Rakendustarkvara: R. Sügis 2017, 1. praktikum*

1 Lühike sissejuhatus

R on programmeerimiskeel ja -keskkond, mis on arendatud statistiliseks andmetöötluseks. R-i kasutavate inimeste hulk on viimase kümmekonna aasta jooksul oluliselt kasvanud nii ülikoolides kui ka ettevõtetes[1]. Eelkõige on populaarsuse põhjus vaba ligipääs, lai valik lisapakette ja kvaliteetsete jooniste tegemise võimalus.

R-i kodulehelt saab vajaliku tarkvara alla laadida: http://www.r-project.org/

1.1 Kasutajaliides

Windowsis on R-il äärmiselt minimalistlik kasutajaliides. Programmi käivitades on näha ainult konsooliaken. Rea ees olev märk `>` näitab, et R ootab uut käsku. Käsud saab kirjutada sümboli `>` järele, sealjuures võib üks käsk paikneda mitmel järjestikusel real. Kui R-i arvates on käsu sisestamine pooleli, on konsoolirea alguses sümbol `+`. Kui käsk on sisestatud, siis `enter`-klahvi vajutamise järel käsk täidetakse ja tulemus trükitakse konsooliaknasse. Sageli on siis rea alguses kantsulgudes mingi arv, mis näitab, mitmes tulemuse element on rea alguses – nimelt võib tulemus mõnikord koosneda mitmest elemendist.

```
> 4 * 6
```

```
[1] 24
```

```
> 4 *
+ 6
```

```
[1] 24
```

```
> 4 * (1:40)
```

```
 [1]   4   8  12  16  20  24  28  32  36  40  44  48  52  56  60  64  68
[18]  72  76  80  84  88  92  96 100 104 108 112 116 120 124 128 132 136
[35] 140 144 148 152 156 160
```

Kui konsooliaknas olles vajutada üles- või allanooleklahvi, saab sirvida eelnevalt täidetud käske.

Konsooli käskude sisestamise asemel on neid mõistlik kirjutada skriptifaili, et hiljem (nt järgmisel päeval) saaks tehtud töö (näiteks mingi analüüsi) kiiresti uuesti üle teha. Skriptifaili tekitamiseks tuleks valida menüüst `File` valik `New script`. Avatakse skriptiaken, kuhu saab käske kirjutada. Kui skriptiaknas olles vajutada klahve `Ctrl+R`, siis vastaval real olev käsk saadetakse R-ile täitmiseks. Kui käsk on kirjutatud mitmel real või on soov mitut käsku järjest jooksutada, võib vastavad read skriptiaknas hiirega ära märkida ning seejärel `Ctrl+R` vajutada. Valides menüüst `File` valiku `Save`, salvestatakse `.R` lõpuga skriptifail kasutaja poolt määratud asukohta.

Kommentaare saab R-i koodis lisada ainult rea lõppu, kasutades trellide-sümbolit:

```
log(5.9)  # võtame naturaallogaritmi arvust 5,9
```

```
## [1] 1.774952
```

```
# terve see rida on kommentaar, sest rea alguses on # ehk trellide sümbol
```

Pane tähele, et **kümnendmurru eraldamiseks** kasutatakse R-is punkti, mitte koma.

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

[1] http://r4stats.com/articles/popularity/

Kahjuks on R-i kasutajaliides mõnevõrra ebamugav. Näiteks konsooliaknas ei saa hiirega kursori asukohta muuta. Samuti on kogu tekst sama värvi (koodi värvimine puudub). Üks alternatiivne kasutajaliides R-i kasutamiseks on **RStudio**[2], kus neid puuduseid pole ning millega on natuke hõlpsam koodi kirjutada.

