480220054468 , Error = 0.339713475507358 %"
```

Για τιμές του ε μεγαλύερες από 0.7 η συνάρτηση δεν ολοκληρωνεται καθώς κάθε σημείο, ή τα περισσότερα που προτείνονται βρίσκονται εκτός του κυλίνδρου και άρα απορρίπτονται, με αποτελεσμα να μην ολοκληρωνεται η διαδικασία. Για τις τιμές που δοκιμάσαμε ξεκινήσμαμε από πολύ μικρό ε, το οποίο έδωσε μεγάλο σφάλμα και ο λόγος είναι διότι δεν κινείται αρκετα γρήγορα η μαρκοβιανή αλυσίδα μέσα στον χώρο, με άλλα λόγια παραμένει πολύ κοντα στο σημείο εκκίνησης και δεν εξερευνά όλο το χώρο. Οι τιμές 0.1 και 0.3 είναι πολύ καλές, με την 0.3 να ξεχωρίζει ακόμα και σε 10^4 βήματα. Ιδανικά για να βρουμε τις βέλτιστες τιμές για τις παραμέτρους θα έπρεπε να έχουμε ένα πολύ πιο πυκνά διαμερισμένο διάστημα και να δοκιμάσουμε πολλές τιμές, όμως ο υπολογιστικός χρόνος είναι μεγάλος και μια σύντομη σύγκριση των τιμών αρκεί.

Τέλος, θα αναπαραστίσουμα γραφικά, σε 3 διαστάσεις για να είναι κατανοητό, κάθε κατάσταση της μαρκοβιανής αλυσίδας.

```
result <- ask1b(3, 10^5, 0.3)
scatterplot3d(result$states[, 1], result$states[, 2], result$states[, 3],
              main="3D Trace Plot", xlab="Dim 1", ylab="Dim 2", zlab="Dim 3")
```



Με χρήση του παραπάνω κώδικα κάνουμε τη γραφική αναπαράσταση των συντεταγμένων των σημείων της μαρκοβιανής αλυσίδας και όπως βλέπουμε τα σημεία βρίσκονται όλα εντός του κυλίνδρου, ο οποίος είναι ο χωρος καταστάσεων. Επαληθευουμε έτσι την διαδικασία που πραγματοποιήσαμε καθώς είναι φανερο πως το σύνολο των σημείων μέσα στη σφαίρα προς τα συνολικά σημεία, δηλαδή μέσα στον κύλινδρο, είναι μια εκτίμηση του λόγου των όγκων των δύο αυτών σχημάτων.

2 Άσκηση 2

Αρχικά η εκτιμημένη συνάρτηση πυκνότητας πιθανότητας για ένα τυχαίο δείγμα $x_1, x_2, ..., x_n$ με την μέθοδο του πυρήνα υπολογίζεται από τον τύπο:

$$\hat{f}(x) = \frac{1}{nh} K\left(\frac{x - x_i}{h}\right) \tag{2.1}$$

Ο όρος $\frac{x-x_i}{h}$ συμβολίζει την απόσταση του σημείου x από την παρατήρηση x_i τυποποιημένο ως προς τη σταθερά h την οποία θα υπολογίσουμε στη συνέχεια. Το μέγεθος του δείγματος συμβολίζεται με n ενώ η συνάρτηση του πυρήνα με K. Θα χρησιμοποιήσουμε τον κανονικό πυρήνα για τον οποίο ισχύει:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\left(\frac{x^2}{2}\right)} \tag{2.2}$$

Για να υπολογίσουμε το h θα μεγιστοποιήσουμε την cross-validated πιθανοφάνεια, η οποία δίνεται από τον τύπο:

$$L(h) = \prod_{i=1}^{n} \hat{f}_{-i}(x_i) \tag{2.3}$$

όπου με $\hat{f}_{-i}(x)$ συμβολίζουμε την εκτίμηση της πυκνότητας χωρίς την i- οστή παρατήρηση. Τελικά λογαριθμίζοντας την πιθανοφάνεια έχουμε να μεγιστοποιήσουμε ως προς h της εξής σχέση:

$$l(h) = \sum_{i=1}^{n} \ln\left(\sum_{j=1, j\neq i}^{n} K\left(\frac{x - x_i}{h}\right)\right) - n\ln\left((n-1)h\right) \tag{2.4}$$

Τα δεδομένα της άσκησης λαμβάνονται από την βιβλιοθήκη MASS την οποία θα κατεβάσουμε και θα φορτώσουμε στην R . Τα δεδομένα περιέχουν 133 προσομοιώσεις ατυχήματος μοτοσυκλέτας, όπου κάθε προσομοίωση παρέχει δύο τιμές, την επιτάχυνση κεφαλής σε g που θα είναι η μεταβλητή απόκρισης στο μοντέλο παλινδρόμισης και τον χρόνο σε χιλιοστά δευτερολέπτου από την στιγμή της κρούσης που θα είναι η επεξηγηματική μεταβλητή.

```
1
2 library(MASS) #load MASS package
3 data(mcycle)  #acces the mcycle data
4 y<-mcycle$accel #response variable y
5 x<-mcycle$times #predictor variable x
```

α. Αρχικά θα ορίσουμε την συνάρτηση kernel στην R και στη συνέχεια την συνάρτηση likelihood η οποία θα υπολογίζει την λογαριθμική πιθανοφάνεια ενός σημείου x για δοσμένο h.

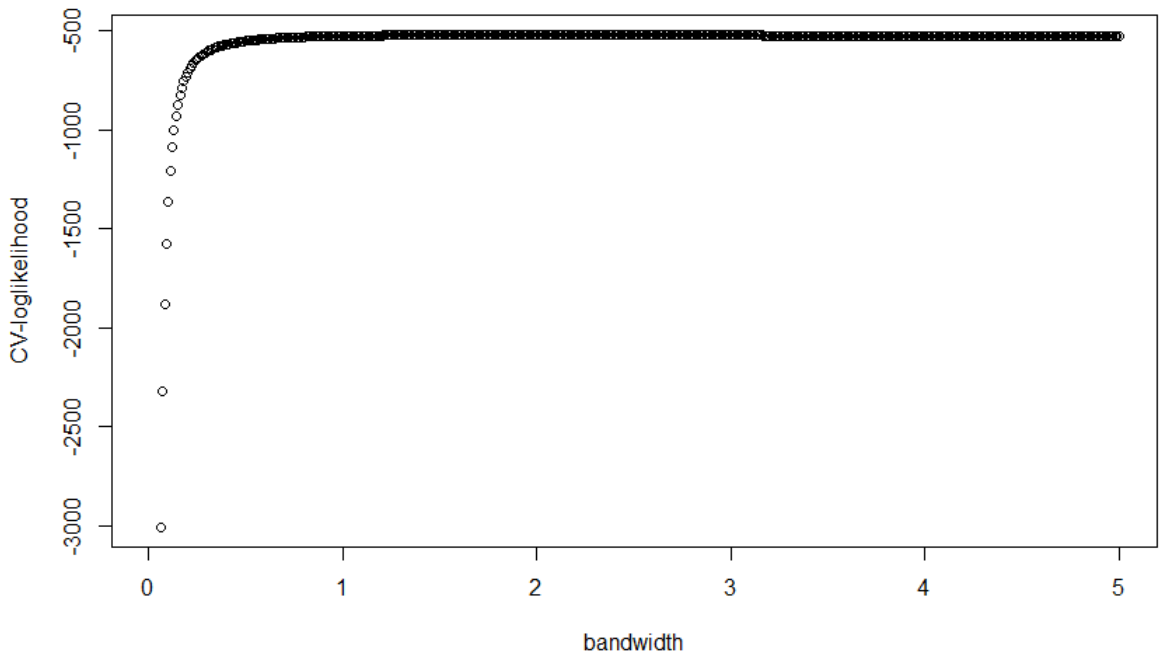
```
1
2 kernel_funct<-function(xi,xj,h){
3   x<-(xi-xj)/h
4   return(exp(-0.5*x^2)/sqrt(2*pi)) #Gaussian kernel
5 }
6
7 likelihood<-function(x,h){
```

```
  n<-length(x) #data size
  kernel_sum<-rep(0,n) #initiate vector

  #Iterate over each element of x
  for( i in 1:n){
    temp<-0
    # Compute the kernel sum for the current data point
    for(j in x[-i]){
      temp<-temp + kernel_funct(x[i],j,h)
    }
    kernel_sum[i]<-temp #store the value
  }

  return(sum(log(kernel_sum))-n*log((n-1)*h))
}
```

Η συνάρτηση kernel δέχεται ως ορίσματα τα x_i, x_j, h υπολογίζει τον όρο $\frac{x_i - x_j}{h}$ και επιστρέφει την τιμή του κανονικού πυρήνα K(x) . Η συνάρτηση likelihood δέχεται ως ορίσματα τα x, h, υπολογίζει το Kernel sum για κάθε i αφού θα χρειαστεί για την επιστροφή της τιμής της λογαριθμικής πιθανοφάνειας για τα x, h.

Στη συνέχεια για να υπολογίσουμε το βέλτιστο h θα διαμερίσουμε το διάστημα [0.01,10] με βήμα 0.01 και έτσι θα έχουμε 500 h για τα οποία θα υπολογίσουμε την τιμή της συνάρτησης likelihood με δεδομένα τις παρατηρήσεις της επεξηγηματικής μας μεταβλητής x. Το διάστημα διαμέρισης επιλέγεται από εμάς και καλό είναι να ξεκινήσουμε από ένα μεγάλο διάστημα με λίγο μεγαλύτερο βήμα και σταδιακά να συγκλινουμε σε μικρότερο διάστημα μειώνοντας το βήμα και άρα αυξάνοντας την ακρίβεια. Παρακάτω δίνεται ο κώδικας που υπολογίζει το βέλτιστο h καθώς και η γραφική αναπαράσταση της λογαριθμικής πιθανοφάνειας συναρτήσει του h.

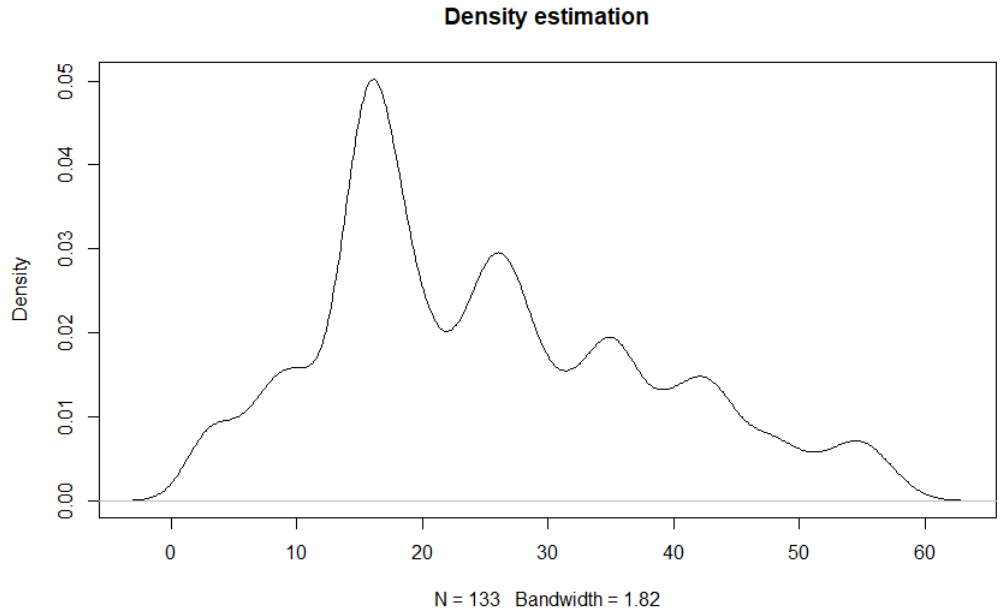
```
h<-seq(from = 0.01, to = 5, by = 0.01)
length(h)
likelihood_vec<-rep(0,length(h)) #initialise vetor
#compute likelihood for each h
for(k in 1:length(h)){
  likelihood_vec[k]<-likelihood(x,h[k])
}
plot(h,likelihood_vec,xlab = "bandwidth",ylab = "CV-
  ↪ loglikelihood")
h_opt<-h[which.max(likelihood_vec)]
```



Η εντολή which.max() μας δίνει το index του h για το οποίο μεγιστοποιείται η πιθανοφάνεια και έτσι παίρνουμε το βέλτιστο h. Στη συνέχεια για να αναπαραστήσουμε γραφικά την εκτιμώμενη $f(x)$ με το βέλτιστο h θα χρησιμοποιήσουμε την παρακάτω εντολή:

