ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

«ΑΝΑΛΥΤΙΚΗ ΔΕΔΟΜΕΝΩΝ, 6ο Εξάμηνο»

Ακαδημαϊκό Έτος 2018 – 2019

Διορία	**28/06/2019**
Ονοματεπώνυμο Φοιτητή **(Αριθμός Μητρώου)**	Σκάρος Γεώργιος **(Π15128)** Γκιολής Δημήτρης **(Π16023)** Κωστάκη Ειρήνη **(Π16064)**

Πρόλογος

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού **R** και το πρόγραμμα **Rstudio**. Επίσης, η δημιουργία της βάσης δεδομένων του πρώτου ερωτήματος έγινε σε **postgreSQL.**

Η εκδόσεις που χρησιμοποιήθηκαν είναι οι εξής:

- R version 3.6.0
- PostgreSQL 9.6.6

Οι βιβλιοθήκες στην R που χρησιμοποιήθηκαν είναι οι εξής

- library(Metrics)
- library(class)
- library(caret)
- library(pROC)
- library(mlbench)
- library(factoextra)
- library(cluster)
- library(NbClust)
- library(arules)
- library(arulesViz)

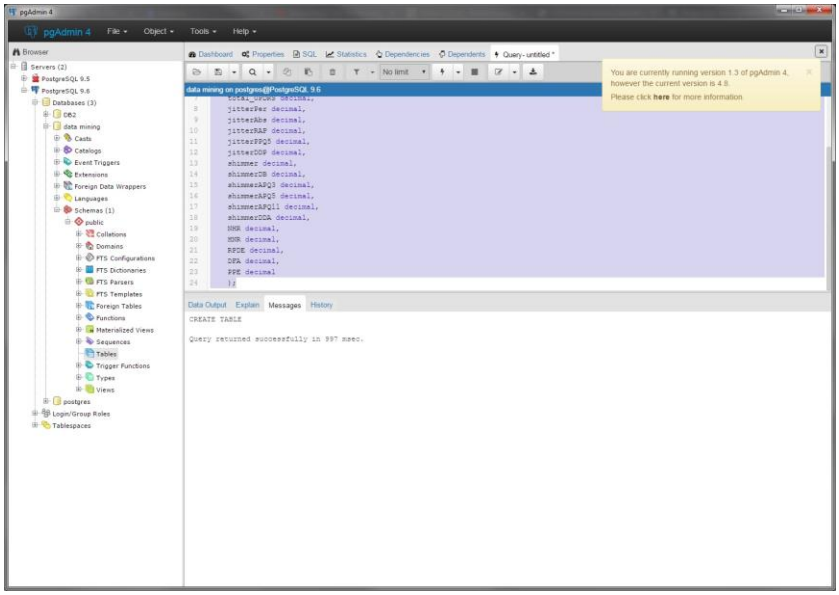
Αν δεν είναι εγκατεστημένη οποιαδήποτε από τις παραπάνω βιβλιοθήκες αυτό μπορεί να γίνει με την εντολή install.packages(“”)

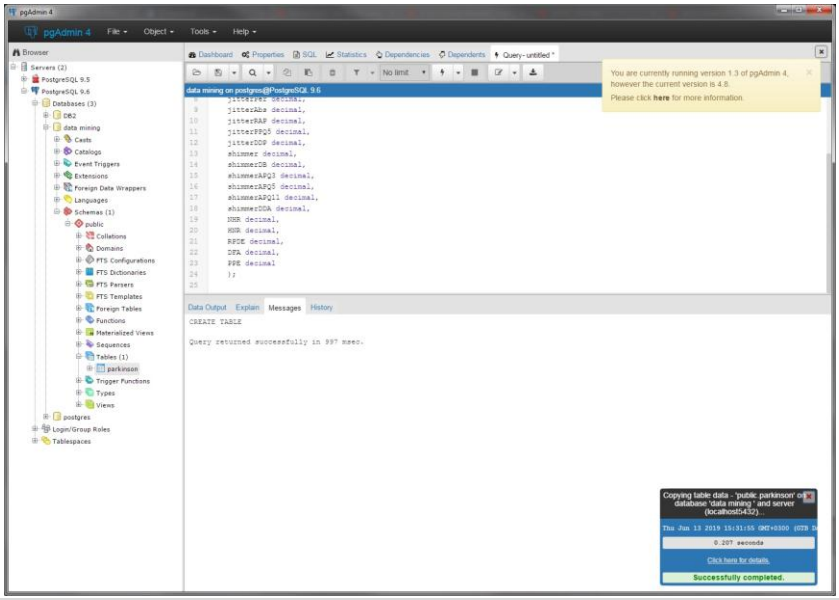
Η εργασία αποτελείται από 7 R scripts.

Εισαγωγή δεδομένων

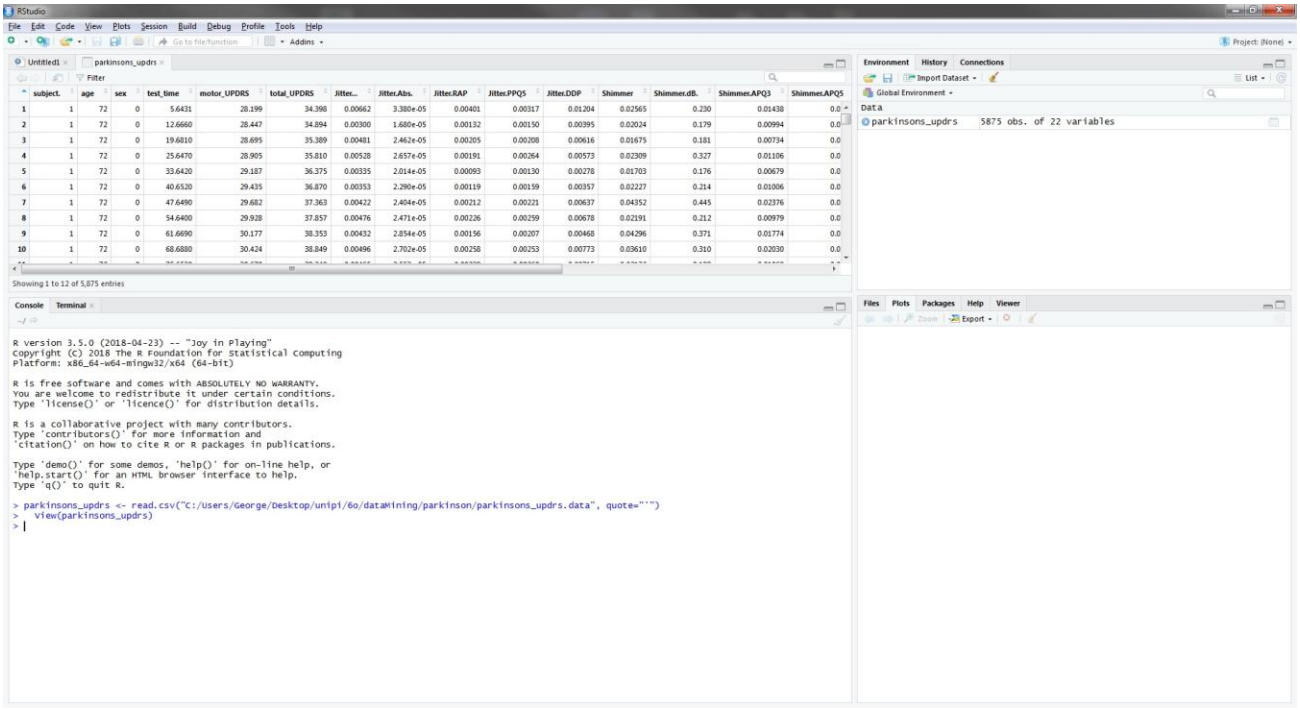
Εισαγωγή στην PostgreSQL

Μετά την δημιουργία της κατάλληλης βάσης δεδομένων στο περιβάλλον μας (το CreateTable υπάρχει στα αρχεία) έγινε η εισαγωγή των δεδομένων στο περιβάλλον μας. Όλες οι στήλες της βάσης επιλέχθηκαν να είναι τύπου decimal μιας και όλα είναι δεκαδικοί αριθμοί με μόνες εξαιρέσεις την ηλικία του ασθενή , το φύλο του και το identifier του.





Στην συνέχεια έγινε η εισαγωγή στο περιβάλλον της R.

