Rakendustarkvara: R. Kevad 2018, 4. praktikum*

1 Toimingud andmestikuga – jätk

1.1 Andmestike täiendamine ja ühendamine (mestimine)

Oleme tutvunud, kuidas andmestikust saab eraldada alamhulki (nt ridu või veerge, mis vastavad teatud kriteeriumitele). Ühekaupa saab uusi veerge `data.frame`-tüüpi objektile lisada nii dollarimärgi kui kantsulgude abil, kui anname ette uue veeru nime.

Vaatame siin näiteandmestikuna andmestikku `mk` (*maakonnad.txt*), mis sisaldab infot USA 5 osariigi mõnede maakondade kohta (425 maakonda). Loeme andmed sisse

```
mk <- read.table("http://kodu.ut.ee/~annes/Rkursus/maakonnad.txt", sep = " ", header = TRUE)
```

Mitut uut veergu või rida saab andmestikule lisada `cbind(.)` ja `rbind(.)` käskudega. Neid kasutades tuleb aga olla ettevaatlik: lisatavas reas peab olema sama palju elemente kui on andmestikus veerge ja lisatavas veerus peab olema sama palju elemente, kui on andmestikus ridu. Tunnuste nimed lisatavates ridades peavad vastama andmestiku omale, samuti peab lisatavates veergudes objektide järjestus olema sama kui esialgses andmestikus.

```
# teeme kaks uut tunnust
suurus <- cut(mk$pop_estimate, c(0, 1000, 10000, 1000000, Inf),
              labels = c("mikro", "väike", "keskmine", "suur"), include.lowest = T)
sooylekaal <- ifelse(mk$fem > 50, "F", "M")
# lisame need vektorid andmestiku lõppu veergudeks:
mk <- cbind(mk, suurus, sooylekaal)
# või
lisatabel<- data.frame(suurus, sooylekaal)
mk <- cbind(mk, lisatabel)

# ridade lisamine, praegu tekitame sellega dubleeritud vaatlusi!
lisa <- rep(seq(1, nrow(mk), by = 60),c(1:3, 1:3, 1, 1))
mktopelt <- rbind(mk, mk[lisa, ])
```

Sageli on analüüsiks vajaminevad andmed mitmes erinevas andmetabelis, mis võivad olla objektide arvu poolest ka erinevad. Näiteks jooksuvõistluste andmete puhul võivad ühes tabelis kirjas isikuandmed kohta (nimi, sugu, vanus), teises tabelis aga võistlustulemuste andmed. Kui sooviks analüüsida tulemuste jaotust sugude vahel või sõltuvalt vanusest, oleks mugav need tabelid mestida käsuga `merge(.)`. Andmete ühendamiseks peaks mõlemas tabelis olema nn võtmetunnus, mille abil read vastavusse pannakse.

```
isikud <- data.frame(nimi = c("Peeter", "Mari", "Tiina", "Laine"),
      sugu = c("M", "N", "N", "N"), vanus = c(30, 22, 25, 20))
tulemused <- data.frame(nimi = c("Mari", "Peeter", "Tiina", "Peeter", "Toomas"),
      tulemus = c(30.1, 22.5, 18.4, 25.3, 20.4),
      voistlus = c("jooks1", "jooks1", "jooks2", "jooks2", "jooks2"))
```

```
isikud
```

```
##     nimi sugu vanus
## 1 Peeter    M    30
## 2   Mari    N    22
## 3  Tiina    N    25
```

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

```
## 4  Laine    N    20
```

```
tulemused
```

```
##     nimi tulemus voistlus
## 1   Mari    30.1   jooks1
## 2 Peeter    22.5   jooks1
## 3  Tiina    18.4   jooks2
## 4 Peeter    25.3   jooks2
## 5 Toomas    20.4   jooks2
```

```
(kokku <- merge(isikud, tulemused, by = "nimi", all = TRUE))
```

```
##     nimi sugu vanus tulemus voistlus
## 1  Laine    N    20      NA     <NA>
## 2   Mari    N    22    30.1   jooks1
## 3 Peeter    M    30    22.5   jooks1
## 4 Peeter    M    30    25.3   jooks2
## 5  Tiina    N    25    18.4   jooks2
## 6 Toomas <NA>    NA    20.4   jooks2
```

1.1.1 Ülesanded

1. Kuidas peaks `merge` käsu kirja panema, kui `isikud` ja `tulemused` andmestikes on võtmetunnusel erinev nimi?
2. Proovi, mis muutub `merge` käsu tulemuses, kui kasutada `all = TRUE` argumendi asemel `all.x = TRUE` või `all.y = TRUE`.