1.2 Lihtsam aritmeetika

R-is on võimalik teha kõiki lihtsamaid aritmeetilisi tehteid, sealjuures järgitakse matemaatikas kasutatavat tehete järjekorda (sulgusid lisades on võimalik seda muuta):

- `1 + (2 - 3) * 4 / 5`
- `2^3 - 2**3` – astendamiseks saab kasutada sümboleid `^` ja `**`
- `5 %% 3` – modulo (jääk jagamisel)
- `log(exp(1)) * cos(-pi) * sqrt(9) + factorial(4) - choose(4, 2) * min(4, 5, 2)` on sama, mis $\ln(e^1) \cdot \cos(-\pi) \cdot \sqrt{9} + 4! - \binom{4}{2} \cdot \min(4, 5, 2)$
- `1/0` annab tulemuseks `Inf` (*infinity*)
- `0/0` annab tulemuseks `NaN` (*not a number*)

1.2.1 Ülesanded

1. Arvuta enda kehamassiindeks: kaal (kg) / pikkus^2 (m^2). (Õppejõule ei pea näitama.)

1.3 Käsud ja abi saamine

Ülalolevad aritmeetikaavaldised `sqrt(9)`, `choose(4, 2)` jne on tegelikult **käsud** ehk **funktsioonid**, mis oskavad teatud asju teha (praeguses näites teatud arvutusi teha). Kõigil käskudel on sarnane süntaks:

`käsk(argumendid)`

Üldiselt on käsu argumentidel ka nimed, näiteks käsk `choose` tahab täpselt kahte argumenti: `n` ja `k`. Mõnikord on kasulik argumentide nimed välja kirjutada, et hiljem koodi üle lugedes oleks aru saada, mida miski tähendab:

```
choose(n = 4, k = 2)
```

Kui kasutada argumentide nimesid, siis ei ole tähtis, millises järjekorras argumendid käsule ette anda. Ent kui argumentide nimesid ei kasuta, tuleb olla ettevaatlik:

```
choose(k = 2, n = 4)
```

```
## [1] 6
```

```
choose(2, 4)
```

```
## [1] 0
```

Kui mingi konkreetse käsu kohta soovitakse abi saada (näiteks kontrollida, mis on käsu argumentide nimed ja millises järjekorras need tuleks ette anda), võib konsooli trükkida `?käsu_nimi` või ka `??otsitav_tekst`:

```
?choose
??"logarithm"
```

[2] http://www.rstudio.com/

Tasub tähele panna, et R on (nagu suurem osa programmeerimiskeeli) **tõstutundlik** – see tähendab, et programm eristab suuri ja väikeseid tähti. Kui ajada käsu nimes mõni suur- ja väiketäht segi, siis on tulemuseks veateade:

```
Log(5)
```

```
## Error in Log(5): could not find function "Log"
```

1.3.1 Ülesanded

1. Vaata, kuidas töötab käsk `log`, mis on selle argumendid.

2 Muutujatega töötamine

2.1 Väärtuste omistamine ja töökeskkond (*environment*)

Sageli on mugav, kui töös kasutatavatele muutujatele nimi anda – siis saame neid edaspidi nimepidi kutsuda. Näiteks kui nimetada `kaal <- 70` ja `pikkus <- 185`, siis saaks kehamassiindeksit arvutada nii:

```
kaal / (pikkus / 100)**2
```

```
## [1] 20.45289
```

Tekitasime töökeskkonda kaks objekti, mille nimed on `kaal` ja `pikkus`. Täpsemini: tekitasime objektid `kaal` ja `pikkus`, millele omistasime väärtused 70 ja 185. Sümboliühendit `<-` nimetatakse omistamisoperaatoriks. Üldjuhul töötab omistamisoperaatorina ka võrdusmärk `=`, ent on mõned erandjuhtumid, kus need erinevalt töötavad; lisaks on võrdusmärk kasutusel käskude argumentidele väärtuse andmisel.

Kui nüüd muuta `kaal` väärtust: `kaal <- 90` , siis konsoolis ülesnoolega üle-eelmise käsu (KMI arvutamise) üles otsides saab seda lihtsasti uuesti jooksutada.

```
kaal / (pikkus / 100)**2
```

```
## [1] 26.29657
```

Siit maksab tähele panna ka seda, et muutujale uut väärtust omistades kirjutatakse vana lihtsalt üle, mingit hoiatust R ei anna.

Töökeskkonnas olevatest objektidest saab ülevaate käsuga `ls()`. Lisaks skriptifailile on võimalik ka töökeskkonda salvestada ning hiljem see uuesti sisse laadida. Selleks peab konsooliakna aktiivseks tegema ja valima menüüst `File` vastavalt `Save workspace` või `Load workspace`.

