Rakendustarkvara: R. Sügis 2017, 7. praktikum*

1 Andmetöötlus paketiga plyr

Tihti soovime teostada mingit analüüsi andmestiku erinevatel alamosadel (nt arvutada meeste ja naiste keskmist palka koos usaldusvahemikuga). Selleks tuleb:

- andmestik jagada alamosadeks
- igal alamosal teostada soovitud toiming
- kombineerida tulemused kokku lihtsasti loetavaks ja töödeldavaks

R-i baaspaketiga on kaasas tükiviisiliseks andmetöötluseks sobilikud käsud `by(.)`, `apply(.)` jt. Neil käskudel on erinev süntaks ning erinev väljund (mis sageli on raskesti töödeldav). Paketis **plyr** on aga proovitud neid käske ühtlustada (baaspaketi funktsioonidele on loodud nn *wrapper*-funktsioonid). Selle paketi käsud kujul `?!ply`, kus ? on ette antava sisendi tüüp (maatriks, andmetabel, list) ja ! on soovitava väljundi tüüp (väljundit võib ka mitte olla). Allolevas tabelis on toodud ära käsud, mida erineva sisendi ja väljundi puhul kasutada:

sisend/väljund	array	data frame	list	nothing
array	`aaply(.)`	`adply(.)`	`alply(.)`	`a_ply(.)`
data frame	`daply(.)`	`ddply(.)`	`dlply(.)`	`d_ply(.)`
list	`laply(.)`	`ldply(.)`	`llply(.)`	`l_ply(.)`

1.1 Sisendi jagamine tükkideks

Kui sisendiks on list, siis seda saab jagada ainult elementideks.

Kui sisendiks on maatriks (*array*), siis seda saab jagada üksikuteks ridadeks või üksikuteks veergudeks, milleks tuleks `a!ply(.)` käsu argumendi `.margins` väärtuseks anda vastavalt 1 või 2.

Kui sisendiks on andmetabel (see on kõige tüüpilisem olukord), siis seda võib tükkideks jagada mingi tunnuse väärtuste alusel (nt haridustase) või erinevate tunnuste väärtuste kombinatsioonide järgi (nt sugu ja haridustase). Selleks tuleb `d!ply(.)` käsu argumendi `.variables` väärtuseks anda vastavate veergude nimed.

1.2 Teostatava funktsiooni määramine

Kui andmestik on jagatud tükkideks, siis peame neile rakendama mingit operatsiooni. Kõikidel `?!ply(.)` funktsioonidel on argument `.fun`, millega saab määrata tükil rakendatava funktsiooni nime (nimi kirjutada ilma jutumärkide ja argumentideta). Tihtipeale on väga kasulikud lihtsad funktsioonid nagu `length`, `nrow` või `mean`. Kui need rakendatavad funktsioonid vajavad lisaargumente (nt `na.rm`), siis saab ka need ette anda.

```
library(plyr) # Kui pole, tuleb installida
# teeme näite jaoks andmetabeli
vr <- data.frame(nimi = c("Mari", "Jaan", "Jüri"), kaal = c(68, 65, 100),
                 pikkus = c(170, 180, 190), sugu = c(2, 1, 1), pulss0m = c(70, 80, 80),
                 pulss10m = c(130, 120, 190), pulss30m = c(150, 120, NA))
```

*Praktikumijuhendid põhinevad aine MTMS.01.092 Rakendustarkvara: R (2 EAP) materjalidel, mille autorid on Mait Raag ning Raivo Kolde

```
# laply arvutab iga elemendi(andmeveeru) keskmise (sealjuures na.rm = T) ja tagastab vektori
laply(vr[, 5:7], mean, na.rm = T) # data.frame on tegelikult list, elementideks veerud
```

```
## [1]  76.66667 146.66667 135.00000
```

Tihti soovime, et ühel tükil tehtaks mitu operatsiooni. Siis on võimalik ise koostada funktsioon (tutvume sellega hiljem) ja see argumendile `.fun` ette anda. Paketiga **plyr** on aga kaasas üks mugav funktsioon `summarise(.)`, millele saab anda mitu erinevat funktsiooni ning määrata ära, kuidas nende funktsioonide tulemused nimetatakse:

```
# leiame pulsi baasmõõtmise kohta karakteristikud
summarise(vr,
          n = length(pulss0m),
          naisi.osakaal = round(sum(sugu == 1)/sum(!is.na(sugu)), 2),
          kesk0 = round(mean(pulss0m), 1),
          sd0 = round(sd(pulss0m), 1),
          haare0 = max(pulss0m) - min(pulss0m))
```

```
##   n naisi.osakaal kesk0 sd0 haare0
## 1 3          0.67  76.7 5.8     10
```