1.2 Sorteerimine

Väga sageli soovime, et andmestiku read oleks mingi tunnuse (nt inimese vanuse) alusel sorteeritud. Sorteerimiseks on R-is kaks olulist käsku:

- `order(.)` tagastab etteantud vektori elementide järjekorranumbrid sellises järjekorras, et saaksime järjestatud vektori, mh argumendiga `decreasing` saab määrata, kas see on kahanev või kasvav;
- `sort(.)` tagastab etteantud vektori elemendid kasvavas (või kahanevas) järjekorras.

```
x <- c(8, 1, NA, 7, 7)
order(x)
```

```
## [1] 2 4 5 1 3
```

```
sort(x, na.last = TRUE)
```

```
## [1]  1  7  7  8 NA
```

```
x[order(x)]
```

```
## [1]  1  7  7  8 NA
```

```
#andmestiku ridade sorteerimine kahe vektori järgi:
kokku[order(kokku$vanus, kokku$tulemus, decreasing = TRUE), ]
```

```
##     nimi sugu vanus tulemus voistlus
## 4 Peeter    M    30    25.3   jooks2
## 3 Peeter    M    30    22.5   jooks1
## 5  Tiina    N    25    18.4   jooks2
## 2   Mari    N    22    30.1   jooks1
## 1  Laine    N    20      NA     <NA>
## 6 Toomas <NA>    NA    20.4   jooks2
```

1.2.1 Ülesanded

1. Loe sisse kaks andmestikku ja tutvu nendega (dimensioon, tunnused):
   `link <- "http://kodu.ut.ee/~annes/Rkursus/"`
   `visiidid <- read.table(paste0(link, "visiidid.txt"), sep = "\t", header = TRUE)`
   `inimesed <- read.table(paste0(link, "isikud.txt"), sep = "\t", header = TRUE)`

- Andmestikus `visiidid` on neli veergu: `ik` – isikukood, `visiidi_kp` – arsti külastamise kuupäev, `vererohk` – süstoolne vererõhk antud arstivisiidil (mmHg), `crv` - C-reaktiivse valgu kontsentratsioon (nn näpuveri) (mg/L). Visiitide andmestik kirjeldab 10 aasta jooksul arsti külastamisel tehtud vererõhu ja CRV mõõtmisi.
- Andmestikus `inimesed` on isikukoodid ja isikukoodist tuletatud info(sugu, sünnikuupäev, vanus).

2. Järjesta visiitide andmestik kasvavalt isikukoodi ja arstivisiidi kuupäeva järgi.
3. Ühenda isikukoodide ning visiitide andmestik, ühendatud andmestik peab sisaldama kõik inimesed mõlemast andmestikust. Mitu vaatlust on ühendatud andmestikus? Mis tunnuste osas esineb puuduvaid väärtusi.

