ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Γλώσσες Προγραμματισμού

Πλαίσια δεδομένων - R

Χαρά Πρασσά, chprassa@gmail.com

Acknowledgments

- Οι διαφάνειες αντιστοιχούν στο 4ο κεφάλαιο του βιβλίου ‘Ανάλυση δεδομένων με την R. Νικολάου, Χ., Εκδόσεις Δίσιγμα, 2019’.

Οι βασικές προϋποθέσεις για τη δημιουργία ενός πλαισίου δεδομένων είναι:

1. Κάθε στήλη πρέπει να περιέχει τον ίδιο αριθμό στοιχείων δεδομένων.
2. Τα ονόματα των στηλών δεν μπορούν να είναι κενά. Ακόμα και αν οι στήλες δεν έχουν όνομα, η R θα τους αποδώσει αυτόματα ένα που θα προκύπτει από τον αύξοντα αριθμό τους μετά το πρόθεμα V (Value), π.χ. $V1, $V2, κ.ο.κ.
3. Τα ονόματα των γραμμών δεν είναι υποχρεωτικά αλλά αν υπάρχουν θα πρέπει να είναι μοναδικά.
4. Τα δεδομένα που είναι αποθηκευμένα σε ένα πλαίσιο δεδομένων μπορούν να έχουν αριθμητική τιμή ή χαρακτήρα, ωστόσο θα πρέπει κάθε στήλη να περιέχει αποκλειστικά δεδομένα ενός είδους. Στην αντίθετη περίπτωση η R θα μετατρέψει τα δεδομένα σε ένα είδος τιμών μέσω εξαναγκασμού.

Το ακόλουθο dataframe είναι ένα έτοιμο σετ δεδομένων της R που περιέχει μετεωρολογικές μετρήσεις (airquality):

```
> str(airquality)
'data.frame':	153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

- Από τη δομή του dataframe βλέπουμε ότι πρόκειται για ένα σετ 153 παρατηρήσεων για 6 αριθμητικές μεταβλητές (ακέραιες ή κινητής υποδιαστολής). Η δομή του dataframe επίσης μας επιτρέπει να δούμε τα ονόματα των μεταβλητών που φαίνονται στο αριστερό μέρος και είναι στην ουσία τα ονόματα των στηλών. Καθεμιά από τις μεταβλητές είναι ένα διάνυσμα το οποίο μπορούμε να καλέσουμε με το όνομα του με τον τελεστή του $:

```
> airquality$Ozone
```

Κενές τιμές (missing values)

Παρατηρώντας το dataframe airquality, κάποια από τα στοιχεία των μεταβλητών δεν περιέχουν τιμές οπότε και εμφανίζονται με το σύμβολο NA (Non-assigned). Η ύπαρξη κενών τιμών μπορεί να δημιουργήσει προβλήματα στην εφαρμογή ακόμα και των πιο απλών συναρτήσεων. Για παράδειγμα αν θέλουμε να υπολογίσουμε το άθροισμα τιμών του διανύσματος Ozone με την συνάρτηση sum() προκύπτει το παρακάτω πρόβλημα:

```
> sum(airquality$Ozone)
 [1] NA
```

- Βλέπουμε ότι η ύπαρξη κενών τιμών στο διάνυσμα δεν επιτρέπει την εξαγωγή του αθροίσματος. Πολλές συναρτήσεις έχουν ενσωματωμένες παραμέτρους που τους επιτρέπουν να αγνοήσουν τις κενές τιμές. Για παράδειγμα:

```
> sum(airquality$Ozone,na.rm=T)
[1] 4887
```

NA removal = TRUE/εκτέλεση συνάρτησης σε πλήρεις τιμές

- Ένας εύκολος τρόπος να δούμε ποιες τιμές σε ένα διάνυσμα είναι κενές είναι η συνάρτηση is.na()

```
> is.na(airquality$Ozone)
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
[12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[23] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
```

Ισοδύναμα, με χρήση της διανυσματικής εφαρμογής θα γράφαμε:

```
> sapply(airquality$Ozone,is.na)
```

Βασικές συναρτήσεις ελέγχου τιμών:

Συνάρτηση	**Έλεγχος**
is.na	Ύπαρξη τιμής
is.infinite	Προσδιορισμένη τιμή
is.nan	Τιμή που απειρίζεται
is.integer	Η τιμή είναι ακέραιος
is.character	Η τιμή είναι χαρακτήρας
is.numeric	Η τιμή είναι αριθμητικό
is.factor	Η τιμή είναι παράγοντας
is.element	Σύγκριση συνόλων

- is.infinite(), is.nan(): ελέγχουν κατά πόσο σ' ένα διάνυσμα μια τιμή έχει αποδοθεί ως άπειρο, πραγματικός αριθμός ή αλφαριθμητικό. Η πρώτη συνάρτηση συνδέεται με την απόδοση τιμής απείρου σε κάποιες πράξεις όπως η διαίρεση με το 0, ενώ η δεύτερη ορίζει την απροσδιοριστία.

Παράδειγμα απροσδιόριστων τιμών:

```
> x<-5
> y<- 1/(x-5)
> y
[1] Inf
> is.infinite(y)
[1] TRUE
> is.nan(y)
[1] FALSE
```

```
> z<-log(0)
> z
[1] -Inf
> is.infinite(z)
[1] TRUE
> is.nan(z)
[1] FALSE
```

Παράδειγμα που προκύπτουν απροσδιοριστίες (πράξεις μεταξύ τιμών απείρου ή η εφαρμογή τιμών εκτός πεδίου ορισμού συναρτήσεων):

```
> x<-seq(-10,10,2)
> z<-sqrt(x)
Warning message:
In sqrt(x) : NaNs produced
> z
 [1]      NaN      NaN      NaN      NaN      NaN 0.000000 1.414214
 [8] 2.000000 2.449490 2.828427 3.162278
> is.nan(z)
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
> is.nan(log(0)+1/0) #πράξη μεταξύ τιμών απείρου
[1] TRUE
```

- is.integer(), is.character(), is.numeric(): ελέγχουν τον τύπο της μεταβλητής που εξετάζεται. Στα dataframes όλα τα διανύσματα είναι υποχρεωτικά του ίδιου τύπου λόγω του εξαναγκασμού που εφαρμόζει.

```
> is.numeric(iris$Petal.Length)
[1] TRUE
```

- is.factor(): ελέγχει μια ειδική κατηγορία δεδομένων που ονομάζονται παράγοντες (factors). Οι παράγοντες είναι κατηγορικές μεταβλητές που η R χρησιμοποιεί για να οργανώσει πιο πολύπλοκα αντικείμενα. Οι παράγοντες είναι διανύσματα χαρακτήρων που όμως μεταφράζονται πάντα σε κατηγορικά δεδομένα όταν χειριζόμαστε πλαίσια δεδομένων.

```
> is.factor(iris$Species)
[1] TRUE
```

- is.element(): χρησιμοποιείται για να συγκρίνει διανύσματα σε ό,τι αφορά την παρουσία κοινών στοιχείων. Εκτελεί έτσι μια σύγκριση συνόλων. Στην πιο απλή μορφή απλά ελέγχει την ύπαρξη ενός στοιχείου σε ένα διάνυσμα:

```
> specific_name <- 'Mary'
> names <- c('George','John','Mary','Peter')
> is.element(specific_name,names)
[1] TRUE
```

