27/02/2007

תקציר הדרכה על R

כתובת לשאלות: ניתן לשלוח מיילים ל torenizh@post.tau.ac.il. במידה ונפתח פורום יעודי עבור שאלות ותשובות יש להשתמש בו ולא לשלוח שאלות במייל.

1. התקנת המערכת

1.1. **Windows:** יש להוריד מהאתר :http://cran.r-project.org, את התוכנה עבור "Windows (95 and later)" ואת החבילה "base". הקובץ נקרא R-?.?.?-win32.exe, כאשר במקום סימני השאלה יופיעו מספרים (למשל 2.4.0). יש להוריד את הגרסה העדכנית ביותר.

1.2. **UNIX מקומי:** מומלץ להתקין ע"י YUM / RMP. שם החבילה הוא "R". למשתמשי Debian על נגזרותיו השונות ניתן להוסיף Repository (למשל עבור Ubuntu edgy)
deb http://cran.r-project.org/bin/linux/ubuntu edgy
שם החבילה הוא r-base.

1.3. **UNIX באוניברסיטה:** ישנו קובץ הוראות באתר הקורס או ב http://www.cs.tau.ac.il/faq כמו כן יפורסם באותם מקומות קובץ הנחיות על השימוש ב emacs כתחליף לממשק הסטנדרטי.

2. R for Windows

2.1. מסך הטרמינל: זהו המסך הראשון העולה עם פתיחת התוכנה.

2.1.1. כתיבת פקודה: הקלדת הפקודה ליד הסימן "<" המופיע על המסך ו Enter.
אם שכחנו לסגור סוגריים / מרכאות וכו' יופיע הסימן "+" ויאפשר לנו לתקן את הטעות. כדי לבטל הקלדת פקודה אפשר להקיש Esc.

2.1.2. היסטוריה: ניתן להציג את הפקודות שהוקלדו ע"י החצים מעלה / מטה.

2.1.3. עצירה באמצע תהליך: על ידי לחיצה על הלחצן Stop.

2.1.4. copy / paste: ניתן לסמן קטעים עם העכבר ולהעתיק על ידי Ctrl+C. גזירה (Ctrl+X) תעתיק ותדביק את הקטע בשורת הפקודה.

2.2. חלון הסקריפט: בחלון זה ניתן לשמור רצף של פקודות (script)

2.2.1. פתיחת סקריפט: File -> New script / Open script.

2.2.2. הרצת שורות מהסקריפט: סימון הקטע ו Ctrl+R

2.2.3. שמירת סקריפט: Crtl+S כאשר אנחנו בחלון הסקריפט.

3. **סוגי משתנים:**

משתנה הוא היחידה הבסיסית לשמירה ועיבוד מידע ב R.

3.1. הצבת ערך למשתנה: כדי להציב ערך למשתנה בשם "x" יש להקליד: `x = ערך`

או (כיוון החץ חשוב !): `x <- ערך`

3.2. מספר:

ישנם סוגים שונים של משתנים מספריים, החל מ integer (ספרה בודדת) וכלה בכל מיני רמות דיוק עשרוני (ניתן לראות פירוט ע"י הקשת "`? double`").

ניתן להציב ערך מספרי `x = 6.776…`

או מספר מרוכב `x = 3 + 4.5i`

3.3. טקסט:

משתנה המכיל ערך טקסט נקרא string או "מחרוזת טקסט". כדי להציב ערך טקסט יש לדאוג לזוג תואם של מרכאות משני צידי הערך (" ", ' ', ` `) ולהציב: `x = "a text string"`

3.4. לוגי:

משתנה המקבל את הערכים TRUE / FALSE (אפשר גם להציב T / F). למשתנה זה חשיבות רבה בהגדרה של תנאים לוגיים (למשל שימוש בתנאי if). למעשה הוא דומה בתכונותיו למשתנה המקבל את הערכים 0/1. מציבים (ללא מרכאות): `x = T / x = F`

3.5. **וקטורים , מטריצות, מערכים:**

3.5.1. וקטורים:

וקטור הוא רשימה של משתנים *מאותו הסוג*. ניתן להגדיר וקטור של מספרים, טקסט, וקטור לוגי וכו'. ווקטור מוגדר על ידי הפונקציה ()c: `x = c(1,3,7,34,7)`

