Rakendustarkvara: R. Kevad 2018, 2. praktikum*

1 Väärtuste tüübid.

Nagu eelmises praktikumis 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 vahel 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.numeric`, `is.character` jne).

Eelmises praktikumis moodustasime mõned vektorid. Tekitame need uuesti

```
kaalud <- c(7, 3.5, 0.4, 2, 3.2, 20.2)
liik <- c("koer", "kass", "rott", "kass", "kass", "koer")
vanused <- c(7, NA, 3, 53, 53, 95)
```

ja kontrollime väärtuste tüüpe:

```
is.integer(kaalud)
```

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

```
is.numeric(kaalud)
```

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

```
class(liik)
```

```
## [1] "character"
```

```
class(vanused)
```

```
## [1] "numeric"
```

Eraldi käsk on ka puuduvate väärtuste esinemise kontrolliks:

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

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

1.0.1 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?
3. Rakenda funktsiooni `is.na(.)` vektorile `z <- c("a", "NA", NA, 0)`. Kas tulemus on oodatav?

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

1.1 Tõeväärtused ja tõeväärtusvektorid

Kui käsuga `mean` kasutada argumenti `na.rm`, siis selle 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
```

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

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

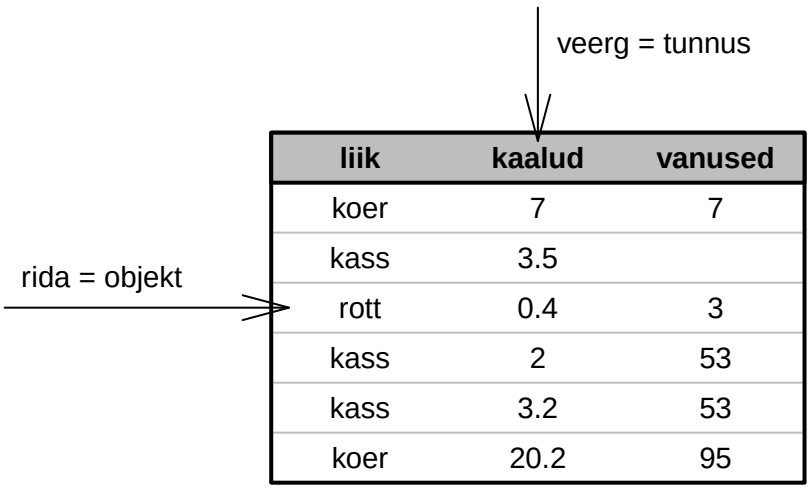
Konverteerimine toimib automaatselt. Kui on endal soov tõeväärtuseid arvuliseks muuta, saab rakendada käsku `as.numeric(.)` tõeväärtustega vektorile. Saab ka vastupidi: kui on soov arvusid tõeväärtusteks muuta, saab seda teha käsuga `as.logical(.)`. Proovi!

1.1.1 Ülesanded

1. Tekita käsku `c(.)` kasutades viiest elemendist koosnev tõeväärtusvektor, sealjuures neljas element olgu `NA`. Konverteeri see vektor arvuliseks käsuga `as.numeric`. Missuguse arvulise väärtuse 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.

2 Andmestik, andmete import

Tavaliselt koosnevad andmestikud ridadest ja veergudest (tulpadest), kus iga rida vastab mingile mõõtmisobjektile ja iga veerg vastab mingile mõõdetud omadusele (tunnusele).



liik	kaalud	vanused
koer	7	7
kass	3.5	
rott	0.4	3
kass	2	53
kass	3.2	53
koer	20.2	95

Erinevad statistikaprogrammid kasutavad andmestike säilitamiseks eri failiformaate. Et andmestikke ühest programmist teise saada, on üheks võimalikuks lahenduseks andmetabel salvestada vahepeal tekstiformaadis failiks, näiteks `.txt` või `.csv` tüüpi failiks ja importida tekstifail.

MS Exceli faile ning statistikatarkvarade spetsiifilisi andmeformaate (nt Stata `.dta`, SPSS-i, `.sav`) ei saa R-i importida baaspaketi käskude abil. Selliste failide impordiks on vaja kasutada mõnda R-i lisapaketti.

Tänases praktikumis vaatame nii tekstifailis andmete kui ka mõne teise statistikaprogrammi andmefailide importimist R-i.

2.1 Andmete sisselugemine ja faili kirjutamine (tekstifail)

Kõige olulisem käsk tekstikujul andmestike sisselugemiseks on `read.table()`. Sellel on mitmeid **argumente**, mille abil saab täpsustada erinevaid asjaolusid, mis antud andmestikku puutuvad:

`file`	-	faili asukoht ja nimi koos faililaiendiga (peab olema jutumärkides)
`header`	-	kas faili esimeses reas on veerunimed? (TRUE = jah, FALSE = ei)
`sep`	-	millise sümboliga on veerud eraldatud? (nt `"\t"` - tabulaatori sümboliga)
`dec`	-	kuidas on märgitud kümnendmurru eraldaja? (nt `"."` või `","`)
`quote`	-	millega on tekstilised väärtused ümbritsetud? (nt `"\""` tähendab, et kahekordsete jutumärkidega)
`dec`	-	kuidas on märgitud kümnendmurru eraldaja? (nt `"."` või `","`)
`na.strings`	-	kuidas on failis puuduvad väärtused tähistatud?
`fileEncoding`	-	mis tähekodeeringut kasutatakse (Windowsis sageli `"latin9"`, Macis/Linuxis sageli `"utf8"`)

Faili asukoht võib olla kõvakettal või võrgus. Kui käsu `setwd` abil on määratud, milline on käimasoleva töösessiooni **töökataloog**, ja andmefail on selles kataloogis, siis piisab käsule `read.table` ainult faili nime etteandmisest (täispikka asukohta ei pea andma). Windowsis on kombeks kaustastruktuuri tähistamiseks kasutada kurakaldkriipsu (tagurpidi kaldkriipsu) \, aga R-is on sellisel kaldkriipsul eriline tähendus – sellega märgitakse, et järgneb erisümbol (nt `\t` on tabulaatori sümbol). Seepärast tuleb kataloogitee märkimisel kasutada kahekordseid kurakaldkriipse, näiteks:

```
setwd("C:\\Users\\mina\\Rkursus\\")
```

Teine võimalus on kasutada tavalist kaldkriipsu, nagu MacOS-is ja Linuxites:

```
setwd("C:/Users/mina/Rkursus/")
```

Andmete sisselugemisel tuleks anda andmetabelile nimi (salvestada see mingi objektina), vastasel juhul andmestik trükitakse lihtsalt ekraanile ja sellega enam midagi muud teha ei saa:

```
näide1 <- read.table("http://kodu.ut.ee/~annes/Rkursus/esimene.txt",
                     header = T, sep = "\t", dec = ",")
```

Töökeskkonnas olevaid andmetabeleid saab faili kirjutada käsuga `write.table(.)`, andes käsule ette salvestatava andmestiku nime ning loodava faili nime jutumärkides (vajaduse korral koos kataloogiteega):

```
write.table(näide1, "failinimi.txt", sep = "\t")
```

2.1.1 Ülesanded

1. Vaata `read.table` käsu argumentide täielikku loetelu abifailist.
2. Aadressil http://www.ut.ee/~annes/Rkursus/ leiad failid *tabel1.csv*, *tabel2*, *tabel3.txt*, *tabel4.tab*. Tutvu nendega (Notepadi vmt kasutades) ja proovi need seejärel R-i korrektselt sisse lugeda. Veendu, et R-is olevates tabelites on sama palju veerge kui originaalandmestikes.

3. USA riiklikud institutsioonid võimaldavad päris sageli andmekogudele vaba ligipääsu. Selles aines kasutame näidisandmestikena USA Rahvaloendusbüroo[1] poolt kogutud andmeid, millele saab ligi IPUMS-USA[2] liidese kaudu[3]. Aadressil http://www.ut.ee/~annes/Rkursus/ on fail *mass.txt*, milles on Massachusettsi osariigis ühe valikuuringuga kogutud andmed. Loe see R-i sisse, andmestiku objektile pane nimeks `andmed`. Tutvu ka tunnuste kirjeldusega: http://www.ut.ee/~annes/Rkursus/descr.txt. Andmestikuobjektist esmase ülevaate saamiseks kasuta käsku `str(andmed)`.

2.2 Lisapakettide kasutamine

Üks R-i populaarseks muutumise põhjuseid on rikkalik lisapakkettide olemasolu. Tõenäoliselt leidub iga praktilise statistika-alase (ja ka mõne muu valdkonna) probleemi jaoks omaette pakett (*package*). Näiteks paketid `foreign`, `readstata13`, `haven`, `readxl` sisaldavad funktsioone, mis aitavad teistes kui tekstiformaatides andmestikke hõlpsamini R-i sisse lugeda. Andmete impordiks sobilikke lisapakette on lisaks nimetatutele veel. Siinkohal vaatame aga kahe paketi `haven` ja `readxl` võimalusi.

Lisapaketid ei ole tavaliselt R-iga kaasas, vaid need tuleb installeerida. Kui paigaldatav pakett vajab omakorda mingeid muid pakette, siis installeeritakse ka need. Kui kasutatakse R-i installeerimisel paika pandud vaikeseadistusi, siis töösessiooni korral esimest korda mingit paketti installeerides küsitakse, millisest serverist soovib kasutaja neid alla laadida. Järgmistel kordadel sama töösessiooni jooksul pakette paigaldades seda enam ei küsita.

Installeerime lisapaketid `readxl` ja `haven`

```
install.packages("readxl")
install.packages("haven")
# NB! Arvutiklassi arvuti korral võib olla vajalik salvestuskoha ettemääramine
# install.packages("readxl", lib = "C:/kataloog-installimiseks")
```

Kui pakett on installitud, siis järgmisel töösessioonil seda enam uuesti installeerima ei pea. Arvutisse paigaldatud paketid tuleb iga kord uut R-i sessiooni alustades kasutamiseks sisse laadida käsuga `library` (või `require`), mille argumendiks on laaditava paketi nimi:

```
library(readxl)
# Kui installimisel oli käsitsi määratud salvestuskoht:
# library(readxl, lib.loc = "C:/kataloog-installimiseks")
```

2.3 Andmete import programmide MS Excel, SAS, Stata, SPSS failidest

2.3.1 MS Excel failid

Lisapaketi `readxl` käsu `read_excel(.)` abil saab importida MS Exceli failide (nii *.xlsx* kui *xls*) töölehti. Salvesta aadressilt http://www.ut.ee/~annes/Rkursus/ oma töökataloogi fail *tudengite-arv.xlsx*, mis sisaldab infot TÜ tudengite arvude kohta aastate lõikes. Ava esmalt fail MS Exceli abil ning vaata millised töölehed failis on. Seejärel vaata ka R abil millised on töölehtede nimed ja impordi teine tööleht, millel on andmed avatud ülikooli õpilaste arvude kohta.

```
list.files() # vaata, mis nimega failid on töökaustas
excel_sheets("tudengite-arv.xlsx") # töölehtede nimed MS Exceli failis
AY <- read_excel("tudengite-arv.xlsx", sheet = "avatud ylikool") # importimine
```

[1] http://www.census.gov/data.html

[2] Steven Ruggles, J. Trent Alexander, Katie Genadek, Ronald Goeken, Matthew B. Schroeder, and Matthew Sobek. Integrated Public Use Microdata Series: Version 5.0 [Machine-readable database]. Minneapolis, MN: Minnesota Population Center [producer and distributor], 2010.

[3] https://usa.ipums.org/usa-action/variables/group

```
# AY <- read_excel("tudengite-arv.xlsx", sheet = 2)  # sama
str(AY)  # vaata tulemust
```

MS Exceli failis ei pruugi andmed alata alati esimesest reast, näiteks töölehel *tabel* on esimeses reas tabeli nimi ning teine rida on tühi, andmetabel algab kolmandalt realt. Selle, et esimesed kaks rida tuleks andmete impordil vahele jätta, saab käsule `read_excel(.)` ette anda argumendi `skip` abil:

```
tabel <- read_excel("tudengite-arv.xlsx", sheet = "tabel", skip = 2)
str(tabel)
```

2.3.2 SAS, Stata ja SPSS failid

Programmide SAS, Stata ja SPSS failide importimiseks saab kasutada paketi `haven` vastavaid funktsioone `read_sas(.)`, `read_stata(.)` ja `read_spss(.)`. Salvesta aadressilt http://www.ut.ee/~annes/Rkursus/ oma töökataloogi kolm faili nimega *effort* ja loe need R-i:

```
library(haven)
andmestik_SAS <- read_sas("effort.sas7bdat")
andmestik_Stata <-  read_stata("effort.dta")
andmestik_SPSS <- read_spss("effort.sav")

str(andmestik_SAS)
str(andmestik_Stata)
str(andmestik_SPSS)
```

R-i andmestiku saab `haven` paketi abil ka eksportida vaadeldud programmide andmefailiks käsuga `write_<...>(.)`, kus `<...>` tuleks siis asendada sobiliku programminimega.

2.3.3 Ülesanded

1. Impordi failist *tudengite-arv.xlsx* viimane tööleht *kokku*, kus on tudengite arvud aastate lõikes (liidetud statsionaar ja avatud ülikool). Vaata esmalt ka abilehte kasutatava funktsiooni kohta: `?read_excel`

2.4 Esmase ülevaate saamine andmetabelist

Käsuga `read.table(.)` sisse loetud andmestik on erilist tüüpi, **`data.frame`**-tüüpi objekt. Andmestikust saab kiire ülevaate järgmiste käskudega:

`nrow(andmed)`	-	mitu rida on andmestikus
`ncol(andmed)`	-	mitu veergu on andmestikus
`dim(andmed)`	-	mitu rida ja mitu veergu on andmestikus
`str(andmed)`	-	andmestiku struktuur: mis tüüpi iga tunnus on ja esimesed väärtused
`summary(andmed)`	-	lühike kirjeldav statistika iga tunnuse kohta
`names(andmed)`	-	veergude nimed
`head(andmed)`	-	andmestiku mõned esimesed read

Käsu `summary(.)` puhul on näha, et mõne tunnuse puhul arvutatakse keskmine, miinimum, maksimum jne, aga teise tunnuse puhul esitatakse sagedused. See, millist kirjeldusviisi kasutatakse, tuleneb tunnuse tüübist. Kui käsuga `str(.)` vaadata, mis on tunnuste tüübid selles andmestikus, on näha kahte tüüpi tunnuseid: `int` ja `Factor`. On näha, et `summary(.)` käsk arvutab `int`-tüüpi tunnustele keskmisi jne, `Factor`-tüüpi tunnustele aga sagedusi. Faktortüüpi tunnus sarnaneb sõnele (*character*), kuid üheks erinevuseks on näiteks fikseeritud väärtuste hulk. Faktortunnust vaatame järgmises praktikumis ka täpsemalt.