```
plot(density(x,kernel = "gaussian", bw=h_opt),main = "Density
    ↪ ␣estimation")
```

Η συνάρτηση density() της R κάνει αυτό ακριβώς που χρειαζόμαστε. Για ορίσματα βάζουμε τα δεδομένα x, το είδος του πυρήνα και το βέλτιστο h.



Η εικόνα είναι αρκετά ικανοποιητική καθώς δεν παρουσιάζεται πρόβλημα underfitting ή overfitting. Αυτό μπορούμε να το ελέγξουμε αν πειραματιστούμε με την τιμή του h όπου για μικρότερο h το γράφημα γίνεται πολύ απότομο και χάνει την προβλεπτική του ικανότητα για τιμές που δεν έχουν παρατηρηθεί, ενώ για μεγαλύτερο h το γράφημα γίνεται πολύ λείο και δεν παρέχει αρκετές πληροφορίες για την εκτίμηση. Επίσης στον άξονα x αναγράφεται και η τιμή του βέλτιστου h η οποία είναι $bandwidth = 1.82$.

β. Στη συνέχεια θα προσαρμόσουμε το μοντέλη μη παραμετρικής παλινδρόμισης Nadaraya-Watson με τις ίδιες μεταβλητές όπως και πριν. Το h θα υπολογιστεί με την μέθοδο του leave-one-out CV.

Ας υποθέσουμε πως έχουμε δύο τυχαίες μεταβλητές Y, X, τότε το απλό γραμμικό μοντέλο ορίζεται ώς εξής:

$$m(x) = E[Y|X = x] = \int yf(y|x)dy = \int y\frac{f(x,y)}{f_X(x)}dy \tag{2.5}$$

Όμως οι $f(x,y)$ και $f_X(x)$ είναι άγνωστες, οπότε θα χρησιμοποιήσουμε τις ακόλουθες προσεγγίσεις.

$$\hat{f}(x,y) = \frac{1}{nh_xh_y}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_x}\right)K_Y\left(\frac{y-y_i}{h_y}\right) \tag{2.6}$$

$$\hat{f_X}(x) = \frac{1}{nh_x}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_x}\right) \tag{2.7}$$

Εύκολα αποδυκνείεται πως το μοντέλο Nadaraya-Watson είναι το εξής:

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_x}\right)y_i}{\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_x}\right)} = \sum_{i=1}^{n} w_iy_i = \hat{m}_{NW}(x) \tag{2.8}$$

Για την εκτιμήτρια της διασποράς του εν λόγω μοντέλου ισχύει:

$$\hat{\sigma}^2(x) = \frac{\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_x}\right) e_i^2}{\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_x}\right)}, \quad e_i^2 = y_i - \hat{m}(x) \tag{2.9}$$

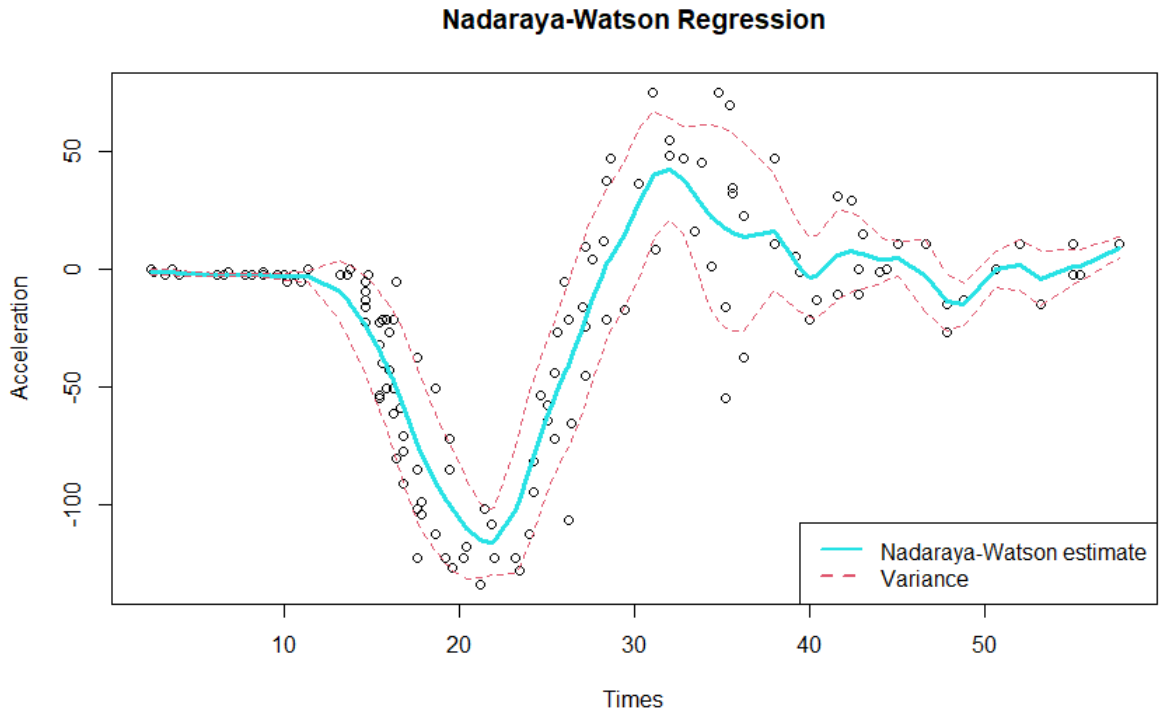
Για τον υπολογισμό του βέλτιστου h όπως αναφέρθηκε και πριν θα χρησιμοποιήσουμε τη μέθοδο leave-one-out CV.

```
h<-seq(1.01,10,0.01) #sequence of h to choose from
cv_mse<-rep(0,length(h)) #Initialise mse vector for each h
n<-length(x)

for( i in 1:length(h)){ #loopoing over every h
  total_mse<-0

  for(j in 1:n){    #leave-one-out CV
    x_train<-x[-j]
    y_train<-y[-j]
    x_test<-x[j]
    y_test<-y[j]

    w<-(1/sqrt(2*pi))*exp(-0.5*((x_train-x_test)/h[i])^2) #
      ↪ calculating the kernel
    estimate<-sum(w*y_train)/sum(w)

    mse<-(y_test-estimate)^2
    total_mse<-total_mse + mse #sum of mse for each
      ↪ observation missing
  }
  cv_mse[i]<-total_mse/n #average of the total sum for
    ↪ current h
}
min_mse<-which.min(cv_mse)
h_opt<-h[min_mse]
```

Αρχικά δημιουργούμε ένα διάνυσμα με h ξεκινώντας απο το 1.01 έω; το 10 με βήμα 0.01 όπως ζητείται στην άσκηση. Στη συνέχεια θα δημιουργήσουμε ενα κένο διάνυσμα μεγέθους ίσου με το μήκος των h για να αποθηκεύσει τις τιμές των MSE για κάθε h. Ξεκινάμε τη διαδικάσια με loop για κάθε h και δημιουργούμε εσωτερικό βρόγχο που αφαιρεί κάθε φορα μια διαφορετική παρατήρηση από το x και υπολογίζει το MSE για αυτά τα δεδομένα. Τέλος παίρνει το άθροισμα και βρίσκει τη μέση του τιμή αποθηκεύοντας στο στο διάνυσμα που θα περιέχει τα MSE για κάθε h. Οι τελευταίες δύο εντολές μας δίνουν το βέλτιστο h το οποίο είναι μάλιστα και πρώτο στο διάνυσμα και είναι το 1.01. Τώρα μπορώ να προσαρμόσω το τελικό μοντέλο και να κάνω τη γραφική αναπαράσταση.

```
pred_values<-rep(0,length(x)) #Initialize vectors for
  ↪ predicted values
variance<-rep(0,length(x))    #and variance

```

```
for( i in 1:length(x)){
  weights<-(1/sqrt(2*pi))*exp(-0.5*((x-x[i])/h_opt)^2)