לאחר ההגדרה ניתן לגשת לתא מס' i בווקטור על ידי הנוסח: `x[i]`

או לגשת לתחום ערכים: `x[1:5], x[c(7,3)]`

(יוצגו המקומות הראשון עד החמישי או המקומות השביעי והשלישי בלבד, בהתאמה)

הערה - ניתן להגדיר וקטור של רצף מספרים מ i עד j על ידי `x = i:j`

3.5.2. פקטור:

פקטור הוא מקרה מיוחד של וקטור טקסט שהוגדרה לו מראש רשימה סופית של ערכים אפשריים. הטיפול בו דומה לטיפול בווקטור טקסט, אך יש לו תכונות נוספות שניתן להשתמש בהן, והוא דרוש לפעמים כדי להפעיל פונקציות של מיון.

ניתן להמיר וקטור טקסט לפקטור על ידי: `x = as.factor(x)`

כעת ניתן לראות את רשימת הערכים האפשריים על ידי `levels(x)`

(זהו בעצמו וקטור טקסט של ערכים).

ניתן גם לשנות בצורה גלובלית את הערכים על ידי הצבת ערך לפונקציה `levels(x)`.

למשל עבור הווקטור `x = c("boy","girl","girl","boy","boy")`

נפעיל `x = as.factor(x)`

אם נקליד `levels(x)` נקבל את הערכים `"boy", "girl"`

נוכל להפוך את הערכים בווקטור ל - 1 עבור "girl" ו - 0 עבור "boy"

על ידי הפקודה: `levels(x) = c(0,1)`

(הסדר חשוב !). התוצאה תהיה הווקטור `(0,1,1,0,0)`.

חשוב להעיר שללא קשר לתוכן הערכים האפשריים (מספר / טקסט / לוגי) הטיפול בערכי הווקטור הוא כווקטור טקסט (לא ניתן למשל לבצע סכימה גם אם אלו מספרים).

לפרטים נוספים יש להקליד "`? factor`" או "`? levels`".

3.5.3. מטריצות / מערכים:

מטריצה היא רשימה של משתנים מאותו הסוג הבנויה כטבלה בעלת אורך ורוחב ידועים מראש. מטריצה מוגדרת על ידי הפונקציה `matrix()`

למשל : `x = matrix(1:12, ncol=3)`

הביטוי ncol הוא *פרמטר* של הפונקציה (ראה סעיף 6 להסבר) ()matrix המגדיר לפונקציה לכמה עמודות לחלק את רשימת הערכים.

כדי להגיע לתא בשורה ה i בעמודה ה j : `x[i,j]`

גם כאן ניתן להגדיר טווחים בדומה לווקטור: `x[1:10,3:15]`

(יחזיר את המינור המורכב מעמודות 1 עד 10 ושורות 3 עד 15)

לפרטים נוספים על הפונקציה ואפשרויות נוספות להגדרת מטריצות ניתן להקליד "`? matrix`" ולקבל חלון עזרה.

* מערך הוא מטריצה בעלת יותר משני ממדים, ומוגדר בצורה דומה.

(ניתן לקבל עזרה על ידי הקלדת "`? array`").

3.6. Dataset – טבלת נתונים:

טבלת נתונים היא הדרך הסטנדרטית לטיפול בנתונים ב R. טבלת נתונים, בדומה למטריצה, היא אוסף של רשימות ערכים בעלות אורך זהה. בשונה ממטריצה ניתן לאגד בטבלת הנתונים סוגים שונים של רשימות ולגשת אליהם בדרך "חכמה" יותר. לכל עמודה בטבלה יש שם והיא נקראת "משתנה".טבלת נתונים מוגדרת על ידי הפונקציה `data.frame`. למשל:

```
data = data.frame(var1=c(1,4,5),var2=c("me","you","him"))
```

תיתן טבלה הנקראת `data` ובה שני משתנים בעלי השמות `var1,var2`.

