Rakendustarkvara: R. Sügis 2014

Contents

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 lihtsus.

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

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

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. Seda saab sümboli `>` järele kirjutada, sealjuures võib seda teha mitmel 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
# terve see rida on kommentaar, sest rea alguses on # ehk trellide sümbol
```

Kümnendmurru eraldamiseks kasutatakse R-is punkti, mitte koma.

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] http://www.rstudio.com/

- `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*)
- `sum(1, 5, 6)` – summeerib kõik arvud
- `prod(1, 5, 6)` – korrutab kõik arvud

1.2.1 Ülesanded

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

1.3 Käsud ja abi saamine

Kõige lihtsam on abi saada interneti otsimootorite kaudu.

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 tegelikult 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"
```

Tasub tähele panna, et R on (nagu suurem osa programmeerimiskeeli) **tõstutundlik** – see tähendab, et suured ja väikesed tähed on erinevad:

```
Log(5)
```

```
## Error in eval(expr, envir, enclos): could not find function "Log"
```

1.3.1 Ülesanded

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

1.4 Objektid ja töökeskkond (*environment*)

Sageli on mugav, kui töös kasutatavatele asjadele nimi anda – siis saame neid asju 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.

Mõistlik on jätta omistamisoperaatori (ja ka muude, nt võrdlusoperaatorite) ümber tühikud. Näiteks `x<-3` teostab omistamise, kuigi võib-olla kasutaja tahtis objekti `x` võrrelda arvuga `-3`. Võrdlemise jaoks peaks kirjutama `x < -3`, omistamise jaoks peaks kirjutama `x <- 3`.

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

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

Töökeskkonnas olevaid objekte saab kustutada käsuga `rm(.)`:

```
rm(kaal, pikkus)
```

Töökeskkonda on võimalik salvestada RData failiks käsuga `save(.)` või 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 formaati üldiselt teised statistikaprogrammid lugeda ei oska.

2 Andmestikud

2.1 Andmete sisselugemine ja faili kirjutamine

Tavaliselt koosnevad andmestikud ridadest ja veergudest (tulpadest), kus iga rida vastab mingile mõõtmisobjektile ja iga veerg vastab mingile mõõdetud omadusele (tunnusele). Erinevad statistikaprogrammid kasutavad andmestike säilitamiseks eri failiformaate. Et andmestikke ühest programmist teise saada, on mõistlik andmetabel salvestada tekstiformaadis failiks, näiteks `.txt` või `.csv` tüüpi failiks.

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 (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)
`na.strings`	kuidas on 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 ainult faili nime andmisest (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 Windowsis 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:

```
andmed <- read.table("http://kodu.ut.ee/~maitraag/rtr/mass.txt", sep = "\t")
```

Töökeskkonnas olevaid andmetabeleid saab tekstiformaadis faili (mida teised programmid sisse lugeda oskavad) kirjutada käsuga `write.table`.

Mõnd enamlevinud statistikatarkvara-spetsiifilist formaati (nt Stata `.dta`, SPSS-i, `.sav`) on võimalik lisapaketti `foreign` kasutades samuti sisse lugeda. Lisapakettidest tuleb juttu hiljem.

2.1.1 Ülesanded

1. Vaata `read.table` käsu argumentide täielikku loetelu abifailist.
2. Aadressil http://www.ut.ee/~maitraag/rtr/ leiad failid tabel1.csv, tabel2, tabel.3.txt, tabel4.tab. Tutvu nendega (Notepadi 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[3] poolt kogutud andmeid, millele saab ligi IPUMS-USA[4] liidese kaudu[5]. Aadressil http://www.ut.ee/~maitraag/rtr/ on fail mass.txt, milles on Massachusettsi osariigis ühe valikuuringuga kogutud andmed. Loe see R-i sisse.
4. Tutvu tunnuste kirjeldusega: http://www.ut.ee/~maitraag/rtr/descr.txt.

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

[4] 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.

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

2.2 Andmestiku kirjeldamine.

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 millised on mõned esimesed väärtused
`summary(andmed)`	lühike kirjeldav statistika iga tunnuse kohta
`names(andmed)`	veergude nimed
`head(andmed)`	andmestiku mõned esimesed veerud

Käsu `summary` puhul on näha, et mõne tunnuse puhul arvutatakse keskmine, miinimum, maksimum jne, aga teise tunnuse puhul esitatakse sagedused. Kuidas R teab, mida teha? Väga lihtsalt: tunnuse tüübi järgi. 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.

2.3 Veergude ja ridade eraldamine andmestikust: indeksi ja nime järgi

Kui tahame andmestikust ainult üht veergu uurida, siis kõige mugavam on kasutada dollari-sümbolit:

```
vanused <- andmed$AGEP
median(andmed$AGEP)
median(vanused)
```

Üldiselt on `data.frame` kahemõõtmeline tabel, mis tähendab, et iga elemendi asukoht selles tabelis on ära määratud rea ja veeru numbriga. Rea- ja veerunumbrite abil andmestikust infot eraldades tuleb kasutada kantsulgusid:

```
andmed[3, 2]  # kolmas rida, teine veerg
andmed[ , 2]  # kogu teine veerg
andmed[3,  ]  # kogu kolmas rida
```

Korraga on võimalik eraldada ka mitut rida või veergu, kasutades selleks vahest kõige olulisemat käsku `c(.)`:

```
andmed[, c(2, 4)]  # teine JA kolmas veerg, trükitakse ekraanile
teinekolmas <- andmed[, c(2, 4)]  # teine ja kolmas veerg salvestatakse uude andmestikku / uue objektin
huvipakkuvad_veerud <- c(2, 4)  # tekitame objekti, milles on kirjas huvipakkuvate veergude numbrid
andmed[, huvipakkuvad_veerud]  # kasutame seda objekti andmestikust veergude eraldamiseks
andmed[c(5, 9), ]  # viies JA üheksas rida
```

Tihti on veeruindeksite asemel mugavam kasutada veergude nimesid, mida samuti kantsulgudes kasutada (peavad olema jutumärkides):

```
andmed[, c("AGEP", "WAGP")]  # eraldame veerud "AGEP" ja "WAGP"
```

Mõnikord harva on ka ridadel nimed, siis saab neid sama moodi kasutada ridade eraldamisel.

2.4 Lihtsam kirjeldav statistika

Allpool on mõned käsud, mille abil huvipakkuvat tunnust (veergu) kirjeldada.

- `min(tunnus)`, `max(tunnus)`, `median(tunnus)`, `mean(tunnus)`, `sd(tunnus)` – arvulise tunnuse karakteristikud
- `quantile(tunnus, kvantiil)` – saab leida kvantiile ehk protsentiile arvulisele tunnusele
- `length(tunnus)` – mitu elementi on antud veerus
- `table(tunnus)` – saab koostada sagedustabelit (kasulik `Factor`-tüüpi tunnuse kirjeldamisel) `table(tunnus1, tunnus2)` – koostab kahemõõtmelise sagedustabeli `table(tunnus1, tunnus2, tunnus3)` – kolmemõõtmeline sagedustabel; inimesel ebamugav lugeda, arvutiga töödelda aga mugav
- `t(tabel)` – vahetab tabeli read ja veerud (transponeerib)
- `ftable(tunnus1 + tunnus2 ~ tunnus3 + tunnus4, data = andmed)` – teeb mitmemõõtmelise sagedustabeli, mis on inimesele lihtsasti loetav

Kui on soov korraga mitme arvulise tunnuse keskmisi arvutada, sobib selleks käsk `colMeans(tunnus)`, sarnane käsk on `rowMeans(.)`. Ridade või veergude summasid saab leida käskudega `rowSums(.)` ja `colSums(.)`. Seda saab näiteks ära kasutada sagedustabeli põhjal protsentide arvutamiseks:

```
sagedustabel <- table(andmed$SEX, andmed$LANX)
sagedustabel / rowSums(sagedustabel)  #proovi, mis juhtub, kui kasutada /colSums(.)
```

```
##
##          No, speaks only English Yes, speaks another language
##   Female               0.8031949                    0.1968051
##   Male                 0.8147019                    0.1852981
```

Sagedustabeli põhjal protsentide arvutamiseks (jaotustabeli arvutamiseks) on mugavam kasutada käsku ‘prop.table(.):

```
prop.table(sagedustabel)  # ühisjaotus
prop.table(sagedustabel, margin = 1)  # iga rida kokku 1 (ehk 100%)
prop.table(sagedustabel, margin = 2)  # iga veerg kokku 1 (ehk 100%)
```

2.5 Puuduvad väärtused

Käsu `summary(andmed)` väljatrükis oli näha (tunnuse MARHT – mitu korda abielus olnud – järgi), et 3026 inimest oli abielus olnud ühe korra, 575 inimest kaks korda ning 63 inimest kolm korda või rohkem. Lisaks oli 2760 inimese juurde märgitud `NA`. Nimelt on puuduv väärtus kodeeritud sümboliga `NA`. Üldiselt tasub puuduvate väärtuste olemasolul olla ettevaatlik. Sageli saab R-i käskudele argumenti `na.rm` kasutades öelda, mida puuduvate väärtusega ette võtta. Kui on soov teada saada, mitu väärtust on puudu, saab kasutada käskude kombinatsiooni `sum(is.na(tunnus))`:

```r
mean(andmed$WAGP)   # mõnel inimesel pole palganumbrit öeldud -- ei oska keskmist arvutada
```

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

```r
mean(andmed$WAGP, na.rm = TRUE)  # ignoreerime keskmise arvutamisel puuduvaid väärtuseid
```

```
## [1] 33162.34
```

```r
table(andmed$LANX)
```

```
##
##      No, speaks only English Yes, speaks another language
##                         4919                         1163
```

```r
table(andmed$LANX, useNA = "always")  # table käsuga on teistmoodi...
```

```
##
##      No, speaks only English Yes, speaks another language
##                         4919                         1163
##                         <NA>
##                          342
```

```r
sum(is.na(andmed$LANX))
```

```
## [1] 342
```

2.5.1 Ülesanded

1. Mitu protsenti on andmestikus mehed?
2. Milline on palk, millest väiksemat palka saab 80% inimestest?
3. Kas lahutatute osakaal on suurem meeste või naiste hulgas?

2.6 Vektorid. Tehted vektoritega

Eelpool oleme mitme veeru (või rea) korraga eraldamiseks kasutanud käsku `c(.)`. Tegemist on 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.

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

```
## [1] "987"                      "-Inf"
## [3] "siin on jutumärkides tekst"
```

Arvudest koosneva vektori tekitamiseks on ka muid mooduseid:

```r
1:5  # täisarvud 1-st 5-ni
```

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

```r
2:-6 # täisarvud 2-st -6-ni
```

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

```r
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
```

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

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

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

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

```r
rep(x=3, times=2)  # arvu 3 korratakse 2 korda
```

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

Käsku `rep` saab kasutada ka **sõnede** kordamisel: `rep("as_9", 4)`.

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.

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

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

```r
1:3 + 9:7   # kahe vektori elemendid liidetakse paarikaupa
```

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

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

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

```r
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
```

2.7 Tõeväärtused ja tõeväärtusvektorid

Kui käsuga `sum` 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 kasutamisel.

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

Sageli on tulemuseks tõeväärtus 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 eelpool mainitud, siis R-is tehakse tehteid vektoritega elementhaaval. Seepärast ka siis, kui võtame mingi arvulise vektori (näiteks palkade veeru) 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).

```
vaikepalk <-  andmed$WAGP < 1000
table(vaikepalk)
```

```
## vaikepalk
## FALSE  TRUE
##  3410  1903
```

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(.)`.

Mõnikord on soov kontrollida, kas mingi väärtus leidub etteantud hulgas. Sobilik on siis 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 või muid aritmeetilisi tehteid teha, siis konverteeritakse tõeväärtused arvudeks: TRUE muutub arvuks 1 ja FALSE muutub arvuks 0.

```
sum(vaikepalk, na.rm = T)
```

```
## [1] 1903
```

```
sum(is.na(vaikepalk))  # mitmel inimesel on palk puudu
```

```
## [1] 1111
```

```
sum(vaikepalk == NA)   # see ei tööta nii, nagu sooviks...
```

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

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

2.7.1 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. Mitu inimest töötab nädalas üle 40 tunni (tunnus `WKHP`)? Mitmel inimesel on töötundide arv puudu?

2.8 Ridade eraldamine

Tõeväärtusvektoreid on väga mugav kasutada nn filtritena. Nimelt on `data.frame` puhul võimalik ridu (ja ka veerge) eraldada mitte ainult reaindeksi või -nime järgi, vaid ka tõeväärtusvektori abil. Vastav tõeväärtusvektor peab olema sama pikk kui on andmestikus ridu, iga väärtus selles vektoris näitab, kas vastavat rida kasutada või mitte. Tõeväärtusvektorite kombineerimisel saab andmestikust väga spetsiifilisi alamhulki eraldada.

```
mehed <- andmed[andmed$SEX == "Male", ]  # eraldame kõik read, kus SEX == "Male" ning
#salvestame selle uueks objektiks
filter_kod <- andmed$CIT == "Not a citizen of the U.S."
filter_vanus <- andmed$AGEP >= 80
alamandmestik <- andmed[filter_kod & filter_vanus, ]   # Ära unusta: [read, veerud]
```

2.8.1 Ülesanded

1. Mitu üle 74 aasta vanust doktorikraadiga naist on Massachusettsi andmestikus?
2. Mitmel inimesel on bakalaureuse-, magistri- või doktorikraad?
3. Milline on keskmine aastapalk meestel, milline naistel?
4. Kui suur osa (protsentides) ilma kõrghariduseta inimestest saab suuremat aastapalka kui keskmine palk?
5. Selekteeri andmestikust iga 5. rida ja salvesta see alamandmestik uue nimega. Kas selles alamandmestikus on meeste ja naiste keskmised aastapalgad samasugused kui kogu andmestikus?

2.9 Väärtuste tüübid. Faktortunnus

Nägime eespool `str(.)` käsku kasutades, et andmestiku veerud on erinevad tüüpi. Massachusettsi andmestikus oli erinevaid tüüpe kaks: `int` ja `Factor`. R-is on veel teisigi väärtuste tüüpe, mõned sagedamini esinevad on:

- `int` / `integer` – täisarvud (ka negatiivsed)
- `numeric` – reaalarvud
- `char` / `character` – sõned (tähemärgid ja muud tekstilised sümbolid)
- `logical` – tõeväärtused (ainult kaks väärtust: TRUE või FALSE)
- `Factor` – faktortunnus

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

Faktortunnus pole tegelikult nn elementaartüüp (nagu näiteks `integer`), vaid keerulisem konstruktsioon. Nimelt faktortunnus on sildistatud täisarvude tunnus. Kui andmestik R-i sisse loetakse, siis vaikimisi pannakse tähelisi väärtuseid sisaldavate tunnuste tüübiks `factor` (seda võib ka `read.table(.)` argumendiga `stringsAsFactors` keelata) ning iga erinev väärtus kodeeritakse mingi täisarvuga, aga lisaks tehakse kodeerimistabel, kus on kirjas iga täisarvu (kodeeringu) tekstiline väärtus (ehk silt).

