Rakendustarkvara: R. Kevad 2018, 9. praktikum*

1 Programmeerimine R-is

R on programmeerimiskeele S[1] implementatsioon; selles on olemas enamus teistest programmeerimiskeeltest tuttavaid konstruktsioone nagu tsüklid, `if-else` tingimuslaused ning võimalus kirjutada lisafunktsioone.

1.1 Tsüklid

Tsükkel on programmikonstruktsioon, mis kordab teatud tegevust mitu korda järjest. See on kasulik näiteks siis, kui tahame vektori ükshaaval järjest üle vaadata ja vastavalt elemendi väärtusele midagi teha. R-is on võimalik kasutada kahte tüüpi tsükleid:

- `for`-tsükkel teeb ettemääratud arvu samme. Tsükli defineerimisel antakse ette mingi vektor, mille elemente ükshaaval hakatakse läbi käima. Enamasti koosneb see vektor järjestikustest täisarvudest, näiteks täisarvudest 1:10, kuid võib olla ka tekstiväärtustega.

```
for (i in 1:10) {  # NB! siinne 'in' on ilma protsendimärkideta!
  print(i)
}

for (loom in c("kass", "kaamel", "kilpkonn", NA, "kaan")) {
  print(loom)
}
```

- `while`-tsüklit korratakse seni, kuni teatud tingimus saab täidetud. Nn peatumistingimus tuleb kindlasti tsükli defineerimisel ette anda, sealjuures tuleb tsükli kirjutamisel olla ettevaatlik, et tsükkel lõpmatult kaua korduma ei jääks. `while`-tsükkel on kasulik siis, kui realiseerida mingit algoritmi, mille peatumine sõltub algoritmi koondumisest. Lõpmatu kordumise ohu vältimiseks on mõistlik tsükli päisesse lõpetamistingimuste hulka lisada tsüklisammude loendur.

```
a <- 1
while(a < 10) {
  print(a)
  a <- a + 1  # kui seda rida poleks, jääks tsükkel lõpmatult korduma
}
```

1.1.1 Ülesanded

1. Loe sisse Massachusetts'i andmestik:
   `mass <- read.table("http://kodu.ut.ee/~annes/Rkursus/mass.txt", sep = "\t", header = TRUE)`
   Kasutades tsüklit leia tabel, kus oleks üle 15 aastaste korral arvutatud ametialade (`OCCP`) kaupa keskmine ja summaarne nädala töötundide (`WKHP`) arv. Vaata ainult neid, kel on ametiala teada ja kes pole `OCCP` järgi töötud.
2. Tee sama tabel **data.table** teegi käske kasutades; võrdle kulunud aega kasutades käsku `system.time({.})`.

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

[1] http://en.wikipedia.org/wiki/S_(programming_language)

1.2 Tingimuslause `if`

Nagu praktiliselt kõigis programmeerimiskeeltes, on ka R-is tingimuslause `if`, millega saab kontrollida erinevate loogiliste tingimuste kehtimist ja vastavalt sellele, kas tulemus on TRUE või FALSE, rakendada erinevaid tegevusi. `if`-ploki sisu täidetakse juhul, kui tingimuse väärtus on `TRUE`; võib defineerida ka `else`-ploki, mis täidetakse siis, kui `if`-tingimuse väärtus on `FALSE`.

```
if (väärtus %% 2 == 0) {
  print(väärtus)
} else {              # kui else-lauset kasutada, peab see olema if-lause suluga samal real
  print(väärtus * 10)
}
```

Meenutuseks mõned olulisemad võrdlustehted:

- `==` – kahe elemendi võrdsus
- `!=` – kahe elemendi erinevus
- `<`, `<=` – kas üks element on väiksem (või võrdne) kui teine
- `>`, `>=` – kas üks element on suurem (või võrdne) kui teine
- `%in%` – kas element kuulub mingi vektori elementide hulka
- `is.na(.)` – kas väärtus on NA
- `is.factor(.)`, `is.numeric(.)`, `is.character(.)`, `is.logical(.)` – kas objekt on antud tüüpi

Sageli tasub `if` lause tingimuste hulka lisada ka väärtuse või tüübi kontrollimine.

```
if (!is.na(väärtus) & is.numeric(väärtus) & väärtus %% 2 == 0) {  }
```

Kui `if`-ploki sisu on üherealine, võib loogelised sulud ka ära jätta. Samuti võib mitu käsku kirjutada ühele reale, eraldades need semikooloniga. Üldiselt aga pole see koodi loetavuse seisukohast soovitatav.

Kasulikud on ka `ifelse(.)` ja `switch(.)` käsud. Käsu `ifelse(.)` esimene argument on tõeväärtusvektor, teine argument tulemus, mis vastab tõesele väärtusele, ja kolmas, mis vastab väärale väärtusele. Käsu `switch(.)` esimene argument on tavaliselt täisarv (ainult üks täisarv!) ning ülejäänud argumendid erinevatele täisarvudele vastavad toimingud (kasvavas järjekorras).

```
table(ifelse(is.na(mass$WAGP), "palgatu", "palgaga")  )
```

1.2.1 Ülesanded

1. Loe sisse arstivisiitide andmestik:
   `visiidid <- read.table("http://kodu.ut.ee/~annes/Rkursus/visiidid.txt", sep = "\t", header = TRUE)`
   `for`-tsüklit kasutades käi läbi kõik inimesed ning trüki ekraanile nende inimeste isikukoodid, kellel esimese ja viimase vererõhu mõõtmise vahe on > 5.