1.3 Unikaalsed ja mitmekordsed elemendid. Hulgatehted.

Mõnikord on üks objekt andmestikus mitu korda, ent me soovime seda ainult ühel korral analüüsi kaasata. Etteantud vektorist saab unikaalsed elemendid kätte käsuga `unique(.)`. Käsuga `duplicated(.)` saab teada, kas ette antud vektoris/andmetabelis on element esimest või juba mitmendat korda (tulemuseks on tõeväärtusvektor, kus `TRUE` tähendab seda, et antud väärtus on juba mitmendat korda).

```
# vaatame andmestikku 'kokku', andmestike ühendamise alapunktist
kokku$nimi
```

```
## [1] Laine  Mari   Peeter Peeter Tiina  Toomas
## Levels: Laine Mari Peeter Tiina Toomas
```

```
unique(kokku$nimi)  # unikaalsed nimed
```

```
## [1] Laine  Mari   Peeter Tiina  Toomas
## Levels: Laine Mari Peeter Tiina Toomas
```

```
duplicated(kokku$nimi)  # mitmes element nimede vektoris on dubleeriv
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE FALSE
```

```
duplicated(kokku)       # mitmes rida andmestikus on dubleeriv (sellist pole!)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

R-is on realiseeritud ka elementaarsed hulgaoperaatorid: ühisosa, ühend ja vahe. Käsk `union(x, y)` tagastab ette antud kahe vektori `x` ja `y` elementidest koostatud uue vektori, sealjuures mõlema vektori kõik elemendid on esindatud. Käsk `intersect(x, y)` tagastab vektori elementidest, sealjuures on esindatud ainult need elemendid mis on nii vektoris `x` kui ka `y`. Käsk `setdiff(x, y)` tagastab vektori, kus on ainult need `x` elemendid, mida vektoris `y` ei ole. Kõik hulgatehete käsud tagastavad sellised vektorid, kus igat elementi on ainult üks kord

```
x <- c(1:5, 1:5)
y <- 3:7
union(x, y)
intersect(x, y)
setdiff(x, y)
```

1.3.1 Ülesanded

1. Kontrolli kas alapunktis **1.1** tekitatud andmestikus `mktopelt` on dubleeritud andmeid. Mitu objekti on korduvad? Kas mõni maakond on rohkem kui 2 korda korratud? Kui jah, siis millised?
2. Vaata visiitide ja isikukoodide ühendatud andmestikku. Kas arsti mitte külastanud isikute osakaal (protsentuaalselt) on suurem meeste või naiste hulgas?

2 Andmestiku teisendused

2.1 Pikk ja lai andmetabel. Pakett reshape2

Teise praktikumi materjali alguses näiteks toodud traditsioonilist andmetabeli kuju ehk *objekt* × *tunnus*-maatriksit nimetatakse vahel ka nn **laias formaadis andmestikuks**, tabeli iga rida vastab ühele objektile, infoliiasus on viidud miinimumini. Näide laias formaadis andmestikust:

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

Pikas formaadis andmestiku puhul proovitakse hoida kõiki sama omadust kirjeldavaid andmeid ühes veerus; üks objekt võib kajastuda mitmel real. Sellisel kujul andmestikku on mõnikord mugav arvuti abil analüüsida (nt segamudelite hindamine, joonised `ggplot2` paketiga):

```
##   nimi kaal pikkus sugu variable value
## 1 Mari   68    170    2  pulss0m    70
## 2 Jaan   65    180    1  pulss0m    80
## 3 Jüri  100    190    1  pulss0m    80
## 4 Mari   68    170    2 pulss10m   130
## 5 Jaan   65    180    1 pulss10m   120
## 6 Jüri  100    190    1 pulss10m   190
## 7 Mari   68    170    2 pulss30m   150
## 8 Jaan   65    180    1 pulss30m   120
## 9 Jüri  100    190    1 pulss30m    NA
```

Eriti äärmuslik on pika formaadi puhul hoida kõiki arvulisi tunnuseid ühes veerus:

```
##    nimi variable value
## 1  Mari     kaal    68
## 2  Jaan     kaal    65
## 3  Jüri     kaal   100
## 4  Mari   pikkus   170
## 5  Jaan   pikkus   180
## 6  Jüri   pikkus   190
## 7  Mari     sugu     2
## 8  Jaan     sugu     1
## 9  Jüri     sugu     1
## 10 Mari  pulss0m    70
## 11 Jaan  pulss0m    80
## 12 Jüri  pulss0m    80
## 13 Mari pulss10m   130
## 14 Jaan pulss10m   120
## 15 Jüri pulss10m   190
## 16 Mari pulss30m   150
## 17 Jaan pulss30m   120
## 18 Jüri pulss30m    NA
```

Andmestiku ühest formaadist teise viimiseks tutvume paketiga **reshape2**, milles olulisimad käsud `melt(.)` ja `dcast(.)` aitavad vastavalt teisendada andmestikku laiast formaadist pikka ja vastupidi, ning teha veel täiendavaid toiminguid/arvutusi.

Funktsioon `melt(.)` teisendab andmed laiast formaadist pikka. Argumendiga `measure.vars` saab sellele ette anda veerunimede või -indeksite vektori, milles olevad tunnused pannakse kõik ühte veergu. Vaikimisi pannakse kõik arvulised väärtused ühte veergu nimega `value` ning teises veerus nimega `variable` on kirjas, mida antud väärtus tähendab (millises veerus see väärtus esialgses andmestikus oli).