Et teada saada, mitu erinevat väärtust antud faktortunnusel võib üldse olla, kasutatakse käsku `levels(.)`. Sealjuures ei pruugi kõigi tasemega väärtuseid andmestikus üldse olla:

```
levels(mehed$SEX) # meeste andmestikust võtame soo veeru
```

```
## [1] "Female" "Male"
```

Faktortunnuse tekitamiseks saab kasutada käsku `factor(.)`. Vaikimisi pannakse faktortunnuse tekitamisel faktori tasemed tähestiku järjekorda. Seda saab aga muuta käsu `factor(.)` argumenti `levels` kasutades:

```
table(andmed$MARHT) # Mitu korda abielus olnud?
andmed$MARHT <- factor(andmed$MARHT, levels = c("One time", "Two times", "Three or more times"))
table(andmed$MARHT)
```

Mõnikord on arvulised tunnused faktorkujul sellepärast, et ühes lahtris oli mingi tekst. Kui nüüd proovida `as.numeric(.)` käsuga see tunnus arvuliseks teisendada, tekib segadus: `Factor`-tüüpi tunnus on juba täisarvude tunnus (kuigi neil on sildid juures) ning seetõttu antakse tulemuseks need täisarvud ehk kodeeringud. Segadust ei teki, kui faktortunnus kõigepealt sõneks teisendada (kodeeringud kaotatakse, jäävad ainult sildid) ning alles seejärel arvudeks.

```
(x <- factor(c("1", "8", "ei vastanud", "12"))) # välimised sulud tingivad väljatrüki
```

```
## [1] 1           8           ei vastanud 12
## Levels: 1 12 8 ei vastanud
```

```
(as.numeric(x))
```

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

```
(as.numeric(as.character(x)))
```

```
## Warning: NAs introduced by coercion
```

```
## [1]  1  8 NA 12
```

Mõnikord soovime arvulist tunnust muuta nn ordinaaltunnuseks (st selliseks, kus on mõned üksikud kategooriad, mis on omavahel järjestatud). Näiteks palkade statistika esitamisel on soov teada infot palgavahemike kaupa. Arvulist tunnust aitab lõikudeks tükeldada käsk `cut(.)`. Selle käsu tulemusel tekib faktortunnus, mille silte saab `cut(.)` käsu argumendiga `labels` ette anda:

```
palgad = cut(x=andmed$WAGP, breaks=c(0, 999, 4999, Inf), include.lowest=T,
             labels=c("0--999", "1000--4999", ">= 5000"))
```

2.9.1 Ülesanded

1. Tekita tunnus, kus oleks kirjas, millisesse vanusgruppi inimene kuulub: 0–17, 18–49, 50–64, 65 või vanem.
2. Kas USA kodakonsust mitteomavate naiste ja meeste hulgas on vanus erinevalt jaotunud?

3 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 paketis `foreign` on olemas käsud `read.spss(.)`, `read.dta(.)` jm, mis aitavad teistes formaatides andmestikke hõlpsamini R-i sisse lugeda. Paketis `car` on aga käsk `recode(.)` millega saab faktortunnuse tasemeid mugavalt ümber kodeerida (ja ka kokku grupeerida).

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.

```
install.packages("ggplot2")
```

Kui pakett on paigaldatud, siis seda enam uuesti samasse arvutisse paigaldama ei pea. Ent iga kord uut R-i sessiooni alustades tuleb vajaminevad paketid sisse laadida käsuga `library(paketinimi)` või `require(paketinimi)`:

```
library(ggplot2)
```

3.0.1 Ülesanded

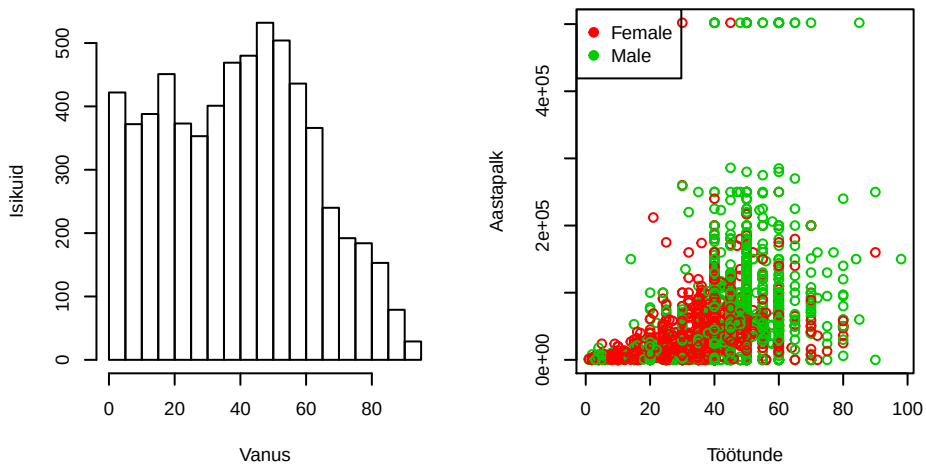
1. Paigalda oma arvutisse pakett ggplot2 (koos nende pakettidega, mis on ggplot2 jaoks vajalikud).

4 Joonised paketiga ggplot2

4.1 Graafika R-is. ggplot2 ja graafikute grammatika

R-i üks tugevaid külgi on tema jooniste tegemise võimekus. Paar kõige rohkem kasutatavat joonistamise käsku:

```
par(mfrow = c(1, 2), cex = 0.6)  # see rida võimaldab kaks joonist kõrvuti panna
hist(andmed$AGE, xlab="Vanus", ylab="Isikuid", main="")
plot(andmed$WKHP, andmed$WAGP, xlab="Töötunde", ylab="Aastapalk", col=as.numeric(andmed$SEX)+1)
legend("topleft", pch=19, col=2:3, legend=levels(andmed$SEX))
```



Kuigi R-i baasgraafika on peensusteni konfigureeritav ja sellega saab teha ülikeerulisi jooniseid, peab tüüpilisemate andmeid kirjeldavate jooniste saamiseks tegema palju lisatööd ja -arvutusi, nagu ülalolevastki näha. Näiteks `barplot(.)` käsk soovib argumendiks saada sagedustabelit, mis tuleks siis eelnevalt `table(.)` käsu abil arvutada.

R-i kasutajate hulgas on viimastel aastatel muutunud populaarseks pakett `ggplot2`, mis võimaldab lihtsamini joonistada andmestikke kirjeldavaid jooniseid, sealjuures on tulemused visuaalselt üsna apetiitsed. Nimelt on `ggplot2` arendamisel pandud tähele statistiku E. Tufte soovitusi värvide valikul ning eriti L. Wilkinsoni struktureeritud käsitlust andmejoonistest, nn graafikute grammatikat (*grammar of graphics*).

Üldised soovitused (Tufte):

- vähema tindiga edastada rohkem infot;
- ei tohiks rõhutada mõnda elementi, kui see pole teistest oluline (nt kõik värvid võiksid olla sama intensiivsusega);

- eelistada silmaga lihtsamini hinnatavaid kujundeid (nt tulba kõrgus tulpdiagrammil on lihtsamini hinnatav kui nurgakraad või sektori pindala ringdiagrammil);
- eemaldada infot mitte sisaldavad komponendid (nt liba-3D).

Graafikute grammatika (Wilkinson) on kontseptsioon, mille kohaselt graafiku ehitamisel ei tuleks lähtuda mitte graafiku tüübist, vaid andmetest. Iga joonis koosneb järgnevatest osadest:

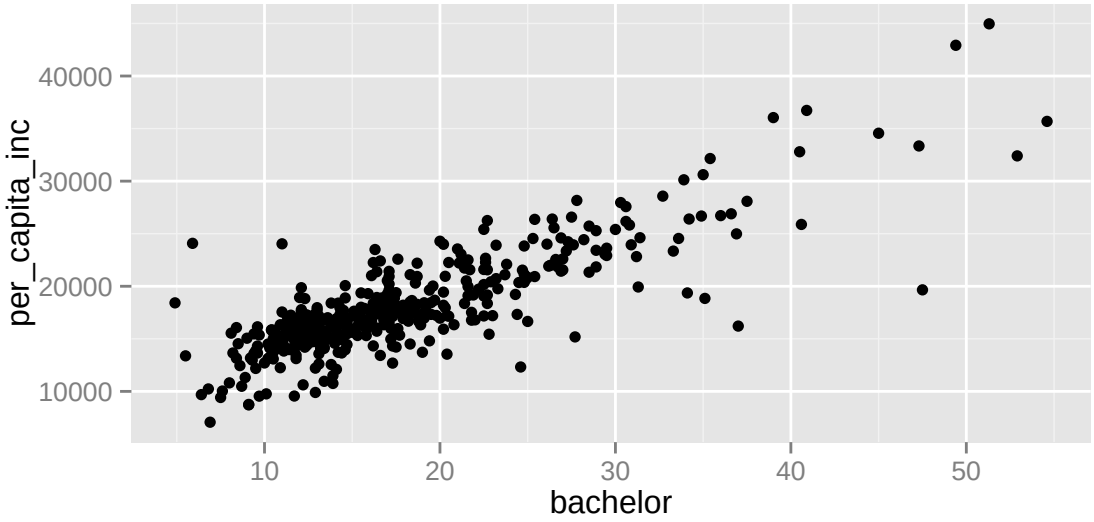
- andmed (inimeste palk ja sugu);
- skaalad (kas esitada palk värviskaalal või x-teljel? kas palga esitamiseks sobib nt log-skaala?);
- statistikud (kas palga puhul kujutada keskmist või summat);
- geomeetrilised kujundid (kas keskmine peaks olema märgitud tulba kõrgusega või hoopis punktikesega);
- koordinaadid (äkki sobib polaarkoordinaadistikus?);
- tahud (joonis jagatud erinevateks alamjoonisteks);
- üldkujundus (font jms).

Paketis `ggplot2` on kaks graafikute tekitamise käsku: `ggplot(.)` (keerukam) ja `qplot(.)` (*quick plot*, lihtsam). Meie kasutame praegu ainult `qplot(.)` käsku. Selle paketi käskude jaoks on kõige põhjalikum dokumentatsioon internetis: http://docs.ggplot2.org/current/ . Kuna tegemist on endiselt väga noore paketiga, siis uuemate versioonide käskude süntaks või toimimine võib olla erinev vanemate versioonide omast (viimati oli oluline muutus opts(.) käsu asendamine theme(.) käsuga). Ka dokumentatsioon pole veel täielik (aga oluliselt parem, kui paar aastat tagasi).

4.2 ggplot2: hajuvusdiagramm; skaalad

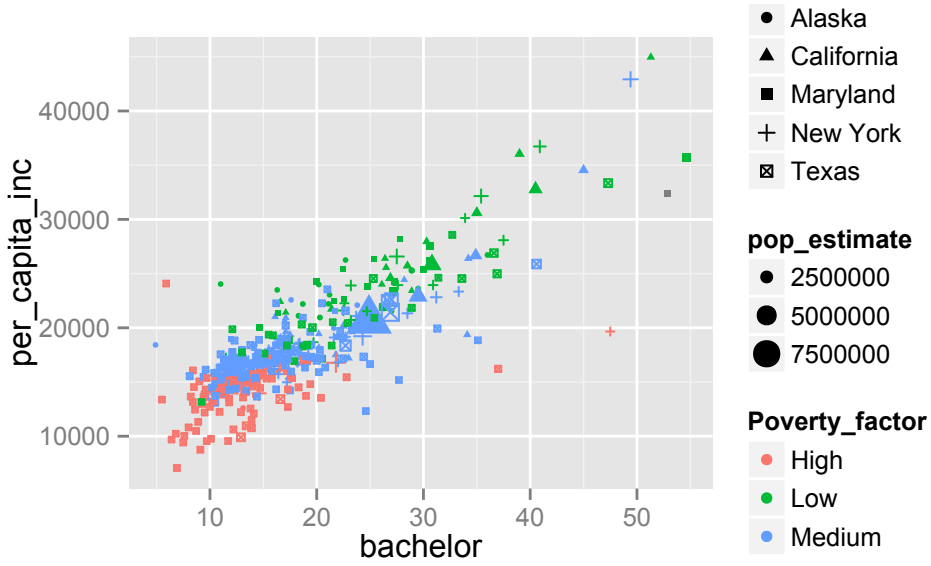
Failis `http://kodu.ut.ee/~maitraag/rtr/maakonnad.txt` on info USA viie osariigi mõnede maakondade (regioonide) kohta (425 maakonda). Uurime kõrgharidusega inimeste osakaalu ja keskmise sissetuleku vahelist seost. Mõistlik on seda kujutada hajuvusdiagrammina, kus iga punkt on maakond ning ühel teljel on kõrgharidusega inimeste osakaalu märkiv tunnus `bachelor` ja teisel teljel keskmine sissetuleku tunnus `per_capita_inc`:

```
mk <- read.table("http://kodu.ut.ee/~maitraag/rtr/maakonnad.txt", sep = " ")
qplot(x = bachelor, y = per_capita_inc, data = mk)
```



Lisaks koordinaatidele saab üks punkt veel edasi anda infot näiteks värvi, kuju ja suurusega:

```
qplot(x = bachelor, y = per_capita_inc, data = mk, colour = Poverty_factor,
      size = pop_estimate, shape = State)
```



Igat tüüpi tunnuseid ei saa suvaliste jooniseühikutega seostada, näiteks pideva tunnusega ei saa siduda punkti kuju (nii palju erineva kujuga punkte pole lihtsalt olemas). Samas aga saab värviga kujutada nii faktortunnust (nt osariik) kui ka arvulist tunnust (nt kõrgus merepinnast). Täpsemini on hajuvusdiagrammi ühel punktil järgmised omadused, millega saab infot edasi anda:

- `x` – (kohustuslik) asukoht x-teljel [numeric, character, logical, Factor]
- `y` – (kohustuslik) asukoht y-teljel [numeric, character, logical, Factor]
- `alpha` – läbipaistvus, väiksem väärtus tähendab suuremat läbipaistvust [numeric, character, logical, Factor]
- `colour` – värvus [numeric, character, logical, Factor]
- `fill` – sisemuse värvus (ainult mõne `shape` väärtuse korral) [numeric, character, logical, Factor]
- `shape` – punkti kuju (kuni 25 erinevat + ise määratavad sümbolid) [logical, character, Factor]
- `size` – punkti suurus [numeric, character, logical, Factor]

4.2.1 Ülesanded

1. Loe sisse maakondade andmestik: `mk <- read.table("http://kodu.ut.ee/~maitraag/rtr/maakonnad.txt", sep = " ")` .
2. Joonista hajuvusdiagramm keskkooli lõpetanute protsendi (`high_scl`) ja ülikooli lõpetanute protsendi (`bachelor`) vahel. Kas on näha mingit seost?
3. Lisa joonisele osariigi (`State`) kohta käiv info; katseta erinevaid variante (värv, kuju jne).
4. Kujuta joonisel mingil moel ka maakonna rahvaarvu (`pop_estimate`).