Kombineerides `summarise(.)` käsku `ddply(.)` käsuga, saame mugava vahendi mitmesuguste grupikokkuvõtete leidmiseks.

```
# esmalt paneme kõik pulsi mõõtmistulemused ühte veergu
m <- melt(vr, measure.vars = 5:7) # reshape2:melt

# nüüd leiame  kõigi kolme pulsi mõõtmise kohta karakteristikud
ddply(m, "variable", summarise,
          n.kokku= length(value),
          naisi.osakaal = round(sum(sugu == 1)/sum(!is.na(sugu)), 2),
          kesk =round( mean(value, na.rm  = T), 1),
          sd = round(sd(value, na.rm = T), 1),
          haare = max(value, na.rm = T) - min(value, na.rm = T))
```

```
##   variable n.kokku naisi.osakaal  kesk   sd haare
## 1  pulss0m       3          0.67  76.7  5.8    10
## 2 pulss10m       3          0.67 146.7 37.9    70
## 3 pulss30m       3          0.67 135.0 21.2    30
```

```
# arvutame iga inimese keskmise ja minimaalse pulsi
ddply(m, "nimi", summarise,
      keskmine = round(mean(value, na.rm = T), 1),
      miinimum = min(value, na.rm = T))
```

```
##   nimi keskmine miinimum
## 1 Jaan    106.7       80
## 2 Jüri    135.0       80
## 3 Mari    116.7       70
```

1.3 Tulemuste ühendamine

Mistahes tüüpi tulemusi saab alati kokku panna ühte listi. Enamasti soovime väljundiks aga saada andmetabelit. Sellisel juhul argumendile `.fun` ette antav funktsioon peab tagastama `data.frame` tüüpi objekti (`summarise(.)` funktsioon seda ka teeb).

1.4 Näiteid

Näidetes on andmestikuks Massachusettsi osariigi valikuuringu andmed

```
# leida igale inimesele, mis tunnuste osas on tal puuduvad väärtused
# tulemuseks on list, sest vastusevektorid võivad olla eri pikkusega
# argument 1 näitab, et töödelda tuleb igat RIDA eraldi
alply(mass, 1, function(x) names(x)[is.na(x)])[1:4]
```

```
## $`1`
## [1] "COW"  "WKHP" "INDP" "OCCP"
##
## $`2`
## [1] "MARHT"
##
## $`3`
## [1] "COW"  "WKHP" "INDP" "OCCP"
##
## $`4`
## character(0)
```

