Rakendustarkvara: R. Sügis 2017, 4. praktikum*

1 Andmestruktuurid

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. Käsuga `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
##
```

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

```
## $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
```

2 Sõnetöötlus paketiga stringr

R- i baaspaketiga on kaasas mitmeid sõnede töötlemise käske, näiteks `grep(.)` ja `substr(.)`; pikemat loetelu näeb, kui trükkida konsooli `?grep` ja `?substr`. Kahjuks nende käskude süntaks pole päris ühesugune ning mõned neist ei ole täielikult vektoriseeritud (nt `substr(.)` argumendid `start` ja `stop` ei tohi olla vektorid). Pakett **stringr** proovib seda puudust kõrvaldada, pakkudes sarnase süntaksiga rohkem vektoriseeritud käske (tegemist on nn *wrapper*-funktsioonidega baaspaketi sõnetöötluskäskudele).

```
#install.packages("stringr") # vaja ainult siis, kui arvutisse seda pole installitud
library(stringr) # vaja iga kord, kui stringr paketiga töötamist alustatakse
```

2.1 Sõne pikkus, sõnede kokkukleepimine ja eraldamine, alamsõne eraldamine

- Sõnede **pikkust** saab teada käsuga `str_length(.)`.
- Sõnesid saab **kokku kleepida** üheks käsuga `str_c(.)`, millel saab argumendiga `sep` määrata, milline sümbol pannakse kokkukleebitavate sõnede vahele. Kui käsule `str_c(.)` kirjutada argumendi `collapse` väärtuseks mingi sümbol, siis kleebitakse kõik sõned üheks ainsaks sõneks, mis on selle sümboliga eraldatud.
- Sõne saab **tükeldada** käsuga `str_split(.)`, mille argumendiga `pattern` saab määrata, mis on tükkide eraldaja. Selle käsu tulemusena tekib **list** , mida saab vektoriks muuta käsuga `unlist(.)`, kui aga lisada käsku argument `simplify = TRUE` on tulemuseks **maatriks**. Kui `str_split(.)` käsule anda `pattern = ""`, siis tükeldatakse sõna üksikuteks tähtedeks.
- **Alamsõne eraldamiseks** on stringr paketis käsk `str_sub(.)`, mille argumentidega `start` ja `end` saab määrata alamsõne alguse ja lõpu tärgi indeksid. Andes neile argumentidele negatiivsed indeksi väärtused alustatakse loendamist sõne lõpust st indeks `-2` märgib sõne eelviimast kohta.

```
sõnad <- c("Õun", "Apelsin", "Porrulauk", NA, "")
str_length(sõnad)
```

```
## [1]  3  7  9 NA  0
```

```
str_c(sõnad, 1:5, sep = "=")
```

```
## [1] "Õun=1"       "Apelsin=2"   "Porrulauk=3" NA            "=5"
```

```
(x <- str_c(sõnad, 1:5, sep = "=", collapse = ". "))
```

```
## [1] NA
```

```
(x <- str_c(str_replace_na(sõnad), 1:5, sep = "=", collapse = ". "))
```

```
## [1] "Õun=1. Apelsin=2. Porrulauk=3. NA=4. =5"
```

```
str_split(x, ". ")
```

```
## [[1]]
## [1] "Õun=1"       "Apelsin=2"   "Porrulauk=3" "NA=4"        "=5"
```

```
unlist(str_split(x, ". "))
```

```
## [1] "Õun=1"       "Apelsin=2"   "Porrulauk=3" "NA=4"        "=5"
```

```
str_split(sõnad, "")
```

```
## [[1]]
## [1] "Õ" "u" "n"
##
## [[2]]
## [1] "A" "p" "e" "l" "s" "i" "n"
##
## [[3]]
## [1] "P" "o" "r" "r" "u" "l" "a" "u" "k"
##
## [[4]]
## [1] NA
##
## [[5]]
## character(0)
```