4.3 ggplot2: tulpdiagramm; elementide asukoht

Graafiku teljed võivad olla seotud ka diskreetse tunnusega (nt Factor), näiteks võiksime maakondi kujutades siduda x-teljega osariigi:

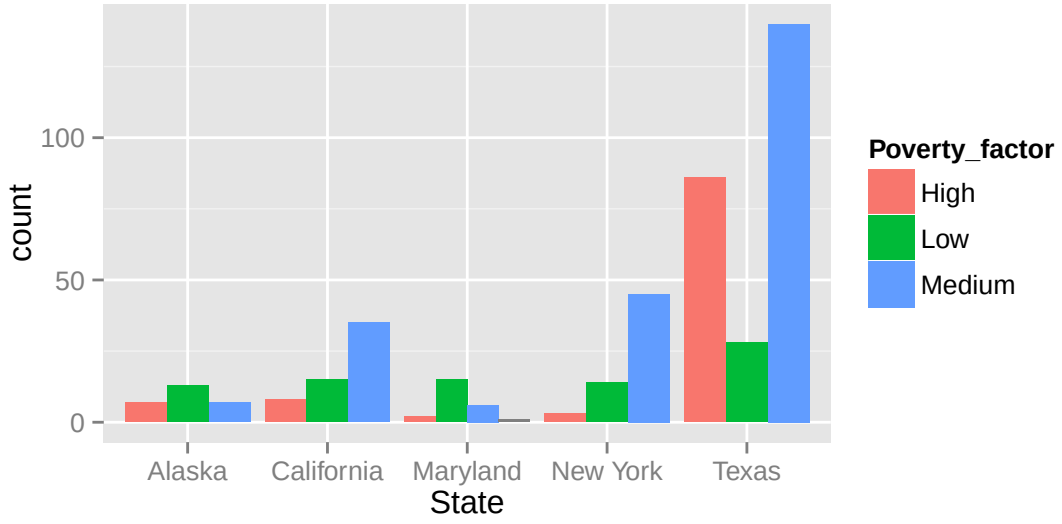
```
qplot(x = State, data = mk)
```

Tekkis tulpdiagramm, kus iga tulp näitab maakondade arvu vastavas osariigis. Ent maakonnad ühe tulba sees võivad olla erinevad, näiteks vaesustaseme (`Birth_factor`) poolest. Selleks võib iga tulba vastavate maakondade arvu järgi ära värvida. Kuna iga joonise element koosneb piirjoonest ning sisemisest osast, tuleb vastava osa värvi muutmiseks kasutada kas argumenti `colour` või `fill`:

```
qplot(x = State, data = mk, fill = Poverty_factor, colour = State)  # praegu on colour
# argument ebavajalik,  sama info on juba x-teljel olemas
```

Samas võib iga tulba ka tükkideks jaotada ja panna üksteise kõrvale, andes argumendile `position` väärtuse `position="dodge"`. (NB! Jutumärgid)

```
qplot(x = State, data = mk, fill = Poverty_factor, position = "dodge")
```



Kuna suuremates osariikides on rohkem maakondi, oleks võib-olla informatiivsem näha suhtelisi sagedusi. See tähendab, et iga osariigi tulp võiks olla sama kõrge (100%):

```
qplot(x = State, data = mk, fill = Poverty_factor, position = "fill")
```

Tasub tähele panna jutumärke argumendi `position` väärtuste ümber. Nimelt selle argumendi väärtust ei saa siduda mõne tunnusega andmestikus (poleks eriti mõttekas, et näiteks ühe osariigi erineva vaesustaseme tulbad paiknevad kõrvuti, teisel aga üksteise otsas). Üldse on argumendil `position` neli võimalikku väärtust:

- `"dodge"` – paneb elemendid üksteise kõrvale
- `"fill"` – skaleerib elemendid sama suureks (kasutatakse protsendiskaalat)
- `"identity"` – (vaikimisi) mitte midagi ei tehta
- `"stack"` – paneb elemendid üksteise otsa (kumulatiivsed summad)
- `"jitter` – nihutab elemente juhuslikult üles-alla, paremale-vasakule

Kui kujutaksime maakondi hajuvusdiagrammil, kus x-teljega seome osariigi ning y-teljega keskmise sissetuleku, siis sellisel joonisel oleksid paljud maakondi kujutavad punktid üksteise peal. Andes `qplot(.)` käsu argumendile `position` väärtuse `position = "jitter"`, saame punkte natukene hajutada, nii et jooniselt on näha sissetulekute jaotus igas maakonnas:

```
qplot(x = State, y = per_capita_inc, data = mk, position = "jitter")
```

Võib-olla on "jitter" punktikesi natuke liiga palju laiali hajutanud. Andes argumendile `position` väärtuseks hoopis funktsiooni `position_jitter(width=, height=)`, saame kontrollida, kui palju hajutamist horisontaal- ja vertikaalsuunal tehakse.

```
qplot(x = State, y = per_capita_inc, data = mk, position = position_jitter(width = 0.2))
```

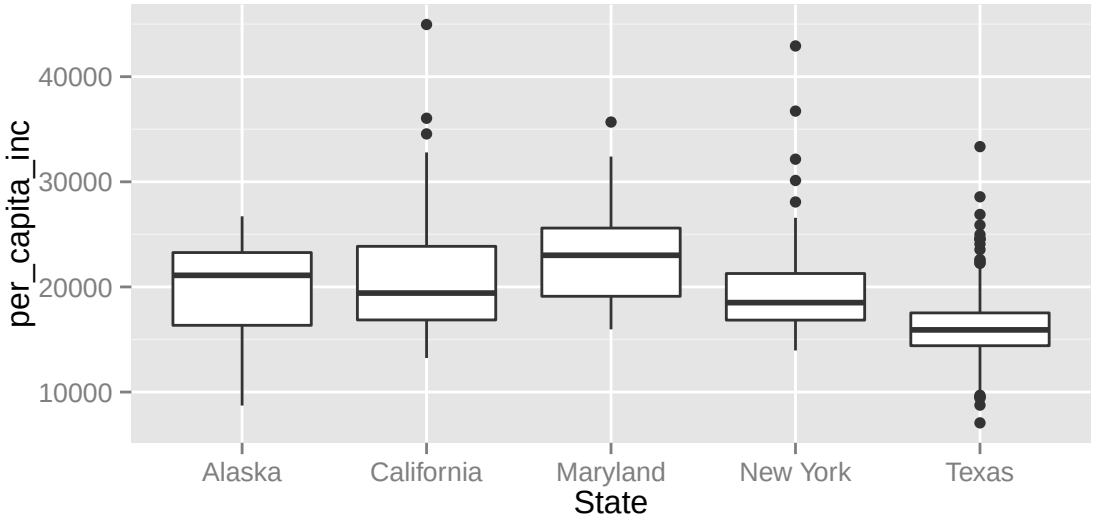
4.3.1 Ülesanded

1. Tee joonis iseloomustamaks sünnitamistaseme (`Birth_factor`) ja vaesuse taseme (`Poverty_factor`) vahelist seost.
2. Tee joonis, mis kirjeldaks sissetulekut (`per_capita_inc`) erinevates osariikides.
3. Tee joonis, mis kirjeldaks sissetulekut (`per_capita_inc`) erinevates osariikides, nii et sissetulek oleks x-teljel.

4.4 Jooniste tüübid

Enamjaolt joonistab `qplot(.)` käsk just sellise joonise, nagu on soovitud – tunnuste tüübi põhjal saab see käsk aru, kas sooviti histogrammi või tulpdiagrammi. Ent mõnikord võib olla soov teistsugust joonist näha. Näiteks kui x-teljele panna diskreetne tunnus ja y-teljele pidev, siis vaikimisi joonistatakse hajuvusdiagrammilaadne joonis (iga objekt on kujutatud punktiga). Aga mõttekam on sellises olukorras hoopis kasutada karpdiagramme. Käsu `qplot` puhul saab argumendiga `geom` seda määrata:

```
qplot(State, per_capita_inc, data = mk, geom = "boxplot")
```



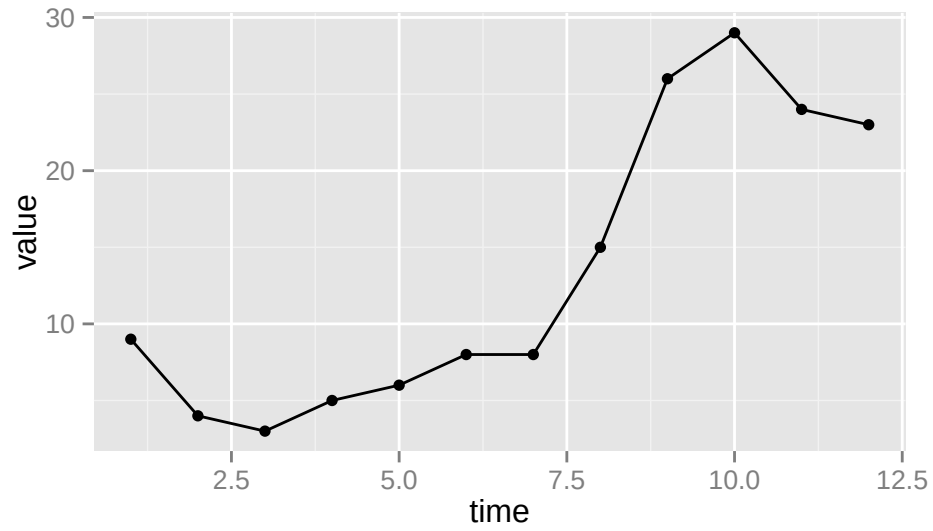
Meeldetuletus joonise tüüpide valikust:

Tunnus1	Tunnus2	Sobivad joonised	`geom = ...`
pidev/arvuline	-	histogramm (tulbad)	`"bar"`
kategooriline	-	tulpdiagramm (tulbad)	`"bar"`

Tunnus1	Tunnus2	Sobivad joonised	`geom = ...`
pidev/arvuline	pidev/arvuline	hajuvusdiagramm (punktid)	`"point"`
pidev/arvuline	kategooriline	karpdiagramm (karbid)	`"boxplot"`
pidev/arvuline	aeg vms järgnevus	joondiagramm (jooned)	`"line"`

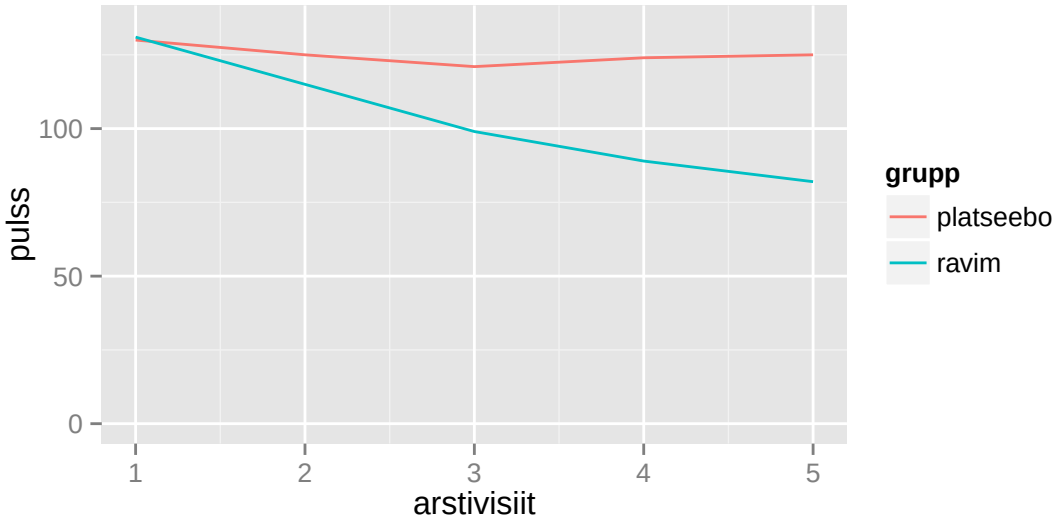
Erinevaid elemente saab joonsel korraga kujutada (nt aegridade puhul):

```
time = 1:12
value = c(9, 4, 3, 5, 6, 8, 8, 15, 26, 29, 24, 23)
qplot(time, value, geom = c("point", "line"))
```



Joondiagrammi puhul saab argumendiga `group` öelda, milline tunnus määrab, millise joonega antud info kokku läheb.

```
# data.frame käsk aitab ise käsitsi andmestikku tekitada
d <- data.frame(grupp = rep(c("platseebo", "ravim"), each=5),
                pulss = c(130, 125, 121, 124, 125, 131, 115, 99, 89, 82),
                arstivisiit = rep(1:5, 2))
qplot(arstivisiit, pulss, data = d, geom = "line", group = grupp, colour = grupp, ylim = c(0, 135))
```



Mõnikord harvem muidugi võib olla soov esitada pideva ja kategoorilise tunnuse vahelise seose kirjeldamiseks mitte karpdiagramme, aga näiteks keskmisi koos usaldusvahemikega. Sellistest keerukamatest võimalustest tuleb hiljem juttu.

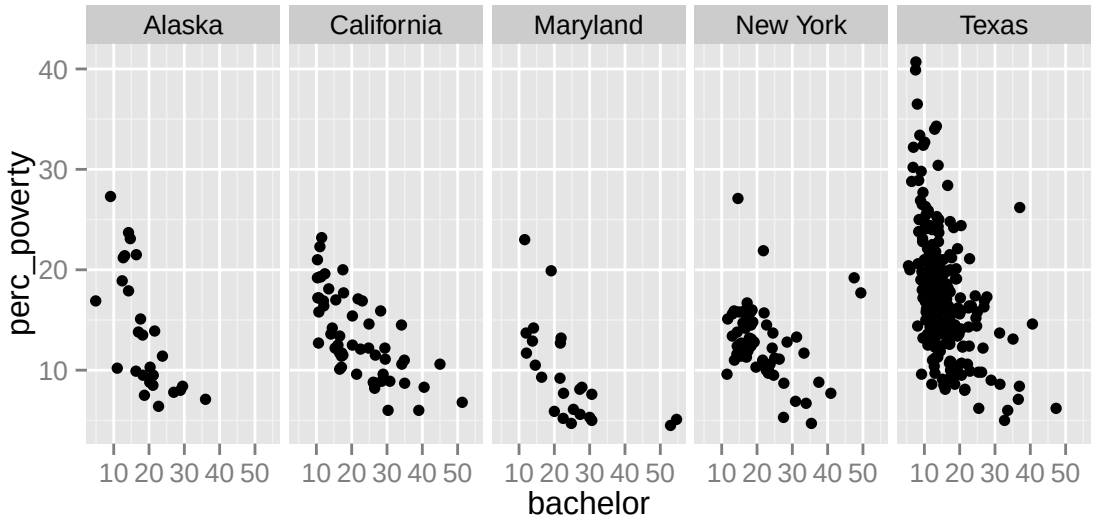
4.4.1 Ülesanded

1. Joonista tunnuse `bachelor` histogramm.
2. Lisa eelmisele joonisele värvi kasutades ka tunnus `State`.
3. Uuri tunnuse `high_scl` jaotust erinevates osariikides, kasutades karpdiagrammi.