Töökeskkonnas olevaid objekte saab kustutada käsuga `rm(.)` (sulgudes olev punkt asendada sobivate argumentidega):

```
rm(kaal, pikkus)
ls() #kontrolli, kas objektid kustutati töölaualt
```

Töökeskkonda on võimalik salvestada RData failiks käsuga `save.image()` või R programmis töötades valides menüüst `File` -> `Save workspace...`. Siis salvestatakse kõik antud töökeskkonnas olnud objektid sellesse faili. Hiljem saab kõik need objektid jälle RData failist töökeskkonda laadida käsuga `load(".RData")`. RData formaati üldiselt teised statistikaprogrammid lugeda ei oska. Kui salvestatud on käskude ehk skripitfail (laiendiga .R), siis objektide eraldi salvestamine pole enamasti vajalik.

2.2 Vektorid. Tehted vektoritega

Eelmises punktis vaatasime juhtu, muutujate väärtuseks oli üks arv. Objekti võib aga moodustada ka mitmest väärtusest, näiteks omistame kuue uuritava kaalud muutujale nimega `kaalud` ja muutujale `liik` uuritavate liigid:

```
kaalud <- c(7, 3.5, 0.4, 2, 3.2, 20.2)
liik <- c("koer", "kass", "rott", "kass", "kass", "koer")
```

Funktsiooni `c(.)` korral on tegemist käsuga, mis sellele antud argumendid kombineerib (*combine*) kokku üheks vektoriks (järjendiks). Kui kombineeritakse erinevat tüüpi väärtuseid (näiteks teksti ja arve), siis muudetakse kõik väärtused selliseks, mis võimaldab võimalikult palju infot säilitada – kõik vektori elemendid peavad olema sama tüüpi.

```
c(987, -Inf, "jutumärkides tekst", kaalud) # pane tähele jutumärke allolevas väljatrükis
```

```
## [1] "987"                "-Inf"               "jutumärkides tekst"
## [4] "7"                  "3.5"                "0.4"
## [7] "2"                  "3.2"                "20.2"
```

Kindla mustriga arvujada tekitamiseks on ka muid mooduseid kui funktsioon `c`:

```
1:5  # täisarvud 1-st 5-ni
```

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

```
2:-6 # täisarvud 2-st -6-ni
```

```
## [1]  2  1  0 -1 -2 -3 -4 -5 -6
```

```
seq(from=0, to=11, by=2) #arvud 0, 0+2, 0+2+2, ..., kuni jõutakse väärtuseni 11
```

```
## [1]  0  2  4  6  8 10
```

```
seq(0, 1, 0.1) # saab ka komaga arvudest järjendeid teha
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
seq(0, 1, length.out=4)
```

```
## [1] 0.0000000 0.3333333 0.6666667 1.0000000
```

```
rep(x=c(1, 3), times=2)  # vektorit korratakse 2 korda
```

```
## [1] 1 3 1 3
```

```
rep(x=c(1, 3), each=2)   # vektori iga elementi korratakse 2 korda
```

```
## [1] 1 1 3 3
```

Käsku `rep(.)` saab kasutada ka **sõnede** kordamisel: `rep(c("a", "b", "c"), c(3, 2, 0))`.

R-i puhul on huvitav see, et sageli tehakse mitmesuguseid tehteid elemendiviisiliselt. Mõnikord tasub olla ettevaatlik: kui asjaosalised vektorid on erineva pikkusega, siis lühemat pikendatakse automaatselt `rep(.)` käsu laadselt

```
1:3 * 4     # iga element korrutatakse 4-ga läbi
```

```
## [1]  4  8 12
```

```
1:3 + 9:7   # elemendid liidetakse paarikaupa
```

```
## [1] 10 10 10
```

```
1:6 * c(1, 2)  # iga teine element korrutatakse 2-ga, ei hoiatata
```

```
## [1]  1  4  3  8  5 12
```

```
1:7 * c(1, 2)  # antakse küll hoiatus, aga 7. element korrutatakse 1-ga
```

```
## Warning in 1:7 * c(1, 2): longer object length is not a multiple of shorter
## object length
```

```
## [1]  1  4  3  8  5 12  7
```

2.2.1 Ülesanded

1. Moodusta vektor nimega `y`, mille väärtuseks oleks arvud `1 1 1  2 2 2 2  3 3 3 3 3` kasutades käsku `rep`.
2. Leia vektor `z`, mille liitmisel `y`-le on tulemuseks vektor, mille kõik koordinaadid on võrdsed `7`.
3. Omista muutujale `u` väärtus 8. Proovi läbi käsud `1:u - 1` ja `1: (u-1)`. Milles on erinevus?