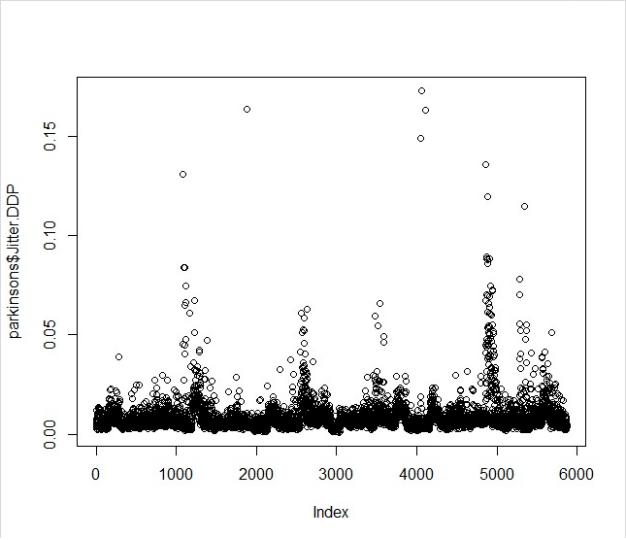


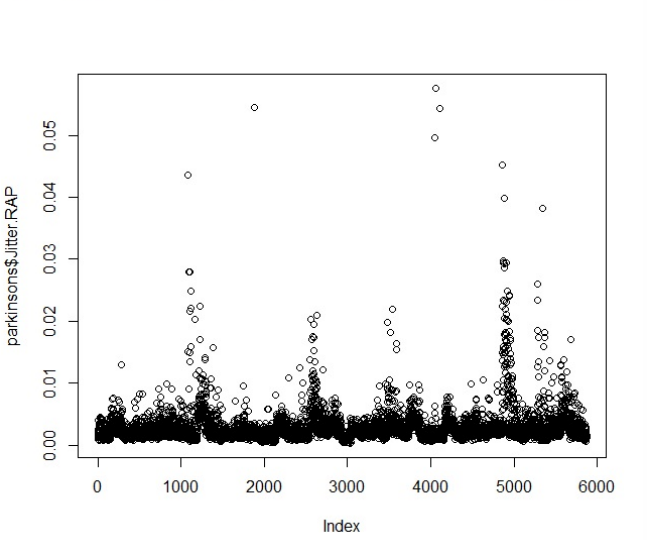
Προεπεξεργασία δεδομένων

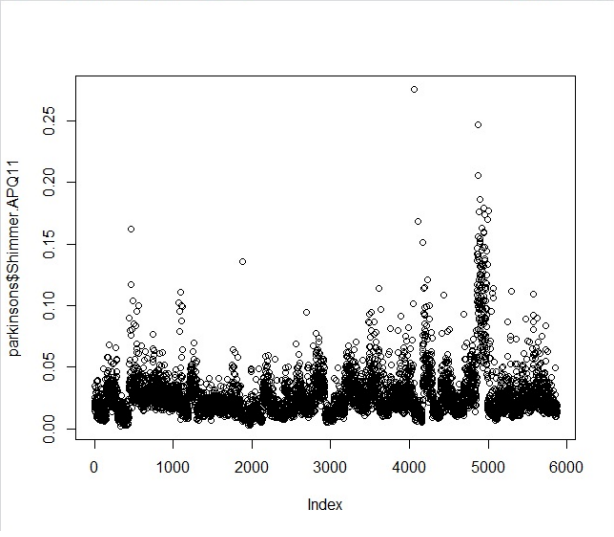
Εύρεση και διαχείριση μη λογικών τιμών Outliers

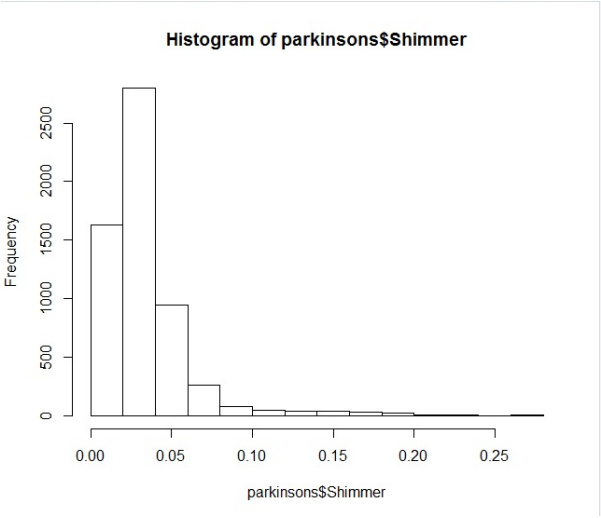
Εύρεση τιμών

Σε αυτό το βήμα έγιναν scatter plots για όλες τις στήλες των δεδομένων με αποτέλεσμα να βρεθούν μη λογικές και ακραίες τιμές. Παρακάτω θα παραθέσω μερικά από τα plots που έγιναν ενδεικτικά. Τα υπόλοιπα screenshots υπάρχουν στον φάκελο screens του παραδοτέου.





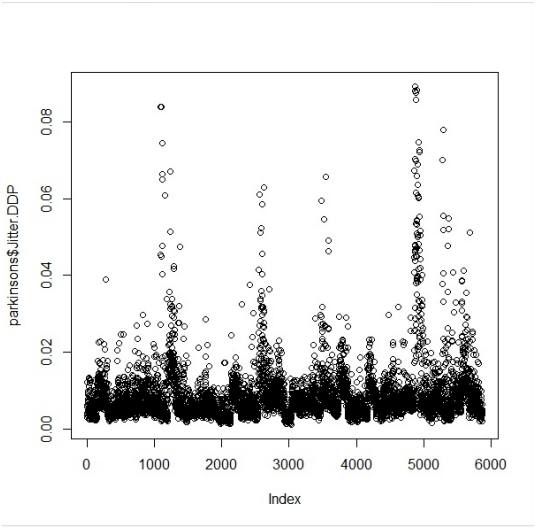


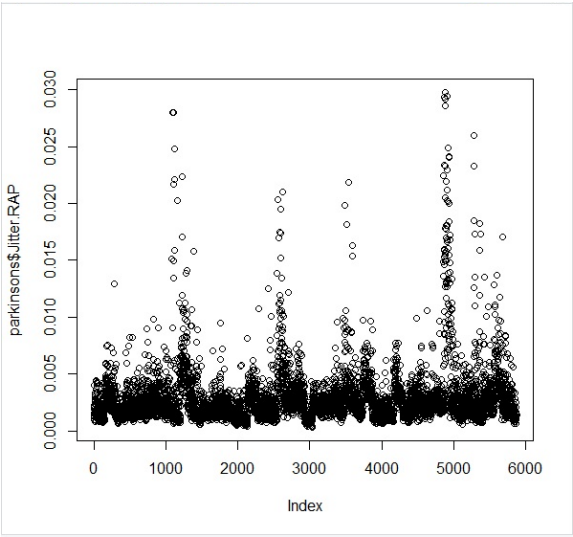


Διαχείριση τιμών

Στην περίπτωση μας βλέπουμε ότι υπάρχουν αρκετές τιμές που είναι μακριά από το σύνολο των δεδομένων. Ο τρόπος διαχείρισης που επιλέχθηκε για την περίπτωση μας είναι να ανατεθεί σε αυτές τις τιμές η μέση τιμή της στήλης αντί να τις διαγράψουμε διότι το σύνολο των δεδομένων μας είναι αρκετά μικρό. Παρακάτω παραθέτω δύο από τα «καθαρισμένα» παραδείγματα.

-*παρατήρηση*: Σε αυτό το σημείο μπορούμε να παρατηρήσουμε ότι οι τιμές των δεδομένων ανάμεσα 4900 και 5000 έχουν πολύ ακραίες τιμές σε σύγκριση με τις υπόλοιπες. Αν το δείγμα μας ήταν μεγαλύτερο αυτές οι τιμές θα έπρεπε να αφαιρεθούν τελείως ή να κανονικοποιηθούν αλλά στην περίπτωση μας αυτό θα άλλαζε πολύ το δείγμα μας οπότε αφέθηκαν ως έχει





Κοινωνικοποίηση του πίνακα

Σε αυτό το βήμα χρησιμοποιήθηκε η συνάρτηση scale για να κανονικοποιηθούν όλες οι στήλες πέραν του φύλου. Σε αυτό το σημείο έκρινα σωστό να παραλείψω την στήλη με το identifier του κάθε ασθενή καθώς μας είναι άχρηστη την υπόλοιπη διαδικασία.

Μείωση του όγκου δεδομένων

Πέρα από την παραπάνω παρατήρηση που έγινε για την μείωση του πλήθους εγγραφών, ένας ακόμα τρόπος για να μειωθούν τα δεδομένα είναι να παραλείψουμε κάποια από τις στήλες με κριτήριο της συσχέτιση της (correlation) που έχει με τις άλλες. Αν το correlation είναι κοντά στο 1 αυτό σημαίνει ότι οι 2 στήλες είναι ανάλογες με αποτέλεσμα να μην χρειάζονται και οι δύο στο data set μειώνοντας έτσι την πολυπλοκότητα.