שימו לב לאופן ההגדרה של המשתנים : `ווקטור = שם משתנה`

כעת ניתן לקרוא לכל משתנה בנפרד על ידי: `data$var1 / data$var2`

* כדי להציג או לשנות שמות של כותרות בטבלה ניתן להשתמש בפונקציה `colnames()`.
השימוש בה דומה לשימוש בפונקציה `levels()`, במובן שהיא משמשת גם להצגה של השדות בטבלה וגם לשינוי או הגדרה של שמות המשתנים. (הקלידו `? colnames` לפרטים).
* הגישה לתא ספציפי היא כמו במטריצה / ווקטור – אם נרצה לגשת לתא מס' 3 במשתנה השני
נוכל לרשום `data$var2[3]`
או לחילופין `data[3,2]`

3.7. ערכים חסרים – NA:

ב R ישנו ערך מיוחד השמור ל"ערכים חסרים". ערכים חסרים הם נתונים שלא נאספו מכל מיני סיבות, למשל אם ספרנו מלאי בחנות ודילגנו על מוצר מסוים לא נרשום את המלאי כ – 0 אלא נציב אותו כערך חסר. כך למשל נדע לא להכליל אותו בממוצעים או בסכומים. ניתן גם להציב ערך זה למשתנה (ללא מרכאות): `x = NA`
חשוב להדגיש כי הטיפול בערכים חסרים תלוי בפונקציה בה משתמשים ויש לבדוק ספציפית לכל פונקציה ופונקציה. טיפול לא נכון בערכים חסרים יגרום בד"כ להודעת שגיאה או להחזרת הערך NA במקום הערך המצופה. למשל אם נסכום את הערכים בווקטור המספרי x המכיל ערכים חסרים על ידי הפונקציה `sum(x)` נקבל NA. ניתן לתקן את הבעיה על ידי הוספת הפרמטר `sum(x, na.rm=T)`, אשר מורה לפונקציה להתעלם מהערכים החסרים.

3.8. המרת סוגי משתנים:

סעיף זה דרוש בעיקר לכתיבה מתקדמת של קוד ב R ולכן לא יינתן פירוט נרחב במדריך זה. ב R קיימת צורה כללית של פונקציות הממירות משתנים מסוג לסוג – למשל להמיר את מחרוזת הטקסט "12" למספר 12. הצורה הכללית היא `as.type(x)`
כאשר type יכול להיות:
* numeric – להמרת טקסט למספר או משתנה לוגי לערכים 0/1
* character – להמרת משתנים לטקסט
* logical – ממיר ווקטור של 0/1 לווקטור לוגי (למשל לצורך שימוש כפילטר – ראו פרק 4)
* matrix – ממיר טבלאות נתונים למטריצות. דרוש לפעמים כקלט לפונקציות (ראו פרק 6)
* data.frame – ממיר מטריצות לטבלאות נתונים, דרוש גם הוא לפונקציות מסוימות

כרגיל – ניתן להקליד `? as.type`
עם הערך המבוקש כדי לקבל מסך עזרה מפורט.

3.9. <u>יחסים בין משתנים :</u>

ניתן להשוות בין ערכים של משתנים. התוצאה תהיה בעצמה משתנה לוגי בעל הערך FALSE/TRUE. ישנם סימנים ברורים להשוואה בין משתנים מספריים:

<, >, <=, >=

כדי לבדוק שוויון יש להשתמש ב סימן "==": `1 == 2` (נקבל FALSE)

(הסימן "=" שמור להצבת ערכים).

כדי לבדוק אי שוויון יש להשתמש ב "=!": `1 != 2` (נקבל TRUE)

סימן נוסף הוא `%in%` הבודק האם x מתאים לאחד מהערכים בווקטור y: `x %in% y`

ניתן כמובן להציב את התוצאה למשתנה לוגי : `x <- (1==2)`

(מומלץ להשתמש בסוגריים כדי למנוע שגיאות)

פעולה זו זהה למעשה להצבה : `x = F`

* <u>השוואה בין וקטורים / מטריצות / טבלאות נתונים:</u> ההשוואה מתבצעת ערך-ערך. ניתן להשוות רק בין רשימות בעלות ממדים זהים. התוצאה שתתקבל תהיה וקטור / מטריצה באותם ממדים כאשר בכל תא ישנו הערך F/T לפי ערך התא המתאים.

4. <u>פילטרים:</u>

פילטר הוא משתנה לוגי המשמש לסינון ערכים. כך נוכל "לחלץ" מתוך וקטור או משתנה בטבלה את כל הערכים העונים לקריטריון מסוים. למשל - var1 הוא משתנה בטבלה data ואנחנו רוצים לקבל את כל הערכים ב var1 הגדולים מ 3 וקטנים או שווים ל 5.

הנוסחה לפילטר תראה כך : `data$var1>3 & data$var1<=5`

(& הוא סימן "וגם", | הוא סימן "או". ניתן להשתמש בכל תנאי ההשוואה מסעיף 3.9)

נוסחה זו תחזיר לנו וקטור באורך של var1 עם הערכים T/F לפי הערך בכל תא ב var1.