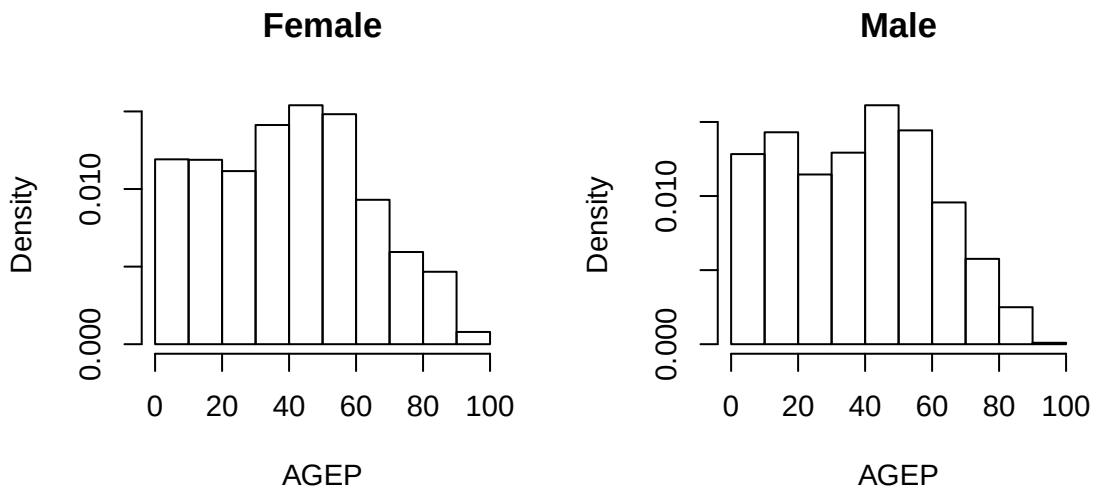
```
# Mitu puuduvat väärtust on igas veerus:
aaply(mass, 2, function(x) sum(is.na(x)))
```

```
##  AGEP   CIT   COW  DDRS   ESR    id  INDP  INTP  LANX   MAR MARHT   MIG
##     0     0  2292   342  1182     0  2582  1111   342     0  2760    68
##   MIL  OCCP PINCP RAC1P  SCHL   SEX  WAGP  WKHP
##  1274  2950  1111     0  1095     0  1111  2678
```

```
# Erineva soo ja päritoluga inimeste arv, keskmine palk ja töötundide arv:
ddply(mass, c("SEX", "CIT"), summarise,
           n = length(WAGP),
        palk = round(mean(WAGP, na.rm = T), 1),
       tunde = round(mean(WKHP, na.rm =T), 1))
```

```
##       SEX                              CIT    n    palk tunde
## 1  Female Born abroad of American parent(s)   31 24903.8  32.9
## 2  Female    Born in PR, Guam, the U.S. VIs   46 13863.9  36.8
## 3  Female                  Born in the U.S. 2785 24878.6  34.7
## 4  Female         Not a citizen of the U.S.  211 21523.2  36.5
## 5  Female    U.S. citizen by naturalization  226 23148.7  35.9
## 6    Male Born abroad of American parent(s)   31 32420.8  40.4
## 7    Male    Born in PR, Guam, the U.S. VIs   33 10842.4  39.8
## 8    Male                  Born in the U.S. 2637 43492.9  40.6
## 9    Male         Not a citizen of the U.S.  237 43005.0  43.1
## 10   Male    U.S. citizen by naturalization  187 40199.4  40.5
```

```
# vanuse histogramm, eraldi meetele ja naistele (baasgraafika)
par(mfrow = c(1, 2), cex = 0.9)
d_ply(mass, "SEX", summarise, joonis = hist(AGEP, freq = F, main = unique(SEX)))
```



Paketis **plyr** on ka kasulik funktsioon `mutate(.)`, mille abil saab andmestikku uusi tunnuseid lisada (+ neid kohe järgmise tunnuse leidmisel kasutada)

```
vr1 <- mutate(vr,
              muutus1 = pulss10m - pulss0m,
              muutus2 = pulss30m - pulss10m,
             keskmine = (muutus1 + muutus2)/2 )
vr1
```

```
##   nimi kaal pikkus sugu pulss0m pulss10m pulss30m muutus1 muutus2 keskmine
## 1 Mari   68    170    2      70      130      150      60      20       40
## 2 Jaan   65    180    1      80      120      120      40       0       20
## 3 Jüri  100    190    1      80      190       NA     110      NA       NA
```

Kombineerituna `ddply(.)`-ga on käsu `mutate(.)` abil võimalik lisada andmestikku grupipõhiseid näitajaid:

```
# laiale andmetabelile lisame soo kaupa leitud näitajad
ddply(vr, "sugu", mutate,
      keskmine = mean(pikkus),
            sd = round(sd(pikkus), 1),
            cv = round(sd/keskmine, 2))
```

```
##   nimi kaal pikkus sugu pulss0m pulss10m pulss30m keskmine  sd   cv
## 1 Jaan   65    180    1      80      120      120      185 7.1 0.04
## 2 Jüri  100    190    1      80      190       NA      185 7.1 0.04
## 3 Mari   68    170    2      70      130      150      170  NA   NA
```

```
# pikk andmestik: lisame igale inimesele kehamassindeksi väärtuse
ddply(m, "nimi", mutate, KMI = round(kaal/(pikkus/100)^2, 1))
```

```
##   nimi kaal pikkus sugu variable value  KMI
## 1 Jaan   65    180    1  pulss0m    80 20.1
## 2 Jaan   65    180    1 pulss10m   120 20.1
## 3 Jaan   65    180    1 pulss30m   120 20.1
## 4 Jüri  100    190    1  pulss0m    80 27.7
## 5 Jüri  100    190    1 pulss10m   190 27.7
## 6 Jüri  100    190    1 pulss30m    NA 27.7
## 7 Mari   68    170    2  pulss0m    70 23.5
```

```
## 8 Mari   68    170    2 pulss10m   130 23.5
## 9 Mari   68    170    2 pulss30m   150 23.5
```

Kui võrrelda funktsioone `summarise()` ja `mutate(.)` siis esimene agregeerib andmed teine mitte.