  pred_values[i]<-sum(weights*y)/sum(weights)
  variance[i]<-sum(weights*(y-pred_values[i])^2)/sum(weights)
}
#plotting data points
plot(x,y,main="Nadaraya-Watson Regression",xlab = "Times",
  ↪ ylab = "Acceleration")
lines(x,pred_values,col=5,lw=3) #regression line on cyan
  ↪ colour
lines(x,pred_values+sqrt(variance),col=2,lty=2) #variance
  ↪ dotted lines on red
lines(x,pred_values-sqrt(variance),col=2,lty=2)
legend("bottomright",legend = c("Nadaraya-Watson estimate","
  ↪ Variance"),col=c(5,2),
lty = c(1,2),lwd = 2)
```



Στον άξονα x έχουμε την επεξηγηματική μεταβλητή, χρόνος σε χιλιοστά του δευτερολέπτου από την στιγμή της κρούσης και στον άξονα y την μεταβλητή απόκρισης, επιτάχυνση κεφαλής σε g. Για να δημιουργήσουμε αυτήν την είκόνα αρχικά δημιουργήσαμε δύο διανύσματα που θα αποθηκεύσουν τις εκτιμήσεις και τις διασπορές. Στη συνέχεια προσαρμόζουμε το μοντέλο μη παραμετρικής παλινδρόμισης Nadaraya-Watson με το βέλτιστο h που υπολογίσαμε και αποθηκεύουμε για κάθε παρατήρηση την εκτίμηση και την διασπορά της. Για την γραφική αναπαράσταση βάζουμε πρώτα τις παρατηρήσεις μας. Στη συνέχεια τις εκ-

τιμήσεις σε μορφή συνεχούς γραμμής και προσθέτουμε και αφαιρούμε μια διασπορά στις εκτιμήσεις. Το μοντέλο μας φαίνεται να προσαρμόζεται πολύ καλά στα δεδομένα.

3 Άσκηση 3

α. Έστω $x_1, x_2, \ldots, x_n$ ανεξάρτητες αυτόνομες τυχαίες μεταβλητές που ακολουθούν την κανονική κατανομή 2 διαστάσεων $N(m, \Sigma)$. Θέλουμε να υπολογίσουμε τις εκτιμήτριες μέγιστης πιθανοφάνειας των παραμέτρων, $\mathbf{m}$ και $\mathbf{\Sigma}$ της κανονικής κατανομής σε 2 διαστάσεις. Η συνάρτηση πυκνότητας πιθανότητας της διδιάστατης κανονικής κατανομής δίνεται από τον τύπο:

$$f_X(x_j; m, \Sigma) = (2\pi)^{-1}|\Sigma|^{-1/2} exp\left(-\frac{1}{2}(x_j - m)^T\Sigma^{-1}(x_j - m)\right) \tag{3.1}$$

Όπου m το 2×1 διάνυσμα των μέσων τιμών, Σ ο 2×2 πίνακας συνδιακύμανσης, $|\Sigma|$ η οριζουσα του και x^T συμβολιζουμε τον ανάστροφο πίνακα. Η συνάρτηση πιθανοφάνειας υπολογίζεται ως εξής:

$$\begin{aligned}
&L(m, \Sigma; x_1, x_2, \ldots, x_n) \\
&= \prod_{i=1}^{n} f_X(x_i; m, \Sigma) \\
&= \prod_{i=1}^{n} (2\pi)^{-1}|\Sigma|^{-1/2} exp\left(-\frac{1}{2}(x_j - m)^T\Sigma^{-1}(x_j - m)\right) \\
&= (2\pi)^{-n}|\Sigma|^{-n/2} exp\left(-\frac{1}{2}\sum_{i=1}^{n}(x_j - m)^T\Sigma^{-1}(x_j - m)\right)
\end{aligned} \tag{3.2}$$

Λογαριμίζοντας έχουμε την log-likelihood συνάρτηση:

$$l(m, \Sigma; x_1, x_2, \ldots, x_n) = -\frac{n}{2}\ln 2\pi + \frac{n}{2}\ln|\Sigma|^{-1} - \frac{1}{2}\sum_{i=1}^{n}(x_j - m)^T\Sigma^{-1}(x_j - m) \tag{3.3}$$

Πρωτού υπολογισουμε τις εκτιμήτριες υπενθυμίζουμε μερικά πράγματα.

- $\mathrm{tr(a)} = \mathrm{a}$, όταν a αριθμός
- $\mathrm{tr(AB)} = \mathrm{tr(BA)}$
- $\mathrm{tr(aA + bB)} = \mathrm{atr(A) + btr(B)}$
- $\nabla_A(\ln|A|) = (A^{-1})^T$
- $\nabla_A(tr(BA)) = B^T$
- $\nabla_x(x^TAx) = 2Ax$

Για να βρούμε τις εκτιμήτριες μέγιστης πιθανοφάνειας των m και Σ θα μηδενίσω τις παραγώγους ως προς κάθε παράμετρο και θα λύσω ως προς αυτές. Ξεκινάω για την $\hat{m}$.

$$
\begin{aligned}
&\nabla_m \left(l(m, \Sigma; x_1, x_2, \ldots, x_n)\right) \\
&= \nabla_m \left(-\frac{n}{2} \ln 2\pi + \frac{n}{2} \ln |\Sigma|^{-1} - \frac{1}{2} \sum_{i=1}^{n} (x_j - m)^T \Sigma^{-1} (x_j - m) \right) \\
&= \sum_{i=1}^{n} \Sigma^{-1} (x_j - m) \\
&= \Sigma^{-1} \sum_{i=1}^{n} (x_j - m) = 0
\end{aligned} \tag{3.4}
$$

Το οποίο ισχύει μόνο όταν

$$
\begin{aligned}
\sum_{i=1}^{n} x_j - nm = 0 \\
\hat{m} = \frac{1}{n} \sum_{i=1}^{n} x_j
\end{aligned} \tag{3.5}
$$

Ακολουθώ την ίδια διαδικασία και για το $\hat{\Sigma}$.

$$
\begin{aligned}
&\nabla_{\Sigma^{-1}} \left(l(m, \Sigma; x_1, x_2, \ldots, x_n)\right) \\
&= \nabla_{\Sigma^{-1}} \left(-\frac{n}{2} \ln 2\pi + \frac{n}{2} \ln |\Sigma|^{-1} - \frac{1}{2} \sum_{i=1}^{n} (x_j - m)^T \Sigma^{-1} (x_j - m) \right) \\
&= \frac{n}{2} \nabla_{\Sigma^{-1}} (\ln |\Sigma|^{-1}) - \frac{1}{2} \nabla_{\Sigma^{-1}} \left(\sum_{i=1}^{n} tr[(x_j - m)^T \Sigma^{-1} (x_j - m)] \right) \\
&= \frac{n}{2} \Sigma^T - \frac{1}{2} \nabla_{\Sigma^{-1}} \left(\sum_{i=1}^{n} tr[(x_j - m)(x_j - m)^T \Sigma^{-1}] \right) \\
&= \frac{n}{2} \Sigma^T - \frac{1}{2} \nabla_{\Sigma^{-1}} \left(tr \left[\sum_{i=1}^{n} (x_j - m)(x_j - m)^T \Sigma^{-1} \right] \right) \\
&= \frac{n}{2} \Sigma^T - \frac{1}{2} \left(\sum_{i=1}^{n} (x_j - m)(x_j - m)^T \right)^T \\
&= \frac{n}{2} \Sigma - \frac{1}{2} \left(\sum_{i=1}^{n} (x_j - m)(x_j - m)^T \right) = 0
\end{aligned}
$$

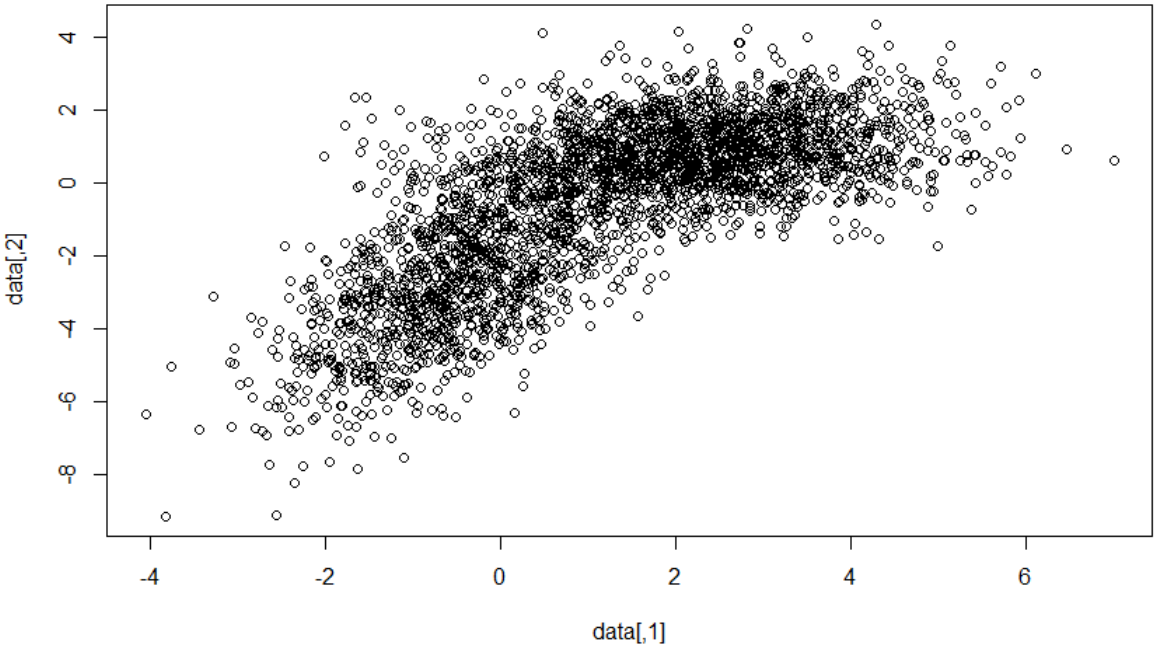
Οπότε τελικά έχω:

$$
\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_j - \hat{m})(x_j - \hat{m})^T \tag{3.6}
$$

Αφού βρήκαμε αναλυτικά τις εκτιμήτριες μέγιστης πιθανοφάνειας, θα προσαρμόσουμε στα δεδομένα μας ένα μοντέλο κανονικής κατανομης δύο διαστάσεων. Αρχικα φορτώνουμε τα δεδομένα καθώς και την βιβλιοθήκη MASS στην R.

```
library("MASS")
data<-unname(as.matrix(read.csv2("datafile.csv")))
```

Τα δεδομένα μας περιέχουν συντεταγμένες 3000 σημείων τα οποία απεικονείζονται στο παρακάτω σχήμα.




```
means<-c(mean(data[,1]),mean(data[,2]))
s11<-sum((data[,1]-means[1])^2)/n
s22<-sum((data[,2]-means[2])^2)/n
s12<-sum((data[,1]-means[1])*(data[,2]-means[2]))/n
cov_matrix<-matrix(c(s11,s12,s12,s22),nrow = 2)
```

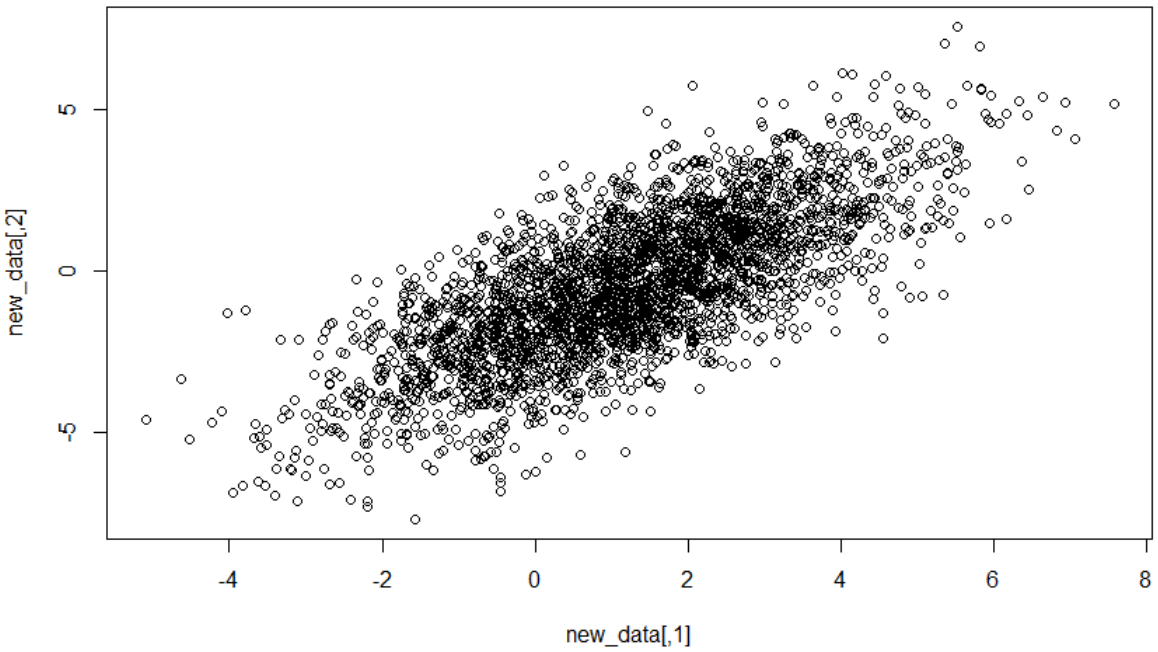
Με τον παραπάνω κώδικα αρχικά δημιουργούμε το διάνυσμα που περιέχει τους μέσους των στηλών των παρατηρήσεων και στη συνέχεια υπολογιζουμε τη διασπορά κάθε στήλης καθώς και τη συνδιασπορά τους. Τελικά τα βάζουμε σε έναν πίνακα και έχουμε τον πίνακα συνδιακύμανσης.