נוכל כעת ליישם את הפילטר: `x.filtered = x[data$var1>3 & data$var1 <=5]`

(שימו לב – הפילטר מופיע בתוך סוגריים מרובעים)

* שימוש חכם בפילטר יכול להיות סינון ערכים מרשימה אחת לפי ערכים המופיעים ברשימה אחרת. למשל, אם יש לנו טבלה data ובה משתנים sex ו age, ניתן על ידי הפילטר לסנן את כל הגילאים המשויכים לבנים בנפרד: `data$age[data$sex == "Boy"]`

ניתן להשתמש בפילטרים כדי לסנן טבלאות ומטריצות ע"י הצבת הפילטר במקום המשמש לקריאה לשורות (שימו לב לפסיק אחרי הפילטר): `data[data$sex == "Boy",]`

התוצאה תהיה טבלה בעלת אותו מספר עמודות שתכיל רק את השורות העונות על הקריטריון.

5. **פקודות מתמטיות:**

לכל אחת מהפונקציות המתמטיות יש חלון עזרה המסביר מהי הפקודה וכיצד להשתמש בה (וראו סעיף 7 על השימוש המפורט בעזרה). יש להקליד "`? command`" (ללא המרכאות, כש command הוא שם הפקודה) על מנת לקבל הסבר מפורט.

* הפעולות המתמטיות הבסיסיות דומות לאלה שבמחשבון / אקסל (^, +, - וכו').

* חיבור/ כפל/ חזקה וכו' בוקטורים או מטריצות נעשים תא – תא.

* `? Math` - מציג מסך עזרה לפעולות מתמטיות מתקדמות יותר כמו כפל מטריצות.

* ממוצע – `mean`, סכום – `sum`, שונות – `var`. שימו לב כי אלו פונקציות ולא פקודות ולכולן יש פרמטר `rm.na` המאפשר התעלמות מערכים חסרים.

6. **פונקציות:**

פונקציה היא רצף מוכן מראש של פעולות הפועל על הקלט של הפונקציה.

6.1. סינטקס:

ברוב המקרים פונקציה מקבלת קלט (משתנה / וקטור / ...) ומחזירה פלט (משתנה / וקטור / ...). פונקציה יכולה לקבל גם פרמטרים המגדירים לה כיצד לטפל בקלט, וראינו כבר כיצד הוספת rm.na=T לפונקציה sum גורמת להתעלמות מערכים חסרים. שימו לב כי פונקציה יכולה לקבל יותר מקלט אחד ומספר גדול של פרמטרים.

הפונקציה מופעלת כך : `y = function(x, param1=T, param2=10)`

או `y <- function(x, param1=T, param2=10)`

כאשר : function הוא שם הפונקציה, x הוא הקלט ו param1, param2 הם פרמטרים (ויכולים להיות מסוגים שונים). נראה בסעיף 8 פונקציות כמו `plot` הדורשות מספר גדול של פרמטרים.

חשוב: הקלט והפלט של פונקציות שונות מוגבל מראש לסוג מסוים של משתנים. למשל פונקציות מסוימות מקבלות אך ורק מטריצות וניסיון להפעיל אותן על טבלאות יחזיר שגיאה. יש להקפיד להתאים בין הפלט והקלט לפונקציה.

6.2. **נוסחאות:**

פונקציות רבות ב R דורשות נוסחה לשם הפעלתן (למשל הפונקציה lm המטפלת במודלים ליניאריים).נוסחה היא סוג של משתנה העומד בפני עצמו, וניתן להמיר מחרוזות טקסט לנוסחאות על ידי הפונקציה `as.formula`.

לנוסחה פורמט קבוע בד"כ: `y ~ x(1) יחס ... יחס x(k-1) יחס x(k)`

* y,x(1),…,x(k) יכולים להיות וקטורים, שדות, מטריצות וכו', והיחסים הם בד"כ מתמטיים.

* הסימן "~" הוא הסימן החשוב בנוסחה ומגדיר בד"כ את ה"כיוון" - למשל ברגרסיה הוא מגדיר מהוא המשתנים המסבירים (מצד ימין של "~") ומהו המשתנה המוסבר (מצד שמאל של "~").