```
#install.packages("reshape2") # kui arvutis pole paketti reshape2, siis esmalt installida
library(reshape2)
vr <- data.frame(nimi = c("Mari", "Jaan", "Jyri"),
                 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))
(m <- melt(vr, measure.vars = 5:7)) # välimised sulud tingivad ekraanile trükkimise
```

```
##   nimi kaal pikkus sugu variable value
## 1 Mari   68    170    2  pulss0m    70
## 2 Jaan   65    180    1  pulss0m    80
## 3 Jyri  100    190    1  pulss0m    80
## 4 Mari   68    170    2 pulss10m   130
## 5 Jaan   65    180    1 pulss10m   120
## 6 Jyri  100    190    1 pulss10m   190
## 7 Mari   68    170    2 pulss30m   150
## 8 Jaan   65    180    1 pulss30m   120
## 9 Jyri  100    190    1 pulss30m    NA
```

Funktsioon `dcast(.)` aitab andmeid pikast formaadist laia teisendada, sealjuures tuleb argumendiga `formula` kindlasti ette öelda, millised tunnused määravad ära tulemusandmestiku read ning millised määravad ära veerud: `formula = reatunnus1 + reatunnus2 ~ veerutunnus1 + veerutunnus2`. Väga kasulik on argument `fun.aggregate`, mille abil saab määrata, kas ja millist funktsiooni peaks kasutama veerutunnustega antud väärtuste agregeerimiseks. Funktsiooniga `recast(.)` saab korraga ära teha töö, mille teeks ära järjestikku rakendatud `melt(.)` ja `dcast(.)` käsud.

```
dcast(m, formula = nimi ~ variable)
```

```
##   nimi pulss0m pulss10m pulss30m
## 1 Jaan      80      120      120
## 2 Jyri      80      190       NA
## 3 Mari      70      130      150
```

```
dcast(m, nimi + kaal ~ variable + sugu)
```

```
##   nimi kaal pulss0m_1 pulss0m_2 pulss10m_1 pulss10m_2 pulss30m_1 pulss30m_2
## 1 Jaan   65        80        NA        120         NA        120         NA
## 2 Jyri  100        80        NA        190         NA         NA         NA
## 3 Mari   68        NA        70         NA        130         NA        150
```

```
dcast(m, nimi ~ . , fun.aggregate = mean, na.rm = TRUE, value.var = "value")
```

```
##   nimi        .
## 1 Jaan 106.6667
## 2 Jyri 135.0000
## 3 Mari 116.6667
```

2.1.1 Ülesanded

1. Kasuta arstivisiitide ja isikute andmestiku ühte alamosa:
   `link <- "http://kodu.ut.ee/~annes/Rkursus/"`
   `valik <- read.table(paste0(link, "valik.txt"), sep = "\t", header = TRUE)`. Vaata andmestik üle `head(valik)`. Tegu on valikuga isikutest, tunnuste osas on lisatud visiidi järjekorranumber.
   - Vii andmed laiale kujule, kus iga isiku jaoks oleks andmestikus üks rida, kus on kirjas isikukood, sugu, sünnikuupäev, vanus ning vererõhu väärtus esimesel ja teisel visiidil.
   - Vii andmed laiale kujule, kus iga isiku jaoks oleks andmestikus üks rida, kus on kirjas isikukood, sugu, sünnikuupäev, vanus, crv väärtus esimesel ja teisel visiidil ning vererõhu väärtus esimesel ja teisel visiidil.
2. Kasuta arstivisiitide andmestikku.
   `visiidid <- read.table(paste0(link, "visiidid.txt"), sep = "\t", header = TRUE)`
   Leia iga isiku keskmine vererõhk ja CRV.

3 Veel andmestruktuure

Oleme varem tutvunud sellega, mis tüüpi võivad olla üksikud andmeelemendid (`int`, `char`, `Factor` jm). Üksikuid elemente saab kokku panna üheks vektoriks näiteks käsuga `c(.)`, millega oleme samuti tuttavad. Samuti oleme tuttavad `data.frame`-tüüpi objektiga – see on R-is andmetabeli tüüp. Andmeid on aga võimalik R-is hoida ka teistsugustes struktuurides kui vektor või `data.frame`.

Maatriks on sisuliselt vektor, mille elemendid on paigutatud ridadesse ja veergudesse. Kuna tegemist on vektoriga, siis peavad kõik elemendid olema sama tüüpi. Maatriksit saab luua mitmel moel. Käsule `matrix(.)` tuleks ette anda vastav vektor ning see, mitmesse ritta ja veergu selle vektori elemendid paigutatakse. Olemasolevaid reavektoreid saab omavahel ühendada käsuga `rbind(.)`, veeruvektoreid käsuga `cbind(.)`.