```
lause1 <- "see ja teine ja kolmas ja neljas"; lause2 <- "üks või kaks või kolm"
str_split(c(lause1, lause2), c("ja", "või"))
```

```
## [[1]]
## [1] "see "     " teine "  " kolmas " " nel"     "s"
##
## [[2]]
## [1] "üks "   " kaks " " kolm"
```

```
str_sub(sõnad, 1:4, 3:6)
```

```
## Warning in stri_sub(string, from = start, to = end): longer object length
## is not a multiple of shorter object length
```

```
## [1] "Õun" "pel" "rru" NA    ""
```

```
str_sub(sõnad, end = -3)
```

```
## [1] "Õ"       "Apels"   "Porrula" NA        ""
```

2.1.1 Ülesanded

1. Loe sisse Massatchusettsi andmestik:
   `andmed <- read.table("http://kodu.ut.ee/~annes/Rkursus/mass.txt", sep = "\t", header=T)`
   Andmetabelis on veerus OCCP iga inimese amet, sealjuures kolme esimese tähega on kodeeritud vastav valdkond; näiteks kõik puhastusteenustega seotud ametid algavad tähtedega CLN. Mitu erinevat valdkonda on selles andmetabelis?

2.2 Alamsõne otsimine ja muutmine

Et teada saada, kas üks sõne sisaldub teises sõnes, saab kasutada käsku `str_detect(.)` argumendiga `pattern`. Juhul, kui on soov otsitava alamsõne teksti kujul leida, saab kasutada käsku `str_extract(.)`. Et teada saada, millisel positsioonil asub otsitav alamsõne, võiks kasutada käsku `str_locate(.)`, mis tagastab kõige esimesel positsioonil leitud alamsõne (kui sellist alamsõne üldse leidub) algus- ja lõpuindeksid **matrix**-tüüpi objektina. Kui tahame kätte saada kõigil positsioonidel olevate alamsõnede algus- ja lõpuindeksid, sobib käsk `str_locate_all(.)`, mis tagastab listi.

```
str_detect(sõnad, "r")
```

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

```
str_extract(sõnad, "r")
```

```
## [1] NA  NA  "r" NA  NA
```

```
str_locate(sõnad, "r")
```

```
##      start end
## [1,]    NA  NA
## [2,]    NA  NA
## [3,]     3   3
## [4,]    NA  NA
## [5,]    NA  NA
```

```
str_locate_all(sõnad, "r")
```

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

```
##      start end
##
## [[2]]
##      start end
##
## [[3]]
##      start end
## [1,]     3   3
## [2,]     4   4
##
## [[4]]
##      start end
## [1,]    NA  NA
##
## [[5]]
##      start end
```

Mõnikord on sõnede alguses või lõpus liiga palju tühikuid, neid saab eemaldada käsuga `str_trim(.)`. Käsuga `str_pad(.)` aga saab sõne algusesse või lõppu panna tühikuid (või muid sümboleid) juurde, nii et sõne saavutaks argumendiga `width` ette antud pikkuse.

```
str_trim("    siin on palju tühjust    ")
str_pad(sõnad[-4], width = 9, side = "both", pad = "_") # NA-ga ei saa str_pad hästi hakkama
```

```
## [1] "siin on palju tühjust"
## [1] "___Õun___" "_Apelsin_" "Porrulauk" "_________"
```

Kõige üldisem sõnede muutmise käsk on `str_replace(.)`, mis proovib argumendiga `pattern` ette antud ja leitud mustrit asendada argumendiga `replacement` määratud mustriga; asendatakse ainult esimene leidumine. Kõiki leidumisi saab asendada käsuga `str_replace_all(.)`