```
 > means
[1]   1.1797288 -0.5346472
> cov_matrix
          [,1]     [,2]
[1,] 3.412038 3.068844
[2,] 3.068844 5.054024
```

Οπότε τα δεδομένα μας ακολουθούν διδιάσταση κανονική κατανομή με διάνυσμα μέσων το means και πίνακα συνδιακύμανσης τον cov_matrix.

β. Στη συνέχεια θέλουμε να προσομοιώσουμε 3000 νέες παρατηρήσεις από την κατανομή που εκτιμήσαμε στο προηγούμενο ερώτημα. Θα χρησιμοποιήσουμε την εντολή mvrnorm η οποία κάνει την ίδια δουλειά με την rnorm αλλά για πολυδιάστατη κανονική κατανομή. Για ορίσματα θα βάλουμε το μέγεθος του δείγματος που θέλουμε, μετά το διάνυσμα των μέσων και τέλος τον πίνακα συνδιακύμανσης.

```
n<-nrow(data)
new_data <- mvrnorm(n = n, mu = means, Sigma = cov_matrix)
plot(new_data)
```



Οι εικόνες είναι παρόμοιες και υπάρχει σίγουρα μια συχέτιση αλλά όπως βλέπουμε η καμπύλη που υπήρχε στην πρώτη γραφική δεν υπάρχει εδώ κάτι που μας οδηγεί στο να πιστέψουμε πως τα δεδομένα δεν ακολουθούν κανονική κατανομή.

γ. Θέλουμε να προσαρμόσουμε τωρα στα δεδομένα μας ένα μοντέλο μείξης κανονικών κατανομών

$$f_x(x) = pf_1(x) + (1-p)f_2(x) \tag{3.7}$$

Όπου f_1, f_2 είναι συναρτήσεις πυκνότητας πιθανότητας κανονικών κατανομών δύο διαστάσεων $\mathrm{N}(m_1,\Sigma_1)$ και $\mathrm{N}(m_2,\Sigma_2)$ και $p \in (0,1)$. Επειδή δεν μπορούμε να υπολογίσουμε αναλυτικά τις εκτιμήτριες μέγιστης πιθανοφάνειας των παραμέτρων με τις παρατηρήσεις που έχουμε, θα θεωρήσουμε την λανθάνουσα μεταβλητή $Z \sim Bernulli(p)$, η οποία κωδικοποιει την κατανομή μείξης από την οποία προέρχεται η κάθε παρατήρηση. Για να εκτιμήσουμε τις παραμέτρους θα υλοποιήσουμε την μέθοδο Expectation-Maximization. Για την Z ισχύει

ότι

$$z = \begin{cases} 1, \text{ αν η παρατήρηση x ανήκει στην κατανομή } f_1 \\ 0, \text{αλλιώς} \end{cases} \tag{3.8}$$

με $P[z=1] = p$ και $P[z=2] = 1-p$. Επίσης θα ισχύει $(X|z=i) \sim N(m_i, \Sigma_i), i = 1, 2$, οπότε η από κοινού συνάρτηση πυκνότητας πιθανότητας των X,Z είναι

$$f(X, Z) = [f(X, z=1)P(z=1)]^z [f(X, z=2)P(z=2)]^{1-z} \tag{3.9}$$

Θέλουμε να υπολογίσουμε την πιθανοφάνεια αυτής της κατανομής.

$$\begin{aligned} & L(p, m_1, m_2, \Sigma_1, \Sigma_2; x_1, x_2, \dots x_n) \\ & = \prod_{i=1}^{n} f(x_i, z_i) \\ & = \prod_{i=1}^{n} [f(x_i, z_i=1)P(z_i=1)]^{z_i} [f(x_i, z_i=2)P(z_i=2)]^{1-z_i} \\ & = \prod_{i=1}^{n} [pN(x_i; m_1, \Sigma_1)]^{z_i} [(1-p)N(x_i; m_2, \Sigma_2)]^{1-z_i} \end{aligned} \tag{3.10}$$

Παίρνω τον λογάριθμο της πιθανοφάνειας.

$$\begin{aligned} & l(p, m_1, m_2, \Sigma_1, \Sigma_2; x_1, x_2, \dots x_n) \\ & = \sum_{i=1}^{n} z_i \ln N(x_i; m_1, \Sigma_1) + \sum_{i=1}^{n} (1-z_i) \ln N(x_i; m_2, \Sigma_2) + \ln p \sum_{i=1}^{n} z_i + \ln (1-p) \sum_{i=1}^{n} (1-z_i) \end{aligned} \tag{3.11}$$

Επειδή δεν γνωρίζω τα z_i θα τα αντικαταστήσω με την τιμή $E[z_i|x_i, \theta^{(0)}]$, όπου $\theta^{(0)}$ είναι οι αρχικές τιμές των παραμέτρων που επιλέγουμε. Από το θεώρημα του Bayes έχουμε

$$\begin{aligned} & E[z_i|x_i, \theta^{(0)}] = P[z_i = 1|x_i, \theta^{(0)}] \\ & = \frac{P[z_i=1] f(X_i, z_i=1)}{\sum_{j=1}^{2} P[z_i=j] f(X_i, z_i=j)} \\ & = \frac{p^{(0)} N(x_i; m_1^{(0)}, \Sigma_1^{(0)})}{p^{(0)} N(x_i; m_1^{(0)}, \Sigma_1^{(0)}) + (1-p^{(0)}) N(x_i; m_2^{(0)}, \Sigma_2^{(0)})} = \gamma_i \end{aligned} \tag{3.12}$$

Τον όρο γ_i τον ονομάζουμε responsibility και το αντικαθιστούμε στη σχέση (3.11)

$$\begin{aligned} & Q(\theta|X, \theta^{(0)}) \\ & = \sum_{i=1}^{n} \gamma_i \ln N(x_i; m_1, \Sigma_1) + \sum_{i=1}^{n} (1-\gamma_i) \ln N(x_i; m_2, \Sigma_2) + \ln p \sum_{i=1}^{n} \gamma_i + \ln (1-p) \sum_{i=1}^{n} (1-\gamma_i) \end{aligned} \tag{3.13}$$

Θα βρούμε τώρα τις παραμέτρους που μεγιστοποιούν την ποσότητα Q παραγωγίζοντας και μηδενίζοντας ως προς κάθε παράμετρο. Οι πράξεις είναι παρόμοιες με του ερωτήματος α και

τελικά καταλήγουμε:

$$\hat{m_1} = \frac{\sum_{i=1}^{n} \gamma_i x_i}{\sum_{i=1}^{n} \gamma_i} \tag{3.14}$$

$$\hat{m_2} = \frac{\sum_{i=1}^{n} (1-\gamma_i) x_i}{\sum_{i=1}^{n} \gamma_i} \tag{3.15}$$

$$\hat{\Sigma_1} = \frac{\sum_{i=1}^{n} \gamma_i (x_i - \hat{m_1})(x_i - \hat{m_1})^T}{\sum_{i=1}^{n} \gamma_i} \tag{3.16}$$

$$\hat{\Sigma_2} = \frac{\sum_{i=1}^{n} (1-\gamma_i) (x_i - \hat{m_2})(x_i - \hat{m_2})^T}{\sum_{i=1}^{n} \gamma_i} \tag{3.17}$$

$$\hat{p} = \frac{\sum_{i=1}^{n} \gamma_i}{n} \tag{3.18}$$

Το μόνο που έχει αλλάξει είναι ότι αθροίζουμε κάθε παρατηρηση επί την ευθύνη (responsibility) της προς το συνολικό άθροισμα των γ_i. Για την εκτιμήτρια του p παίρνουμε την μέση τιμή των γ_i. Οπότε ξεκινάμε τον αλγοριθμό μας ορίζοντας τις αρχικές τιμές, στη συνέχεια υπολογίζουμε τα γ_i βάση των αρχικών μας παραμέτρων και μετά ανανεώνουμε τις παραμέτρους βάση των σχέσεων (3.14) έως (3.18) για να μεγιστοποιήσουμε την πιθανοφάνεια. Παρακάτων δίνεται ο κώδικας που υλοποιεί αυτή τη μέθοδο.

```
library(mvtnorm)
EM <- function(data, tolerance = 1e-6){

  n <- nrow(data)
  d <- ncol(data)
  m1<-c(-1,-2) #initial values
  m2<-c(1.5,1)
  S1<-matrix(c(0.7,1,1,3),nrow = 2)
  S2<-matrix(c(1.8,0.5,0.5,1),nrow = 2)
  lambda<-0.5

  converged <- FALSE
  old.params <- c(lambda, m1, as.vector(S1), m2, as.vector(S2
    ↪ ))
  params.history <- list()

  while (!converged){

    params.history[[length(params.history) + 1]] <- list("
      ↪ lambda" = lambda, "m1" = m1, "S1" = S1, "m2" = m2,
      ↪ "S2" = S2) #for error calculation
    # E-step: calculate the responsibility
    prob1 <- lambda * dmvnorm(data, mean = m1, sigma = S1,log
      ↪  = TRUE)  #log likelihoods
    prob2 <- (1 - lambda) * dmvnorm(data, mean = m2, sigma =
      ↪ S2,log = TRUE)
    tau1 <- prob1 / (prob1 + prob2) #responsibilities
```