Classification/regression

Linear regression

Αρχικά με την συνάρτηση Sample χώρισα το data set σε test και train(20% test 80% train). Σε αυτό το σημείο πρέπει να σημειωθεί ότι το data set έχει δύο στήλες στόχους πράγμα που σημαίνει ότι δοκίμασα όλους τους πιθανούς συνδυασμούς για να φτιάξω το γραμμικό μοντέλο. Εξηγώντας, οι 4 επιλογές είναι οι εξής :

- στήλη στόχος το motor_UPDRS
- στήλη στόχος το total_UPDRS
- στήλη στόχος το motor_UPDRS ,συμπεριλαμβάνοντας την στήλη total_UPDRS
- στήλη στόχος το total_UPDRS, συμπεριλαμβάνοντας την στήλη motor_UPDRS

Αφού έφτιαξα το μοντέλο με την συνάρτηση lm() και για τις 4 περιπτώσεις εκπαίδευσα με το ίδιο train (πειράζοντας βέβαια τις στήλες οπού χρειαζόταν) τα 4 μοντέλα μου.

```
> summary(m_motor)

Call:
lm(formula = motor_UPDRS ~ age + sex + test_time + Jitter... +
    Jitter.Abs. + Jitter.RAP + Jitter.PPQ5 + +Jitter.DDP + Shimmer +
    Shimmer.dB. + Shimmer.APQ3 + Shimmer.APQ5 + Shimmer.APQ11 +
    Shimmer.DDA + NHR + HNR + RPDE + DFA + PPE, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-3.5345 -0.7311 -0.1056  0.7535  2.3457

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.05938    0.01721   3.451 0.000564 ***
age            0.19631    0.01421  13.816  < 2e-16 ***
sex           -0.19062    0.03422  -5.571 2.68e-08 ***
test_time      0.08245    0.01341   6.149 8.46e-10 ***
Jitter...      0.17131    0.03554   4.820 1.48e-06 ***
Jitter.Abs.   -0.40331    0.04004 -10.072  < 2e-16 ***
Jitter.RAP   -12.83119   13.03694  -0.984 0.325059
Jitter.PPQ5   -0.01025    0.03006  -0.341 0.733180
Jitter.DDP    13.06298   13.03680   1.002 0.316391
Shimmer        0.07877    0.06216   1.267 0.205121
Shimmer.dB.    0.05758    0.08368   0.688 0.491451
Shimmer.APQ3  -0.01588    0.12010  -0.132 0.894847
Shimmer.APQ5  -0.20976    0.08258  -2.540 0.011116 *
Shimmer.APQ11  0.26144    0.04945   5.287 1.30e-07 ***
Shimmer.DDA   -0.24525    0.12243  -2.003 0.045214 *
NHR           -0.15106    0.03494  -4.323 1.57e-05 ***
HNR           -0.22925    0.03124  -7.339 2.52e-13 ***
RPDE           0.02681    0.01932   1.388 0.165275
DFA           -0.21210    0.01711 -12.393  < 2e-16 ***
PPE            0.17615    0.02819   6.249 4.50e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9177 on 4674 degrees of freedom
Multiple R-squared:  0.1622,    Adjusted R-squared:  0.1588
F-statistic: 47.61 on 19 and 4674 DF,  p-value: < 2.2e-16

>
```

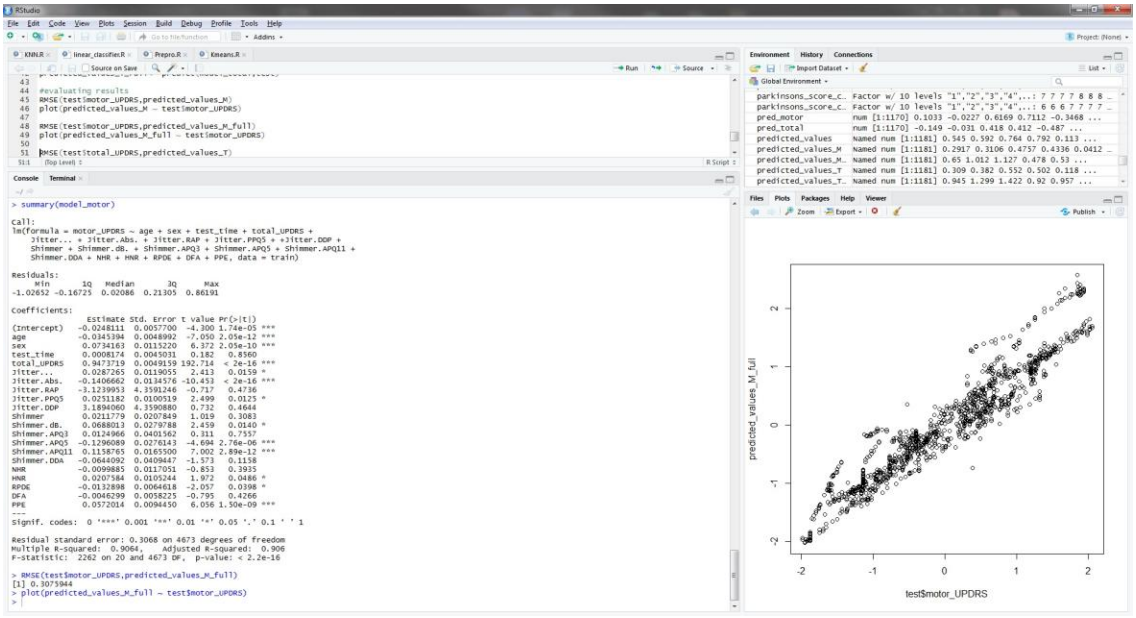
```
> summary(m_total)

Call:
lm(formula = total_UPDRS ~ age + sex + test_time + Jitter... +
    Jitter.Abs. + Jitter.RAP + Jitter.PPQ5 + +Jitter.DDP + Shimmer +
    Shimmer.dB. + Shimmer.APQ3 + Shimmer.APQ5 + Shimmer.APQ11 +
    Shimmer.DDA + NHR + HNR + RPDE + DFA + PPE, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6473 -0.6977 -0.1735  0.6927  2.4758

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.08887    0.01712   5.191 2.18e-07 ***
age            0.24367    0.01413  17.239  < 2e-16 ***
sex           -0.27870    0.03404  -8.188 3.40e-16 ***
test_time      0.08617    0.01334   6.460 1.16e-10 ***
Jitter...      0.15050    0.03536   4.257 2.11e-05 ***
Jitter.Abs.   -0.27724    0.03984  -6.959 3.89e-12 ***
Jitter.RAP   -10.24644   12.96939  -0.790   0.4295
Jitter.PPQ5   -0.03733    0.02990  -1.248   0.2120
Jitter.DDP    10.42207   12.96925   0.804   0.4217
Shimmer        0.06079    0.06184   0.983   0.3256
Shimmer.dB.   -0.01185    0.08325  -0.142   0.8868
Shimmer.APQ3  -0.02995    0.11948  -0.251   0.8021
Shimmer.APQ5  -0.08461    0.08216  -1.030   0.3031
Shimmer.APQ11  0.15365    0.04919   3.124   0.0018 **
Shimmer.DDA   -0.19089    0.12180  -1.567   0.1171
NHR           -0.14891    0.03476  -4.284 1.87e-05 ***
HNR           -0.26390    0.03108  -8.492  < 2e-16 ***
RPDE           0.04232    0.01922   2.202   0.0277 *
DFA           -0.21899    0.01703 -12.862  < 2e-16 ***
PPE            0.12555    0.02804   4.477 7.74e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.913 on 4674 degrees of freedom
Multiple R-squared:  0.1769,    Adjusted R-squared:  0.1736
F-statistic: 52.88 on 19 and 4674 DF,  p-value: < 2.2e-16
```