```
> specific_names <- c('Mary','Anna','Peter')
> names <- c('George','John','Mary','Peter')
> is.element(specific_names,names)
[1]  TRUE FALSE  TRUE
```

- Οι παράγοντες είναι στοιχεία των dataframes που αντιστοιχούν σε κατηγορικές μεταβλητές. Παρά το γεγονός ότι η R θα αποδώσει την ιδιότητα παράγοντα σε κάθε διάνυσμα χαρακτήρων μέσα σε ένα dataframe, αυτό έχει νόημα μόνο στην περίπτωση που ο αριθμός των μοναδικών τιμών μέσα στο διάνυσμα είναι περιορισμένος ώστε να επιτρέπει την ομαδοποίηση. Το πλήθος των μοναδικών τιμών μπορεί κανείς να το δει με την χρήση της συνάρτησης levels(), η οποία επιστρέφει τις τιμές σε ένα διάνυσμα χαρακτήρων:

```
> levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
```

- Στην πράξη οποιοδήποτε διάνυσμα χαρακτήρων μπορεί να μετατραπεί σε παράγοντα με την χρήση της συνάρτησης factor():

```
> a_vector <- c(rep('X',5),rep('Y',3),rep('Z',2))
> a_factor <- factor(a_vector)
> levels(a_factor)
[1] "X" "Y" "Z"
```

❑ Όταν παίρνουμε τα επίπεδα ενός παράγοντα, αυτά οργανώνονται πάντα με αλφαβητική σειρά. Στο ακόλουθο παράδειγμα δημιουργούμε ένα πλαίσιο δεδομένων με τρία διανύσματα (εισόδημα, ηλικία και επίπεδο εκπαίδευσης), από τα οποία τα δύο πρώτα είναι αριθμητικές τιμές και το τελευταίο κατηγορική μεταβλητή που μετατρέπεται αυτόματα σε παράγοντα κατά την δημιουργία του dataframe.

```
income <- c(32,51,12,23,26,27,24,26,25,18,30,22,24,21,25,27,18)
age <- c(48,59,26,23,37,32,20,25,45,55,44,42,51,30,35,29,19)
education<- c('middle','high','basic','high','high','middle','high','high','middle','basic','high',
'basic','basic','middle','middle','middle','basic')
#create the data_frame
income_data <- data.frame(income,age,education)
income_data$education <-factor(income_data$education)
levels(income_data$education)
[1] "basic"  "high"   "middle"
```

- Σε αρκετές περιπτώσεις θέλουμε μια διαφορετική σειρά στα επίπεδα του παράγοντα. Αυτό γίνεται με την αντικατάσταση του παράγοντα από έναν παράγοντα όπου τα επίπεδα ορίζονται αναλυτικά όπως στο παράδειγμα:

```
> income_data$education <-factor(income_data$education,levels=c('basic','middle','high'))

> levels(income_data$education)

[1] "basic"  "middle" "high"
```

- Η βασική χρησιμότητα των παραγόντων είναι η κατηγοριοποίηση των δεδομένων σε ένα dataframe. Η R αναγνωρίζει τις μεταβλητές που μπορούν να χρησιμοποιηθούν ως παράγοντες και μπορεί να κατατάξει τις υπόλοιπες μεταβλητές με βάση τα επίπεδα τους πριν κάνει υπολογισμούς σε διανυσματικό επίπεδο. Χαρακτηριστικές συναρτήσεις που επιτρέπουν τους υπολογισμούς αυτού του είδους είναι η by() και η aggregate().

```
> by(income_data$income, income_data$education,mean)

income_data$education: basic
[1] 18.8
------------------------------------------------------------
income_data$education: middle
[1] 26.16667
------------------------------------------------------------
income_data$education: high
[1] 30
```

- **income_data$income**: διάνυσμα στο οποίο πραγματοποιείται ο υπολογισμός
- **income_data$education**: διάνυσμα παραγόντων με το οποίο θα κάνει την κατηγοριοποίηση
- **mean**: η συνάρτηση που θα εφαρμοστεί για τους υπολογισμούς

- Αντίστοιχη λειτουργία πραγματοποιεί η aggregate() με τη βασική διαφορά ότι τα διανύσματα παραγόντων που θα χρησιμοποιηθούν δίνονται με τη μορφή λίστας. Στο ακόλουθο παράδειγμα υπολογίζουμε την μέση τιμή των ηλικιών ανά μορφωτικό επίπεδο:

```
> aggregate(income_data$age, by=list(Education=education),mean)
  Education        x
1     basic 38.60000
2      high 34.66667
3    middle 36.50000
```

- Ένα βασικό πλεονέκτημα της aggregate() είναι ότι μπορεί να εφαρμοστεί σε συνδυασμούς παραγόντων που οδηγούν σε πιο πολύπλοκες κατηγοριοποιήσεις. Η R επιτρέπει την κατηγοριοποίηση μέσω πολλαπλών παραγόντων κάνοντας χρήση της συνάρτησης table(). Η συγκεκριμένη συνάρτηση δέχεται ως είσοδο δύο ή περισσότερα διανύσματα παραγόντων και δημιουργεί έναν πίνακα σύμπτωσης που περιέχει τον αριθμό των στοιχείων των συνδυασμών τους.

```
age_class <- c('higher','higher','lower','lower','higher','lower','lower','lower','higher','higher',
'higher','higher','higher','lower','higher','lower','lower')
income_data <- data.frame(income_data,age_class)
income_data$age_class <- factor(income_data$age_class)
str(income_data)
```

Δημιουργία νέου διανύσματος factor age_class, το οποίο και προστίθεται στο dataframe income_data

```
'data.frame':          17 obs. of  4 variables:
$ income   : num  32 51 12 23 26 27 24 26 25 18 ...
$ age      : num  48 59 26 23 37 32 20 25 45 55 ...
$ education: Factor w/ 3 levels "basic","middle",..: 2 3 1 3 3 2 3 3 2 1 ...
$ age_class: Factor w/ 2 levels "higher","lower": 1 1 2 2 1 2 2 2 1 1 ...
```

- Το dataframe περιέχει τώρα δύο διανύσματα παραγόντων τα education και age_class. Η table() μας βοηθάει να απαντήσουμε σε ερωτήματα του τύπου ‘πόσοι συνδυασμοί ηλικιακών ομάδων και μορφωτικών επιπέδων υπάρχουν;’ ή ‘πόσα άτομα ανήκουν στο υψηλότερο μορφωτικό επίπεδο ενώ είναι νέοι;’ Η εφαρμογή της γίνεται πάνω στο σύνολο των διανυσμάτων ενδιαφέροντος:

```
> table(income_data$education,income_data$age_class)

        higher lower
basic        3     2
middle       3     3
high         3     3
```

Ο 3x2 πίνακας που προκύπτει περιέχει όλη την πληροφορία σχετικά με την κατανομή των στοιχείων του dataframe μεταξύ των δύο κατηγορικών μεταβλητών.

- Στη συνέχεια βλέπουμε πως μπορούμε να χρησιμοποιήσουμε την aggregate() για να υπολογίσουμε χαρακτηριστικές τιμές για αυτούς τους συνδυασμούς, κάτι που δεν μπορούμε να κάνουμε με την by().

```
> aggregate(income_data$income, by=list(Education=education,Age=age_class),mean)
```

```
  Education    Age        x
1     basic higher 21.33333
2      high higher 35.66667
3    middle higher 27.33333
4     basic  lower 15.00000
5      high  lower 24.33333
6    middle  lower 25.00000
```

Παρατηρούμε ότι υψηλότερο μέσο εισόδημα έχουν τα μεγαλύτερης ηλικίας και μορφωτικού επιπέδου άτομα.

- Μια από τις πιο σημαντικές λειτουργίες στα πλαίσια δεδομένων είναι η κατηγοριοποίηση τους σε επιμέρους σύνολα. Η συνάρτηση summary() επιστρέφει τα βασικά στατιστικά χαρακτηριστικά των δεδομένων που δέχεται ως όρισμα:

```
> summary(income_data)
     income          age        education  age_class
 Min.   :12.00   Min.   :19.00   basic :5   higher:9
 1st Qu.:22.00   1st Qu.:26.00   middle:6   lower :8
 Median :25.00   Median :35.00   high  :6
 Mean   :25.35   Mean   :36.47
 3rd Qu.:27.00   3rd Qu.:45.00
 Max.   :51.00   Max.   :59.00
```

Παρατηρούμε ότι η συνάρτηση επιστρέφει τις τιμές των κύριων τάσεων και ποσοστημορίων για κάθε αριθμητικό διάνυσμα, ενώ για τους παράγοντες επιστρέφει τις συχνότητες των στοιχείων ανάλογα με τα επίπεδα τους.

- Αν θέλουμε να δούμε με βάση ποια αριθμητική τιμή έχει γίνει ο διαχωρισμός σε age_class: higher, lower, μπορούμε να εφαρμόσουμε τη συνάρτηση εύρους στον συγκεκριμένο παράγοντα by():

```
> by(income_data$age,income_data$age_class,range)
income_data$age_class: higher
[1] 35 59
------------------------------------------------------
income_data$age_class: lower
[1] 19 32
```

- Έστω ότι μας ενδιαφέρει η σύγκριση εισοδημάτων μεταξύ άνω και κάτω των 30. Σε αυτήν την περίπτωση θα πρέπει να δημιουργήσουμε νέα υποσύνολα με βάση αυτό το αριθμητικό όριο. Χωρίς να κάνουμε χρήση των παραγόντων μπορούμε να εφαρμόσουμε απευθείας έναν λογικό, αριθμητικό έλεγχο στο διάνυσμα που μας ενδιαφέρει.

```
> income_data$age[income_data$age<=30]
[1] 26 23 20 25 30 29 19
```

- Η παραπάνω διαδικασία μπορεί να γίνει όσο πιο σύνθετη επιθυμούμε με την ενσωμάτωση λογικών πράξεων πάνω στους ελέγχους. Οι λογικές πράξεις βασίζονται στους τελεστές άλγεβρας Boole που παρουσιάζονται στον ακόλουθο πίνακα:

Τελεστής	**Πράξη**
==	Έλεγχος ισότητας
!=	Έλεγχος διαφοράς
>	Μεγαλύτερο από
<	Μικρότερο από
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο
&	Σύνδεση ενδεχομένων με ένωση
|	Σύνδεση ενδεχομένων με διάζευξη
!	Συμπλήρωμα ενδεχομένου

- Έστω για παράδειγμα ότι θέλουμε να πάρουμε τις τιμές εισοδήματος από τα άτομα με ηλικία άνω των 30 που έχουν χαμηλό μορφωτικό επίπεδο. Μπορούμε να διαμορφώσουμε ένα λογικό έλεγχο ως ακολούθως:

```
> income_data$income[income_data$age<=30 & income_data$education=='basic']
[1] 12 18
```

Λήψη υποσυνόλων από πλαίσια δεδομένων με την συνάρτηση subset()

- Ένας επιπρόσθετος εύκολος τρόπος να συνδυάσουμε λογικούς/αριθμητικούς ελέγχους με ταυτόχρονη λήψη του υποσυνόλου είναι η συνάρτηση subset() η οποία δρα πάνω σε ένα dataframe με ταυτόχρονο 'πέρασμα' των ελέγχων σαν δεύτερη παράμετρο. Αν για παράδειγμα επιθυμούμε τα δεδομένα του data_income για άτομα ηλικίας κάτω των 30, θα γράψουμε:

```
> subset(income_data, age<=30)
   income age education age_class
3      12  26     basic     lower
4      23  23      high     lower
7      24  20      high     lower
8      26  25      high     lower
14     21  30    middle     lower
16     27  29    middle     lower
17     18  19     basic     lower
```

- Αν δεν θέλουμε όλα τα διανύσματα μπορούμε να επιλέξουμε αυτά που θέλουμε:

```
> subset(income_data, age<=30,c(income,education))
   income education
3      12     basic
4      23      high
7      24      high
8      26      high
14     21    middle
16     27    middle
17     18     basic
```

Λήψη υποσυνόλων από πλαίσια δεδομένων με την συνάρτηση subset()

- Μπορούμε ακόμα να εξαιρέσουμε τα διανύσματα που δεν θέλουμε:

```
> subset(income_data, age<=30,-c(age,age_class))
  income education
3     12     basic
4     23      high
7     24      high
8     26      high
14    21    middle
16    27    middle
17    18     basic
```

- Μπορούμε τέλος να συνδυάσουμε ελέγχους με την χρήση των λογικών τελεστών σύνδεσης που είδαμε παραπάνω:

```
> subset(income_data, age<=30 & education=='high',c(age_class,income,education))
  age_class income education
4     lower     23      high
7     lower     24      high
8     lower     26      high
```

Λήψη θέσεων στοιχείων υποσυνόλων με την συνάρτηση which()

- Η συνάρτηση which() επιστρέφει τη θέση των στοιχείων που ικανοποιούν κάποιο λογικό έλεγχο. Για παράδειγμα, θέλουμε να εντοπίσουμε το υποσύνολο των ατόμων με εισόδημα κάτω από 20 και να προσθέσουμε στις τιμές του εισοδήματος ένα bonus (+2).

```
> which(income_data$income<=20)->i
> i
[1]  3 10 17
> income_data$income
 [1] 32 51 12 23 26 27 24 26 25 18 30 22 24 21 25 27 18
> income_data$income[i]<- income_data$income[i]+2
> income_data$income
 [1] 32 51 14 23 26 27 24 26 25 20 30 22 24 21 25 27 20
```

- Μπορούμε τέλος να πραγματοποιήσουμε πράξεις σε ήδη υπάρχοντα σύνολα όπως ένωση, τομή, συμπλήρωμα και διαφορές μεταξύ συνόλων με μια σειρά από συναρτήσεις που δρουν πάνω σε δύο σύνολα.

Συνάρτηση	**Πράξη**
union(a,b)	Ένωση a, b
intersect(a,b)	Τομή a, b
setdiff(a,b)	Στοιχεία του a που δεν υπάρχουν στο b
setdiff(b,a)	Στοιχεία του b που δεν υπάρχουν στο a
setequal(a,b)	Έλεγχος ταύτισης συνόλων a, b

- Δημιουργούμε 2 σειρές αριθμών που περιέχουν τους πρώτους 10 στην σειρά πρώτους αριθμούς (primes) και τους πρώτους 10 όρους της ακολουθίας Fibonacci.

```
> primes<-c(2,3,5,7,11,13,17,19,23,29)
> fibonacci<- c(1,1,2,3,5,8,13,21,34,35)
```

- Αρχικά, θα συγκρίνουμε τα 2 σύνολα με τα setequal(a,b):

```
> setequal(primes,fibonacci)
[1] FALSE
```

← FALSE γιατί τα 2 σύνολα δεν ταυτίζονται

- Στη συνέχεια, θα εξάγουμε την ένωση των 2 συνόλων με την union(a,b):

```
> union(primes,fibonacci)
 [1]  2  3  5  7 11 13 17 19 23 29  1  8 21 34 35
```

← το 1 εμφανίζεται μία φορά

- Ακολούθως, θα εξάγουμε την τομή των 2 συνόλων με την intersect(a,b):

```
> intersect(primes,fibonacci)
[1]  2  3  5 13
```

Τόσο η πράξη της τομής όσο και της ένωσης διενεργούνται σε μοναδικές τιμές των συνόλων.

- Οι αντίστοιχες πράξεις που μας δίνουν τις διαφορές των δύο συνόλων είναι:

```
> setdiff(primes,fibonacci)
[1]  7 11 17 19 23 29
> setdiff(fibonacci,primes)
[1]  1  8 21 34 35
```

οι οποίες μας επιστρέφουν και εδώ τα μοναδικά μη κοινά στοιχεία του κάθε συνόλου.