6.3. אובייקטים מורכבים:

יצירה ושימוש מתקדם באובייקטים ב R לא יידון במדריך זה. אולם פונקציות רבות ב R משתמשות בטכניקה זו על מנת ליצור פלט מורכב. לצורך הפשטה, ניתן להסתכל על "אובייקט" כעל אוסף של משתנים, טבלאות, מטריצות וכו'. בניגוד לטבלאות נתונים אין חשיבות לאורך המשתנה או לסוג וניתן לאגד בתוך אובייקט משתנים מסוגים שונים.

למשל, הפונקציה lm יוצרת אובייקט של מודל ליניארי המכיל בתוכו מספר רב של נתונים.

האובייקט הוא הפלט של הפונקציה (y): `y = lm(x1 ~ x2 + x3)`

כעת ניתן לגשת את הנתונים המאוחסנים בתוך y. הגישה אל הנתונים היא כמו אל משתנים בטבלת נתונים על ידי הסימן $: `... y$residuals / y$rank`

הערכים האפשריים תלויים כמובן בסוג הפונקציה והם מופיעים במערכת העזרה תחת הסעיף Value (מקלידים `? function` ומקבלים חלון עזרה).

6.4. הצגה "חכמה" של פלט:

במקרים בהם פונקציה מחזירה אובייקט מורכב, נרצה לפעמים לקבל סיכום מסודר של הנתונים.

ב R ישנה מערכת המאפשרת הצגה של סיכום הנתונים באובייקט מורכב. תפקוד המערכת וסוג הפלט שיוצג תלוי כמובן בפלט של הפונקציה הספציפית.

אם השתמשנו בפונקציה: `y = function(x, param1,param2)`

יש להקליד: `summary(y)`

בד"כ יתקבל פלט מסודר של כל הנתונים שמכיל האובייקט y. כעת ניתן להעתיק אותו ולהשתמש בו באפליקציות אחרות (למשל Word, וראו סעיף 8 לפרטים).

כמו כן ניתן להשתמש בפונקציה plot בצורה לא סטנדרטית: `plot(y)`

במידה והוגדרו מראש גרפים בתוך האובייקט הפקודה תציג בחלון הגראפי מספר גרפים רלוונטיים (וראו סעיף 9 לדרך בה ניתן לייצא / לשמור גרפים).

שימו לב: כדי לדפדף בין הגרפים האפשריים יש לחזור לחלון הטרמינל ולהקיש Enter.

6.5. חבילות חיצוניות של פונקציות:

ב R קיימת האפשרות לטעון חבילות של פונקציות אשר אינן כלולות באוסף הפונקציות הבסיסי של התוכנה. קיים מגוון עצום של חבילות של פונקציות הניתנות להורדה באמצעות האינטרנט או להתקנה עצמית.

* על אופן הטיפול בחבילות חיצוניות ראו סעיף 10.

* על בעיות ופתרונות בהצגת עזרה או בחיפוש פונקציות כאלו ראו סעיף 7 (בפרט 7.3).

7. <u>שימוש בעזרה:</u>

R מכיל אלפי פונקציות שונות והחוכמה היא לדעת באילו פונקציות להשתמש כדי לקצר תהליכים. פונקציה טובה יכולה להיות תחליף לשורות קוד רבות, וברוב המקרים תעשה את העבודה טוב יותר מקוד שנכתב על ידנו. הדרך הנוחה ביותר להתמצא בסבך הפונקציות ב R היא שימוש במערכות העזרה השונות.

7.1. עזרה ישירה:

במידה וידוע שם הפונקציה ניתן להקליד `? function`

ולקבל מסך עזרה מפורט הכולל דוגמה פשוטה בסופו.

<u>העזרה מחולקת בד"כ ל 10 חלקים :</u>

- **Description:** תיאור כללי של הפונקציה.
- **Usage:** סינטקס – פירוט כיצד יש לכתוב את הפונקציה בשורת הפקודות.
- **Arguments:** הסבר על כל אחד מהפרמטרים. חשוב – יש להיעזר בסעיף זה כדי להתאים את סוג הקלט לפונקציה.
- **Details:** הסבר מפורט יותר על הדרך שבה יש להשתמש בפונקציה.
- **Value:** מפרט מהו הפלט שמחזירה הפונקציה. אם מדובר באובייקט מורכב (כמו אובייקט רגרסיה) יפורטו מהם השדות אותם מכיל האובייקט. ניתן לקרוא לשדות על ידי שימוש בסימן ה $ אחרי שם האובייקט (כמו בטבלת נתונים): `object$value`
- **Notes, Authors, References**: ...
- **See Also:** מפרט מספר פונקציות רלוונטיות שניתן לחפש עליהן עזרה (בד"כ שימושי מאוד!)
- **Examples:** בד"כ הסעיף החשוב ביותר. נותן דוגמה קצרה לדרך בה ניתן להשתמש בפונקציה בתוך סקריפט.