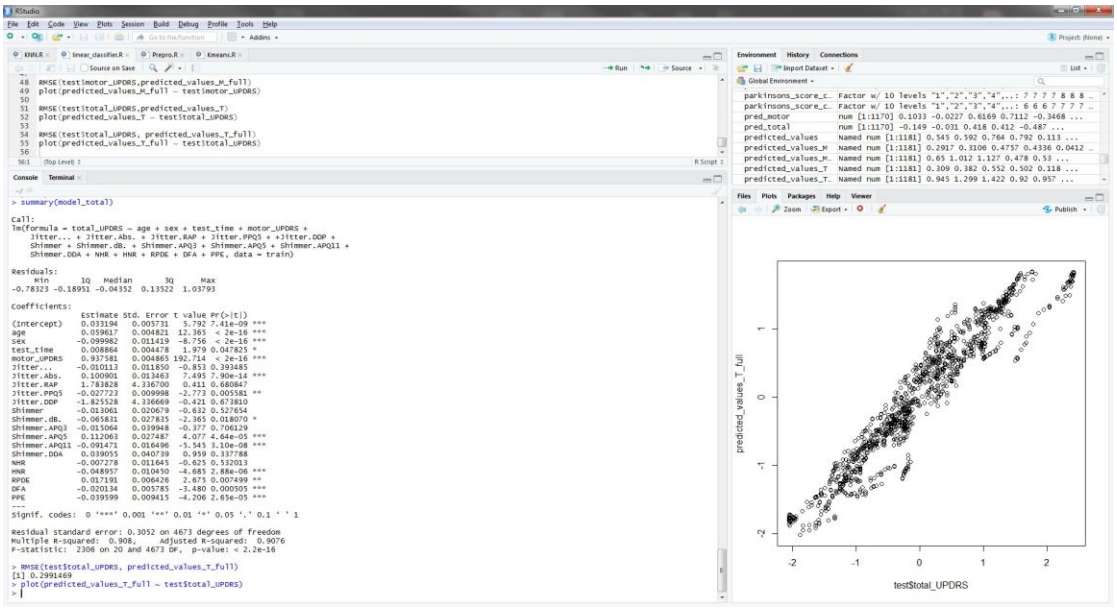
Σημαντικό ρόλο στην αξιοπιστία του μοντέλου παίζουν οι τιμές **p-value** και **R-squared** που βλέπουμε κάτω δεξιά στις εικόνες. Το **p-value θέλουμε να είναι μικρότερο του 0,05** και **το R-squared όσο το δυνατόν πιο κοντά γίνεται στο 1.**

Με τα μοντέλα που έφτιαξα κάνω πρόβλεψη και για τις 4 περιπτώσεις με την συνάρτηση predict() και δημιουργώ 4 προβλεπτές και προσπαθώ να προβλέψω την αντίστοιχη τιμή της κάθε στήλης στόχου κάθε φορά. Οι δύο παραπάνω προβλεπτές έχουν πολύ κακό ποσοστό

πρόβλεψης αφού το R-squared είναι πολύ κοντά στο 0 οπότε δεν θα δείξω παραδείγματα από scatter plot από αυτούς τους δύο προβλεπτές.

Οι άλλες δύο περιπτώσεις έχουν μια ικανοποιητική πρόβλεψη αλλά και πάλι όχι αρκετά καλή έτσι ώστε να είναι ο αλγόριθμος που θα προτιμηθεί από αυτούς που υλοποιήθηκαν. Παρακάτω παραθέτω τα Summary ,RMSE values ,plots για τις δύο πιο καλά προσαρμοσμένες περιπτώσεις του προβλήματος μας.





KNN

Πριν το μοντέλο του προβλεπτή

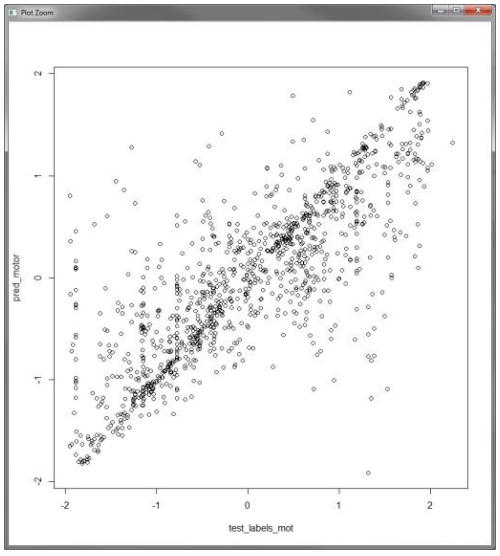
Σε αυτό το ερώτημα εργάστηκα παρόμοια με το προηγούμενο ως αναφορά την στήλη στόχο.

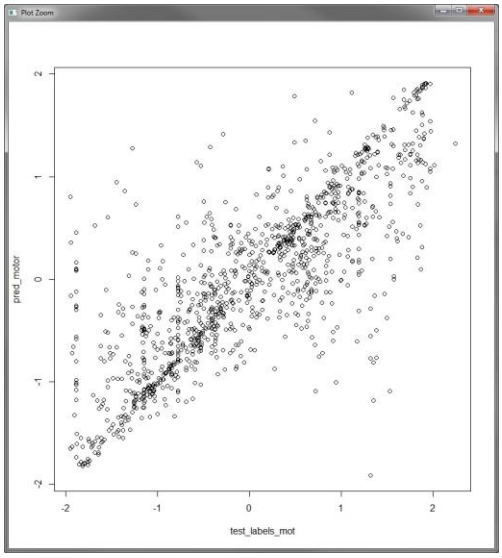
Αρχικά δημιουργήθηκε μια συνάρτηση για την διαδικασία του cross validation η οποία παίρνει το train set το κόβει σε 10 κομμάτια και ελέγχει να βρει το καλύτερο ποσοστό λάθους αφού έχει βρει το καθένα ξεχωριστά. Αυτή η διαδικασία επαναλαμβάνεται με 3 διαφορετικά διπλώματα (folds) του σετ μας ψάχνοντας να βρει όσο το δυνατόν καλύτερο αποτέλεσμα.