Paketis **plyr** on ka spetsiaalne käsk andmestike ridade sorteerimiseks: `arrange(.)`. Sellega on sorteerimise kirjapanek mugavam kui baaspaketi käskudega:

```
arrange(vr, pulss0m, pulss10m)
vr[order(vr$pulss0m, vr$pulss10m),]
```

1.4.1 Ülesanded

1. Loe sisse Massachusettsi andmestik:
   `mass <- read.table("http://kodu.ut.ee/~annes/Rkursus/mass.txt", sep = "\t", header=T)`.
2. Koosta 10-aastased vanusgrupid ja arvuta iga vanusgrupi korral keskmine palk (veerus `AGEP` on täpne vanus, `WAGP` on palganumber), palga miinimum ja maksimum. Lisaks leia vanusgrupi inimeste arv ja inimeste arv vanusgrupis, kel on palgaväärtus puudu.
3. Arvuta samad näitajad igas vanus-soogrupis. Kujuta palga keskväärtust ning miinimumi-maksimumi ka sobiva joonisega.

1.5 Veel mõni lisafunktsioon paketist plyr

Paketis **plyr** on veel paari tüüpi käske. `r!ply(.)` kordab suvalist R-i käsku etteantud arv kordi ja pärast ühendab tulemused; see on kasulik juhul, kui etteantavas käsus on midagi juhuslikku. `m!ply(.)` annab meid huvitavale funktsioonile ette erinevaid argumentide komplekte, mis on kirjas andmetabeli kujul.

```
# Tekitame viis korda kolme-elemendilist juhuarvude vektorit ning igast vektorist leiame
# minimaalse juhuarvu (standardnormaaljaotusest)
rdply(5, min(rnorm(3)))
```

```
##   .n         V1
## 1  1 -0.9550650
## 2  2 -2.9118382
## 3  3  0.4633130
## 4  4 -0.2491168
## 5  5 -0.9847358
```

```
# Tekitame maatriksi, kus on kahe-elemendilised vektorid (kokku kolm vektorit)
raply(3, rnorm(2))
```

```
##              1           2
## [1,] 0.5113884 -0.23374356
## [2,] 1.0171957  0.44797607
## [3,] 0.3018056 -0.09225555
```

```
# Defineerime andmetabeli funktsiooni argumendi väärtustega:
argumendid <- data.frame(mean = c(0, 10, 100), sd = c(100, 10, 0))
# Genereerime viie-elemendilised vektorid meie poolt määratud normaaljaotuse parameetritega
mdply(argumendid, rnorm, n = 5)
```

```
##   mean  sd        V1          V2        V3         V4         V5
## 1    0 100 111.10566 107.9274849  18.15558  40.365097  33.528055
## 2   10  10  17.87753   0.8824069  14.20301   1.277302  -4.557281
## 3  100   0 100.00000 100.0000000 100.00000 100.000000 100.000000
```

2 Andmetöötlus paketiga dplyr

dplyr pakett on paketi **plyr** käsu `ddply(.)` edasiarendus, aitamaks suuri andmestikke kiiremini töödelda/analüüsida. Põhjalik ülevaade selle paketi käskudest on paketi dokumentatsioonis[1]. Põhifunktsioonid paketis on järgmised

- `mutate()` – lisab andmestikku uusi tunnuseid, mis on leitud olemasolevate põhjal.
- `select()` – valib andmestikust nime põhjal tunnused.
- `filter()` – valib andmestikust vaatlused/read loogilise tingimuse põhjal.
- `summarise()` – summaarsete näitajate leidmine.
- `arrange()` – andmestiku ridade sorteerimine.

Kõigi funktsioonide esimene argument on andmetabel, järgmised argumendid täpsustavad, mida andmestikuga teha tuues ära tunnuste nimed, mille kirjapanekuks siin käskudes jutumärke pole vaja kasutada. Kõik funktsioonid väljastavad tulemuseks omakorda andmetabeli. Kõiki funktsioone saab kombineerida käsuga `group_by()`, mis määrab toimingute tegemise gruppide kaupa.

Nagu näha, siis mitu funktsiooninime langeb kokku paketiga **plyr**. Seega pakette kasutades peaks neist korraga valima ühe. Või aktiveerida mõlemad paketid, siis kindlasti enne **plyr** pakett, seejärel **dplyr**.