7.2. חיפוש עזרה:

ניתן לבצע חיפוש מורחב בדפי העזרה על ידי הפונקציה `help.search()`. למשל לחיפוש כל הפונקציות הקשורות לפעולות על מטריצות נקליד: `help.search("matrix")`

החיפוש נעשה על מחרוזת טקסט ולכן יש להזין את הערך לחיפוש עם מרכאות.

מתקבלת רשימה של כל הפונקציות הקשורות. לאחר קבלת הרשימה ומציאת הפונקציה הרלוונטית יש לחזור לחלון הטרמינל ולחפש עזרה במערכת החיפוש הפנימית.

שימו לב – בצמוד לשם של כל פונקציה (בתוך סוגריים) מופיע שם החבילה אליה הפונקציה שייכת (וראו סעיף 10 על השימוש בחבילות).

7.3. עזרה על פונקציות מחבילות חיצוניות:

על מנת להציג את מסך העזרה של פונקציה מחבילה חיצונית (package) ראשית יש לטעון את החבילה. אם נבקש עזרה על פונקציה מחבילה חיצונית לפני שנטענה החבילה אליה היא שייכת נקבל הודעת שגיאה כאילו הפונקציה לא קיימת.

זוהי כמובן בעיה – כיצד ניתן לדעת לאיזו חבילה שייכת הפונקציה אותה אנו מחפשים ?
התשובה לכך היא פשוטה – יש לבצע חיפוש מורחב: `help.search("function")`
נקבל חלון ובו רשימה של פונקציות כאשר ליד כל פונקציה מופיע בתוך סוגריים שם החבילה אליה היא שייכת. כעת ניתן לחזור לחלון הטרמינל ולטעון את החבילה: `library(package)`
להסבר מפורט יותר על השימוש בחבילות ב R ראו סעיף 10.

7.4. מערכות חיצוניות:

ישנן מספר מערכות חיפוש המכילות שאלות ותשובות לגבי השימוש ב R. אחת מהן נמצאת באתר http://www.r-project.org - תחת Documentation ניתן למצוא מדריך כתוב (manual) ואוסף שאלות ותשובות (FAQ). כמו כן חיפוש ב – google עם המלה "R-help" והשאלה תחזיר תוצאות ממאגרים נוספים.

8. קריאה וכתיבה לקבצים

8.1. sink:

הפקודה ()sink מאפשרת לשמור את הפלט של פונקציות לתוך קובץ טקסט חיצוני.
מרגע הפעלת הפקודה `sink(file="myfile.txt")`
כל הפלט שניצור יכנס לתוך הקובץ myfile.txt. כדי לסגור את הקובץ ולחזור לתצוגה רגילה בטרמינל יש להפעיל את הפקודה `sink()`
הערה: הפקודה sink שימושית בעיקר כאשר עובדים בסביבת UNIX. בסביבת Windows קל יותר להציג את הפלט בטרמינל, לסמן עם העכבר ולהעתיק עם Ctrl+C. מומלץ להשתמש בפקודה summary כדי להציג תוצאות בפורמט מסודר (ראו סעיף 6.4).

8.2. read.table: משמשת לקריאה של קבצי טקסט או טבלאות בפורמט מוגדר מראש.

הפלט של הפונקציה read.table הוא טבלת נתונים. למשל, אם נשמור טבלה ב Excel בפורמט "Text (tab delimited)" בקובץ mytable.txt, נוכל לקרוא אותו לתוך הטבלה table1 על ידי הפקודה:

```
table1=read.table(file="mytable.txt", sep="\t", header=T, na.string=NA)
```

הסבר קצר על הפרמטרים:

* file - שם הקובץ כמחרוזת טקסט (יש לכתוב בין מרכאות)
* sep – הסימן המפריד בין עמודות בקובץ. בפורמט "Text (tab delimited)" ב Excel זהו סימן ה tab שמיוצג ב R על ידי המחרוזת "\t".
* header – לוגי. קובע האם השורה הראשונה בקובץ תשמש כשמות למשתנים בטבלה.
* na.string – משמש לקביעת הערך של הערכים החסרים. כברירת מחדל זהו הערך NA אך ניתן לציין ערכים אחרים בהתאם לצורך.