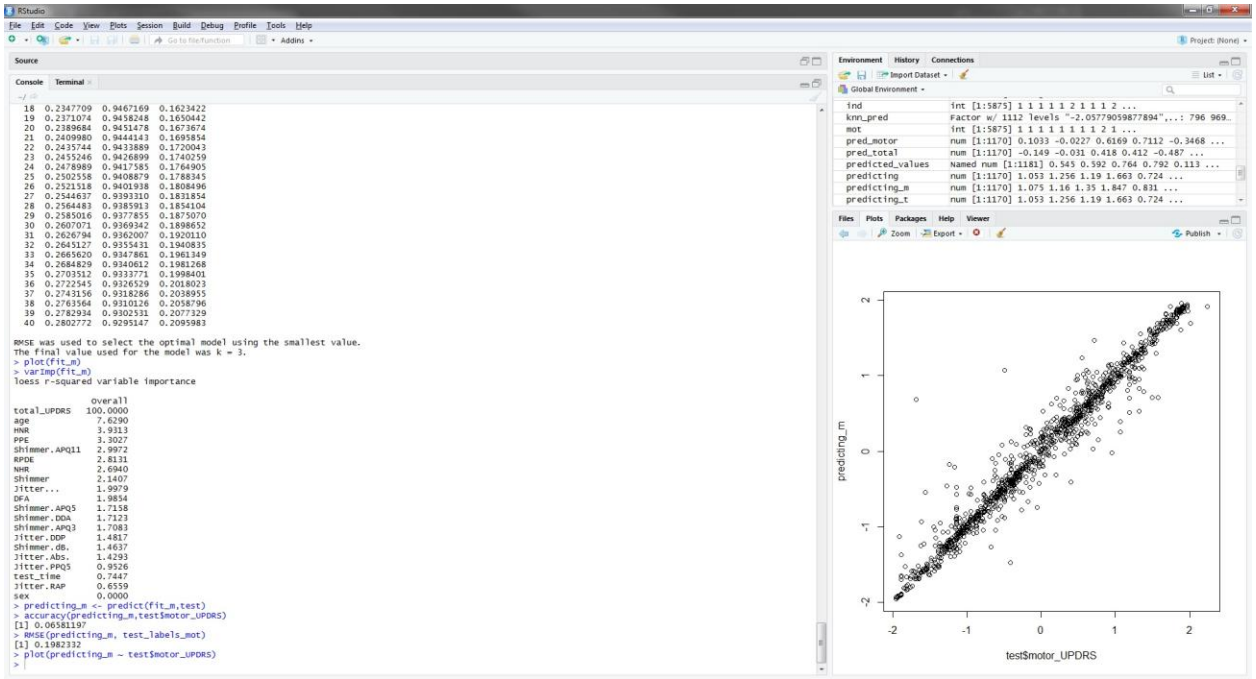
Προβλεπτής

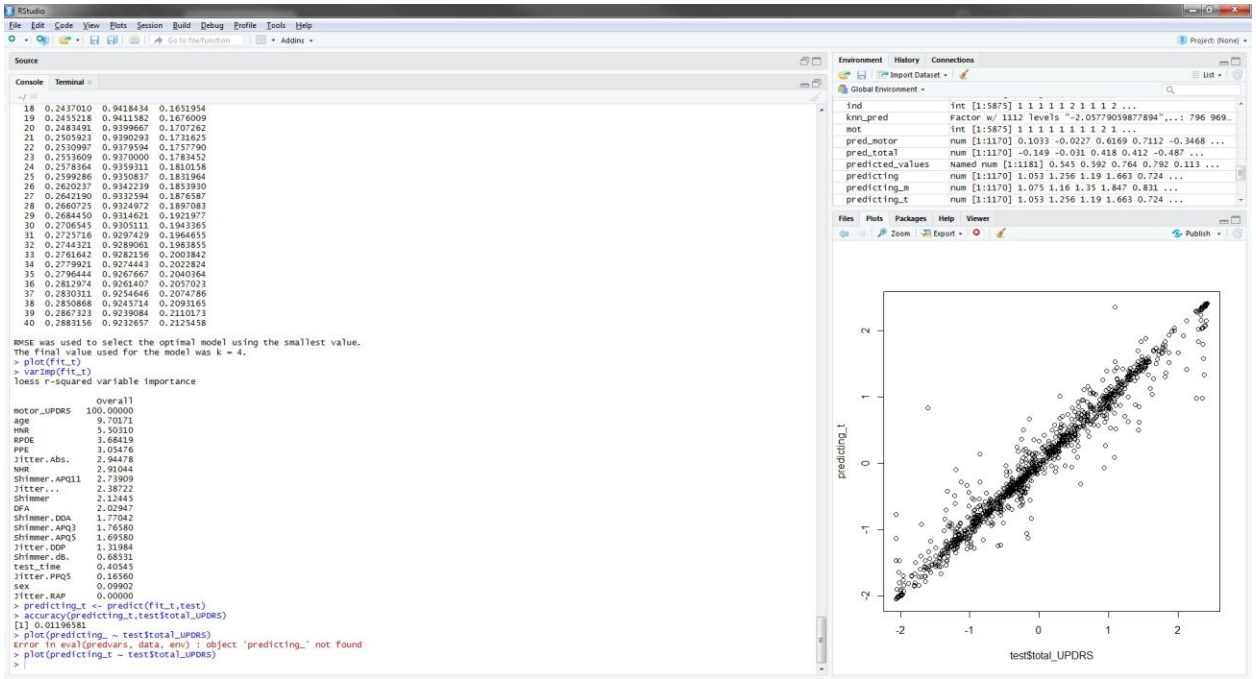
Όμοια με τον γραμμικό προβλεπτή φτιάχνω 4 διαφορετικά μοντέλα για τις ανάγκες μου και τα υλοποιώ και συγκρίνω τα αποτελέσματα είτε οπτικά (με κάποιο plot) είτε από την τιμή της συνάρτησης accuracy. Το k επιλέχθηκε από τον αλγόριθμο αφού ήταν έτσι φτιαγμένος ώστε να δοκιμάζει διαφορετικά k και να επιλέγει αυτό που δίνει το καλύτερο αποτέλεσμα.

Παρακάτω δείχνω plots από τα μοντέλα και τα accuracy scores όπως και τα Variable importance της κάθε περίπτωσης. Από το τελευταίο μπορούμε να δούμε ποιες από τις μεταβλητές δεν παίζουν πολύ ρόλο στην πρόβλεψη που γίνεται. Τα δύο πρώτα plots είναι από τις περιπτώσεις που δεν είχαμε την δεύτερη μεταβλητή στόχο στις μεταβλητές πρόβλεψης και τα επόμενα είναι με τις μεταβλητές. Είναι φανερό το γεγονός ότι όταν βάζουμε τις μεταβλητές στόχους μέσα τα αποτελέσματα είναι πολύ καλύτερα.









ΣΚΑΡΟΣ ΓΕΩΡΓΙΟΣ (Π15128)

ΓΙΟΛΗΣ ΔΗΜΗΤΡΗΣ (Π16023)

ΚΩΣΤΑΚΗ ΕΙΡΗΝΗ (Π16064)

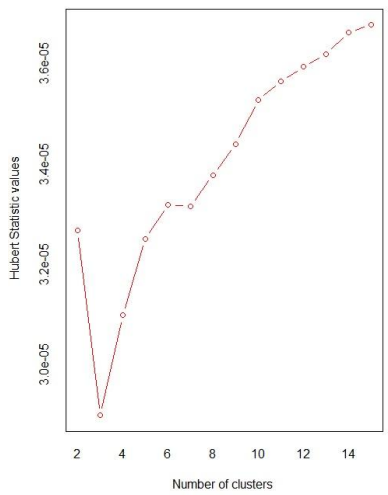
Clustering

Kmeans

Αρχικά παραλήφθηκε η στήλη στόχος με όλους τους πιθανούς συνδυασμούς.

Ο kmeans είναι ένας αλγόριθμος ο οποίος λειτουργεί με κεντροειδή, αυτό σημαίνει ότι η πρώτη διαδικασία είναι να τρέψουμε έναν αλγόριθμο για να μας υπολογίσει αυτά τα κεντροειδή. Μετά από δύο αλγορίθμους που χρησιμοποιήθηκαν το ιδανικό k υπολογίστηκε ότι είναι 3 , δηλαδή ότι το σύνολο τον δεδομένων μας χωρίζεται σε 3 clusters.

```
> NbClust(data = parkinsons_clust, diss = NULL, distance = "euclidean",
+         min.nc = 2, max.nc = 15, method = "kmeans")
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

*******************************************************************
* Among all indices:
* 6 proposed 2 as the best number of clusters
* 12 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 1 proposed 7 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 2 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3
```



Αφού έτρεξα τον ολόθερμο για κάθε ένα από τα σύνολα πηρά την πρωτοβουλία να χωρίσω κάθε ένα από τα χαρακτηριστικά στόχους σε bins (ίσα σαν μέγεθος διαστήματα) για να μπορέσω να παρατηρήσω ποσοστά ευστοχίας του αλγορίθμου ,το οποίο θα παραθέσω παρακάτω μαζί με τα clusters που δημιουργήθηκαν.