2.1 Paketi dplyr käskude kasutamine

Käsk `filter(.)` aitab selekteerida teatud kriteeriumitele vastavaid ridu. See on kiirem kui kantsulgude kasutamine, sest kantsulgusid kasutades vaadatakse üksikud elemendid ükshaaval üle, ent `filter(.)` käsu puhul kasutatakse nutikamaid algoritme (enamasti andmed sorditakse mingil moel enne kui hakatakse üldse filtris määratud kriteeriumeid kontrollima).

```
library(dplyr) # Kui pole, tuleb installida
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# rakendame filtrit ja selekteerime tunnuseid
select(filter(mass, AGEP > 70, WAGP > 100000), contains("G"))
```

```
##   AGEP                                       MIG   WAGP
## 1   84 No, different house in US or Puerto Rico 150000
## 2   71              Yes, same house (nonmovers) 110000
## 3   77              Yes, same house (nonmovers) 145000
```

Käsk `group_by(.)` aitab andmestiku tükkideks jagada, aga ei tee sellega midagi enamat. Kui tükkidel soovida midagi analüüsida, tuleb `group_by(.)` funktsiooni tulemus ette anda vastavaks analüüsiks kasutatavale funktsioonile. Siin kohal on jälle hea kasutada funktsiooni `summarise(.)`:

[1] http://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

```
summarise(group_by(mass, CIT),
          keskpalk = mean(WAGP, na.rm = T),
                 n = n(),
             notNA = sum(!is.na(WAGP)))
```

```
## # A tibble: 5 x 4
##                                CIT keskpalk     n notNA
##                             <fctr>    <dbl> <int> <int>
## 1 Born abroad of American parent(s) 28588.63    62    51
## 2    Born in PR, Guam, the U.S. VIs 12516.49    79    74
## 3                 Born in the U.S. 33829.36  5422  4361
## 4         Not a citizen of the U.S. 32797.35   448   423
## 5    U.S. citizen by naturalization 30703.34   413   404
```

Lisaks on paketis **dplyr** defineeritud **toru** ehk aheldamisoperaator (kujul `%>%`), millega on võimalik ühe funktsiooni tulemused edasi anda järgmisele funktsioonile, ilma, et vahetulemust ekraanile prinditaks või uuele muutujale omistataks. Ehk koodi, kus on vaja rakenda kahte funktsiooni `f1` ja `f2` järgmiselt `fun2(fun1(x), y)` võib kirja panna kujul `fun1(x) %>% fun2(y)`. Esimese funktsiooni tulemus antakse siin teisele funktsioonile ette **esimeseks** argumendiks.

Toru kasutamine aitab mõnikord muuda koodi loetavamaks. Näiteks, kui on vaja leida andmestikust `mass` kodakondsuse, soo ja perekonnaseisu gruppides keskmist palka ning grupi mahtu ja saadud tulemustabelist näha esimesi ridu, siis aheldamisoperaatori abil saaks selle kirja panna järgmiselt:

```
mass %>% group_by(CIT, SEX, MAR) %>%
       summarise(keskpalk = mean(WAGP, na.rm = T), n = n()) %>% head()
```

```
## # A tibble: 6 x 5
## # Groups:   CIT, SEX [2]
##                                CIT    SEX                     MAR keskpalk     n
##                             <fctr> <fctr>                  <fctr>    <dbl> <int>
## 1 Born abroad of American parent(s) Female                Divorced 37333.33     3
## 2 Born abroad of American parent(s) Female                 Married 16416.67    12
## 3 Born abroad of American parent(s) Female Never, or under 15 years 39071.43    12
## 4 Born abroad of American parent(s) Female               Separated 32500.00     2
## 5 Born abroad of American parent(s) Female                 Widowed     0.00     2
## 6 Born abroad of American parent(s)   Male                 Married 44660.00    15
```

Kui paneksime sama asja kirja "tavaliselt", siis peaksime koodirida alustama nö tagantpoolt ettepoole ehk alustama viimasest operatsioonist, mida andmetele rakendada (`head`)

```
head(summarise(group_by(mass, CIT, SEX, MAR), keskpalk = mean(WAGP, na.rm = T), n = n()))
```

2.1.1 Ülesanded

1. Leia inimeste vanuses 16 aastat kuni 89 aastat, keskmine ja maksimaalne nädala töötundide arv (WKHP), soo ja vanusgruppide(eelmises ülesandeplokis tehtud vanusgrupid) kaupa. Esita tabelis ka gruppide mahud ja vaatluste arv, mida on keskväärtuse/maksimumi leidmisel kasutatud. Leia küsitud tabel kahte moodi, nii "toru" kasutades kui ilma.
2. Tee eelmise ülesande andmete põhjal joondiagramm.
3. Aheldamisoperaatorit kasutades ühenda eelmised kaks ülesannet.