2.3 Väärtuste tüübid.

Nagu öeldud, siis vektori väärtused peavad olema kõik ühte tüüpi. R-is on väärtuste põhitüübid:

- `int` / `integer` – täisarvud (ka negatiivsed)
- `numeric` – reaalarvud
- `cplx` / `complex` – kompleksarvud
- `char` / `character` – sõned (tähemärgid ja muud tekstilised sümbolid, sisestamisel kasutada jutumärke)
- `logical` – tõeväärtused (ainult kaks väärtust: `TRUE` või `FALSE`)

Üht tüüpi väärtust saab teisendada teist tüüpi väärtuseks vastava `as.<tüübi_nimi>` käsuga (`as.integer`, `as.numeric`, `as.character` jne). Kontrollimaks mingi väärtuse tüüpi saab kasutada vastavat käsku `is.<tüübi_nimi>` (`is.integer`, `is.character` jne).

2.3.1 Ülesanded

1. Kas tekstiväärtused saab teisendada arvudeks? Kontrolli vektorite `month.name` ja `x <- c(0:5, "tekst", "T", 234.5, "234,5")` korral, kas need on tekstivektorid st kasuta funktsiooni `is.character(.)`. Seejärel vaata, mis on tulemuseks kui rakendad funktsiooni `as.numeric(.)` nendele vektoritele.
2. Miks annavad käsud `is.integer(1:4)` ja `is.integer(c(1, 2, 3))` erineva tulemuse?

2.4 Puuduvad väärtused

Andmete kogumisel võib juhtuda, et kõigil uuritavatel ei õnnestu vajalikke mõõtmisi teha ja andmetesse tekivad lüngad. Puuduva väärtuse tähistamiseks on R-is tähekombinatsioon `NA` (*Not Available*). Oletame, et analüüsil on vaja teada ka uuritavate vanust, teise vaatlusaluse kohta pole aga see teada:

```
vanused <- c(7, NA, 3, 53, 53, 95)
vanused
```

```
## [1]  7 NA  3 53 53 95
```

Kui teha arvutusi puuduva väärtusega, siis tulemuseks on samuti puuduv väärtus. Proovi:

```
123 + NA
```

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

```
# vanuste vektor teisendada aastateks ja ümardada
round(vanused/12, 1)
```

```
## [1] 0.6  NA 0.2 4.4 4.4 7.9
```

R-i funktsioonidel **võivad** olla lisaargumendid, mis reguleerivad puuduvate väärtusega toimetamist:

```
mean(vanused) # keskmine vanus, kui kaasta NA väärtus on NA
```

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

```
mean(vanused, na.rm = TRUE) # keskmine vanus, kui NA väärtus on eemaldatud
```

```
## [1] 42.2
```

```
table(vanused) # vanuste sagedustabel, NA väärtust ei esitata
```

```
## vanused
##  3  7 53 95
##  1  1  2  1
```

```
table(vanused, useNA = "ifany") # lisame sagedustabelisse ka puuduva väärtuse, kui see esineb
```

```
## vanused
##    3    7   53   95 <NA>
##    1    1    2    1    1
```

```
table(liik, useNA = "always") # lisame sagedustabelisse ka puuduva väärtuse
```

```
## liik
## kass koer rott <NA>
##    3    2    1    0
```

```
summary(vanused) # NA väärtuste arv tuuakse vaikimisi välja
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     3.0     7.0    53.0    42.2    53.0    95.0       1
```

2.4.1 Ülesanded

1. Rakenda funktsiooni `is.na(.)` vektorile `<- c("a", "NA", NA, 0)`. Kas tulemus on oodatav?
2. Kas funktsioonil `which.min(.)` on lisaargument, mis reguleerib puuduvate väärtustega tegelemist?