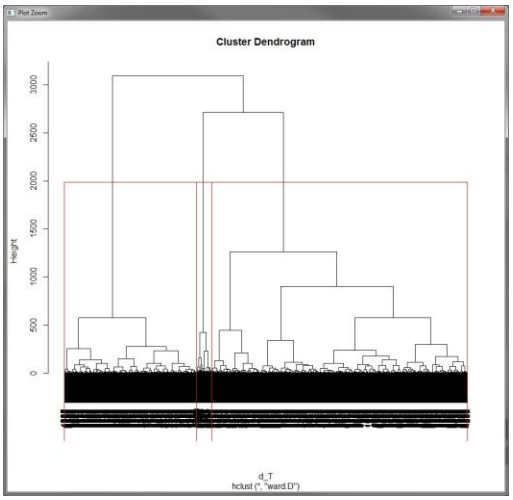
```
> clusters_M <- kmeans(parkinsons_clust_M,3)
> clusters_T <- kmeans(parkinsons_clust_T,3)
> print(clusters_M)
K-means clustering with 3 clusters of sizes 3577, 182, 2116

Cluster means:
         age       sex    test_time total_UPDRS  Jitter... Jitter.Abs. Jitter.RAP Jitter.PPQ5 Jitter.DDP    Shimmer Shimmer.dB. Shimmer.APQ3 Shimmer.APQ
5 Shimmer.APQ11
1 -0.1460725 0.3597987  0.005249682  -0.1198141 -0.3735017  -0.4765347 -0.3983611  -0.3968489 -0.3983588 -0.4853426  -0.4796693   -0.4827582   -0.454307
3    -0.4828605
2 -0.2145716 0.7307692 -0.147266501   0.1684680  3.8607059   3.1119386  3.7602990   3.7111082  3.7603315  3.7684128   3.9865560    3.7793469    3.945468
7     3.4508015
3  0.2653843 0.2112476  0.003792245   0.1880500  0.2993227   0.5378978  0.3499827   0.3516572  0.3499759  0.4963230   0.4679696    0.4910136    0.428630
4     0.5194452
  Shimmer.DDA        NHR        HNR       RPDE         DFA        PPE pred_M
1  -0.4843164 -0.2880669  0.5485796 -0.3945944 -0.29840097 -0.5313499      1
2   3.7573117  4.4105508 -2.9586642  1.5376941 -0.07535145  2.4914275      3
3   0.4955430  0.1076064 -0.6728698  0.5347844  0.51091409  0.6839313      2

Clustering vector:
   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1 1 1 1 3 1 1 1 1 1 3 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1
  [82] 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 3 1 1 1 1 1 1 1 3 3 3 3 1
 3 3 3 3 3 3 3 3
 [163] 3 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 1 1 3 1 1 3 1 3 3 1 3 3 1 3 3 1 1 3 3 1 3 3 1 3 3 3 1 3 1 3 1 3 3 3 3 3 3 1 3 3 1 3 3 1 1 3 3 3 1 1 1 3 3 1 3 3 3
 3 3 3 3 3 3 3 3
 [244] 1 3 1 3 1 1 3 3 3 1 3 1 1 3 3 3 3 3 3 1 1 1 3 3 1 1 1 1 1 3 3 1 1 1 3 3 1 2 1 1 3 3 1 3 3 1 1 3 1 3 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1
 [325] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1
 [406] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 2 1 3 1
 3 1 1 3 1 3 3 1
 [487] 1 3 2 1 3 3 3 1 3 1 2 1 3 3 1 1 3 3 1 1 1 3 1 3 3 1 3 3 3 3 1 1 1 3 3 1 1 1 1 1 3 3 3 1 3 1 3 2 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 3 1 3 3 1 1 1 3 3 1
 3 2 1 1 1 1 3 3
 [568] 3 3 1 1 3 3 1 3 3 3 1 3 1 3 3 3 3 1 1 1 1 3 1 3 3 3 1 1 3 1 3 3 3 3 3 1 1 3 3 1 3 1 1 3 1 3 3 1 1 1 1 1 3 1 3 1 1 3 3 1 1 1 1 3 3 3 1 3 1 3 3 3 1
 3 1 1 1 3 1 3 1
 [649] 3 1 1 3 1 3 3 1 3 1 3 1 3 3 3 1 3 1 3 3 3 1 1 3 1 3 3 3 1 3 3 1 1 3 3 1 3 1 1 1 3 3 3 1 3 3 1 1 3 1 1 3 3 3 3 1 3 3 3 1 1 1 1 3 1 3 3 1 3 1 3 3
 3 3 1 1 1 1 1 1
 [730] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 1 1 1 3 1 1 1 3 1 3 1 1 1 3 3 1 3 3 3 3 3 3 3 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 3 1
 1 3 3 1 3 3 3 3
 [811] 1 3 3 1 1 1 3 3 3 3 3 1 1 3 3 1 3 3 3 3 1 3 3 3 3 3 1 3 1 1 3 1 1 1 3 1 1 1 1 3 3 1 1 3 1 3 1 3 3 3 3 3 1 3 3 1 3 1 3 1 1 1 1 1 1 1 1 1 3 1 1 3 3
 3 3 3 3 3 3 3 3
 [892] 3 3 1 1 3 3 1 3 3 1 1 1 1 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1 1 3 1 1 1 3 1 3 1 1 1 1 3 1 1 1 1 3 3 1 1 1 3 3 1 3 1 1 3 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 3
 1 1 1 3 3 1 1 3
 [973] 1 1 1 1 3 1 3 1 1 3 1 1 1 1 3 3 3 3 1 1 1 1 3 3 3 3 1 3
 [ reached getOption("max.print") -- omitted 4875 entries ]

Within cluster sum of squares by cluster:
[1] 23903.82 11849.80 23360.85
 (between_SS / total_SS =  48.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

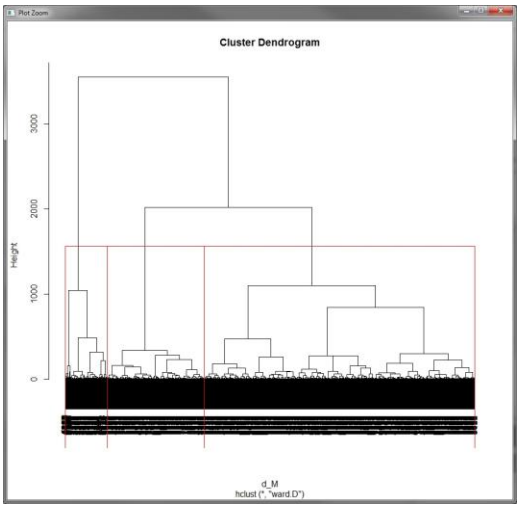
```
> print(clusters_T)
K-means clustering with 3 clusters of sizes 2100, 182, 3593

Cluster means:
         age       sex    test_time motor_UPDRS  Jitter... Jitter.Abs. Jitter.RAP Jitter.PPQ5 Jitter.DDP    Shimmer Shimmer.dB. Shimmer.APQ3 Shimmer.APQ
5 Shimmer.APQ11
1  0.2568736 0.2128571  0.008111712   0.1621872  0.3024898   0.5407840  0.3539221   0.3553362  0.3539148  0.5008642   0.4718771    0.4956231    0.432878
3     0.5243083
2 -0.2145716 0.7307692 -0.147266501   0.2725042  3.8607059   3.1119386  3.7602990   3.7111082  3.7603315  3.7684128   3.9865560    3.7793469    3.945468
7     3.4508015
3 -0.1392659 0.3581965  0.002718594  -0.1085970 -0.3723565  -0.4737042 -0.3973312  -0.3956659 -0.3973285 -0.4836254  -0.4777331   -0.4811160   -0.452858
2    -0.4812394
  Shimmer.DDA        NHR        HNR       RPDE         DFA        PPE pred_T
1   0.5001808  0.1085118 -0.6770139  0.5362720  0.52030775  0.6878608      3
2   3.7573117  4.4105508 -2.9586642  1.5376941 -0.07535145  2.4914275      2
3  -0.4826636 -0.2868342  0.5455625 -0.3913252 -0.30028731 -0.5282347      1

Clustering vector:
   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1 3 3 3 3 1 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3
  [82] 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3 3 1 1 1 1 3
 1 1 1 1 1 1 1 1
 [163] 1 1 1 1 3 1 1 1 1 1 1 1 1 3 1 1 3 3 1 3 3 1 3 1 1 3 1 1 3 1 1 3 3 1 1 3 1 1 3 1 1 1 3 1 3 1 3 1 1 1 1 1 1 3 1 1 3 1 1 3 3 1 1 1 3 3 3 1 1 3 1 1 1
 1 1 1 1 1 1 1 1
 [244] 3 1 3 1 3 3 1 1 1 3 1 3 3 1 1 1 1 1 1 3 3 3 1 1 3 3 3 3 3 1 1 3 3 3 1 1 3 2 3 3 1 1 3 1 1 3 3 1 3 1 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3
 [325] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3
 [406] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 3 1 3
 1 3 3 1 3 1 1 3
 [487] 3 1 2 3 1 1 1 3 1 3 2 3 1 1 3 3 1 1 3 3 3 1 3 1 1 3 1 1 1 1 3 3 3 1 1 3 3 3 3 3 1 1 1 3 1 3 1 2 1 1 1 1 1 3 3 1 1 1 1 1 3 3 3 1 3 1 1 3 3 3 1 1 3
 1 2 3 3 3 3 1 1
 [568] 1 1 3 3 1 1 3 1 1 1 3 1 3 1 1 1 1 3 3 3 3 1 3 1 1 1 3 3 1 3 1 1 1 1 1 3 3 1 1 3 1 3 3 1 3 1 1 3 3 3 3 3 1 3 1 3 3 1 1 1 1 3 3 3 1 3 1 1 1 3
 1 3 3 3 1 3 1 3
 [649] 1 3 3 1 3 1 1 3 1 3 1 3 1 1 1 3 1 3 1 1 1 3 3 1 3 1 1 1 3 1 1 3 3 1 1 3 1 3 3 3 1 1 1 3 1 1 3 3 1 3 3 1 1 1 1 3 1 1 1 3 3 3 3 1 3 1 1 3 1 3 1 1
 1 1 3 3 3 3 3 3
 [730] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 3 3 3 1 3 3 3 1 3 1 3 3 3 1 1 3 1 1 1 1 1 1 1 3 3 3 3 3 1 3 1 3 3 3 3 3 3 1 3
 3 1 1 3 1 1 1 1
 [811] 3 1 1 3 3 3 1 1 1 1 1 3 3 1 1 3 1 1 1 1 3 1 1 1 1 1 3 1 3 3 1 3 3 3 1 3 3 3 3 1 1 3 3 1 3 1 3 1 1 1 1 1 3 1 1 3 1 3 1 3 3 3 3 3 3 3 3 3 1 3 3 1 1
 1 1 1 1 1 1 1 1
 [892] 1 1 3 3 1 1 3 1 1 3 3 3 3 1 1 1 1 3 1 1 1 1 1 1 1 1 1 3 3 1 3 3 3 1 3 1 3 3 3 3 1 3 3 3 3 1 1 3 3 3 1 1 3 1 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 1
 3 3 3 1 1 3 3 1
 [973] 3 3 3 3 1 3 1 3 3 1 3 3 3 3 1 1 1 1 3 3 3 3 1 1 1 1 3 1
 [ reached getOption("max.print") -- omitted 4875 entries ]