8.3. write.table:

משמשת לכתיבה של טבלאות בפורמט ידוע מראש. למשל כדי שנוכל לקרוא טבלה ב Excel בפורמט "Text (tab delimited)" נשתמש בפקודה:

```
write.table(table1, file="mytable2.txt", sep="\t", col.names=T, na=" ")
```

לאחר מכן נפתח את הקובץ ב Excel ונבחר את המפריד tab (separator). פונקציה זו מקבלת גם טבלאות וגם מטריצות.

8.4. קבצים בינאריים:

לאחר שהגדרנו מערכת של משתנים, נתונים וכו' ניתן לשמור את סביבת העבודה בשלמותה לתוך קובץ בינארי על ידי הפונקציה: `save.image(file="myfile.bin")`

או: `save(…, file="myfile.bin")`

על מנת לשמור רשימה ספציפית (במקום ה ...) של משתנים לקובץ.

לאחר מכן ניתן לטעון את כל סט המשתנים על ידי: `load(file="myfile.bin")`

חשוב: שימוש בקובץ בינארי הוא אמנם נוח ומהיר ומאפשר שמירה של סביבת עבודה שלמה, כולל אובייקטים מורכבים יותר מאלה שהכרנו עד עכשיו. אולם, לשימוש בצורה זו חסרון בולט אחד – לא ניתן יהיה לשחזר נתונים מהקובץ באמצעות תוכנות אחרות (למשל לא ניתן יהיה לקרוא את הנתונים ב excel).

9. הצגה גראפית של נתונים

9.1. הפונקציות plot / hist / pie...

ניקח לדוגמה את הפונקציה plot. הפלט של פונקציה זו הוא תיאור גרפי של הקלט. למשל :

```
x = 1:100
y = x^2
plot(x, y, type="l")
```

נקבל קו בקטע [1,100] המתאר את הפונקציה $y = x^2$.

נוכל גם להוסיף קו על גבי הגרף הקיים באמצעות הפונקציה lines

```
lines(x, x^3, type="p")
```

על הגרף יתווספו נקודות (ולא קו) המייצגות את הפונקציה $z=x^3$

9.2. פרמטרים שימושיים:

* main – מחרוזת טקסט. קובע את הכותרת הראשית לגרף

* xlab, ylab – מחרוזת טקסט. קובע את השמות שיוצמדו לציר ה x / y בגרף.

פרמטרים אלה עובדים גם עם הפונקציות lines, points:

* type – מחרוזת טקסט. "type="l יצייר קו חלק, "type="p רק נקודות, "type="b יצייר גם וגם.

* col – מספר מ 1 עד 16. קובע את צבע הקו שיצויר.

* lty – מספר מ 1 עד 16. קובע את סוג הקו (אחיד, נקודות, מקווקוו) שיצויר בגרף

* pch – ערך מספרי. קובע את סוג הנקודה שתצויר (עיגול מלא / ריק, משולש, ריבוע וכו').

9.3. חלון חדש – x11()

בכל פעם שנשתמש בפקודה plot ימחק הגרף בחלון הפעיל ויוצג הגרף החדש שיצרנו. ניתן לעבוד עם מספר רב של חלונות גראפיים על ידי הפקודה: `x11()`

פקודה זו פותחת חלון גרפי ריק ועליו יוצגו הגרפים החדשים.

9.4. par – פרמטרים גראפיים נוספים:

לפונקציות הגראפיות השונות יש עשרות פרמטרים, החל מעובי או צורת הקו בגרף וכלה בקנה המידה של הצירים. כל הפרטים מופיעים במסך העזרה של הפקודה par (מקלידים "par ?").

9.5. יצוא גרף לקובץ (Windows):

ניתן לבחור את חלון הגרף ולהעתיק אותו (Ctrl+C) ולהדביק אותו במקום הרצוי. ניתן גם להקליק עם הכפתור הימני בעכבר על הגרף הפעיל ולבחור הדפסה או שמירה בפורמטים שונים – מומלץ להשתמש בפורמט metafile.

