Rakendustarkvara: R. Kevad 2018, 7. praktikum*

1 Andmetöötlus paketiga dplyr

Pakett **dplyr** mõeldud andmetabelite täienduste, tabelis olevate objektide või tunnuste alamhulkade valiku, grupeerimise ja sorteerimise teostamiseks. Samuti saab selle abil leida erinevaid andmeid agregeerivaid tabeleid, mis sobivad kirjeldava statistilise analüüsi läbiviimiseks. Paketi **dplyr** käskude juures võib välja tuua veel nende töö kiiruse suuremate andmestike korral. 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.

1.1 Paketi dplyr käskude kasutamine

Loeme kõigepealt sisse näiteandmestiku: Massachusettsi valikuuring

```
mass <- read.table("http://kodu.ut.ee/~annes/Rkursus/mass.txt", sep = "\t", header = T)
```

Paketi installimine ja aktiveerimine:

```
#install.packages(dplyr)          # vajadusel installeerida pakett
library(dplyr)
```

Käsu `mutate()` abil saab andmestiku veergudega teisendusi teha, uusi tunnuseid tekitada või ka mingeid tunnuseid kustutada

```
# arvutame kaks uut tunnust, kustutame ühe vana
mass1 <- mutate(mass, kuus = WAGP/12, kuus_euro = kuus * 0.8, MIG = NULL)
```

Käsk `filter(.)` aitab valida välja 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).

```
# rakendame filtrit (ekraanile paari esimese veeru väärtused neil vaatlustel)
filter(mass, AGEP > 70, WAGP > 100000)[,1:3]
```

```
##      id AGEP                            CIT
## 1  9452   84 U.S. citizen by naturalization
## 2 49277   71               Born in the U.S.
## 3 64546   77               Born in the U.S.
```

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

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

Käsu `select(.)` abil saab välja valida tingimustele vastavaid veerge andmestikust. Järgmises näites on rakendatud esmalt ridade filtreerimist, seejärel tehtud valik veergude hulgast: valitakse need veerud, mille nimes sisaldub täht "G"

```
# 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 hea kasutada funktsiooni `summarise(.)`, mis esitab agregeeritud tulemused:

```
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 Puerto Rico, Guam, the U.S. Virgin Islands, 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
```

Käsu `arrange(.)` abil saab andmestikke sorteerida

```
osa <- filter(select(mass, id, SEX, AGEP, WKHP, WAGP), WAGP > 300000, AGEP < 55)
arrange(osa, SEX, desc(AGEP), WKHP)
```

```
##       id    SEX AGEP WKHP   WAGP
## 1   4026 Female   53   50 502000
## 2  47523 Female   44   45 502000
## 3  54710 Female   38   30 502000
## 4  56856   Male   54   65 502000
## 5  45001   Male   52   60 502000
## 6  26473   Male   51   60 502000
## 7  27055   Male   50   60 502000
## 8  51654   Male   49   60 502000
## 9  63966   Male   48   48 502000
## 10 42026   Male   45   40 502000
## 11 48820   Male   45   40 502000
## 12 12981   Male   45   55 502000
## 13 47379   Male   45   60 502000
## 14 48414   Male   45   70 502000
## 15   821   Male   42   50 502000
## 16 12187   Male   40   50 502000
## 17 27263   Male   40   60 502000
## 18 38114   Male   39   50 502000
## 19 62255   Male   37   60 502000
## 20 41999   Male   34   60 502000
## 21 60364   Male   32   55 502000
## 22 59973   Male   29   85 502000
```

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 `fun1` ja `fun2` 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` 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(SEX, MAR) %>%
  summarise(keskpalk = mean(WAGP, na.rm = T), n = n()) %>%
  head()
```

```
## # A tibble: 6 x 4
## # Groups:   SEX [2]
##      SEX                                 MAR  keskpalk     n
##   <fctr>                              <fctr>     <dbl> <int>
## 1 Female                            Divorced 33860.000   281
## 2 Female                             Married 28164.255  1356
## 3 Female Never married or under 15 years old 21621.436  1336
## 4 Female                           Separated 19475.926    54
## 5 Female                             Widowed  4232.647   272
## 6   Male                            Divorced 39443.583   187
```

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, SEX, MAR), keskpalk = mean(WAGP, na.rm = T), n = n()))
```

1.1.1 Ülesanded

1. Leia inimeste vanuses 16 aastat kuni 85 (kaasaarvatud) aastat, keskmine ja maksimaalne nädala töötundide arv (WKHP), soo ja vanusgruppide (5-aastased 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.

2 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.

2.1 Näited data.table süntaksi kasutamisest

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.

```
#install.packages("data.table")    # vajadusel installeerida
library(data.table)

# 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
##  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>
```

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

Ridade ja veergude valik (alamosa andmetabelist)

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

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. Lisatud tühjad kantsulud `[]` rea lõpus prindivad muudetud tabeli `DT` ekraaanile. Üldjuhul neid lisama ei pea.

```
# ühe uue tunnuse lisamine
DT[, d := a + 5][]
# mitme uue tunnuse lisamine
DT[, c("uus1", "uus2") := .(a + 5, b - 2) ][]
# mitme uue tunnuse lisamine, variant 2
DT[, `:=` (uus3 = a + 50,
           uus4 = b - 2,
           b = -b)][]
# väärtuste muudatus valitud ridades
DT[c(2, 4),  a := a*10L][]
# uus tunnus, aga mitte igas reas
DT[c(2, 4),  uus5 := a*10L][]
# tunnuste kustutamine
DT[, uus5 := NULL][]
DT[, c("uus1", "uus2") := NULL][]
```

2.1.1 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.