Within cluster sum of squares by cluster:
[1] 23104.98 11864.34 24150.10
 (between_SS / total_SS =  50.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
> |
```

```
>  table(parkinsons_score_col_M, clusters_M$cluster)/nrow(parkinsons_clust)

parkinsons_score_col_M           1           2           3
                     1 0.222978723 0.001531915 0.096000000
                     2 0.257531915 0.024680851 0.161191489
                     3 0.128340426 0.004765957 0.102978723
> table(parkinsons_score_col_T, clusters_T$cluster)/nrow(parkinsons_clust)

parkinsons_score_col_T           1           2           3
                     1 0.086127660 0.001531915 0.199829787
                     2 0.175829787 0.025191489 0.307574468
                     3 0.095489362 0.004255319 0.104170213
> |
```

Hierarchical

Αυτός ο αλγόριθμος ξεκινά και ενώνει γειτονικά στοιχεία μέχρι να καταλήξει σε ένα σύνολο. Ανάλογα με το ποσό ψηλά ενώνονται τα υποσύνολα καταλαβαίνουμε τόσα είναι τα clusters του αλγορίθμου. Στο παράδειγμα μας έχουν χωριστεί τα σύνολα στο δέντρο με κόκκινο χρώμα και αφού έχει γίνει αυτό κόβουμε το δέντρο και έχουμε τα 3 cluster μας και μπορούμε με τον ίδιο τρόπο που έγινε και στον kmeans να υπολογίσουμε ποσοστά επιτυχίας.






```
> table(parkinsons_hier_score_M, groups_M)
                        groups_M
parkinsons_hier_score_M    1    2    3
                      1  656 1158   69
                      2  469 1720  416
                      3  271 1000  116
> table(parkinsons_hier_score_T, groups_T)
                        groups_T
parkinsons_hier_score_T    1    2    3
                      1  811  867   11
                      2 2016  796  176
                      3  902  266   30
> |
```

```
> table(parkinsons_hier_score_M, groups_M)/nrow(parkinsons)
                        groups_M
parkinsons_hier_score_M         1         2         3
                      1 0.11165957 0.19710638 0.01174468
                      2 0.07982979 0.29276596 0.07080851
                      3 0.04612766 0.17021277 0.01974468
> table(parkinsons_hier_score_T, groups_T)/nrow(parkinsons)
                        groups_T
parkinsons_hier_score_T         1         2         3
                      1 0.138042553 0.147574468 0.001872340
                      2 0.343148936 0.135489362 0.029957447
                      3 0.153531915 0.045276596 0.005106383
> |
```

Association rule mining

Σε αυτό το βήμα επειδή τα δεδομένα μας είναι αριθμητικά επρεπε να τα κάνουμε κατηγορικά χρησιμοποιόντας την συνάρτηση cut(). Αυτή η συνάρτηση μας επιτρέπει να χωρίσουμε ένα σύνολο από numeric δεδομένα σε ίσα διαστήματα έτσι ώστε να μπορεί ο apriori() αλγόριθμος που θα χρησιμοποιήσουμε στην συνέχεια να υπολογίσει πιθανότητες εμφάνισης. Το πλήθος της κάθε κατηγορίας που σχηματίζεται υπολογίστηκε εμπειρικά από τα histograms της κάθε μεταβλητής. Παρακάτω παραθέτω ένα στιγμιότυπο από την εκτέλεση του αλγορίθμου οπού υπάρχουν οι καλύτερες 50 συσχετίσεις ταξινομημένες ως αναφορά την εμπιστοσύνη που δείχνουν στον σύνολο των δεδομένων άλλα και την τιμή lift.

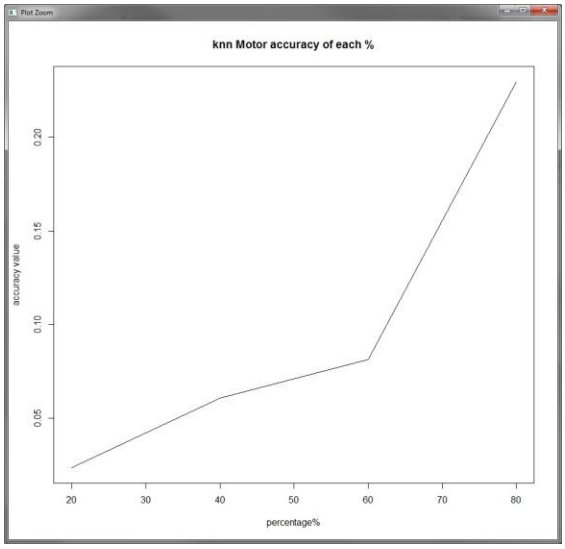
```
> inspect(head(sort(parkinson_rules, by = "lift"), 150))
      lhs                                                  rhs                 support    confidence lift     count
[1]   {Jitter.RAP=3}                                   => {Jitter.DDP=3}   0.01038298 1          96.31148  61
[2]   {Jitter.DDP=3}                                   => {Jitter.RAP=3}   0.01038298 1          96.31148  61
[3]   {age=6,total_UPDRS=1}                            => {motor_UPDRS=1} 0.02144681 1          36.26543 126
[4]   {total_UPDRS=1,HNR=7}                            => {motor_UPDRS=1} 0.01140426 1          36.26543  67
[5]   {age=6,total_UPDRS=1,PPE=2}                      => {motor_UPDRS=1} 0.01157447 1          36.26543  68
[6]   {age=6,total_UPDRS=1,Shimmer=1}                  => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[7]   {age=6,total_UPDRS=1,Shimmer.APQ11=1}            => {motor_UPDRS=1} 0.01259574 1          36.26543  74
[8]   {age=6,total_UPDRS=1,Shimmer.dB.=1}              => {motor_UPDRS=1} 0.01276596 1          36.26543  75
[9]   {age=6,total_UPDRS=1,Shimmer.DDA=1}              => {motor_UPDRS=1} 0.01242553 1          36.26543  73
[10]  {age=6,total_UPDRS=1,Jitter.Abs.=1}              => {motor_UPDRS=1} 0.01514894 1          36.26543  89
[11]  {age=6,total_UPDRS=1,Shimmer.APQ3=1}             => {motor_UPDRS=1} 0.01412766 1          36.26543  83
[12]  {age=6,total_UPDRS=1,Shimmer.APQ5=1}             => {motor_UPDRS=1} 0.01582979 1          36.26543  93
[13]  {age=6,total_UPDRS=1,Jitter.RAP=1}               => {motor_UPDRS=1} 0.02076596 1          36.26543 122
[14]  {age=6,total_UPDRS=1,Jitter.DDP=1}               => {motor_UPDRS=1} 0.02076596 1          36.26543 122
[15]  {age=6,total_UPDRS=1,Jitter.PPQ5=1}              => {motor_UPDRS=1} 0.02093617 1          36.26543 123
[16]  {age=6,total_UPDRS=1,NHR=1}                      => {motor_UPDRS=1} 0.02144681 1          36.26543 126
[17]  {age=6,total_UPDRS=1,Jitter...=1}                => {motor_UPDRS=1} 0.02144681 1          36.26543 126
[18]  {total_UPDRS=1,Jitter.Abs.=1,HNR=7}              => {motor_UPDRS=1} 0.01021277 1          36.26543  60
[19]  {total_UPDRS=1,Shimmer.APQ5=1,HNR=7}             => {motor_UPDRS=1} 0.01021277 1          36.26543  60
[20]  {total_UPDRS=1,Jitter.RAP=1,HNR=7}               => {motor_UPDRS=1} 0.01140426 1          36.26543  67
[21]  {total_UPDRS=1,Jitter.DDP=1,HNR=7}               => {motor_UPDRS=1} 0.01140426 1          36.26543  67
[22]  {total_UPDRS=1,Jitter.PPQ5=1,HNR=7}              => {motor_UPDRS=1} 0.01140426 1          36.26543  67
[23]  {total_UPDRS=1,NHR=1,HNR=7}                      => {motor_UPDRS=1} 0.01140426 1          36.26543  67
[24]  {total_UPDRS=1,Jitter...=1,HNR=7}                => {motor_UPDRS=1} 0.01140426 1          36.26543  67
[25]  {age=6,total_UPDRS=1,Jitter.Abs.=1,PPE=2}        => {motor_UPDRS=1} 0.01123404 1          36.26543  66
[26]  {age=6,total_UPDRS=1,Shimmer.APQ3=1,PPE=2}       => {motor_UPDRS=1} 0.01004255 1          36.26543  59
[27]  {age=6,total_UPDRS=1,Shimmer.APQ5=1,PPE=2}       => {motor_UPDRS=1} 0.01089362 1          36.26543  64
[28]  {age=6,total_UPDRS=1,Jitter.RAP=1,PPE=2}         => {motor_UPDRS=1} 0.01157447 1          36.26543  68
[29]  {age=6,total_UPDRS=1,Jitter.DDP=1,PPE=2}         => {motor_UPDRS=1} 0.01157447 1          36.26543  68
[30]  {age=6,total_UPDRS=1,Jitter.PPQ5=1,PPE=2}        => {motor_UPDRS=1} 0.01157447 1          36.26543  68
[31]  {age=6,total_UPDRS=1,NHR=1,PPE=2}                => {motor_UPDRS=1} 0.01157447 1          36.26543  68
[32]  {age=6,total_UPDRS=1,Jitter...=1,PPE=2}          => {motor_UPDRS=1} 0.01157447 1          36.26543  68
[33]  {age=6,total_UPDRS=1,Shimmer=1,Shimmer.APQ11=1}  => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[34]  {age=6,total_UPDRS=1,Shimmer=1,Shimmer.dB.=1}    => {motor_UPDRS=1} 0.01021277 1          36.26543  60
[35]  {age=6,total_UPDRS=1,Shimmer=1,Shimmer.DDA=1}    => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[36]  {age=6,total_UPDRS=1,Shimmer=1,Shimmer.APQ3=1}   => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[37]  {age=6,total_UPDRS=1,Shimmer=1,Shimmer.APQ5=1}   => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[38]  {age=6,total_UPDRS=1,Jitter.RAP=1,Shimmer=1}     => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[39]  {age=6,total_UPDRS=1,Jitter.DDP=1,Shimmer=1}     => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[40]  {age=6,total_UPDRS=1,Jitter.PPQ5=1,Shimmer=1}    => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[41]  {age=6,total_UPDRS=1,Shimmer=1,NHR=1}            => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[42]  {age=6,total_UPDRS=1,Jitter...=1,Shimmer=1}      => {motor_UPDRS=1} 0.01038298 1          36.26543  61
[43]  {age=6,total_UPDRS=1,Shimmer.dB.=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01174468 1       36.26543  69
[44]  {age=6,total_UPDRS=1,Shimmer.APQ11=1,Shimmer.DDA=1} => {motor_UPDRS=1} 0.01140426 1       36.26543  67
[45]  {age=6,total_UPDRS=1,Jitter.Abs.=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01157447 1       36.26543  68
[46]  {age=6,total_UPDRS=1,Shimmer.APQ3=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01191489 1      36.26543  70
[47]  {age=6,total_UPDRS=1,Shimmer.APQ5=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01259574 1      36.26543  74
[48]  {age=6,total_UPDRS=1,Jitter.RAP=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01259574 1       36.26543  74
[49]  {age=6,total_UPDRS=1,Jitter.DDP=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01259574 1       36.26543  74
[50]  {age=6,total_UPDRS=1,Jitter.PPQ5=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01259574 1      36.26543  74
[51]  {age=6,total_UPDRS=1,Shimmer.APQ11=1,NHR=1}      => {motor_UPDRS=1} 0.01259574 1          36.26543  74
[52]  {age=6,total_UPDRS=1,Jitter...=1,Shimmer.APQ11=1} => {motor_UPDRS=1} 0.01259574 1        36.26543  74
[53]  {age=6,total_UPDRS=1,Shimmer.dB.=1,Shimmer.DDA=1} => {motor_UPDRS=1} 0.01191489 1        36.26543  70
[54]  {age=6,total_UPDRS=1,Jitter.Abs.=1,Shimmer.dB.=1} => {motor_UPDRS=1} 0.01174468 1        36.26543  69
[55]  {age=6,total_UPDRS=1,Shimmer.dB.=1,Shimmer.APQ3=1} => {motor_UPDRS=1} 0.01259574 1       36.26543  74
[56]  {age=6,total_UPDRS=1,Shimmer.dB.=1,Shimmer.APQ5=1} => {motor_UPDRS=1} 0.01276596 1       36.26543  75
```

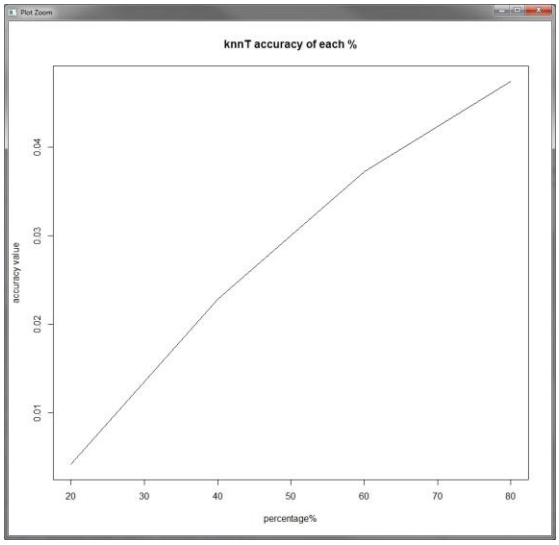
Επίδραση του δείγματος στην ποιότητα του αποτελέσματος

Σε αυτό το ερώτημα κριθήκαμε να ξανατρέξουμε τους αλγορίθμους με κομμάτια από το σύνολο των δεδομένων και να δούμε αν οι αποδώσεις του αλγόριθμου έπεσαν σε σχέση με το αρχικό σύνολο δεδομένων.

Είναι προφανές ότι η απόδοση των αλγορίθμων πέφτει όσο μικραίνουμε το data set, αυτό συμβαίνει διότι το μοντέλο δεν έχει αρκετά δεδομένα έτσι ώστε να μπορέσει να εκπαιδευτεί επαρκώς και να βγάλει ικανοποιητικά αποτελέσματα.

Αφού χώρισα το σύνολο σε κομμάτια της τάξεως των 20,40,60,80% ξαναέτρεξα τους αλγόριθμους και αυτά είναι κάποια από τα αποτελέσματα. Συγκεκριμένα από τον αλγόριθμο knn από τον ιεραρχικό αλγόριθμο.





Τέλος θεώρησα περιττό να κάνω αυτή την διαδικασία για το βήμα 5 μιας και δεν θα μπορέσουμε εύκολα να δούμε διαφορές στα αποτεέσματα αφου κάθε φορα θα είναι διαφορετικές οι συσχετισεις λόγο μικρου μεγέθους δεδομένων.