```
    tau2 <- 1-tau1

    # M-Step: update the parameters to maximize the expected
      ↪ log-likelihood
    lambda <- sum(tau1)/n
    m1 <- colSums(tau1 * data) / sum(tau1)
    m2 <- colSums(tau2 * data) / sum(tau2)
    S1 <- t(tau1 * (data - m1)) %*% (data - m1) / sum(tau1)
    S2 <- t(tau2 * (data - m2)) %*% (data - m2) / sum(tau2)
    new.params <- c(lambda, m1, as.vector(S1), m2, as.vector(
      ↪ S2))
    param.diff <- sum(abs(new.params - old.params))
    if (param.diff < tolerance){
      converged <- TRUE
    }
    old.params <- new.params
  }


  return(list("final" = list("lambda" = lambda, "m1" = m1, "
    ↪ S1" = S1, "m2" = m2, "S2" = S2), "history" = params.
    ↪ history))
}

results<-EM(data)
results$final
results$history
length(results$history)
```

Για αρχικές τιμές έχουμε πολλές επιλογές, μπορούμε να ξεκινήσουμε με τους μέσους κάθε στηλης συν μιας τυχαίας μεταβλητής, μπορουμε να εφαρμόσουμε την μέθοδο k-clusters ή ακόμα και να ξεκινήσουμε από το 0. Ένας πιο αντιαισθητικός τρόπος και αναξιόπιστος, αλλα χρήσιμος στην περίπτωσή μας είναι να παρατηρήσουμε τη γραφική παράσταση των σημείων. Με λίγο πειραματισμό για τα νούμερα μπορούμε να έχουμε μια αρχική εκτίμηση των παραμέτρων δημιουργώντας νέα σημεία και βλέποντας αν το νέο σχήμα μοιάζει σε αυτό που θέλουμε. Η μέθοδος θα εξηγηθεί καλύτερα στο ερωτημα δ αλλά επιλέξαμε για αρχικές τιμές των παραμέτρων αυτές που φαίνονται στον κώδικα. Αφού επιλέξαμε αρχικές τιμές ξεκινάμε ένα while loop που θα τρέχει εως ότου το άθροισμα των σφαλμάτων κάθε παραμέτρου είναι μικρότερο απο 10^{-6}, δημιουργούμε μια λίστα με τις παραμέτρους του μοντέλου ώστε να τις αποθηκεύουμε σε κάθε βήμα. Στη συνέχεια για το E-step της μεθόδου υπολογίζουμε τα responsibilities με χρήση της εξίσωσης (3.12), για την συνάρτηση πυκνότητας πιθανότητας διδιάστατης κανονικής κατανομής χρησιμοποιούμε τη βιβλιοθήκη mvtnorm που περιέχει την εντολή dmvnorm η οποία όταν δέχεται το log = TRUE επιστρέφει τις πιθανοφάνειες για τα δεδομένα. Για το M-step υπολογίζουμε τις νέες τιμές των παραμέτρων. Αποθηκεύουμε τις νέες τιμές των παραμέτρων στη λίστα μας και υπολογίζουμε το άθροισμα των απολύτων διαφορών των παραμέτρων και ελέγχουμε αν είναι μικρότερο του 10^{-6} σε περίπτωση που είναι, τερματίζουμε τη διαδικασία και επιστρέφουμε τη λίστα με τις παραμέτρους σε κάθε βήμα και τις τελευταίες παραμέτρους που υπολόγισε πριν τερμασίσει, αλλιώς συνεχίζει η

διαδικασία. Έπειτα καλούμε τη συνάρτηση και παίρνουμε τα αποτελέσματα:

```
> results$final
$lambda
[1] 0.5723955

$m1
[1]  1.1797288 -0.5346472

$S1
          [,1]     [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624679

$m2
[1]  1.1797288 -0.5346472

$S2
          [,1]     [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624679

#------------------------------
> results$history
[[1]]
[[1]]$lambda
[1] 0.5

[[1]]$m1
[1] -1 -2

[[1]]$S1
     [,1] [,2]
[1,]  0.7    1
[2,]  1.0    3

[[1]]$m2
[1] 1.5 1.0

[[1]]$S2
     [,1] [,2]
[1,]  1.8  0.5
[2,]  0.5  1.0


[[2]]
[[2]]$lambda
[1] 0.5798949

```

```
[[2]]$m1
[1] 1.919493 0.302922

[[2]]$S1
          [,1]     [,2]
[1,] 4.011192 1.768793
[2,] 1.768793 4.323695

[[2]]$m2
[1]  0.1585909 -1.6907915

[[2]]$S2
          [,1]     [,2]
[1,] 4.419030 2.990689
[2,] 2.990689 7.478789


[[3]]
[[3]]$lambda
[1] 0.5706126

[[3]]$m1
[1]  1.0064879 -0.7471182

[[3]]$S1
          [,1]     [,2]
[1,] 5.043982 3.331804
[2,] 3.331804 7.074240

[[3]]$m2
[1]  1.4099485 -0.2522946

[[3]]$S2
          [,1]     [,2]
[1,] 4.729293 2.799306
[2,] 2.799306 5.888067


[[4]]
[[4]]$lambda
[1] 0.5730394

[[4]]$m1
[1]  1.2172759 -0.4850478

[[4]]$S1
          [,1]     [,2]
[1,] 4.904907 3.090795
```

```
[2,] 3.090795 6.486403

[[4]]$m2
[1]  1.1293355 -0.6012163

[[4]]$S2
         [,1]     [,2]
[1,] 5.003782 3.230759
[2,] 3.230759 6.802684


[[5]]
[[5]]$lambda
[1] 0.5722337

[[5]]$m1
[1]  1.171511 -0.545906

[[5]]$S1
         [,1]     [,2]
[1,] 4.960235 3.167631
[2,] 3.167631 6.657321

[[5]]$m2
[1]  1.1907214 -0.5195859

[[5]]$S2
         [,1]     [,2]
[1,] 4.933730 3.133374
[2,] 3.133374 6.580629


[[6]]
[[6]]$lambda
[1] 0.5724354

[[6]]$m1
[1]  1.1815745 -0.5320576

[[6]]$S1
         [,1]     [,2]
[1,] 4.946146 3.149607
[2,] 3.149607 6.616890

[[6]]$m2
[1]  1.1772577 -0.5381141

[[6]]$S2
```

```
          [,1]     [,2]
[1,] 4.952774 3.157768
[2,] 3.157768 6.635089


[[7]]
[[7]]$lambda
[1] 0.572386

[[7]]$m1
[1]  1.1793096 -0.5352435

[[7]]$S1
          [,1]     [,2]
[1,] 4.949668 3.153922
[2,] 3.153922 6.626501

[[7]]$m2
[1]  1.180290 -0.533849

[[7]]$S2
          [,1]     [,2]
[1,] 4.948069 3.152006
[2,] 3.152006 6.622240


[[8]]
[[8]]$lambda
[1] 0.5723977

[[8]]$m1
[1]  1.1798248 -0.5345096

[[8]]$S1
          [,1]     [,2]
[1,] 4.948823 3.152912
[2,] 3.152912 6.624255

[[8]]$m2
[1]  1.1796004 -0.5348313

[[8]]$S2
          [,1]     [,2]
[1,] 4.949201 3.153359
[2,] 3.153359 6.625247


[[9]]
```

```
[[9]]$lambda
[1] 0.572395

[[9]]$m1
[1]  1.1797068 -0.5346789

[[9]]$S1
         [,1]     [,2]
[1,] 4.949022 3.153147
[2,] 3.153147 6.624778

[[9]]$m2
[1]  1.1797583 -0.5346047

[[9]]$S2
         [,1]     [,2]
[1,] 4.948934 3.153044
[2,] 3.153044 6.624548


[[10]]
[[10]]$lambda
[1] 0.5723956

[[10]]$m1
[1]  1.1797339 -0.5346398

[[10]]$S1
         [,1]     [,2]
[1,] 4.948976 3.153093
[2,] 3.153093 6.624657

[[10]]$m2
[1]  1.179722 -0.534657

[[10]]$S2
         [,1]     [,2]
[1,] 4.948996 3.153117
[2,] 3.153117 6.624710


[[11]]
[[11]]$lambda
[1] 0.5723955

[[11]]$m1
[1]  1.1797276 -0.5346489

```

```
[[11]]$S1
          [,1]      [,2]
[1,] 4.948986 3.153105
[2,] 3.153105 6.624685

[[11]]$m2
[1]  1.1797304 -0.5346449

[[11]]$S2
          [,1]      [,2]
[1,] 4.948982 3.153100
[2,] 3.153100 6.624672


[[12]]
[[12]]$lambda
[1] 0.5723955

[[12]]$m1
[1]  1.1797291 -0.5346468

[[12]]$S1
          [,1]      [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624678

[[12]]$m2
[1]  1.1797285 -0.5346477

[[12]]$S2
          [,1]      [,2]
[1,] 4.948985 3.153104
[2,] 3.153104 6.624681


[[13]]
[[13]]$lambda
[1] 0.5723955

[[13]]$m1
[1]  1.1797288 -0.5346473

[[13]]$S1
          [,1]      [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624680

[[13]]$m2
```

```
[1]   1.179729 -0.534647

[[13]]$S2
          [,1]      [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624679


[[14]]
[[14]]$lambda
[1] 0.5723955

[[14]]$m1
[1]   1.1797288 -0.5346471

[[14]]$S1
          [,1]      [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624679

[[14]]$m2
[1]   1.1797288 -0.5346472

[[14]]$S2
          [,1]      [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624680

#-----------------------------
> length(results$history)
[1] 14
```

Παρατηρούμε πως η μέθοδος κάνει τις παραμέτρους να συγκλίνουν προς τις τιμές του ερωτήματος α σαν να έχουμε ένα μοντέλο διδιάστατης κανονικής κατανομής. Μια άλλη εναλλακτική είναι στη συνάρτηση dmvnorm να μην επιλέξουμε log = TRUE και έτσι λαμβάνουμε τα παρακάτω τελικά αποτελέσματα.

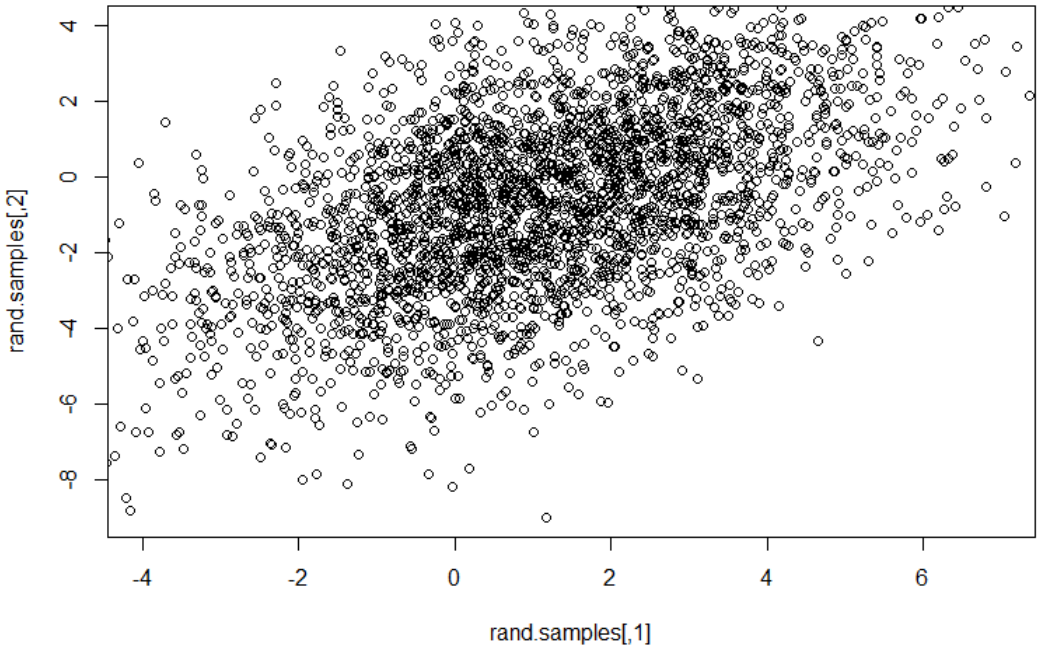
```
> results$final
$lambda
[1] 4.072829e-08