4.5 Joonise jagamine tahkudeks

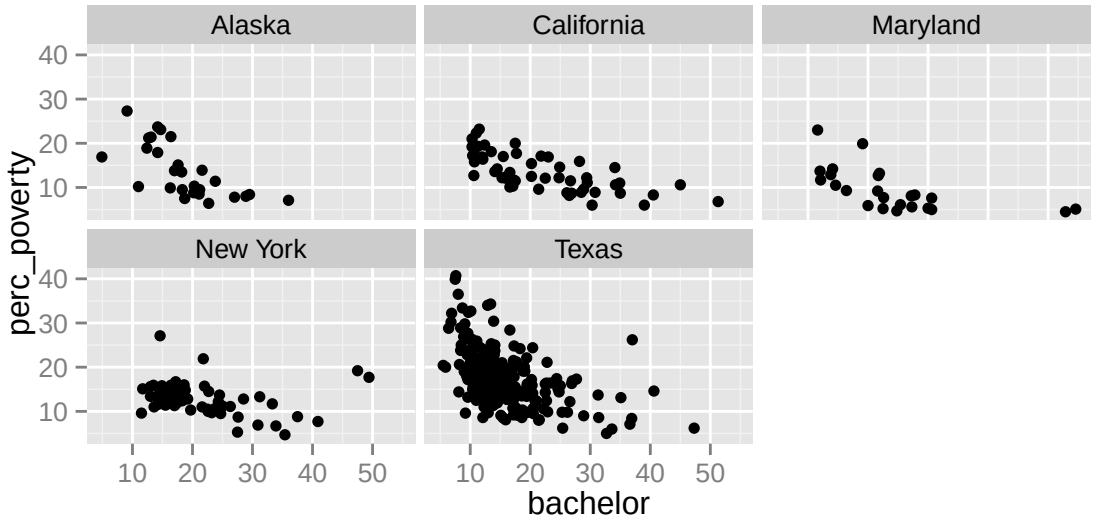
Sageli on mõistlik ühe suure ja kirju pildi asemel joonistada palju väikeseid sarnase sisuga pilte. Selmet erinevate osariikide maakondade vaesus ja haridustaseme seost ühel ja samal pildil kujutada, võiks seda iga osariigi jaoks teha eraldi. Käsus `qplot(.)` on selleks argument `facets`. Sellele argumendile tuleb väärtus anda nn **valemi kujul**: `ridadeks_jagav_muutuja ~ veergudeks_jagav_muutuja`. Kui üks neist ära jätta, siis selle asemele tuleks kirjutada punkt.

```
qplot(bachelor, perc_poverty, data = mk, facets = . ~ State)
```



Kui ridadeks jagava muutuja asemele punkti mitte panna, siis laotakse joonised kõrvuti ritta kuni rida saab täis; ülejäänud joonised pannakse järgmistesse ridadesse:

```
qplot(bachelor, perc_poverty, data = mk, facets = ~ State)
```



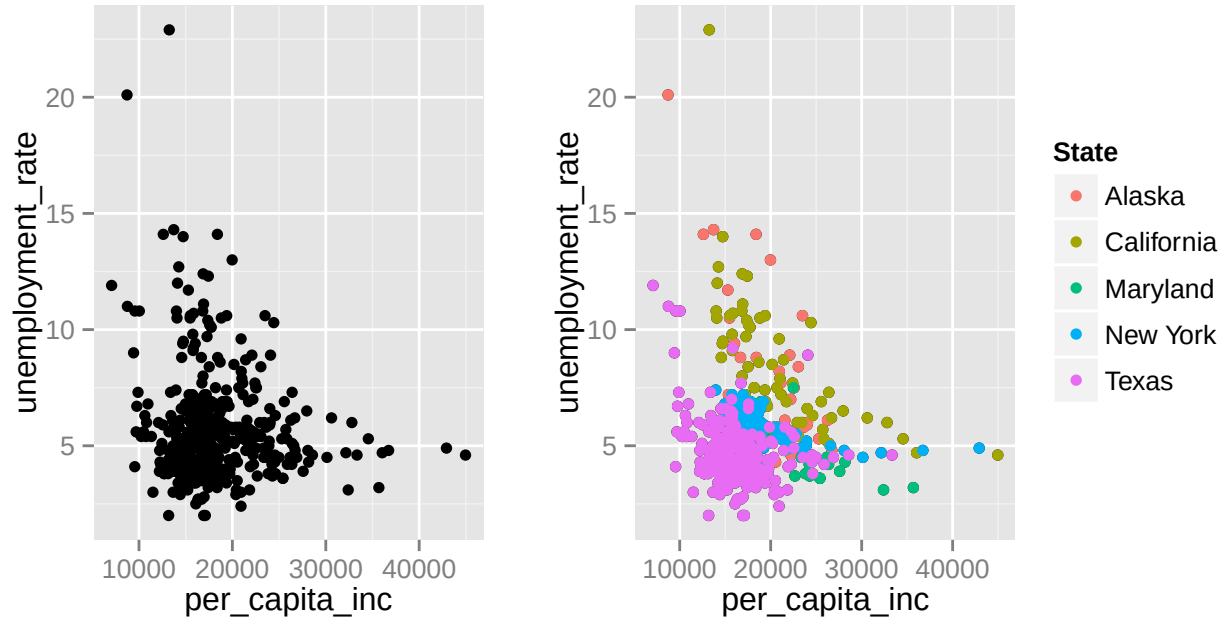
4.5.1 Ülesanded

1. Joonista hajuvusdiagramm keskkooli ja ülikooli lõpetanute protsendi jaoks (tunnused `high_scl` ja `bachelor`) erinevate `Birth_factor` tasemete kaupa.
2. Järjesta `Birth_factor` väärtused `factor(.)` käsu abil mõistlikumas järjekorras ja tee see joonis uuesti.
3. Lisa eelmisele joonisele ka jaotus osariikide kaupa. Kas erinevates osariikides on seosed on samad või erinevad?

4.6 Joonisele kihtide lisamine

Paketiga `ggplot2` koostatud jooniseid saab salvestada objektina; valmisolevaid jooniseid saab seetõttu lihtsasti muuta ja täiendada. Erinevalt baasgraafikast, kus joonise koostamiseks vajalik info tuleb anda ühe käsu piires või mitme täiesti eraldi käsuga, kasutatakse ggplot2 puhul käskude liitmist sümboliga `+` (see ilmestab paremini seda, kuidas joonis elementhaaval üles on ehitatud). Geomeetriliste elementide (punktid, jooned, tulbad jne) lisamiseks/muutmiseks on käsud `geom_<elemendi_nimi>`.

```
p <- qplot(x=per_capita_inc, y=unemployment_rate, data=mk) # tekitatakse joonise objekt, ei kuvata
p # kuvatakse joonis
p + geom_point(mapping = aes(colour = State)) # lisame punktidele värvi
```



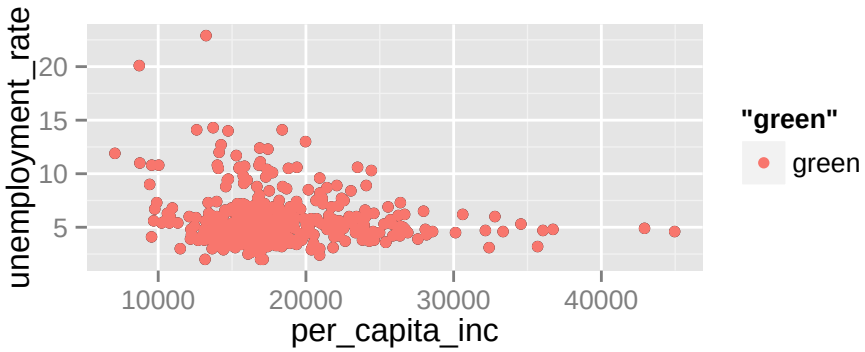
Tasub tähele panna käsku `aes(.)` (nagu *aesthetics*), mis ülaloleval joonisel punktide värvimiseks `geom_point(.)` funktsiooni argumendile `mapping` ette anti. Nimelt `aes(.)` funktsioon aitab siduda graafilisi elemente andmestikus olevate tunnustega. Kui `aes(.)` funktsiooni ei kasutaks, otsitaks vastavate argumentide väärtuseid mitte andmestikust, vaid töökeskkonnast:

```
p + geom_point(colour = State) # objekti State otsitakse töökeskkonnast ja ei leita
```

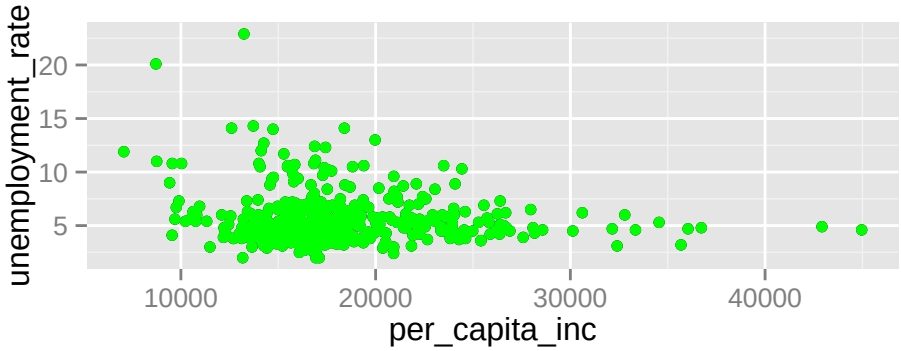
```
## Error in do.call("layer", list(mapping = mapping, data = data, stat = stat, : object 'State' not fou
```

Samas aga käsitsi mingi värvi etteandmiseks ei peaks `aes(.)` funktsiooni kasutama (sest muidu otsitakse seda värvi andmestikust).

```
p + geom_point(aes(colour = "green"))
```

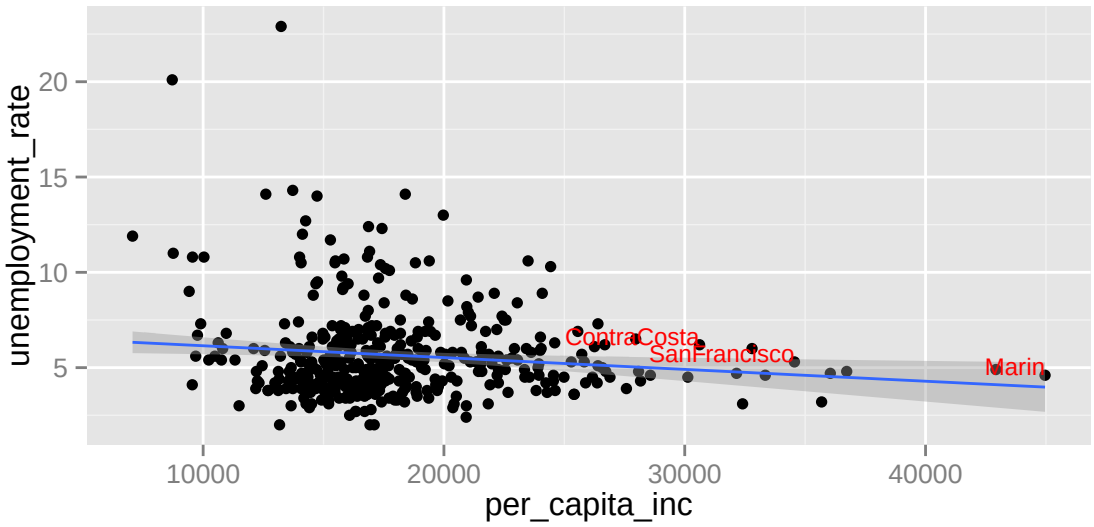



```
p + geom_point(colour = "green")
```



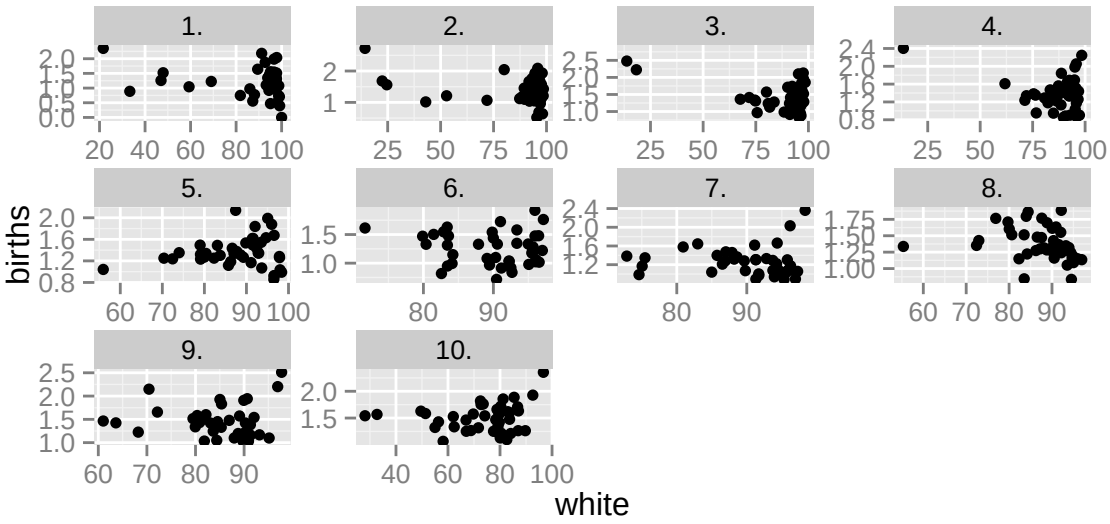
Joonisele saab lisada ka uusi elemente, näiteks regressioonikõveraid või teksti. Uute elementide lisamisel võib ette anda uue andmestiku argumendiga `data`:

```
p + geom_smooth(method = lm) + # lisatakse siluja (praegu lineaarne regressioon)
    geom_text(mapping = aes(label = County), size = 3, data = mk[c(34, 48, 65), ],
              colour = "red", hjust = 1, vjust = 0)
```



Kordusmõõtmiste puhul joonistatakse sageli iga mõõtmisobjekti mõõtmistulemused eraldi tahkudele (*facets*). Et täita tahkudega mitu rida, saab kasutada käsku `+ facet_wrap(.)` või anda `qplot(.)` käsu argumendile `facets` väärtuseks `facets = ~ pop_deciles`. Kui miskipärast on soov tahkudel kasutada erinevaid skaalasid, siis seda saab `facet_wrap(.)` argumendiga `scale` määrata:

```
mk$pop_deciles <- cut(mk$pop_estimate, quantile(mk$pop_estimate, seq(0, 1, 0.1)),
                      labels = c("1.", "2.", "3.", "4.", "5.", "6.", "7.", "8.", "9.", "10."),
                      include.lowest = TRUE)
qplot(white, births, data = mk) + facet_wrap(facets = ~ pop_deciles, scale = "free")
```



4.6.1 Ülesanded

1. Loe sisse andmestik: `read.table("http://kodu.ut.ee/~maitraag/rtr/mk.txt", sep = " ")`
2. Joonista hajuvusdiagramm `high_scl` ja `bachelor` vahel.
3. Lisa neile maakondadele nimed, kus keskkooliharidusega inimeste osakaal on $< 50\%$.
4. Lisa graafikule vertikaaljoon kohale `x = 50` (vihje: kasuta käsku `geom_vline(.)`)

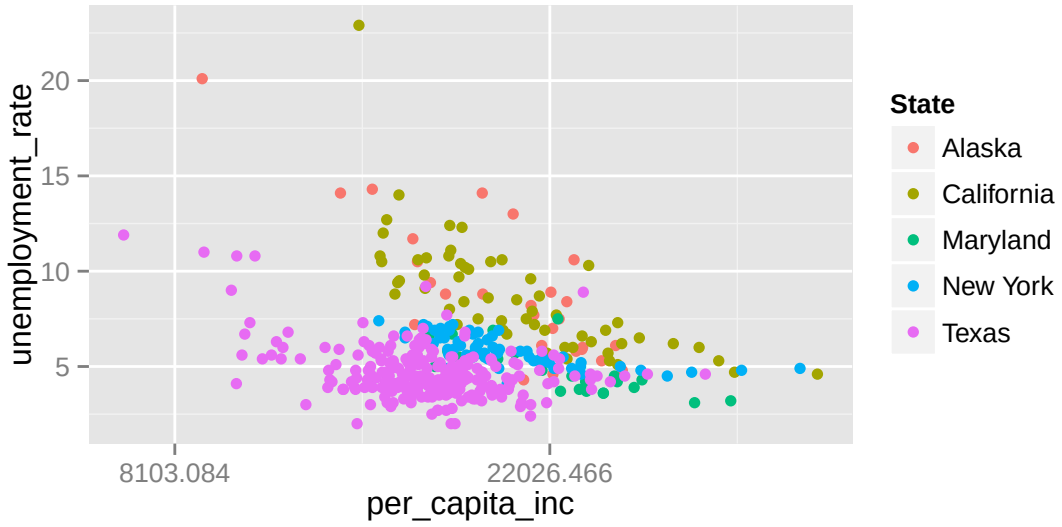
4.7 Skaalade muutmine

Kui `ggplot2`-ga koostada joonis, siis muutujad seostatakse graafiliste elementide omadustega (asukoht x-teljel, asukoht y-teljel, värvus, suurus jne). Vaikimisi kasutatakse küllalt mõistlikke skaalasid, nt värviskaalad on valitud selliselt, et kõik värvid oleks võrdse intensiivsusega. Mõnikord on aga soov skaalasid muuta. Selleks saab kasutada käske `scale_<graafilise elemendi nimi>_<skaala nimi>`. Näiteks `scale_x_continuous(.)` käsuga saab muuta x-telje vahemikku, `scale_colour_grey(.)` muudab värviskaala mustvalgeks.

Kõiki skaalade muutmise funktsioone on võimalik näha ggplot2 kodulehel http://docs.ggplot2.org/current/index.html jaotuse "Scales" all.