1.3 Funktsioonide defineerimine

Uusi käske ehk **funktsioone** saab R-is tekitada funktsiooni `function(.){.}` abil. Funktsiooni **päises** on võimalik defineerida ja vaikeväärtustada argumendid, mida see funktsioon töö jaoks vajab. Argumendina võib defineerida/kasutada mistahes objekti, k.a mingit muud funktsiooni. Loogeliste sulgude vahel paiknevas funktsiooni **kehas** tuleb kirjeldada, mida teha nende argumentidega, mis funktsiooni päises on defineeritud. Hea tava on see, et funktsioon toimetab ainult nende objektidega, mis päises on defineeritud, ega muuda funktsioonivälisеid objekte.

```
minukäsk <- function(argument1, argument2 = "tere", argument3 = mean, ...) {
  # funktsiooni sisu
  tagastatav_objekt <- argument3(argument1, ...)
  return(tagastatav_objekt)
}
```

Enamjaolt on soov, et funktsioon tagastaks midagi käsu `return(.)` abil. Kui funktsioon midagi tagastama ei pea, ei pea `return(.)` käsku kirjutama (siiski tagastatakse sel juhul viimase käsu tulemus).

Funktsiooni argumentidel võivad olla **vaikeväärtused** (nagu `argument2 = "tere"`), ent sageli on vähemalt ühe argumendi väärtus vaja ette anda (antud näites on vaja kindlasti määrata `argument1` väärtus). Argumentideks võivad olla ka funktsioonid (praeguses näites `argument3 = mean`). Eriline argument on `...`, millega saab võimaldada teiste argumentidena kasutatavate funktsioonide lisaargumentide väärtuste lisamist.

```
minukäsk(argument1 = 1:6, argument3 = mean, na.rm = T) # argument na.rm saadetakse käsule mean
minukäsk(1:6, , min)
```

1.3.1 Ülesanded

1. Kirjutada funktsioon, mis teisendab etteantud vektori väärtused Z-skoorideks (igast väärtusest lahutatakse vektori keskmine ja jagatakse standardhälbega).
2. Defineeri uus funktsioon, mida saaks kasutada paketi **dplyr** käsus `summarise(.)` selleks, et leida arstivisiitide andmestikus iga inimese puhul, kui sageli (mitu korda aastas) külastab ta keskmiselt arsti. Andmestikus on genereeritud andmeid vahemikus 2004 kuni 2014, aga arvuta nõutud keskmine igal inimesel selles ajavahemikus milles on tal selles andmestikus andmed. Selleks on vaja defineerida funktsioon, mis etteantud `data.frame`-tüüpi objekti puhul leiab esimese ja viimase visiidi kuupäeva, leiab mitu aastat jääb nende kuupäevade vahele, loendab visiitide arvu ja leiab nende kahe arvu jagatise.

2 Juhuarvud. Simuleerimine

Statistikas tuleb ette olukordi, kus kõige lihtsam viis mingi väite kontrollimiseks on vastavat situatsiooni simuleerida. R-is on palju juhuarvude genereerimise käske kujul `rJAOTUS(.)`, teoreetiliste jaotuste kvantiilide leidmiseks on käsud kujul `qJAOTUS(.)`; tihedus- ja jaotusfunktsiooni väärtuste teadasaamiseks on vastavad käsud `dJAOTUS(.)` ja `pJAOTUS(.)`. Et simulatsioonid oleks korratavad (st iga kord koodi läbi jooksutades tuleks sama tulemus), võiks ette anda pseudojuhuarvude generaatori algväärtuse käsuga `set.seed(algväärtus)`. Näited ühtlase jaotusega:

```
set.seed(1357) # kui seda rida poleks, tuleks järgmiste ridadega iga kord erinev tulemus
(x <- runif(n = 3, min = 0, max = 10))  # kolm arvu ühtlasest jaotusest U(0,10)
```

```
## [1] 6.427499 5.899772 9.613298
```

```
punif(q = x, min = 0, max = 10)  # jaotusfunktsioon vastaval kohal
```

```
## [1] 0.6427499 0.5899772 0.9613298
```

```
qunif(p = c(0.3, 0.75), min = 0, max = 10)  # 30. ja 75. protsentiil jaotusel U(0, 10)
```

```
## [1] 3.0 7.5
```

Mõnede teiste jaotuse juhuarvude genereerimise funktsioonid (kvantiilide, tihedus- ja jaotusfunktsioonide jaoks tuleb esimene täht asendada vastavalt 'q', 'd' või 'p' tähega):

- `rbinom(.)` – binoomjaotus; seda tuleb kasutada ka Bernoulli jaotusest arvude genereerimiseks
- `rpois(.)` – Poissoni jaotus
- `rnorm(.)` – normaaljaotus
- `rt(.)` – t-jaotus
- `rchisq(.)` – hii-ruut jaotus
- `rexp(.)` – eksponentjaotus

Väga kasulik on käsk `sample(.)`, mis aitab lihtsasti teha juhuvalikut etteantud vektori elementide hulgast. Näiteks siis, kui on vaja mingisuguse algoritmi tööd teatud suurel andmestikul kontrollida, on mõistlik võtta sellest andmestikust juhuvalim ja kontrollida selle peal.

2.0.1 Ülesanne

1. Kirjuta funktsioon, mis genereerib kasutaja poolt määratud mõõtmetega arvutabeli (maatriksi) Poissoni jaotusest arvudega, kusjuures Poissoni jaotuse parameeter on samuti kasutaja poolt määratud.
2. Kirjuta funktsioon, mis ette antud 2x2 maatriksi põhjal teeb hii-ruut testi ja tagastab vastava p-väärtuse.