$m1
[1]   0.4746379 -1.6557931

$S1
          [,1]       [,2]
[1,] 6.244159  4.538616
[2,] 4.538616 10.394352

```

```
$m2
[1]   1.1797288 -0.5346471

$S2
          [,1]      [,2]
[1,] 4.948984 3.153103
[2,] 3.153103 6.624679
```

Εδώ παρατηρούμε πως διαφοροποιούνται οι παράμετροι των κανονικών κατανομών αλλά η παράμετρος μείξης μηδενιζεται και πάλι προσομοιώνει μια κατανομή αντί για μείξη.

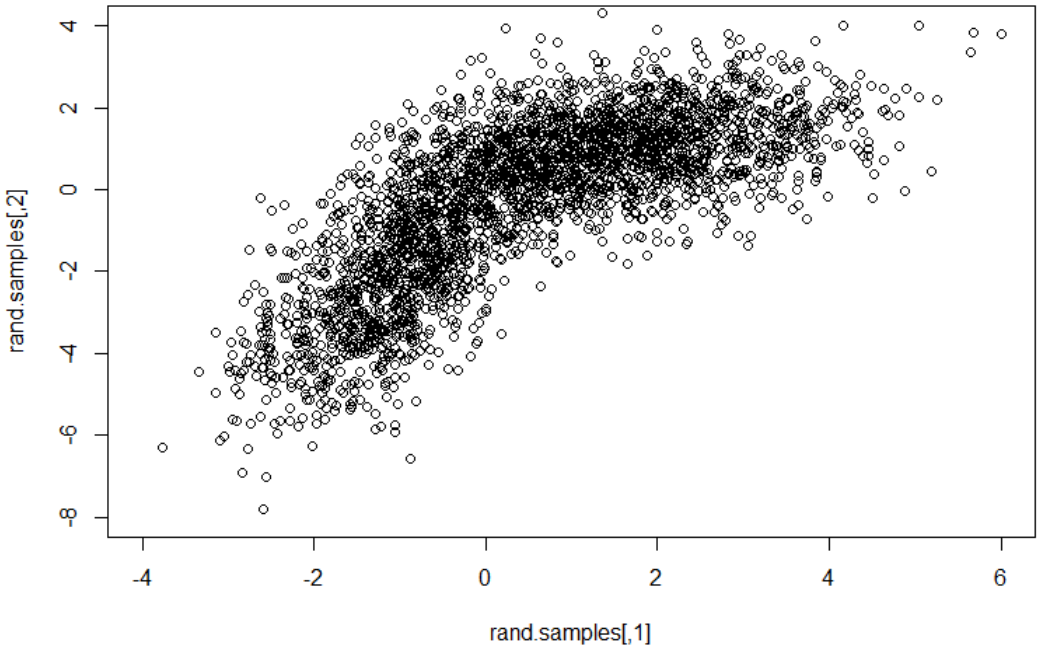
δ. Για να προσομοιώσουμε σημεία από μείξη κανονικών κατανομών θα δημιουργήσουμε 3000 τυχαίους αριθμούς μεταξύ 0 και 1 και όταν είναι μικρότερος του λ θα προσομοιώνει σημείο από την πρώτη κατανομή και αλλίως από τη δεύτερη. Τελικά θα έχουμε 3000 νέα σημεία και θα τα αναπαραστίσουμε γραφικά.

```
N=3000
U =runif(N)
rand.samples = matrix(NA,nrow = N,ncol=2)

for(i in 1:N){
  if(U[i]<0.5){
    rand.samples[i,] = mvrnorm(1,results$final[[2]],results$
      ↪ final[[3]])#results$final[[3]]
  }else{
    rand.samples[i,] = mvrnorm(1,results$final[[4]],results$
      ↪ final[[5]])#results$final[[5]]
  }
}
plot(rand.samples,xlim = c(-4,7),ylim = c(-9,4))
```



Η εικόνα που δημιουργείται δεν είναι ικανοποιητική για τα δεδομένα και μπορούμε να δούμε πως για τις αρχικες τιμές που εκτιμήσαμε η εικόνα είναι πολύ πιο κοντά στα δεδομένα.

```
N=3000
U =runif(N)
rand.samples = matrix(NA,nrow = N,ncol=2)
l1<-c(-1,-2)
l2<-c(1.5,1)
sigma1<-matrix(c(0.7,1,1,3),nrow = 2)
sigma2<-matrix(c(1.8,0.5,0.5,1),nrow = 2)
for(i in 1:N){
  if(U[i]<0.4){
    rand.samples[i,] = mvrnorm(1,l1,sigma1)
  }else{
    rand.samples[i,] = mvrnorm(1,l2,sigma2)
  }
}
plot(rand.samples,xlim = c(-4,6),ylim = c(-8,4))
```


rand.samples[,2]
4
2
0
-2
-4
-6
-8
-4
-2
0
2
4
6
rand.samples[,1]

4 Άσκηση 4

Στην βιβλιοθήκη faraway θα βρούμε τα δεδομένα fat τα οποία θα χρησιμοποιήσουμε για αυτήν την άσκηση. Τα δεδομένα περέχουν πληροφορίες για 252 άννδρες, σχετικά με το ποσοστό σωματικού λίπους χρησιμοποιώνταας την εξίσωση Brozek και 17 ακόμη επεξηγηματικές μεταβλητές. Αρχικά θα φορτώσουμε τα δεδομένα μας και θα αφαιρέσουμε τις μεταβλητές siri, density, free μιας και δεν θα χρησιμοποιηθούν για την άσκηση.

```
library(faraway) #load the library
data<-faraway::fat #store our data
data<-data[,c(-2,-3,-8)]#remove unused columns

for (i in 2:ncol(data)){ #normalising
  data[,i]<-data[,i]-mean(data[,i]) #sum = 0
  data[,i]<-data[,i]/sqrt(sum((data[,i]-mean(data[,i]))^2)) #
    ↪ sum of squares = 1
}
```

Αφού φορτώσαμε τα δεδομένα στη συνέχεια, μόνο για τις τιμές των επεξηγηματικών μεταβλητών πραγματοποιήθηκε τυποποίηση. Συγκεκριμένα, μετα την αλλαγή κάθε μεταβλητή έχει άθροισμα ίσο με 0 και άθροισμα τετραγώνων ίσο με 1. Αυτό το καταφέραμε αφαιρώντας από κάθε παρατήρηση τη μέση τιμή της στήλης που ανήκει και διαιρώντας με την τετραγωνική ρίζα του αθροισματος τετραγώνων της στήλης.

α. Αρχικά θέλουμε να χωρίσουμε το δειγμα μας σε training data και test data με μεγέθη 4/5 του δείγματος και 1/5 αντίστοιχα.

```
set.seed(0) #for reproducibility
deigma<-sample(1:nrow(data),round(4/5*nrow(data)),replace =
  ↪ FALSE)
training_sample<-data[deigma,]
test_sample<-data[-deigma,]