Skaleerimisfunktsiooni rakendamiseks tuleb see lisada joonisele:

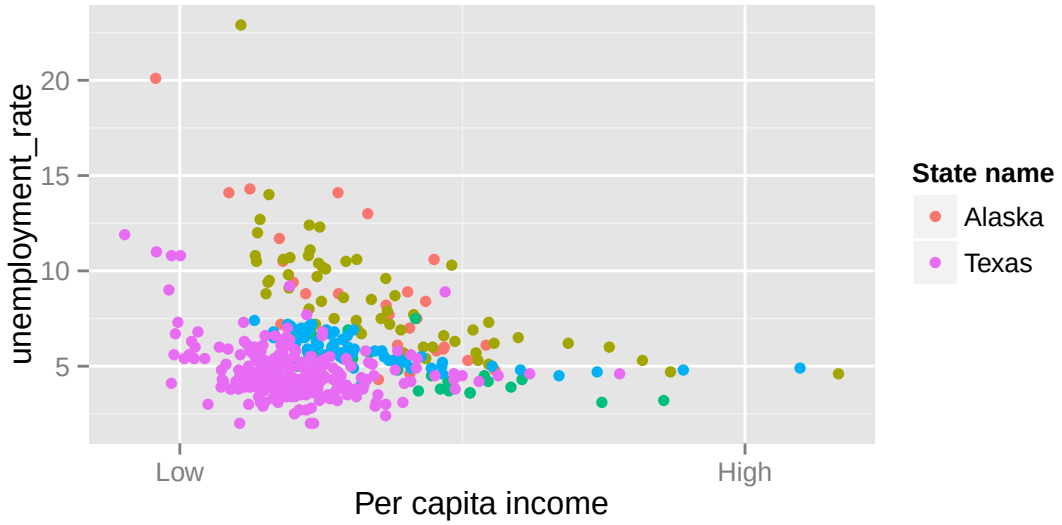
```
p2 <- qplot(per_capita_inc, unemployment_rate, colour = State, data = mk)
p2 + scale_x_continuous(trans = "log")
```



Iga erineva omaduse skaala muutmiseks saab joonisele "juurde liita" uue funktsiooni. Konkreetse funktsiooni parameetrid sõltuvad omaduse tüübist (nt punkti kuju ei saa logaritmida), aga on kõigil skaleerimisfunktsioonidel on kindlasti kolm argumenti:

- `name` – telgede puhul telje nimi; värvide, kuju jm legendis antava info puhul vastava legendi pealkiri
- `breaks` – vektor punktidega, mis määravad, millised väärtused joonisel ära markeeritakse (nt x-telje jaotis)
- `labels` – parameetriga `breaks` määratud punktidele vastavad sildid

```
p2  + scale_x_continuous(name = "Per capita income", breaks = c(10000, 40000),
                         labels = c("Low", "High")) +
      scale_colour_hue(name = "State name", breaks = c("Alaska", "Texas"))
```



Skaalasid (mh nende nimesid) saab muuta vastava `scale`-käsuga, ent mõned `qplot(.)` käsu argumendid muudavad selle natukene lihtsamaks:

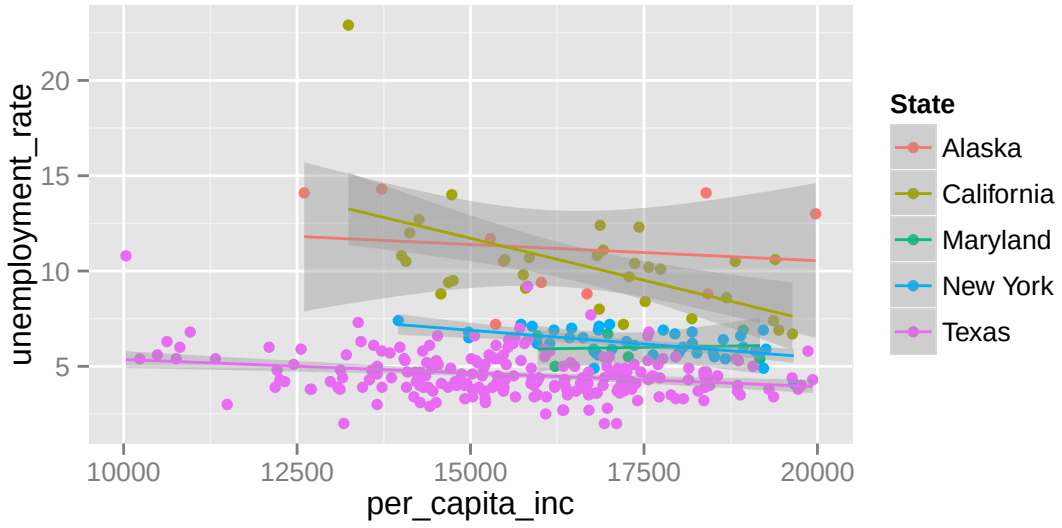
- `main` – joonise pealkiri
- `xlab`, `ylab` – telgede pealkirjad
- `xlim`, `ylim` – kaheelemendilised vektorid, mis määravad telgede väärtusvahemiku.

4.7.1 Pidevate skaalade muutmine

Pidevate skaalade muutmise käskudel (nt `scale_<?>_continuous`, `scale_<?>_gradient`) on mõned spetsifilised argumendid:

- `trans` – skaala transformeerimise funktsiooni nimi, nt "exp", "log", "sqrt".
- `limits` – kahe-elemendiline vektor, mis annab skaala algus- ja lõpp-punkti. Selle argumendi puhul tuleb olla tähelepanelik, sest andmeid, mis vastavast vahemikust välja jäävad, ei kasutata joonise tegemisel (nt ka regressioonisirge arvutamisel).

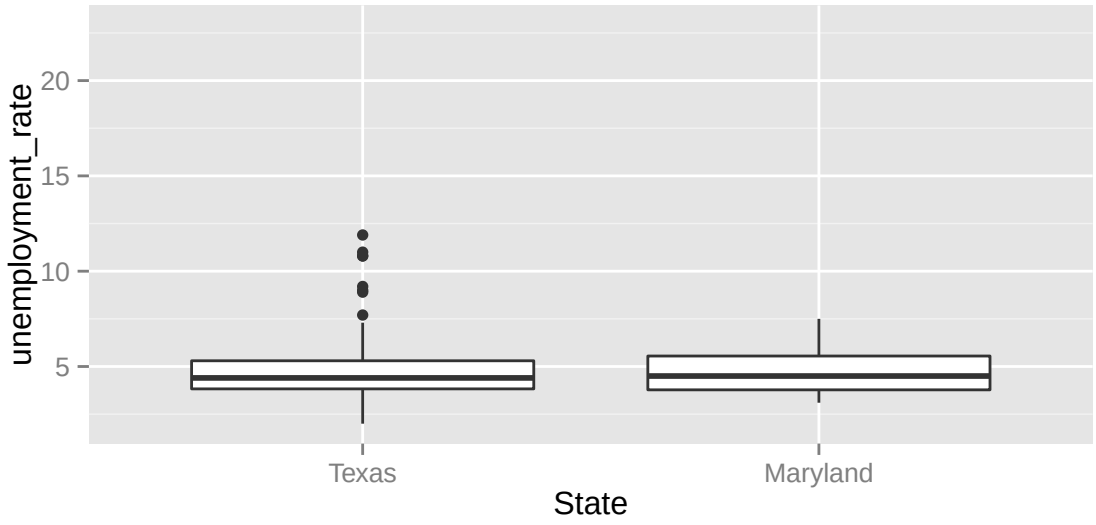
```
p2 + scale_x_continuous(limits = c(10000, 20000)) + geom_smooth(method = lm)
```



4.7.2 Diskreetsete skaalade muutmine

Diskreetsetel skaaladel töötab argument `limits` teistmoodi: nimelt saab sellega ette anda konkreetsed väärtused, mida joonisel kujutatakse, ülejäänud väärtusi siis joonisel ei kujutata.

```
qplot(State, unemployment_rate, geom = "boxplot", data = mk) +
   scale_x_discrete(limits = c("Texas", "Maryland"))  # pane tähele järjekorda
```



4.7.3 Ülesanded

1. Pane eelmises ülesandes tehtud joonisel y-telje nimeks "Higher education percentage"; muuda telje vahemikku (0, 100); muuda teljel olevaid silte ja nende paigutust nii, et need oleksid kujul 0%, 25%, 50%, 75%, 100%.
2. Lisa pildile värviga tunnus `income_class` nii, et näidataks ainult punkte, kus sissetulek on kas väga kõrge või väga madal.

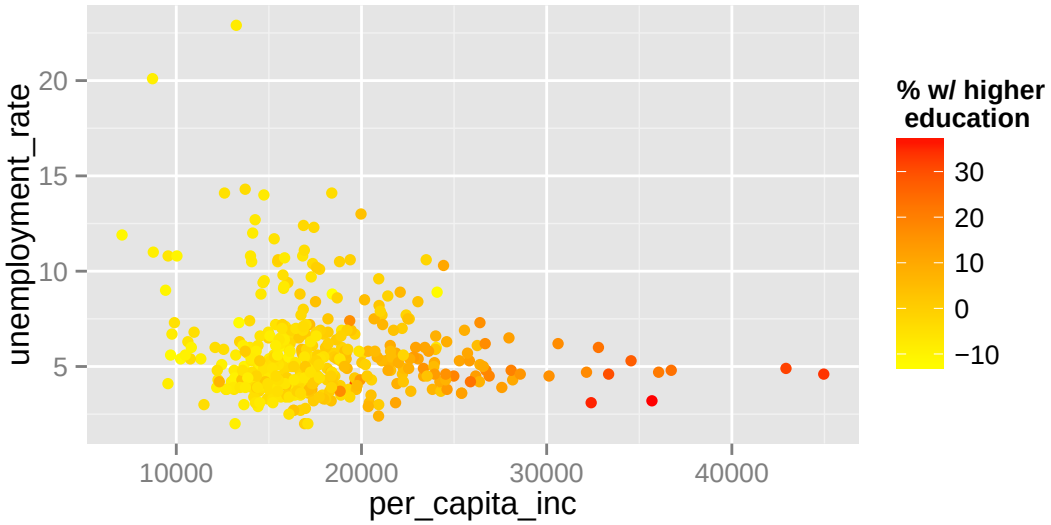
4.7.4 Värviskaala muutmine

Värvide valik joonisel on väga oluline. Õigesti valitud värvidega on võimalik tuua selgemini välja oma sõnumit, muuta joonist loetavamaks ja meeldivamaks. Asjakohane värvus sõltub tunnusest, mida soovitakse kujutada. Üldiselt võib värviskaalad jaotada kolmeks:

- gradient – pidevate tunnuste jaoks; kõige väiksem ja kõige suurem väärtus vastavad mingitele värvidele ning vahepealsed väärtused nende kahe värvi segule;
- lahknev gradient – pidevate tunnuste jaoks, kui pideval tunnusel on mingi selge nullpunkt (nt õhutemperatuur, tsentreeritud skoor vms); kaks ekstreemset väärtust ja nullpunkt vastavad mingile puhtale värvile, vahepealsed väärtused kahe värvi segule;
- kvalitatiivne – diskreetsete tunnuste jaoks; iga väärtuse jaoks kasutatakse võimalikult erinevat värvitooni. Samas on oluline silmas pidada, et heleduselt ja intensiivsuselt oleks kõik värvid võrdsed.

Gradientskaalat saab muuta käsuga `scale_<?>_gradient(.)` (küsimärgi asemel on tavaliselt `fill` või `colour`). Saab kasutada kõiki pideva tunnuse skaala muutmise argumente, lisaks on kaks argumenti, `low` ja `high`, ekstreemsete väärtuste värvi määramiseks gradiendil. Lahknevat gradienti saab muuta funktsiooniga `scale_<?>_gradientn(.)`, mille argumendile `colours` saab ette anda vektori värvidega, mille vahele uued värvid sujuva üleminekuga valitakse.

```
p3=qplot(per_capita_inc, unemployment_rate, colour = bachelor - mean(bachelor), data = mk) +
     scale_colour_gradient(name="% w/ higher \n education", low = "yellow", high = "red")
p3
```



Diskreetsete/kvalitatiivsete värviskaalade kontrollimiseks kasutatakse vaikimisi funktisooni `scale_<?>_hue(.)`, mis valib HCL värviskaalal[6] parameetri *hue* väärtused võimalikult erinevad, jättes värvi tugevuse ja heleduse konstantseks. Nii saadakse võimalikult erinevad värvid, mis samal ajal on ühesuguse intensiivsusega.

Diskreetsete värviskaalade jaoks saab kasutada värvipaletti "Colorbrewer"[7] (algselt arendatud maakaartide värvimiseks). ggplot2-s pääseb sellele paletile ligi funktsiooniga `scale_<?>_brewer(.)`, millel on kaks argumenti:

- `type` – võimalikud väärtused on `"seq"`, `"div"` ja `"qual"`
- `palette` – paleti number (vt http://www.colorbrewer2.org)

4.7.5 Ülesanded

1. Proovi eelmises ülesandes tehtud joonisel valida `income_class` jaoks erinevaid värviskaalasid. Milline sobib kõige paremini ja milline ei sobi üldse?

4.8 Joonise viimistlemine

Vaikimisi joonistab ggplot2 halli taustaga jooniseid – et heledamad värvid oleksid sama silmatorkavad kui tumedad. Kui on soov valge tausta järele, siis seda saab tellida, lisades joonisele käsu `+ theme_bw()`. Veelgi detailsemalt saab joonise kujundust käsuga `+ theme(.)`, mille argumentidest mõned olulisemad on:

Argument	Elemendi tüüp	Selgitus
axis.line	line	joonise teljed
axis.text.x	text	x-telje väärtused
axis.text.y	text	y-telje väärtused
axis.ticks	line	joonise jaotised
axis.title.x	text	x-telje pealkiri
axis.title.y	text	y-telje pealkiri

[6] http://en.wikipedia.org/wiki/Munsell_color_system
[7] http://www.colorbrewer2.org

Argument	Elemendi tüüp	Selgitus
legend.background	rect	legendi taust
legend.key	rect	legendi võtme taust
legend.text	text	legendi tekst
legend.title	text	legendi pealkiri
panel.background	rect	graafiku taust
panel.border	rect	graafikut ümbritsev raam
panel.grid.major	line	jaotise jooned
panel.grid.minor	line	jaotise jooned
plot.background	rect	kogu joonise taust
plot.title	text	joonise pealkiri
strip.background	rect	tahkude pealkirjade taust
strip.text.x	text	horisontaalsete tahkude pealkiri
strip.text.y	text	vertikaalsete tahkude pealkiri

Täielikum loetelu `theme(.)` võimalikest argumentidest on aadressil http://docs.ggplot2.org/current/theme.html .