2.5 Tõeväärtused ja tõeväärtusvektorid

Kui käsuga `mean` kasutada argumenti `na.rm`, siis argumendi väärtus võib olla `TRUE` või `FALSE`. Tegemist on **tõeväärtustega**. Kuna tõeväärtustega saab teha **loogilisi tehteid** (ehk neid kombineerida), siis on neist kasu ka mujal kui `na.rm` argumendi väärtustamisel.

Loogilisi tehteid on kolm: korrutamine (`&`), liitmine (`|`) ja eitus (`!`).

tehe	tulemus
TRUE & TRUE	TRUE
TRUE & FALSE	FALSE
FALSE & TRUE	FALSE
FALSE & FALSE	FALSE
TRUE | TRUE	TRUE
TRUE | FALSE	TRUE
FALSE | TRUE	TRUE
FALSE | FALSE	FALSE
!TRUE	FALSE
!FALSE	TRUE

Tõeväärtus on tulemuseks siis, kui võrdleme kahte objekti/väärtust omavahel.

võrdlus	sümbol	näide
võrdumine	`==`	`2 == 3 ; "a" == "A"`
mittevõrdumine	`!=`	`2 != 3 ; "a" != "A"`
väiksem kui	`<`	`2 < 3 ; "a" < "A"`
väiksem või võrdne	`<=`	`3 <= 2 ; "a" <= "A"`
suurem kui	`>`	`3 > 2 ; "a" > "A"`
suurem või võrdne	`>=`	`3 >= 2 ; "a" >= "A"`

Nagu ennist mainitud, siis R-is tehakse tehteid vektoritega elementhaaval. Seepärast ka siis, kui võtame mingi arvulise vektori ja võrdleme seda mingi arvuga, siis võrreldakse igat elementi eraldi ja tulemuseks on tõeväärtustest koosnev vektor, milles on sama palju elemente kui oli elemendiviisilisi võrdluseid. Sama kehtib ka juhul, kui võrreldakse muud tüüpi väärtuseid (näiteks tekstilisi väärtuseid).

```
kaalud > 10
vanused == 53
liik == "kass"
vanused == kaalud
liik == "kass" | liik == "koer"
```

Kui soovime mingit väärtust võrrelda `NA`-ga, et teada saada, kas tegemist on puuduva väärtusega või mitte, siis topeltvõrdusmärgid ei tööta, vaid tuleb kasutada käsku `is.na(.)`

```
vanused == NA
```

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

```
is.na(vanused)
```

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

Mõnikord on soov kontrollida, kas mingi väärtus leidub etteantud hulgas. Siis on sobilik kasutada operaatorit `%in%` :

```
1:4 %in% c(2, 5)
```

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

Tõeväärtuste puhul on huvitav see, et kui nendega tavalisi arve korrutada või liita vms, siis konverteeritakse tõeväärtused arvudeks: `TRUE` muutub arvuks 1 ja `FALSE` muutub arvuks 0.

```
is.na(vanused) * 1
```

```
## [1] 0 1 0 0 0 0
```

```
sum(is.na(vanused))  # mitmel vaatlusalusel on vanus puudu
```

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

Konverteerimine toimib automaatselt. Kui on endal soov tõeväärtuseid arvuliseks muuta, saab rakendada käsku `as.numeric(.)` tõeväärtustega vektorile. Saab ka vastupidi: kui on soov arvusid tõeväärtusteks muuta, saab seda teha käsuga `as.logical(.)`. Proovi!

2.5.1 Ülesanded

1. Tekita käsku `c(.)` kasutades viiest elemendist koosnev tõeväärtusvektor, sealjuures neljas element olgu `NA`. Konverteeri see vektor arvuliseks käsuga `as.numeric`. Missuguse arvulise väätuse sai `NA`?
2. Selgita välja, millised arvud konverteeritakse väärtuseks `TRUE` ja millised väärtuseks `FALSE`, kui kasutada käsku `as.logical`.
3. Mis on loogiliste tehete tulemus, kui üks tehtes osalev väärtus on `NA`?
4. Proovi läbi järgmised käsud `as.integer(c("tere", 0, 1, TRUE, FALSE))` ja `as.integer(c(0, 1, TRUE, FALSE))` ning mõtle, miks väärtused `TRUE` ja `FALSE` teisendatakse neil juhtudel erinevalt.