```
> scoreTableM20
                          groups_T_20
parkinsons_hier_score_T_20           1           2           3
                         1 0.224211424 0.063086104 0.003410060
                         2 0.371696505 0.105711850 0.025575448
                         3 0.156862745 0.051150895 0.005967604
> scoreTableM20
                          groups_T_20
parkinsons_hier_score_T_20           1           2           3
                         1 0.224211424 0.063086104 0.003410060
                         2 0.371696505 0.105711850 0.025575448
                         3 0.156862745 0.051150895 0.005967604
> scoreTableM40
                          groups_T_40
parkinsons_hier_score_T_40           1           2           3
                         1 0.2313013403 0.0687418936 0.0030263727
                         2 0.3674881107 0.1175961954 0.0259403372
                         3 0.1552096844 0.0562040640 0.0008646779
> scoreTableM40
                          groups_T_40
parkinsons_hier_score_T_40           1           2           3
                         1 0.2313013403 0.0687418936 0.0030263727
                         2 0.3674881107 0.1175961954 0.0259403372
                         3 0.1552096844 0.0562040640 0.0008646779
> scoreTableM60
                          groups_T_60
parkinsons_hier_score_T_60           1           2           3
                         1 0.246490736 0.028635598 0.002245929
                         2 0.363559798 0.113700168 0.029758563
                         3 0.125210556 0.067939360 0.005334082
> scoreTableM60
                          groups_T_60
parkinsons_hier_score_T_60           1           2           3
                         1 0.246490736 0.028635598 0.002245929
                         2 0.363559798 0.113700168 0.029758563
                         3 0.125210556 0.067939360 0.005334082
> scoreTableM80
                          groups_T_80
parkinsons_hier_score_T_80           1           2           3
                         1 0.159293917 0.124202467 0.003190132
                         2 0.180986814 0.290940026 0.038068907
                         3 0.043173118 0.152275627 0.005954913
> scoreTableM80
                          groups_T_80
parkinsons_hier_score_T_80           1           2           3
                         1 0.159293917 0.124202467 0.003190132
                         2 0.180986814 0.290940026 0.038068907
                         3 0.043173118 0.152275627 0.005954913
>
```