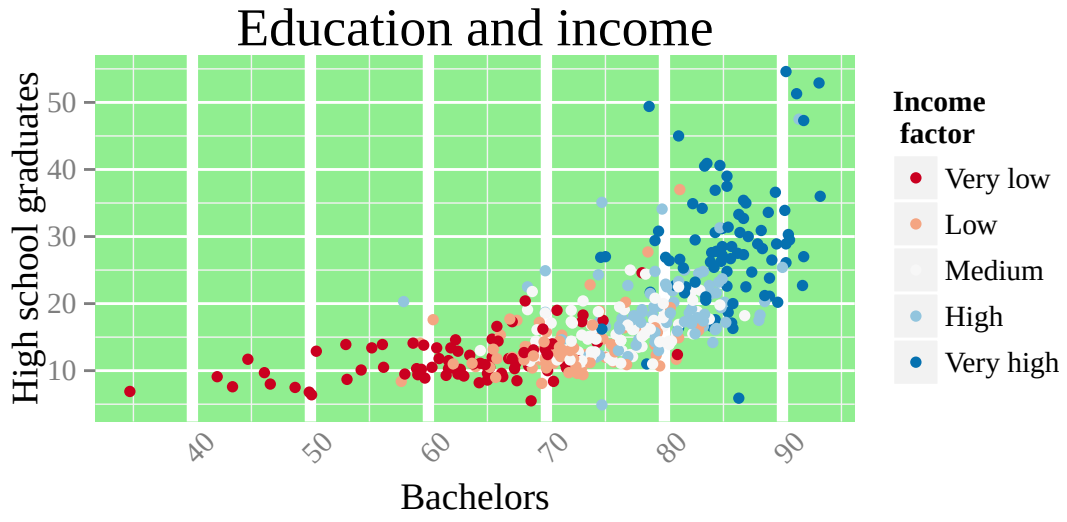
Nagu näha, on kolm peamist elementitüüpi, mida saab muuta käskudega `element_text(.)`, `element_line(.)`, ja `element_rect(.)`. Nendel käskudel on omakorda argumendid:

- `element_text(.)` – `family`, `size`, `colour`, `angle`, `hjust`, `vjust` kontrollivad teksti šrifti, suurust, värvi, kaldenurka ja positsiooni vaikimisi määratud koha suhtes
- `element_line(.)` – `colour`, `size`, `linetype`;
- `element_rect(.)` – `colour`, `size`, `linetype` ääre muutmiseks, `fill` sisemuse värvi muutmiseks

Kui mõnda elementi üldse joonisel ei soovi näha, siis tuleb vastavale argumendile anda väärtus `element_blank()`.

4.8.1 Ülesanded

1. Tee allolevaga võimalikult sarnane joonis.



4.9 Joonise salvestamine

ggplot2-ga tehtud jooniseid saab salvestada käsuga `ggsave(.)`. Kui anda käsule vaid faili nimi (koos laiendiga), siis salvestatakse viimati valmistatud joonis. Kui argumente `widht` ja `height` ei kasuta, siis joonise mõõtmed võetakse joonise akna järgi.

4.10 ggplot2 puudused

Kuna ggplot2 on paljude võimalustega suhteliselt uus pakett, siis on tal ka puuduseid. Olulisim puudus on vahest mittetäielik dokumentatsioon. Sageli häirib ka see, et `identify(.)` käsku ei saa kasutada andmestiku ridade identifitseerimiseks joonisel klikkides. Karpdiagrammi saab teha ainult vertikaalselt ning horisontaalseks tuleb see käsuga `coord_flip(.)` pöörata:

```
qplot(State, births, geom = "boxplot", data = mk) + coord_flip()
```

5 Andmestruktuurid R-is

Oleme varem tutvunud sellega, mis tüüpi võivad olla üksikud andmeelemendid (int, char, Factor jm). Üksikuid elemente saab kokku panna üheks vektoriks näiteks käsuga `c(.)`. 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. Allolevad kolm struktuuri on enimkasutatavad.

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 andmevektor ning see, mitmesse ritta ja veergu selle vektori elemendid paigutatakse. Olemasolevaid reavektoreid saab omavahel ühendada käsuga `rbind(.)`, veeruvektoreid käsuga `cbind(.)`. Maatriksi elemente saab eraldada kantsulgudega.

```
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
```

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 sobilikku elementi 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))
minulist$esimene
minulist["esimene"]
minulist[[2]]
minulist$neljas = c(5, 7) # lisame uue elemendi
```

```
## $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")
##
## [1] "üksainus sõne"
## $esimene
## [1] "üksainus sõne"
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Andmetabel (data.frame) on tegelikult teatud piirangutega list: kõik elemendid peavad olema sama pikad vektorid (võivad olla erinevat tüüpi). Seepärast saab andmetabeli veerge eraldada ka sel moel, nagu listist: `tabel["veerunimi"]`, ent see pole koodi arusaadavuse tõttu soovitatav. 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 teada.

```
class(df)
```

```
## [1] "function"
```

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

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

6 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
```

6.1 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.
- Kokkukleebitud sõne saab **tükeldada** käsuga `str_split(.)`, mille argumendiga `pattern` saab määrata, mis on tükkide eraldaja. Selle käsu tulemusena tekib **list** (sellest tuleb hiljem juttu), mida saab vektoriks muuta käsuga `unlist(.)`. 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, jäetakse vastavast indeksist alates tähed ära sõne algusest või lõpust.
- stringr käsud käsitlevad sõnesid sellisena, et iga sõne alguses on tühisõne "".

```
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=4"        "=5"
```

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

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

```
str_sub(sõnad, 1:4, 3:6)
```

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

```
str_sub(sõnad, end = -3)
```

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

6.1.1 Ülesanded

1. Loe sisse Massatchusettsi andmestik:`andmed <- read.table("http://kodu.ut.ee/~maitraag/rtr/mass.txt", sep = "\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?

6.2 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 (tuleb juttu hiljem). Kui tahame kätte saada kõigil positsioonidel olevate alamsõnede algus- ja lõpuindeksid, sobib käsk `str_locate_all(.)`, mis tagastab listi.

```
str_detect(sõnad, pattern = "r")
```

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

```r
str_extract(sõnad, "r")
```

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

```r
str_locate(sõnad, "r")
```

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

```r
str_locate_all(sõnad, "r")
```

```
## [[1]]
##      start end
##
## [[2]]
##      start end
##
## [[3]]
##      start end
## [1,]     3   3
## [2,]     4   4
##
## [[4]]
##      start end
##
## [[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.

```r
str_trim("    siin on palju tühjust    ")
str_pad(sõnad[-4], width = 9, side = "both", pad = "_") # NA jäetakse vahele
```

```
## [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(.)`

```r
str_replace(sõnad, pattern = "r", replacement = "l")
str_replace_all(sõnad, "r", "l")
```

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

Kõigile stringr paketi käkudele võib argumentidega `pattern` ja `replacement` ette anda ka regulaaravaldisi[8]; R-is on kasutusel selline määratlus, nagu on kirjeldatud käsu `regex(.)` abifailis `?regex`.

```
str_detect(sõnad, "r|l")  # püstkriips = OR
str_replace("telefoninumber: +372 55-549-85", "[a-zA-Z: ]*(\\+[0-9]*)*( )*(.*)", "(\\1) \\3")
```

```
## [1] FALSE  TRUE  TRUE    NA FALSE
## [1] "(+372) 55-549-85"
```

6.2.1 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[9] 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,– haigla, j – sel päeval selles haiglas sündimise järjekord, k – kontrollnumber.
    1. Loe sisse komaga eraldatud isikukoodide jada:
    `isikukoodid <- (read.table("http://www.ut.ee/~maitraag/rtr/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.

7 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 `pattern` 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.

[8] http://en.wikipedia.org/wiki/Regular_expression
[9] http://et.wikipedia.org/wiki/Isikukood

```r
d3 <- Sys.Date() # Tänane kuupäev
paevad <- c(d1, d2, d3)
mean(paevad)
```

```
## [1] "2011-03-25"
```

```r
d1 %in% paevad
```

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

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

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

```
## [1] "kuupäev: 21.01, (2015. aasta)"
```

7.0.1 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).

8 Andmestiku teisendused

8.1 Andmestike ühendamine (mestimine)

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

```r
mk[, "taastootev"] <- mk$births > 2.1   # kas sündimuskordaja on suur?
mk$taastootev <- mk$births > 2.1
```

Mitut uut veergu või rida andmestikule lisada on lihtne `cbind(.)` ja `rbind(.)` käskudega. Neid kasutades tuleb olla ettevaatlik: lisatavas reas peab olema sama palju elemente kui on andmestikus veerge, sealjuures need elemendid peavad olema sobilikku tüüpi (eriti palju tekitab probleeme Factor-tüüpi veergu uute väärtuste lisamine); lisatavas veerus peab olema sama palju elemente kui on andmestikus ridu.

```r
# teeme kaks vektorit (Factor ja character tüüpi):
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)
# paneme andmestiku kaks korda üksteise otsa:
mktopelt <- rbind(mk, mk)
```

Sageli on analüüsiks vaja minevad andmed mitmes erinevas andmetabelis. Näiteks jooksuvõistluste andmete puhul võivad ühes tabelis kirjas isikuandmed kohta (nimi ja sugu), teises tabels aga nende isikute võistlustulemuste andmed. Kui sooviks analüüsida tulemuste jaotust sugude vahel oleks mugav need tabelid mestida käsuga `merge(.)`.

```
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"),
      tulemus = c(30.1, 22.5, 18.4, 25.3),
      voistlus = c("jooks1", "jooks1", "jooks2", "jooks2"))
```

```
isikud[order(isikud$vanus), ]
```

```
##     nimi sugu vanus
## 4  Laine    N    20
## 2   Mari    N    22
## 3  Tiina    N    25
## 1 Peeter    M    30
```

```
tulemused
```

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

8.2 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 on element esimest või juba mitmendat korda (tulemuseks on tõeväärtusvektor, kus TRUE tähendab seda, et antud väärtus on juba mitmendat korda).

```
kokku$nimi
unique(kokku$nimi)
duplicated(kokku$nimi)
```

```
## [1] Laine  Mari   Peeter Peeter Tiina
## Levels: Laine Mari Peeter Tiina
## [1] Laine  Mari   Peeter Tiina
## Levels: Laine Mari Peeter Tiina
## [1] FALSE FALSE FALSE  TRUE 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)
```

```
## [1] 1 2 3 4 5 6 7
```

```
intersect(x, y)
```

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

```
setdiff(x, y)
```

```
## [1] 1 2
```

8.3 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)
```

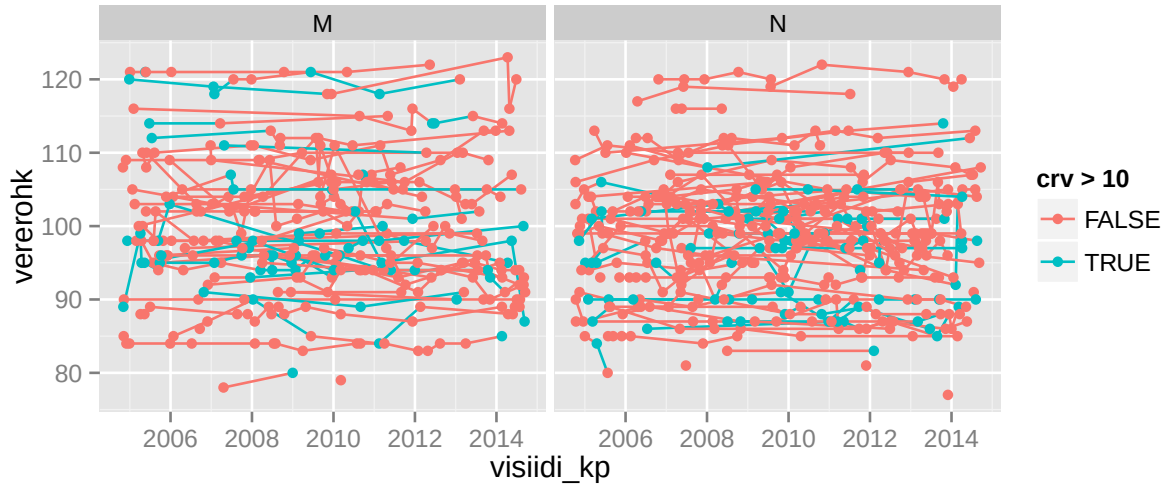
```
## [1]  1  7  7  8 NA
```

```
# saab ka mitme vektori järgi sortida:
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>
```

8.3.1 Ülesanded

1. Loe sisse arstivisiitide andmestik:
   `visiidid <- read.table("http://kodu.ut.ee/~maitraag/rtr/visiidid.txt", sep = "\t", header = TRUE)`. Selles andmestikus 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). Tegemist on samade inimestega, kelle isikukoodid eelmises praktikumis sai andmestikuks muudetud; visiitide andmestik kirjeldab viimase 10 aasta jooksul arsti külastamisel tehtud vererõhu ja CRV mõõtmisi. Kui eelmisest praktikumist pole isikukoodide andmestikku alles, siis saab selle sisse lugeda nii: `inimesed <- read.table("http://kodu.ut.ee/~maitraag/rtr/isikud.txt", sep = "\t", header = TRUE)`.
2. Järjesta visiitide andmestik kasvavalt isikukoodi ja arstivisiidi kuupäeva järgi.
3. Ühenda isikukoodide ning visiitide andmestik. Mitu inimest on käinud viimase 10 a jooksul arsti juures? Kas arsti mitte külastanud isikute osakaal (protsentuaalselt) on suurem meeste või naiste hulgas?
4. Joonista meeste ja naiste jaoks eraldi joondiagrammid nagu alloleval joonisel.



8.4 Pikk ja lai andmetabel. reshape2

Tihti on väiksemat sorti uuringutes kogutud andmeid (nt Exceli vms abil) sellisel moel, et tekkivat andmetabelit oleks inimesel lihtne mõista: tegemist on nn **laias formaadis andmestikuga**, üks 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 segamudelitega, ggplot2 ja lattice pakettidega):

```
##   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
```

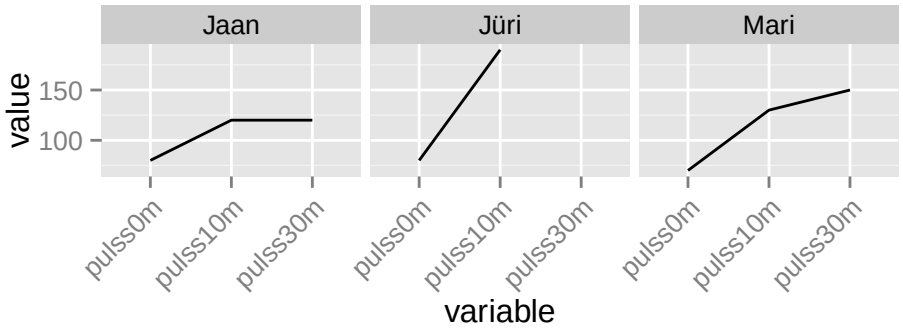
Eriti äärmuslik on pika formaadi puhul hoida kõiki arvulisi tunnuseid ühes veerus:

```
##    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
```

Kuigi R-i baaspaketis on kaasas käsk `reshape(.)`, siis on sellega üsna tüütu teisendada andmestikku ühest formaadist teise. Seetõttu 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).

```
library(reshape2) # kui arvutis pole, siis installida: install.packages("reshape2")
library(ggplot2)
vr <- data.frame(nimi = c("Mari", "Jaan", "Jüri"), 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
qplot(variable, value, data = m, geom = "line", group = nimi, facets = ~ nimi) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
##   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
```

Funktsioon `dcast(.)` aitab andmeid pikast formaadist laia teisendada, sealjuures tuleb argumendiga `formula` kindlasti ette öelda, millised tunnused määravad ära 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 rakendada `melt(.)` ja seejärel `dcast(.)` käsku.