```
str_replace(sõnad, "r", "l")
str_replace_all(sõnad, "r", "l")
```

```
## [1] "Õun"       "Apelsin"   "Polrulauk" NA          ""
## [1] "Õun"       "Apelsin"   "Pollulauk" NA          ""
```

Kõigile `stringr` paketi käskudele võib argumentidega `pattern` ja `replacement` ette anda ka regulaaravaldisi[1].

2.2.1 Ülesanded

1. Massatchusettsi andmestikus on veerus `COW` ära toodud, kelle heaks inimene töötab. Kui tegemist on palgatöötajaga, sisaldab `COW` väärtus vastava inimese puhul sõna *Employee* või *employee*. Milline on palgatöötajate keskmine palk (`WAGP`)?
2. Eesti isikukoodi formaat[2] on järgmine: abcdefghijk, kus a – sugu ja sajand (paaritu arv mees, paarisarv naine, 1,2 – 1800, 3,4 – 1900, 5,6 – 2000); bc – aasta, de – kuu, fg – päev, hij – (haigla) sel päeval (selles haiglas) sündimise järjekord, k – kontrollnumber.
    1. Loe sisse komaga eraldatud isikukoodide jada:
       `isikukoodid <- (read.table("http://www.ut.ee/~annes/Rkursus/isikukoodid.txt"))[1,]`
    2. Eralda isikukoodid üksteisest ja salvesta tekkiv vektor mingi nimega.
    3. Tekita uus andmetabel (`data.frame`), millesse lisa isikukoodide veerg.
    4. Lisa veerg, kus oleks kirjas, mis sugu iga inimene selles andmetabelis on.

[1] http://en.wikipedia.org/wiki/Regular_expression
[2] http://et.wikipedia.org/wiki/Isikukood

3 Kuupäevadega töötamine

R-is on kuupäevade jaoks `Date` andmetüüp. See on omapärane andmetüüp: näiliselt on tegemist tekstiga, ent sisuliselt arvuga – kuupäevi saab liita-lahutada, arvutada keskmist. ISO standardile vastav kuupäev on kujul aasta-kuu-päev, sealjuures aasta on nelja numbriga, kuu ja päev kumbki kahe numbriga. Sageli on sisse loetavates andmestikes aga kuupäev teisiti vormindatud. Suvalises formaadis kuupäevalise sõna saab `Date`-tüüpi väärtuseks teisendada käsuga `as.Date(.)`, mille argumendiga `format` saab määrata, millises formaadis kuupäev ette antakse.

```
d1 <- as.Date("22.04.2009", "%d.%m.%Y")
d2 <- as.Date("30.04.2009", "%d.%m.%Y")
d2 - d1
```

```
## Time difference of 8 days
```

```
as.numeric(d2 - d1)
```

```
## [1] 8
```

Siinkohal `%d` tähendab päeva numbrit, `%m` kuu numbrit ning `%Y` neljakohalist aastanumbrit. Rohkem saab lugeda käsu `strptime(.)` abifailist `?strptime`.

Kuupäevaks formaaditud objektidega saab teha kõiki mõistlikke operatsioone, näiteks leida miinimum, keskmine, võrrelda hulki; küll aga ei saa näiteks leida logaritmi või ruutjuurt.Põhjus on selles, et R talletab kuupäevi tegelikult päevade arvuna alates nullpunktist, sealjuures nullpunktiks loetakse vaikimisi 1970-01-01. Selles võime veenduda `as.numeric(.)` käsku kasutades.

```
d3 <- Sys.Date()
paevad <- c(d1, d2, d3)
mean(paevad)
```

```
## [1] "2012-02-13"
```

```
d1 %in% paevad
```

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

`Date`-tüüpi väärtuse saab sobival kujul sõneks teisendada käsuga `format(.)`:

```
format(Sys.Date(), "kuupäev: %d.%m, (%Y. aasta)")
```

```
## [1] "kuupäev: 21.09, (2017. aasta)"
```

3.0.1 Ülesanded

1. Lisa isikute andmestikku veerg, mis sisaldaks iga inimese sünnikuupäeva.
2. Lisa veerg, mis annab inimese vanuse täisaastates tänasel päeval (aastas on ~365,25 päeva).
3. Leia kuupäev mil said/saad 10 000 päeva vanuseks.