X<-training_sample[,-1]
Y<-training_sample$brozek
x_test<-test_sample[,-1]
y_test<-test_sample[,1]
```

Χρησιμοποιούμε τη συνάρτηση sample() της R η οποία θα μας δώσει τα indices των παρατηρήσεων που θα μπούν στο training sample. Η συνάρτηση δέχεται ώς ορίσματα το εύρος των αριθμών από τους οποίους θα επιλέξει, που σε μας είναι το πλήθος των γραμμών, έπειτα πόσους αριθμούς θα επιστρέψει και ζητάμε τα 4/5 των παρατηρήσεων στρογγυλοποιημένα καθώς μιλάμε για διακριτές τιμές. Επίσης θα επιλέξουμε replace = FALSE για να μην επιλαγεί μια παρατήρηση πολλαπλές φορές. Στη συνέχεια ορίζουμε τα δύο δείγματα παίρνωντας μόνο τις παρατηρήσεις που μας έδωσε η τυχαία συνάρτηση sample() .

β. Η μέθοδος Lasso είναι μια μέθοδος συρρίκνωσης που χρησιμοποιείται για να μειώσει την επίδραση συγκεκριμένων εμταβλητών ενος μοντέλου παλινδρόμισης, ενώ ταυτόχρονα πραγματοποιεί και έμμεση επιλογή μεταβλητών μοντέλου. Ο τρόπος που λειτουργεί είναι πως

ποινικοποιείτους συντελεστές β_j, μέσω της l_1 νόρμας του διανύσματος των συντελεστών. Οι συντελεστές του μοντέλου Lasso βρίσκονται ελαχιστοποιώντας την παρακάτω σχέση:

$$min(y - Z\beta)^T(y - Z\beta), \quad \sum_{i=1}^{p} |\beta_j| \leq t \tag{4.1}$$

Όπου y το διάνυσμα μεταβλητής απόκρισης, Z ο πίνακας των τυποποιημένων επεξηγηματικών μεταβλητών, β το διάνυσμα των συντελεστών του μοντέλου και t είναι η παράμετρος ρύθμισης (regularization parameter). Η σχέση ισοδύναμα γράφεται και ως εξής:

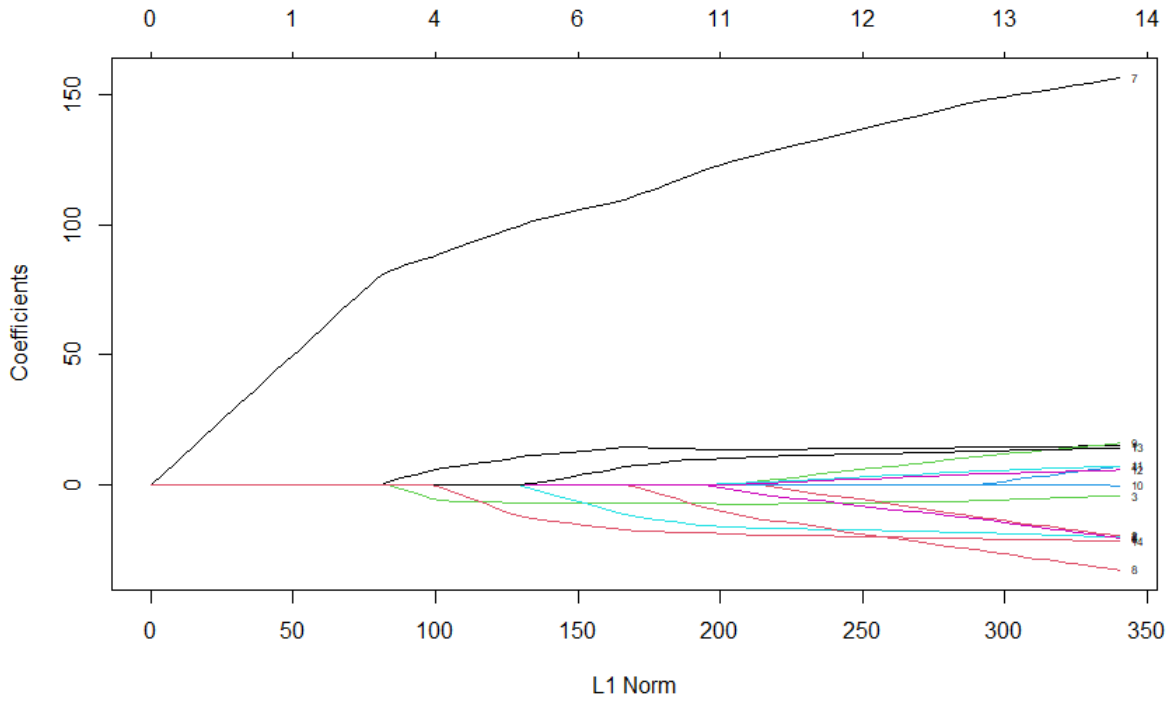
$$min \left\{ (y - Z\beta)^T(y - Z\beta) + \lambda \sum_{i=1}^{p} |\beta_j| \right\} \tag{4.2}$$

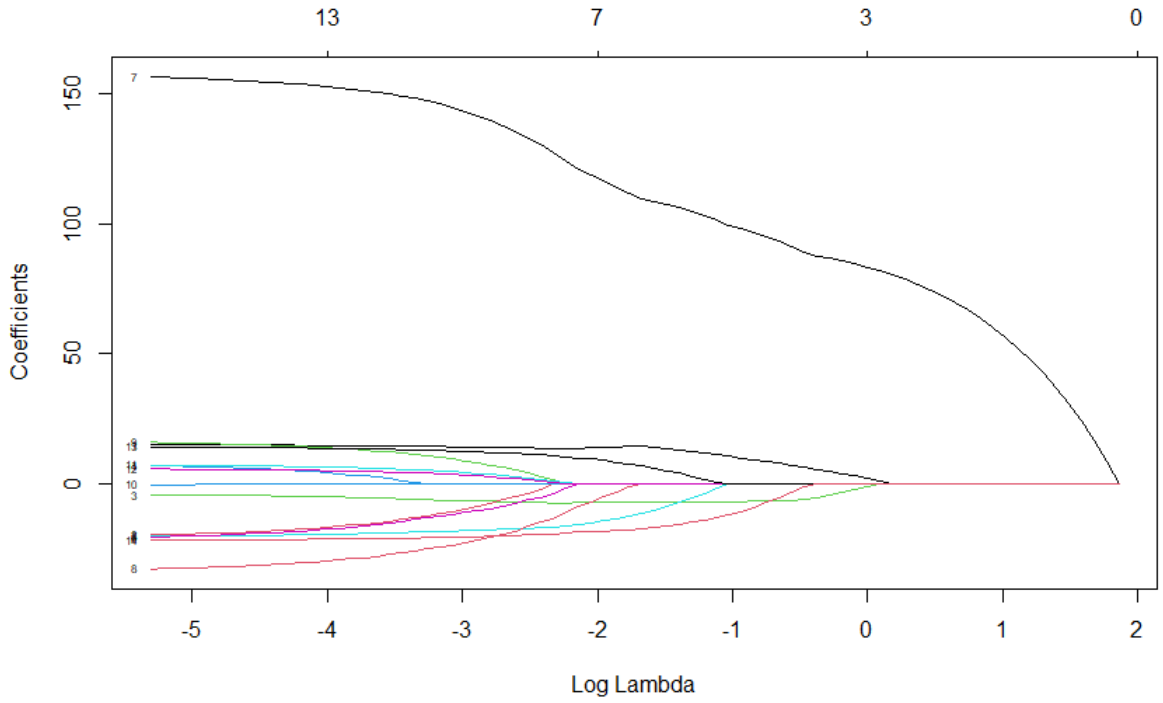
με λ πάλι η παράμετρος σύγκρισης σε άλλη μορφή.

Για να εφαρμόσουμε την μέθοδο θα χρησιμοποιήσουμε τη βιβλιοθήκη glmnet της R στο training sample.

```
library(glmnet) #load the library
lasso1<-glmnet(X,Y)
plot(lasso1,label = T)
plot(lasso1,xvar = 'lambda',label = T) #plotting with log scaling
```

Αρχικά αφού καλέσουμε τη βιβλιοθήκη κάνουμε χρήση της ομόνυμης εντολής $glmnet()$ με ορίσματα τις επεξηγηματικές μεταβλητές και την μεταβλητή απόκρισης. Στη συνέχεια αναπαριστούμε γραφικά το μοντέλο συναρτήσει της l_1 νόρμας και τον λογάριθμο του λ όπως φαίνεται στη συνέχεια.





Τα συμπεράσματα για τα δύο γραφήματα είναι ίδια, αφούμ βλέπουμε το σταδιακό μηδενισμό των συντελεστών των μεταβλητών καθώς το t μειώνεται στην πρώτη γραφική και καθώς ο λογάριθμος του λ αυξάνεται στη δεύτερη. Οι πιο σημαντικές μεταβλητές στο μοντέλο είναι αυτές που μηδενίζονται τελευταίες, όπως οι 7,13 και 3.

γ. Στη συνέχεια θέλουμε να επιλέξουμε, χρησιμοποιώντας cross-validation, την τιμή της παραμέτρου λ που ελαχιστοποιεί το CV-MSE. Θα χρησιμοποιήσουμε την εντολή cv.glmnet() και θα καταλήξουμε σε ένα μοντέλο με λιγότερες επεξηγηματικές μεταβλητές, που θα το καλέσουμε M1. Έπειτα με παρόμοια διαδικασία θα βρουμε την τιμή της λ που ελαχιστοποιεί το CV-MSE με σφάλμα εντός μιας τυπικής απόκλισης από την ελάχιστη τιμή, αυτό το μοντέλο θα το καλέσουμε M2. Σκοπός είναι να συγκρίνουμε τα δύο αυτά μοντέλα στα test data και να επιλέξουμε αυτό εμ το μικρότερο MSE.

```
1 X_matrix<-as.matrix(X)
2 lasso2<-cv.glmnet(X_matrix,Y)
3 M1<-coef(lasso2,s="lambda.min") #find the coefficients of the
    ↪ model
4 M2<-coef(lasso2,s="lambda.1se")
5 M1.ind<-c(1,3,5,7,8,13,14) #store the indices to fit the
    ↪ model
6 M2.ind<-c(1,3,7,14)
```

Αρχικά μετατρέπω τις επεξηγηματικές μου μεταβλητές σε μορφή πίνακα και καλώ την εντολή cv.glmnet(). Έπειτα βρίσκω τους συντελεστές του μοντέλου για κάθε περίπτωση με την εντολή coef() και αποθηκεύω σε μορφή διανύσματος τους δείκτες των μη μηδενικών μεταβλητών. Αυτό το κάνω για να μπορέσω να προσαρμόσω στη συνέχεια τα μοντέλα M1 και M2. Παρακάτω βλέπουμε τους συντελεστές των δύο μοντέλων.

```
> M1
15 x 1 sparse Matrix of class "dgCMatrix"
                     s1
(Intercept)  18.809049
age          14.079613
weight        .
height       -7.050080
adipos        .
neck        -14.019852
chest         .
abdom       116.538211
hip          -4.974351
thigh         .
knee          .
ankle         .
biceps        .
forearm       9.003045
wrist       -18.123980

> M2
15 x 1 sparse Matrix of class "dgCMatrix"
                    s1
(Intercept) 18.840156
age          9.450551
weight        .
height       -6.630711
adipos        .
neck          .
chest         .
abdom        96.501029
hip           .
thigh         .
knee          .
ankle         .
biceps        .
forearm       .
wrist        -8.915406
```

Έπειτα προσαρμόζουμε τα δύο μοντέλα παλινδρόμισης βάση των μεταβλητών που δεν μηδενίστηκαν και υπολογίζουμε τα MSE.

```
model1<-lm(Y~.,data = X[,M1.ind]) #fit the models
model2<-lm(Y~.,data = X[,M2.ind])
MSE.M1<-sum((predict(model1,x_test[,M1.ind])-y_test)^2)/
   ↪ length(y_test)
MSE.M2<-sum((predict(model2,x_test[,M2.ind])-y_test)^2)/
   ↪ length(y_test)
M_Lasso<-model2
```

Το MSE υπολογίζεται ως η διαφορά των προβλεπόμενων τιμών με τις πραγματικές στο τετράγωνο, προς το πλήθος των παρατηρήσεων, για το test sample. Για τις προβλεπόμενες τιμές χρησιμοποιούμε την εντολή predict().

```
> MSE.M1
[1] 15.76858
> MSE.M2
[1] 15.43327
```

Τελικά επιλέγουμε το M2 ώς M_Lasso αφού έχει μικρότερο MSE.

δ. Χρησιμοποιώντας όλα τα δεδομένα θα προσαρμόσουμε το γραμμικό μοντέλο με επεξηγηματικές μεταβλητές ίδιες με αυτές του M_Lasso.

```
x<-data[,-1]
y<-data[,1]
M3<-lm(y~.,data=x[,M2.ind])

> M3
Call:
lm(formula = y ~ ., data = x[, M2.ind])

Coefficients:
(Intercept)          age       height         abdom
   ↪ wrist
     18.938       13.391       -9.552      115.433
       ↪ -28.651
```

ε. Ψάχνουμε στον χώρο όλων των πιθανών μοντέλων, εκείνο το μοντέλο το οποίο ελαχιστοποιεί την τιμή του κριτηριου BIC. Θέλουμε να προσαρμόσουμε δηλαδή μοντέλα με κάθε δυνατό συνδυασμό μεταβλητών και να υπολογίσουμε το BIC για κάθε ένα από αυτά, επιλέγοντας τελικά το μοντέλο που δίνει την μικρότερη τιμή. Αρχικά θα αποθηκεύσω όλες τις μεταβλητές ξεχωριστά καθώς θα μου χρειαστεί στην αναζήτηση μοντέλων.