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

```
##   nimi pulss0m pulss10m pulss30m
## 1 Jaan      80      120      120
## 2 Jüri      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 Jüri  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 Jüri 135.0000
## 3 Mari 116.6667
```

```
#     (    dcast osa, melt osa          , dcast osa    )        -- tulemus sama, mis üleval
recast(vr, nimi ~ . , measure.var = 5:7, fun.aggregate = mean, na.rm = T)
```

8.4.1 Ülesanded

1. Leia iga isiku keskmine vererõhk ja CRV.

9 Andmetöötlus paketiga plyr

Tihti soovime teostada mingit analüüsi andmestiku erinevatel alamosadel (nt arvutada meeste ja naiste keskmist palka koos usaldusvahemikuga). Selleks tuleb:

- andmestik jagada alamosadeks
- igal alamosal teostada soovitud toiming
- kombineerida tulemused kokku lihtsasti loetavaks ja töödeldavaks

R-i baaspaketiga on kaasas tükiviisiliseks andmetöötluseks sobilikud käsud `by(.)`, `apply(.)` jt. Neil käskudel on erinev süntaks ning erinev väljund (mis sageli on raskesti töödeldav). Paketis **plyr** on aga proovitud neid käske ühtlustada (baaspaketi funktsioonidele on loodud nn *wrapper*-funktsioonid). Selle paketi käsud kujul `?!ply`, kus ? on ette antava sisendi tüüp (maatriks, andmetabel, list) ja ! on soovitava väljundi tüüp (väljundit võib ka mitte olla). Allolevas tabelis on toodud ära käsud, mida erineva sisendi ja väljundi puhul kasutada:

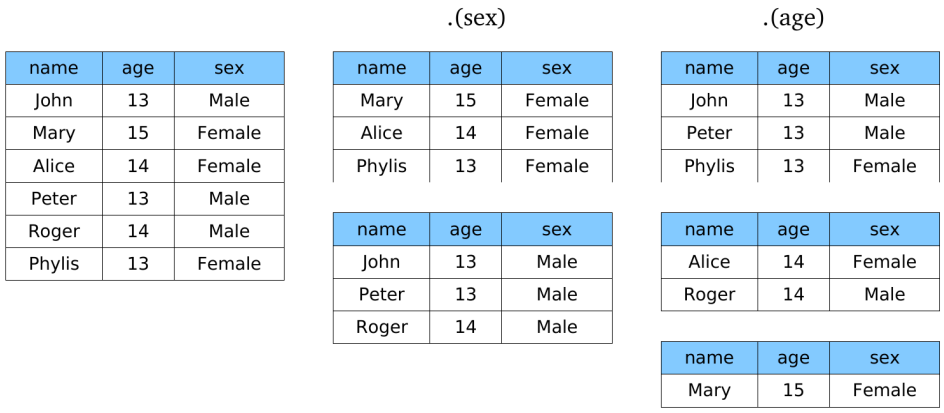
sisend	array	data frame	list	nothing
array	`aaply(.)`	`adply(.)`	`alply(.)`	`a_ply(.)`
data frame	`daply(.)`	`ddply(.)`	`dlply(.)`	`d_ply(.)`
list	`laply(.)`	`ldply(.)`	`llply(.)`	`l_ply()`

9.1 Sisendi jagamine tükkideks

Kui sisendiks on list, siis seda saab jagada ainult elementideks.

Kui sisendiks on maatriks, siis seda saab jagada üksikuteks ridadeks või üksikuteks veergudeks, milleks tuleks `a!ply(.)` käsu argumendi `.margins` väärtuseks anda vastavalt 1 või 2.

Kui sisendiks on andmetabel (see on kõige tüüpilisem olukord), siis seda võib tükkideks jagada mingi tunnuse väärtuste alusel (nt haridustase) või erinevate tunnuste väärtuste kombinatsioonide järgi (nt sugu ja haridustase). Selleks tuleb `d!ply(.)` käsu argumendi `.variables` väärtuseks anda vastavate veergude nimed.



9.2 Teostatava funktsiooni määramine

Kui andmestik on jagatud tükkideks, siis peame neile rakendama mingit operatsiooni. Kõikidel `?!ply(.)` funktsioonidel on argument `.fun`, millega saab määrata tükil rakendatava funktsiooni nime (ilma jutumärkide ja argumentideta). Tihtipeale on väga kasulikud lihtsad funktsioonid nagu `length`, `nrow` või `mean`. Kui need rakendatavad funktsioonid vajavad lisaargumente (nt `na.rm`), siis saab ka need ette anda.

```
library(plyr)
laply(vr[, 5:7], mean, na.rm = T) # data.frame on tegelikult list, elementideks veerud
# laply arvutab iga elemendi keskmise (sealjuures na.rm = T) ja tagastab vektori
```

```
## [1]  76.66667 146.66667 135.00000
```

Tihti soovime, et ühel tükil tehtaks mitu operatsiooni. Siis on võimalik ise koostada funktsioon (tutvume sellega hiljem) ja see argumendile `.fun` ette anda. Paketiga **plyr** on aga kaasas üks mugav funktsioon `summarise(.)`, millele saab anda mitu erinevat funktsiooni ning määrata ära, kuidas nende funktsioonide tulemused nimetatakse:

```
# paneme kõik pulsi mõõtmistulemused ühte veergu
m <- melt(vr, measure.vars = 5:7) # reshape2::melt
# arvutame iga inimese keskmise ja minimaalse vererõhu
ddply(m, "nimi", summarise, keskmine = mean(value, na.rm = T), miinimum = min(value, na.rm = T))
#   (andmestik, kuidas tükeldada, summarise, arvutatava väärtuse nimi ja vastav funktsioon)
```

```
##   nimi keskmine miinimum
## 1 Jaan 106.6667       80
## 2 Jüri 135.0000       80
## 3 Mari 116.6667       70
```

9.3 Tulemuste ühendamine

Mistahes tüüpi tulemusi saab alati kokku panna ühte listi. Enamasti soovime väljundiks aga saada andmetabelit. Sellisel juhul argumendile `.fun` ette antav funktsioon peab tagastama `data.frame` tüüpi objekti (`summarise(.)` funktsioon seda ka teeb).

9.4 Näiteid

```
# Kuidas on tunnused jaotunud:
head(alply(mass, 2, summary))  # 2 näitab, et tahan töödelda igat VEERGU eraldi
```

```
## $`1`
##        id
##  Min.   :    1
##  1st Qu.:16639
##  Median :32446
##  Mean   :32453
##  3rd Qu.:48477
##  Max.   :64916
##
## $`2`
##       AGEP
##  Min.   : 0.00
##  1st Qu.:20.00
##  Median :40.00
##  Mean   :39.68
##  3rd Qu.:56.00
##  Max.   :94.00
##
## $`3`
##                                                    CIT
##  Born abroad of American parent(s)                   :  62
##  Born in Puerto Rico, Guam, the U.S. Virgin Islands,:  79
##  Born in the U.S.                                    :5422
##  Not a citizen of the U.S.                           : 448
##  U.S. citizen by naturalization                      : 413
##
## $`4`
##                                                 COW
##  Employee of a private for-profit company or   :2635
##  Employee of a private not-for-profit,         : 491
##  Local government employee (city, county, etc.): 331
##  Self-employed in own not incorporated         : 289
##  State government employee                     : 157
##  (Other)                                       : 229
##  NA's                                          :2292
##
## $`5`
##    DDRS
##  No  :5879
##  Yes : 203
##  NA's: 342
##
## $`6`
##       INTP
##  Min.   : -9999
##  1st Qu.:     0
##  Median :     0
##  Mean   :  3191
```

```
##  3rd Qu.:     0
##  Max.   :229000
##  NA's   :1111
```

```
# Mitu puuduvat väärtust on igas veerus (defineerin ise uue funktsiooni):
aaply(mass, 2, function(x){return(sum(is.na(x)))})
```

```
##  AGEP   CIT   COW  DDRS   ESR    id  INDP  INTP  LANX   MAR MARHT   MIG   MIL  OCCP PINCP RAC1P
##     0     0  2292   342  1182     0  2582  1111   342     0  2760    68  1274  2950  1111     0
##  SCHL   SEX  WAGP  WKHP
##  1095     0  1111  2678
```

```
# Erineva soo ja päritoluga inimeste arv, keskmine palk ja töötundide arv:
ddply(mass, c("SEX", "CIT"), summarise,
          n = length(WAGP), palk = mean(WAGP, na.rm = T), tunde = mean(WKHP, na.rm =T))
```

```
##       SEX                                                CIT    n     palk    tunde
## 1  Female                  Born abroad of American parent(s)   31 24903.85 32.94737
## 2  Female Born in Puerto Rico, Guam, the U.S. Virgin Islands,   46 13863.90 36.81818
## 3  Female                                  Born in the U.S. 2785 24878.65 34.72685
## 4  Female                       Not a citizen of the U.S.  211 21523.18 36.45082
## 5  Female                  U.S. citizen by naturalization  226 23148.67 35.92466
## 6    Male                  Born abroad of American parent(s)   31 32420.80 40.37500
## 7    Male Born in Puerto Rico, Guam, the U.S. Virgin Islands,   33 10842.42 39.84615
## 8    Male                                  Born in the U.S. 2637 43492.89 40.58805
## 9    Male                       Not a citizen of the U.S.  237 43005.05 43.07865
## 10   Male                  U.S. citizen by naturalization  187 40199.44 40.53731
```

9.5 Kasulikud lisafunktsioonid

Paketis **plyr** on veel paari tüüpi käske. `r!ply(.)` kordab suvalist R-i käsku etteantud arv kordi ja pärast ühendab tulemused; see on kasulik juhul, kui etteantavas käsus on midagi juhuslikku. `m!ply(.)` annab meid huvitavale funktsioonile ette erinevaid argumente, mis on andmetabelis antud.

```
# Tekitame viis korda kolme-elemendilist juhuarvude vektorit ning igast vektorist leiame
# minimaalse juhuarvu (standardnormaaljaotusest)
rdply(5, min(rnorm(3)))
```

```
##   .n         V1
## 1  1 -0.7071305
## 2  2 -0.5669933
## 3  3 -2.5926963
## 4  4 -0.5292715
## 5  5 -1.1611932
```

```
# Tekitame listi, kus on kahe-elemendilised vektorid (kokku kolm vektorit)
rlply(3, rnorm(2))
```

```
## [[1]]
## [1]  0.5357281 -1.6853919
```

```
##
## [[2]]
## [1] 0.7841608 1.6789122
##
## [[3]]
## [1] 0.5114857 0.8497628
```

```
# Defineerime andmetabeli funktsiooni argumendi väärtustega:
argumendid <- data.frame(mean = c(0, 10, 100), sd = c(100, 10, 0))
# Genereerime viie-elemendilised vektorid meie poolt määratud normaaljaotuse parameetritega
mdply(argumendid, rnorm, n = 5)
```

```
##   mean  sd          V1        V2        V3        V4         V5
## 1    0 100 -66.7149416 184.59612 131.21062 -57.32150 -43.012547
## 2   10  10  -0.8540487  13.00069  22.44667  13.40243  -1.066437
## 3  100   0 100.0000000 100.00000 100.00000 100.00000 100.000000
```

9.5.1 Ülesanded

1. Loe sisse Massachusettsi andmestik: `mass <- read.table("http://kodu.ut.ee/~maitraag/rtr/mass.txt", sep = "\t")`.
2. Koosta 10-aastased vanusgrupid ja arvuta iga vanusgrupi keskmine palk (veerus `AGEP` on täpne vanus, `WAGP` on palganumber), minimaalne ja maksimaalne palk.
3. Arvuta keskmine, minimaalne ja maksimaalne palk igas vanus-soogrupis. Kujuta seda ka sobiva joonisega.

10 Veelgi kiirem andmetöötlus.

10.1 dplyr pakett

dplyr pakett on paketi **plyr** käsu `ddply(.)` edasiarendus, aitamaks suuri andmestikke kiiremini töödelda/analüüsida. Põhjalik ülevaade selle paketi käskudest on paketi dokumentatsioonis[10]. Allpool on loetletud mõned kõige vajalikumad/kasulikumad käsud.

Käsk `filter(.)` aitab selekteerida teatud kriteeriumitele vastavaid ridu. See on kiirem kui kantsulgude kasutamine, sest kantsulgusid kasutades vaadatakse üksikud elemendid ükshaaval üle, ent `filter(.)` käsu puhul kasutatakse nutikamaid algoritme (enamasti andmed sorditakse mingil moel enne kui hakatakse üldse filtris määratud kriteerume kontrollima).

```
library(dplyr) # Kui pole, tuleb installida
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise, summarize
##
## The following object is masked from 'package:stats':
```

[10] http://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

```
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Võtame ainult mõned veerud (2=AGEP,14=WAGP) ja alles siis rakendame filtrit
filter(mass[, c(1, 2, 3, 12, 13, 14)], AGEP > 70, WAGP > 100000)
```

```
##      id AGEP                             CIT                                              SCHL  SEX
## 1  9452   84 U.S. citizen by naturalization Professional degree beyond a bachelor's degree Male
## 2 49277   71               Born in the U.S.                                 Master's degree Male
## 3 64546   77               Born in the U.S. Professional degree beyond a bachelor's degree Male
##     WAGP
## 1 150000
## 2 110000
## 3 145000
```

Käsk `group_by(.)` aitab andmestiku tükkideks jagada, aga ei tee sellega midagi enamat. Kui tükkidel soovida midagi analüüsida, tuleb see ette anda vastavaks analüüsiks kasutatavale funktsioonile. Mugav funktsioon on jällegi **dplyr** paketiga kaasas olev `summarise(.)`:

```
mass <- group_by(mass, CIT)
summarise(mass, keskpalk = mean(WAGP, na.rm = T))
```

```
## Source: local data frame [5 x 2]
##
##                                                 CIT keskpalk
## 1                   Born abroad of American parent(s) 28588.63
## 2 Born in Puerto Rico, Guam, the U.S. Virgin Islands, 12516.49
## 3                                Born in the U.S. 33829.36
## 4                          Not a citizen of the U.S. 32797.35
## 5                      U.S. citizen by naturalization 30703.34
```

Lisaks on paketis **dplyr** defineeritud **toru** ehk aheldamisoperaator (Unixi käsurealt tuttav püstkriips `|`), millega on võimalik ühe funktsiooni tulemused edasi anda järgmisele funktsioonile. Toru kasutamine aitab mõnikord muuda koodi loetavamaks. Aheldamisoperaatori kuju dplyr paketis on `%>%`.

```
# Andmestiku 'mass' grupeerime kodakonsuse, soo ja perekonnaseisu kaupa ning
# iga grupi jaoks arvutame keskmise palga (kokku 48 grupp), lõpuks võtame 6 esimest rida
mass %>% group_by(CIT, SEX, MAR) %>% summarise(keskpalk = mean(WAGP, na.rm = T)) %>% head()
```

```
## Source: local data frame [6 x 4]
## Groups: CIT, SEX
##
##                                 CIT    SEX                                MAR keskpalk
## 1 Born abroad of American parent(s) Female                           Divorced 37333.33
## 2 Born abroad of American parent(s) Female                            Married 16416.67
## 3 Born abroad of American parent(s) Female Never married or under 15 years old 39071.43
## 4 Born abroad of American parent(s) Female                          Separated 32500.00
## 5 Born abroad of American parent(s) Female                            Widowed     0.00
## 6 Born abroad of American parent(s)   Male                            Married 44660.00
```

10.2 Paralleelarvutus

Üldiselt jookseb R ühe protsessina – see tähendab, et kõige jaoks, mida R teeb, kasutatakse ainult üht protsessorituuma. Kui arvuti võimaldab ning andmestiku tükkide analüüsis on iga tükk sõltumatu, on mõistlik (eriti suuremate andmestike puhul) kasutada paralleelarvutuse võimalusi: mitmetuumalise protsessori iga tuum panna tegelema omaette andmetükiga. R-is on mitmeid lisapakette, mis aitavad ligi pääseda korraga mitmele protsessorituumale. Meie kasutame paketti **doParallel**. Kui see pakett on paigaldatud, saab käsu `ddply(.)` argumendile `.parallel` anda väärtuseks `TRUE`.

```
library(doParallel) # kui pole, tuleb installeerida; vajab veel ka teisi pakette
# Avame ligipääsu kõigile tuumadele:
registerDoParallel()
# Kasutame paralleelarvutust ddply käsuga, sel juhul tuleb kasutatav
# abifunktsioon ise defineerida
ddply(mass, c("SEX", "CIT"), .parallel = TRUE, .fun = function(df){
  return(data.frame(keskmine = mean(df$WAGP, na.rm = T)))
})
stopImplicitCluster() # Sulgeme tuumad
```

```
##       SEX                                             CIT keskmine
## 1  Female                  Born abroad of American parent(s) 24903.85
## 2  Female Born in Puerto Rico, Guam, the U.S. Virgin Islands, 13863.90
## 3  Female                                  Born in the U.S. 24878.65
## 4  Female                       Not a citizen of the U.S. 21523.18
## 5  Female                  U.S. citizen by naturalization 23148.67
## 6    Male                  Born abroad of American parent(s) 32420.80
## 7    Male Born in Puerto Rico, Guam, the U.S. Virgin Islands, 10842.42
## 8    Male                                  Born in the U.S. 43492.89
## 9    Male                       Not a citizen of the U.S. 43005.05
## 10   Male                  U.S. citizen by naturalization 40199.44
```

Paketis **parallel** (mida kasutab ka pakett **doParallel**) on veel mõned paralleelarvutuseks soblikud käsud – `mclapply(.)` ja `mcmapply(.)` –, mis teostavad tuumade registreerimise ja töö peatamise automaatselt, samas nõuab nende käskude süntaks natukene peamurdmist.