```
(m <- matrix(1:12, nrow = 3, byrow = F))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
cbind(1:3, 5:7, 11:13)
```

```
##      [,1] [,2] [,3]
## [1,]    1    5   11
## [2,]    2    6   12
## [3,]    3    7   13
```

Maatriksi elemente saab eraldada kantsulgudega:

```
m[1:2, 2:3]
```

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    5    8
```

List on universaalne andmestruktuur. Sisuliselt on tegemist erinevatest elementidest koosneva loendiga, sealjuures need elemendid võivad olla täiesti erinevat tüüpi objektid (isegi funktsioonid). Kui listi elementidel on nimi, saab sobiliku elemendi kätte dollarimärgi ja nime abil, üldisemalt saab elementide eraldamiseks kasutada kahekordseid kantsulgusid. Listi saab tekitada käsuga `list(.)`. Uusi elemente saab lisada kantsulgudes indeksi abil või dollarimärgi abil.

```
(minulist <- list(esimene = "üksainus sõne", matrix(1:12, 3), funktsioon = min))
```

```
## $esimene
## [1] "üksainus sõne"
##
## [[2]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## $funktsioon
## function (..., na.rm = FALSE)  .Primitive("min")
```

```
# elementide valik listist
minulist$esimene
```

```
## [1] "üksainus sõne"
```

```
minulist[[2]]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
# elementide lisamine listi
minulist$neljas <- c(5, 7) # lisame uue elemendi
minulist[[5]] <- letters[1:10] # lisame veel ühe uue elemendi
# muudetud listi struktuuri vaatamine
str(minulist)
```

```
## List of 5
##  $ esimene   : chr "üksainus sõne"
##  $           : int [1:3, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
##  $ funktsioon:function (..., na.rm = FALSE)
##  $ neljas    : num [1:2] 5 7
##  $           : chr [1:10] "a" "b" "c" "d" ...
```

Andmetabel (`data.frame`) on tegelikult teatud piirangutega list: kõik elemendid peavad olema sama pikad vektorid (võivad olla erinevat tüüpi). Andmetabelit saab "käsitsi" tekitada käsuga `data.frame(.)`:

```
df <- data.frame(esimene = 1:5,
                 "2. veerg" = 11:15,
                 nimed = c("Peeter", "Mari", "Kaur", NA, "Tiiu"))
```

Kontrollimaks, kas tegemist on maatriksi, listi või andmetabeliga, saab kasutada käske `is.matrix(.)`, `is.list(.)`, `is.data.frame(.)`. Veelgi kasulikum on käsk `class(.)`, millega saab objekti tüübi nime teada.

```
class(df)
```

```
## [1] "data.frame"
```

```
is.list(df)
```

```
## [1] TRUE
```

3.0.1 Ülesanded

1. Tekita list:
   ```
   sõnad <- list(
   a = c("aabits", "aade", "aadel", "aader", "aadlik"),
   b = c("baar", "baas", "baat"),
   c = c("c-vitamiin", "ca", "circa", "cafe"))
   ```

2. Eralda listist teine element, kasutades elemendi nime.
3. Eralda listist teine element, kasutades elemendi indeksit.
4. Eralda listist esimese elemendi kolmas element.