9.6. יצוא והדפסה באמצעות פקודות (Windows ו UNIX):

ניתן לשמור גרפים באמצעות מספר פקודות בסקריפט (ראו פרטים ודוגמאות במערכת העזרה):

* ב Windows ניתן להשתמש ב savePlot לשמירה ו win.printer להדפסה.

* ב UNIX ניתן להשתמש ב postscript להדפסה ושמירה וב png או jpeg לשמירה.

10. התקנת "חבילות" – Packages

סעיף זה מיועד אך ורק לסטודנטים בקורסים מתקדמים הנדרשים להשתמש בפונקציות מתקדמות שאינן קיימות ב R בתצורה הבסיסית. הסעיף מיועד למי שיש לו הרשאות להתקין חבילות על המחשב הפרטי (בעבודה או בבית).

על מערכות המחשב באוניברסיטה לא ניתן להתקין חבילות באופן עצמאי ולשם כך, **לאחר** שווידאתם שחבילה אינה מותקנת, יש לפנות במייל ל - torenizh@post.tau.ac.il ולבקש התקנה של חבילה ספציפית, תוך פירוט שם, שם הקורס והסיבה להתקנה.

מערכת R מאפשרת הוספת "חבילות" של פונקציות ממקור חיצוני.

האתר http://cran.r-project.org מכיל מאגרים של חבילות במגוון נושאים מתקדמים, וכן קיימים אתרים נוספים המאפשרים התקנה של חבילות. ל R יש מערכת פנימית המאפשרת התקנה ותחזוקה אוטומטית של תוספות אלו.

10.1. **לפני התקנת חבילה**: ראשית יש לוודא שהיא אינה מותקנת כבר על המערכת. הדרך הפשוטה ביותר היא לנסות ולטעון את החבילה על ידי הפקודה: `library(pkg_name)`

כאשר pkg_name הוא שם החבילה.

10.2. התקנת חבילות:

באמצעות הפקודה: `install.packages(pkgs = "pkg_name")`

כאשר pkg_name הוא שם החבילה הדרושה (את השם תקבלו מהמרצה או שניתן למצוא אותו תחת Packages באתר http://cran.r-project.org.

לאחר הקשה על Enter יפתח חלון ובו תתבקשו לבחור אתר ממנו תורד החבילה. מומלץ לבחור באתר הנמצא בארה"ב יסומן באותיות USA. אני ממליץ על USA [CA 2].

לאחר מכן יתבצע תהליך ההורדה וההתקנה באופן אוטומטי. יתכן ותתבקשו לאשר את ההתקשרות לאינטרנט בגלל הגדרות אבטחה מקומיות (של ה firewall).

* ל R יש נתיב (path) המשמש כברירת מחדל להתקנה של חבילות. אם ברצונכם להתקין חבילות במקום אחר ניתן לעשות זאת על ידי פירוט נתיב אחר כפרמטר:

```
install.packages(pkgs = "pkg_name", lib="library_name")
```

חשוב ! כאשר מפרטים את הנתיב לספריה יש להקפיד להשתמש בסימן "/" או בסימן "\\" במקום הסימן "\" כמפריד בין הספריות. שימוש במפריד "\", למשל c:\data\r_lib, יחזיר הודעת שגיאה. הפורמט הנכון הוא c:\\data\\r_lib או (עדיף) c:/data/r_lib.

(הסיבה לצורת כתיבה זו קשורה לאופן שבו R מטפל בסימן "\" ולא נדון בה כאן).

10.3. טעינת חבילה:

הפקודה היא: `library(pkg_name)`

שימו לב – לא ניתן להציג עזרה על פונקציות הכלולות בחבילה מסוימת לפני שטוענים אותה.

<u>* טעינת חבילה מספריה לא סטנדרטית:</u>

אם התקנתם את החבילה בספרייה אחרת מספריית ברירת המחדל (עם הפרמטר "lib" בעת ההתקנה) יש להוסיף פרמטר המכיל את מיקום הספרייה.

הקריאה לספריה package1 שהותקנה ב c:\data\r_lib תעשה כך:

```
library(package1, lib.loc="c:/data/r_lib")
```

שימו לב כי גם כאן שם הספרייה הוא במרכאות וללא שימוש בסימן "\", כמו בסעיף 10.2 ניתן גם לקרוא לפונקציה עם הסימן "\\":

```
library(package1, lib.loc="c:\\data\\r_lib")
```