3 Pakett data.table

Suurte andmete filtreerimiseks, teisendamiseks, agregeerimiseks ja grupeerimiseks sobib ka pakett **data.table**[2].

Paketi andmetabel ehk *data.table* on tavalise andmetabeli *data.frame*-i edasiarendus. Nagu tavalise andmetabeli korral on *data.table* tabelist võimalik objekte ja veerge valida kantsulgudes indekseid määrates. Lisavõimalus on see, et *data.table* tabelis saab sel moel uusi tunnuseid arvutada, tunnuseid jooksvalt uuendada ning andmeid grupitunnuse põhjal agregeerida.

Kui `DT` on *data.table* tüüpi andmetabel, siis põhisüntaksi kuju saab kirja panna järgnevalt

```
DT[i, j, by]
```

kus

- `i` määrab read/objektid, mida edasi kasutada
- `j` määrab veerud, mis valitakse, uuendatakse või tekitatakse
- `by` määrab vajadusel grupitunnuse `j` tehtavateks arvutustesse.

Siinkohal mõned näiteid andmestikust objektide või veergude valimisest ja arvutuste kirjapanekust. Proovi need läbi ja uuri tulemust. Erinevalt tavalisest andmestikust ei ole *data.table* tabelites reanimesid, on ainult reanumbrid, seega rida nime järgi valida saa. Teiseks võib välja tuua, et tekstitunnused on neis tabelites vaikimisi *character* tüüpi, mitte *factor* tüüpi.

```
library(data.table)  # Kui pole, tuleb installida
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following objects are masked from 'package:reshape2':
##
##     dcast, melt
```

```
# data.table tüüpi objekti moodustamine
DT <- data.table(a = 1:10, b = rep(1:3, c(3, 3, 4)), c = rep(LETTERS[5:7], c(4, 3, 3)))
DT
```

```
##       a b c
##  1:   1 1 E
##  2:   2 1 E
##  3:   3 1 E
```

[2] https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html

```
##  4:  4 2 E
##  5:  5 2 F
##  6:  6 2 F
##  7:  7 3 F
##  8:  8 3 G
##  9:  9 3 G
## 10: 10 3 G
```

```
str(DT)
```

```
## Classes 'data.table' and 'data.frame':   10 obs. of  3 variables:
##  $ a: int  1 2 3 4 5 6 7 8 9 10
##  $ b: int  1 1 1 2 2 2 3 3 3 3
##  $ c: chr  "E" "E" "E" "E" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Ridade ja veergude valik

```
DT[1:2, ]
DT[, a]
DT[,"a"]
DT[b > 1.5, 3]
DT[c(1, 3:4), .(a, b)]
```

Uue tabeli moodustamine, arvkarakteristikud, loendamine, uued tunnused:

```
DT[, .(kv = mean(a), s = sd(a), mitu = .N), by = c]
DT[a > 3, .N]
DT[, .N, by = c]
DT[, .(d = a + 50)]
```

Olemasoleva tabeli sees muudatuste tegemine

```
DT[, d := a + 50 ]
DT[c(2, 4),  a := a*100L]
DT[, `:=` (b = -b, # olemasoleva vektori väärtuste teisendus
           uus = "lisatunnus" # uue tunnuse lisamine
           )]
DT
DT[, c("d", "uus") := NULL][]
```

3.0.1 Ülesanded

1. Teisenda andmestik `mass` paketi **data.table** andmetüübiks käsuga `as.data.table(.)`.
2. Kasutades `DT[i, j, by]` konstruktsiooni leia mitu meessoost üle 60 aastast on uuritavate hulgas.
3. Kasutades `DT[i, j, by]` konstruktsiooni leia perekonnaseisu sagedustabel.
4. Järjesta eelmine tabel sageduste järgi kasvavalt.
5. Täienda eelmist tabelit gruppide osakaaludega.
6. Tekita tabel, kus oleks soo ja perekonnaseisu gruppides leitud keskmine vanus, vanuse standardhälve ja levinum haridustase.