Paralleelarvutuse võimalusi on mõistlik kasutada just suurte andmestike puhul. Kahjuks nõuab see ka aga mõnevõrra suuremat töömälu (erinevad tuumad tekitavad töödeldavast andmestikust omale isiklikud koopiad). Eriti gigantsete andmete puhul tasub kaaluda SQL-päringuid. R-i jaoks on mitmeid pakette, mis võimaldavad ligi pääseda SQL-andmebaasidele ning neist SQL-i või R-i lausetega väljavõtteid teha. Samuti on mõned paketid, mis muudavad data.frame tüüpi objekti sisuliselt SQL-andmebaasiks ning seetõttu aitavad selliseid andmetabeleid kiiremini töödelda. Üks viimase aja populaarne pakett on **data.table**[11].

11 Programmeerimine R-is

R on programmeerimiskeele S[12] implementatsioon; selles on olemas enamus teistest programmeerimiskeeltest tuttavaid konstruktsioone nagu tsüklid, `if-else` tingimuslaused ning võimalus kirjutada lisafunktsioone[13].

[11] http://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.pdf
[12] http://en.wikipedia.org/wiki/S_(programming_language)
[13] http://cran.r-project.org/doc/manuals/R-intro.pdf ptk 9 ja 10

11.1 Tsüklid

Tsükkel on programmikonstruktsioon, mis kordab teatud tegevust mitu korda järjest. See on kasulik näiteks siis, kui tahame vektori elemendid 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 vektori järjestikustest täisarvudest, näiteks täisarvudest 1:10.

```
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
}
```

11.1.1 Ülesanded

1. Loe sisse maakondade andmestik: `mk <- read.table("http://kodu.ut.ee/~maitraag/rtr/maakonnad.txt", sep = " ", header = TRUE)` . Arvuta ja trüki ekraanile iga osariigi jaoks maakondade rahvastike kogusumma ning keskmine rahvaarv vastava osariigi maakondades; kasuta tsüklit.

11.2 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 vasakpoolne element kuulub parempoolse 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. `ifelse(.)` esimene argument on tõeväärtusvektor, teine arugment tulemus, mis vastab tõesele väärtusele, ja kolmas, mis vastab väärale väärtusele. `switch(.)` käsu esimene argument on tavaliselt täisarv (ainult üks täisarv!) ning ülejäänud argumendid erinevatele täisarvudele vastavad toimingud (kasvavas järjekorras).

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

11.2.1 Ülesanded

1. Loe sisse arstivisiitide andmestik: `visiidid <- read.table("http://kodu.ut.ee/~maitraag/rtr/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.

11.3 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.

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

11.3.1 Ülesanded

1. Kirjutada funktsioon, mis teisendab etteantud vektori väärtused Z-skoorideks (igast väärtusest lahutatakse vektori keskmine ja jagatakse standardhälbega).
2. Leia `ddply(.)` käsu ja enda kirjutatud funktsiooni abil arstivisiitide andmestikus iga inimese puhul, kui sageli (mitu korda aastas) külastab ta keskmiselt arsti. Selleks on vaja defineerida funktsioon, mis etteantud data.frame-tüüpi objekti puhul leiab esimese ja viimase visiidi kuupäeva ja teeb antud inimese puhul vastava arvutuse.

12 Juhuarvud. Simuleerimine

Statistikas tuleb ette olukordi, kus kõige lihtsam mingi väite kontrollimiseks on vastavat situatsiooni simuleerida. R-is on palju juhuarvude genereerimise käske kujul `rJAOTUS(.)`, teoreetiliste jaotuste kvantiilide kontrollimiseks 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.

Põhjalikuma ülevaate R-is realiseeritud jaotustest leiab R-i dokumentatsioonifailist: http://cran.r-project.org/doc/manuals/R-intro.pdf ptk 8.

12.0.1 Ülesanne

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

13 Tulemuste vormistamine. knitr ja rmarkdown

Sageli kõige tüütum andmeanalüüsi juures on tulemuste vormistamine ilusaks dokumendiks. R-i väljund on tavaliselt kas konsoolis või graafikaaknas. Isegi kui tulemused on konsooliaknas tabelina, on seda Wordi vms programmi ümber tõsta tülikas. Õnneks on ka selle probleemi lahendamiseks R-is mitmeid erinevaid pakette, mille tööpõhimõte on sarnane:

1. Kirjutada oma analüüsi tekst endale meeldivas märgenduskeeles (nt LaTeX või Markdown) koos vahepealsete R-i koodiridadega ühte faili.
2. Lasta R-il koodiread asendada arvutustulemustega.
3. Kompileerida märgenduskeeles olev fail loetavamasse formaati (nt PDF või HTML)

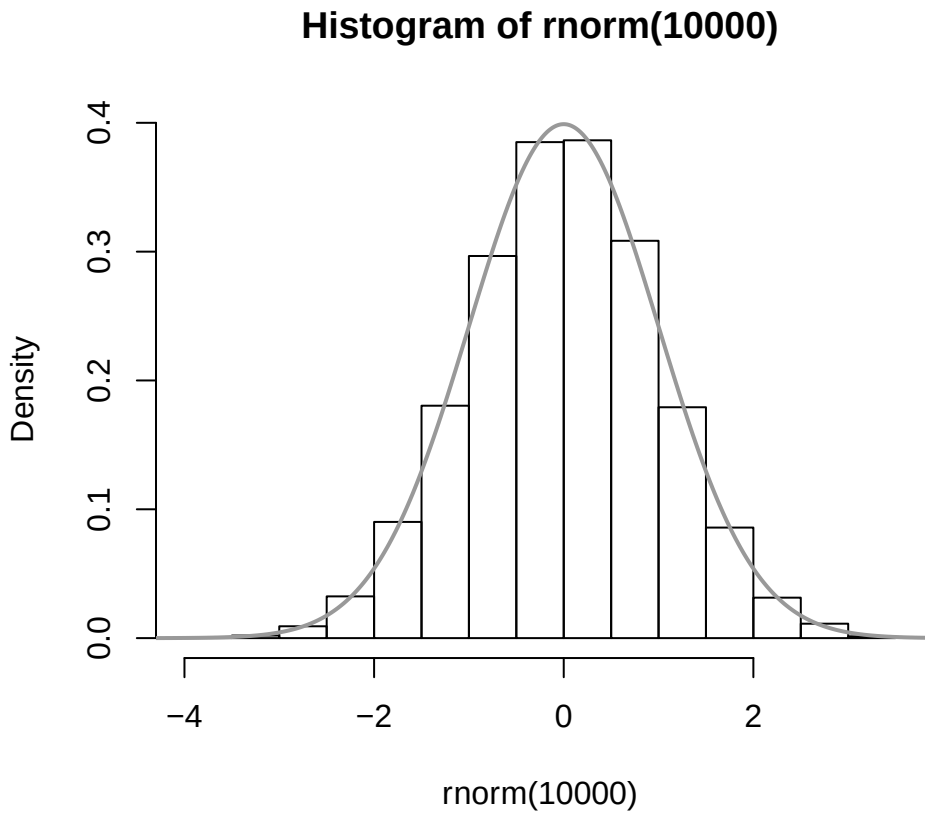
Väga paindlik pakett on **Sweave**[14], mis produtseerib LaTeX formaadis dokumente (mida saab omakorda edasi kompileerida HTML või PDF formaati). Käesolevas kursuses tutvume aga natukene lihtsama paketiga **knitr**[15]. Selle paketi kõige olulisem käsk on `knit(.)`, millele tuleb ette anda puntis 1. koostatud faili nimi. Selles failis on mingis märgenduskeeles tekst ja R-i kood vaheldumisi, näiteks nii:

````
---
output: html_document
---

Arvude 1...10 aritmeetiline keskmine on `r mean(1:10)`.
Standardnormaaljaotusest juhusliku suuruse väärtused
jaotuvad taoliselt:

```{r,echo=FALSE}
hist(rnorm(10000), freq = F)
lines(x <- seq(-5, 5, 0.01), dnorm(x), col = 19, lwd = 2)
```
````

Oletame, et ülaltoodud ridu sisaldava faili nimi on "minuraport.Rmd". Kui R-i konsoolile anda käsk `knit("minuraport.Rmd")`, tunneb R ära, et tegemist on **markdown**-märgenduskeeles koodiga ning töökausta genereeritakse fail `minuraport.md`, kus R-i käsud on asendatud tulemustega (nt `mean(1:10)` on asendatud: `5.5`). Selle faili saab paketis **markdown** oleva käsu `markdownToHTML(.)` või **rmarkdown**[16] paketis oleva käsu `render(.)` abil muuta HTML-failiks, mis veebilehitsejaga vaadates näeb välja umbes selline:

[14] http://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf
[15] http://yihui.name/knitr/
[16] http://rmarkdown.rstudio.com/

Arvude 1...10 aritmeetiline keskmine on 5.5. Standardnormaaljaotusest juhusliku suuruse väärtused jaotuvad taoliselt:



Sama protseduuri on palju lihtsam teostada RStudios[17]: valida menüüst `File -> New file -> R Markdown` ja klõpsata nupul `Knit HTML`.

Ülalolevas näites on teksti vahele pikitud R-i nn koodijuppide ehk *chunk*idega. Jupp võib olla nn reasisene ehk *inline*, või täiesti eraldiseisev. Reasisene R-i koodijupp tuleb piirata graavise sümboliga (*backtick*), mida eesti paigutusega klaviatuuril leiab *backspace* klavhi kõrvalt; koodijupi alguses peab olema täht `r`.

Sageli on mõttekas kasutada täiesti eraldiseisvaid koodiplokke, mis tuleks piirata kolmekordse graavisega. Ploki alguses tuleb loogelistes sulgudes kõigepealt kirjutada täht `r`, seejärel võib kirjutada lisaargumente[18]. Mõned olulisemad:

- `echo` – kas väljundis peaks ka R-i kood olema näha (TRUE/FALSE)
- `fig.width`, `fig.height` – kui koodiplokis tehakse joonis, siis mis mõõtmetega see peab olema.
- `results` – kuidas vormindada väljundit (konsooli väljatrükki); väärtus `'asis'` on paslik sel juhul, kui kasutame mingit spetsiifilist vorminduskäsku (nt `xtable(.)`).

[17] http://www.rstudio.com/
[18] http://yihui.name/knitr/options

Koodiplokkide vahelise teksti puhul mõned olulisemad märgendusvõtted[19]:

- `*kursiiv*` ja `**rõhutus**`
- `# Pealkiri` – esimese taseme pealkiri
- `## alapealkiri` – teise taseme pealkiri jne
- Uue lõigu alustamiseks jätta üks tühi rida vahele
- Nummerdamata loetelu elementide ette näiteks `-` või `*`; loetelu ees peab olema tühi rida
- Nummerdatud loetelu **kõigi** elementide ette `1.` (kindlasti mitte `2.`)
- Käsitsi saab tabelit vormistada miinusmärgi ja püstkriipsude abil
- allmärkuseid saab nii: `mingitekst^[allmärkuse tekst]`

Nii on võimalik R-i väljundit mugavasti ühte faili saada, ilma et peaks pidevalt kopeerima-kleepima. Kuidas aga R-i produtseeritavad tabelid ilusaks saada? Selleks on jälle palju erinevaid pakette; üks levinumaid on **xtable**, mille kõige olulisem käsk `xtable(.)` produtseerib etteantud tabelist (või vähegi tabelit meenutavast objektist, nt data.frame'ist) sobiliku LaTeX või HTML koodi. Sel juhul peaks koodiploki päises olema `results='asis'`, vastasel juhul ei tule vormindatud tabelis midagi välja.

```
library(xtable)
m <- table(mass$SEX, mass$CIT)
print(xtable(m), type = "html")
# type ja comment argumendid on print käsule
# tex->PDF puhul tuleks kirjutada nii:
# print(xtable(m, align = rep("p{2cm}",6)), type = "latex", comment = F)
```

	Born abroad of American parent(s)	Born in Puerto Rico, Guam, the U.S. Virgin Islands,	Born in the U.S.	Not a citizen of the U.S.	U.S. citizen by naturalization
Female	31	46	2785	211	226
Male	31	33	2637	237	187

14 Kordamine

Kursuse lõpetuseks teeme läbi lihtsa analüüsi näidisandmestikuga. Kasutatavas andmestikus on andmed maailma riikide rikkuse kohta, see sisaldab infot riikide rahvaarvu, koguvara ning erinevate maavarade ja loodusressursside osa kapitalist. Kõik rahanumbrid on dollarites inimese kohta.

14.1 Ülesanded

1. Loe sisse andmestik WB.txt aadressilt http://kodu.ut.ee/~maitraag/rtr/. (NB! pane tähele kuidas on tähistatud tühjad lahtrid)
2. Andmeid vaadates on näha, et osade riikide kohta on süstemaatiliselt andmed puudu.
   - Kui palju neid riike on?
   - Tekita uus andmestik, kus neid riigid on välja visatud. Kasuta selle ülseande täitmiseks tunnust `Population`.

[19] http://rmarkdown.rstudio.com/authoring_basics.html

3. Mitu erinevat regiooni on esindatud selles andmestikus? Kui palju erinevaid riike igast regioonist on?
4. Kuidas on jaotunud riigid regiooni ja jõukuse kaupa?
5. Tekita tunnus nn naftariikidest ja neist kellel seda pole (tunnuse `Oil` väärtus peab olema suurem kui 0)
6. Kuidas jagunevad naftariigid regioonide kaupa?
7. Millised on Lõuna-Aasia regiooni naftariigid?
8. Tunnus `Total.wealth` näitab riigi kõikide varade summat inimese kohta. Arvutage keskmine varade maht inimese kohta üle kõigi riikide. (Vihje: lihtsalt keskmise võtmine annab kallutatud tulemusi, kuna rahvaarv riikides on erinev.)
9. Arvutage nüüd keskmine varade maht elaniku kohta kõigis regioonides eraldi. Lisaks näidake sealjuures ära ka iga regiooni rahvaarv.
10. Joonistage graafik, mis võrdleb riikide produtseeritud kapitali ja loodusvarades peituvat kapitali. Kas joonistub välja mingi trend?
11. Proovige lisada pildile ka riikide sissetuleku grupid.
12. Lisage pildile ka mõnede huvitavamate riikide nimed.
13. Eralda andmestikust veerud `Region`, `Population`, `Natural.Capital`, `Intangible.Capital` ja `Produced.Capital`. Leia nagu ennegi iga maailmajao keskmine inimese kohta kõigi sõna “Capital” sisaldava nimega suuruste jaoks. Joonista tulemused välja tulpdiagrammina, kus on üks tulp iga regiooni kohta ja tulba kõrgus näitab kolme erineva kapitalitüübi summat ja värvidega on näidatud eri kapitalitüüpide osakaalud.