```
brozek<-data$brozek
age<-x$age
weight<-x$weight
height<-x$height
adipos<-x$adipos
neck<-x$neck
chest<-x$chest
abdom<-x$abdom
hip<-x$hip
thigh<-x$thigh
knee<-x$knee
ankle<-x$ankle
biceps<-x$biceps
forearm<-x$forearm
wrist<-x$wrist

```

```
predictors<-colnames(x)
response<-"brozek"
best_bic <- Inf
best_model <- NULL

# Loop over all subset sizes
for (k in 1:length(predictors)) {
  # Loop over all subsets of size k
  subsets <- combn(predictors, k, simplify = FALSE)
  for (subset in subsets) {
    # Construct the formula for this subset
    predictors_str <- paste(subset, collapse = "+")
    formula_str <- paste(response, "~", predictors_str)
    formula <- as.formula(formula_str)
    # Fit the model and compute its BIC
    model <- lm(formula, data = x)
    bic <- BIC(model)


    # If this is the best model so far, remember it
    if (bic < best_bic) {
      best_bic <- bic
      M4 <- model
    }
  }
}

summary(M4)
print(best_bic)
```

Έπειτα δημιουργώ ένα διάνυσμα που περιέχει τα ονόματα των επεξηγηματικών μεταβλητων, το ίδιο για την μεταβλητή απόκρισης και ορίζω αρχική τιμή για τι BIC το άπειρο. Το βήμα αυτό όπως και το best_model|- NULL είναι προεραιτικό και βοηθάει στην αρχικοποίηση των δεδομένων. Επειτα κάνουμε iterate σε κάθε πιθανό μέγεθος μοντέλου, πέρα απο το μηδενικό, και σε κάθε στάδιο του βρόγχου με χρήση της εντολής combn παίρνουμε όλους τους δυνατούς συνδυασμούς επεξηγηματικών μεταβλητών εκείνου του μεγέθους. Στη συνέχεια φτιάχνουμε τη φόρμουλα του μοντέλου ενώνοντας τα string πρώτα της μεταβλητής απόκρισης, έπειτα την και τέλος τον συγκεκριμένο συνδυασμό επεξηγηματικών μεταβλητών πυο τις χωρίζουμε μεταξύ τους με +. Έτσι έχουμε καταφέρει να έχουμε για συγκεκριμένο μέγεθος όλα τα πιθανά μοντέλα προσαρμοσμένα και αποθηκεύουμε σε ένα διάνυσμα το κάθε BIC. Αυτό πραγματοποιείται για μεγέθη μοντέλων απο το 1 μέχρι το πλήρες μοντέλο και άρα έχουμε κάνει αναζήτηση στον χώρο όλων των πιθανών μοντέλων. Τέλος με ένα απλό if statement έχουμε την μικρότερη τιμή του κριτηρίου BIC καθώς και το μοντέλο που το αποκαλούμε M4.

```
> summary(M4)
Call:
lm(formula = formula, data = x)

Residuals:
    Min      1Q  Median      3Q     Max
-9.8002 -2.8728 -0.1545  2.8980  8.3845

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  18.9385     0.2533  74.761  < 2e-16 ***
weight      -58.4648    10.6704  -5.479 1.05e-07 ***
abdom       157.4037     8.8692  17.747  < 2e-16 ***
forearm      14.2904     5.3853   2.654 0.008480 **
wrist       -20.5853     6.0628  -3.395 0.000799 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1  1

Residual standard error: 4.021 on 247 degrees of freedom
Multiple R-squared:  0.7351,    Adjusted R-squared:  0.7308
F-statistic: 171.4 on 4 and 247 DF,  p-value: < 2.2e-16

> print(best_bic)
[1] 1444.647
```

Το μοντέλο M4 περιέχει για επεξηγηματικές μεταβλητές τις weight, abdom, forearm, wrist και είναι διαφορετικό από το M3.

στ. Χρησιμοποιώντας την μέθοδο 5-fold cross-validation και την (within fold) ARMSE (Average Root Mean Square Error) συνάρτηση, θα εξεταστεί ποιό από τα μοντέλα M3 και M4 έχει την καλύτερη προβλεπτική ικανότητα. Αρχικά θα χωρίσω το δείγμα σε 5 folds περίπου ίδιου μεγέθους και θα χρησιμοποιήσω τα τέσσερα από αυτά για την προσαρμογή του μοντέλου ενώ το τελευταίο για πρόβλεψη και για τον υπολογισμό του MSE. Θα το κάνω αυτό για κάθε fold και θα υπολογίσω την μέση τιμή των MSE για να προβώ σε σύγκριση των μοντέλων. Έστω T_k ένα από τα folds, n_k το μέγεθος του, y_i είναι οι παρατηρήσεις της μεταβλητής απόκρισης του fold αυτού και $y_{i,-k}$ είναι η πρόβλεψη του y_i με όλα τα δεδομένα ακτός του T_k. Τότε έχουμε:

$$MSE(T_k) = \frac{1}{n_k} \sum_{i \in T_k} (y_i - y_{i,-k})^2 \tag{4.3}$$

$$ARMSE = \frac{1}{5} \sqrt{\sum_{i=1}^{5} MSE(T_k)} \tag{4.4}$$

Δίνεται ο κώδικας στην R.

```
n<-length(y)
set.seed(155)
randomise<-sample(1:n,n)
F1<-randomise[1:50]
F2<-randomise[51:100]
F3<-randomise[101:150]
F4<-randomise[151:200]
F5<-randomise[201:252]
folds<-list(F1,F2,F3,F4,F5)
M3.ind<-c(1,3,7,14)
M4.ind<-c(2,7,13,14)

MSE.M3<-rep(0,5)
MSE.M4<-rep(0,5)

for(i in 1:5){
  fold<-unlist(folds[i])
  xnew_train<-x[-fold,]
  ynew_train<-y[-fold]
  xnew_test<-x[fold,]
  ynew_test<-y[fold]

  M3.mod<-lm(ynew_train~.,data = xnew_train[,M3.ind])
  M4.mod<-lm(ynew_train~.,data = xnew_train[,M4.ind])

  MSE.M3[i]<-sum((predict(M3.mod,xnew_test[,M3.ind])-ynew_
    ↪ test)^2)/length(ynew_test)
  MSE.M4[i]<-sum((predict(M4.mod,xnew_test[,M4.ind])-ynew_
    ↪ test)^2)/length(ynew_test)
}

ARMSE.M3<-sqrt(sum(MSE.M3))/5
ARMSE.M4<-sqrt(sum(MSE.M4))/5
```

Αρχικά για να δημιουργήσουμε τα folds θα χρειαστούμε τη συναρτηση sample() όπου θα επιλέξει από τις παρατηρήσεις μας ένα τυχαίο δείγμα ίδιου μεγέθους και άρα απλά έχει ανακατέψει τις παρατηρήσεις. Στη συνέχεια χωρίζουμε το τυχαίοποιημένο δείγμα σε 5 μέρη και τα τοποθετούμε σε μια λίστα. Θα χρειαστούμε και τους δείκτες των επεξηγηματικών μεταβλητών των δύο μοντέλων και αρχικοποιούμε τα διανύσματα που θα αποθηκεύσουν τις τιμές των MSE. Στη συνέχεια για κάθε i επιλέγουμε το i-οστό fold και το αφαιρούμε απο τις παρατηρήσεις για την προσαρμογή του μοντέλου, ενω θα το χρειαστούμε για προβλέψεις. Προσαρμόζουμε τα δύο μοντέλα στα δεδομένα και στη συνέχεια υπολογίζουμε το MSE για το συγκεκριμένο fold. Τελος αφού έχουμε υπολογίσει 5 τιμές για κάθε MSE θα υπολογίσουμε τις μέσες τιμές τους. Το M3 έχει ARMSE = 1.877849 ένω το M4 έχει ARMSE = 1.814728. Παρατηρούμε πως το μοντέλο M4 έχει μικρότερη τιμή για το ARMSE και άρα έχει την καλύτερη προβλεπτική ικανότητα.

ζ. Χρησιμοποιώντας το M4 ως τελικό μοντέλο σε όλα τα δεδομένα και 1000 Bootstrap δείγματα από τα υπόλειπα θέλουμε να δώσουμε εκτιμήτριες και τυπικά σφάλματα για τους συντελεστές του μοντέλου. Θα χρησιμοποιήσουμε μη παραμετρικό Bootrstrap καθώς δεν γνωρίζουμε τη μέση τιμή και τη διασπορά των υπολοίπων. Συγκεκριμένα από το δείγμα των υπολοίπων θα δημιουργήουμε 1000 νέα δείγματα ίδιου μεγέθους κάνοντας δειγματοληψία με επανατοποθέτηση. Για το καινούριο σύνολο δεδομένων θα ισχύει:

$$\bar{\hat{\theta}}^* \equiv \frac{1}{B}\sum_{i=1}^{B}\hat{\theta}^* \to E[\hat{\theta}] \tag{4.5}$$

$$\sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{\theta}_i^* - \bar{\hat{\theta}}^*)^2} \to se(\hat{\theta}) \tag{4.6}$$

Όπου B είναι το πλήθος των δειγμάτων. Οι πράξεις αυτές θα γίνουν με την βοήθεια της βιβλιοθήκης boot της R.

```
M_final<-M4
library(boot)
res<-M_final$residuals

bootstrap_func<-function(data,indices){

  bootstrap_res<-res[indices]

  bootstrap_response <- brozek + bootstrap_res
  bootstrap_model <- lm(bootstrap_response ~., data = x[M4.
     ↪ ind])
  return(coef(bootstrap_model))
}

results1<-boot(data=res,statistic = bootstrap_func, R=1000)

bootstrap_coefficients <- colMeans(results1$t) #estimates of
   ↪ the coefficients
print(bootstrap_coefficients)
print(M4$coefficients)

bootstrap_se <- apply(results1$t, 2, sd) #estimates of
   ↪ standard error
print(bootstrap_se)
print(summary(M4)$coefficients[,"Std.␣Error"])
```

Αρχικά αποθηκεύουμε σε ένα διάνυσμα τα υπόλοιπα του τελικού μοντέλου, στη συνέχεια θα φτιάξουμε τη συνάρτηση που θα χρησιμοποιήσει η εντολή boot. Η συνάρτηση δέχεται τα αρχικά δεδομένα και τους δείκτες που παράγει η εντολή και αφού επιλέξει το νέο δείγμα υπολοίπων το προσθέτει στην μεταβλητή απόκρισης και προσαρμόζει καινούριο μοντέλο παλινδρόμισης με τις ίδιες επεξηγηματικές μεταβλητές και αποθηκεύει τους συντελεστές του μοντέλου. Έπειτα καλούμε τη συνάρτηση boot με δεδομένα τα υπόλοιπα, συνάρτηση

αυτή που δημιουργήσαμε και πλήθος δειγμάτων 1000. Για να συγκρίνουμε τα αποτελέσματα θα τυπώσουμε τους εκτιμητές καθώς και τους πραγματικούς συντελεστές του μοντέλου και μετα τις εκτιμήσεις των τυπικών σφαλμάτων μαζί με τις πραγματικές τιμές του αρχικού μοντέλου.

```
 > print(bootstrap_coefficients)
[1]  18.94283 -57.80087 157.04111  14.28471 -20.82949
> print(M4$coefficients)
(Intercept)      weight        abdom     forearm       wrist
   18.93849   -58.46475   157.40365    14.29038   -20.58529

> print(bootstrap_se)
[1]  0.2464339 10.3167843  8.3938198  5.1239055  6.0259312
> print(summary(M4)$coefficients[,"Std. Error"])
(Intercept)      weight        abdom     forearm       wrist
  0.2533216  10.6704481   8.8692358   5.3852752   6.0628484
```

Όπως βλέπουμε οι εκτιμήσεις είναι πολύ κοντα στις πραγματικές τιμές και για τους συντελεστές και για τα τυπικά σφάλματα. Σε περίπτωση που θέλουμε να έχουμε μικρότερο σφάλμα στις εκτιμήσεις μας μπορούμε να μεγαλώσουμε το R ή να εφαρμόσουμε κάποια μέθοδο ελάττωσης διασποράς στις εκτιμήσεις.