# Sissejuhatus R’i.

R on andmeanalüüsiks kasutatav vabavara, mis on muuhulgas populaarne ka ökoloogiateaduses. R’i õppimikseks on olemas õppematerjale igale tasemele, sealhulgas:

1. Statistikatarkvara R õpetus (Mait Raag, Raivo Kolde) <http://andmeteadus.github.io/2015/rakendustarkvara_R/> (pikk jutt eesti keeles)
2. Sissejuhatus statistikapaketti R (Märt Möls) [http://www-1.ms.ut.ee/mart/MC2013/SissejuhatusR.pdf (22](http://www-1.ms.ut.ee/mart/MC2013/SissejuhatusR.pdf%20(22) lk eesti keeles)
3. Statistikapakett R (Ants Kaasik) <http://lepo.it.da.ut.ee/~tammarut/Rkasutamine.txt> (eesti keeles 1 lk)
4. An Introduction to R (W. N. Venables, D. M. Smith and the R Core Team) <http://cran.r-project.org/doc/manuals/R-intro.pdf> (väga pikk “sissejuhatus”)

Ja mäleta, et google’st on pea alati kasu.

Käesolev tekst pakub abi esimesteks sammudeks R’i maailmas, sealhulgas

1. R installeerimine
2. põhifunktsioonid
3. andmete saamine R’i ja sealt välja
4. lihtsaimad joonised

## R installeerimine

1. leia <https://cran.r-project.org/>
2. vali “Download R for Windows” (või Linux või Mac, kui asjakohane)
3. vali “base”
4. laadi alla kõige uuema versiooni installeerimisprogramm (R 3.5.2 under <https://cran.r-project.org/bin/windows/base/R-3.5.2-win.exe> )
5. käivita see. Edasi piisab, kui vajutad Next piisavalt mitu korda.

Kui peaksid kunagi tahtma oma R’i **uuendada**, siis tuleb teha samamoodi nagu esmasel paigaldamisel (st järgi ülaltoodud juhiseid).

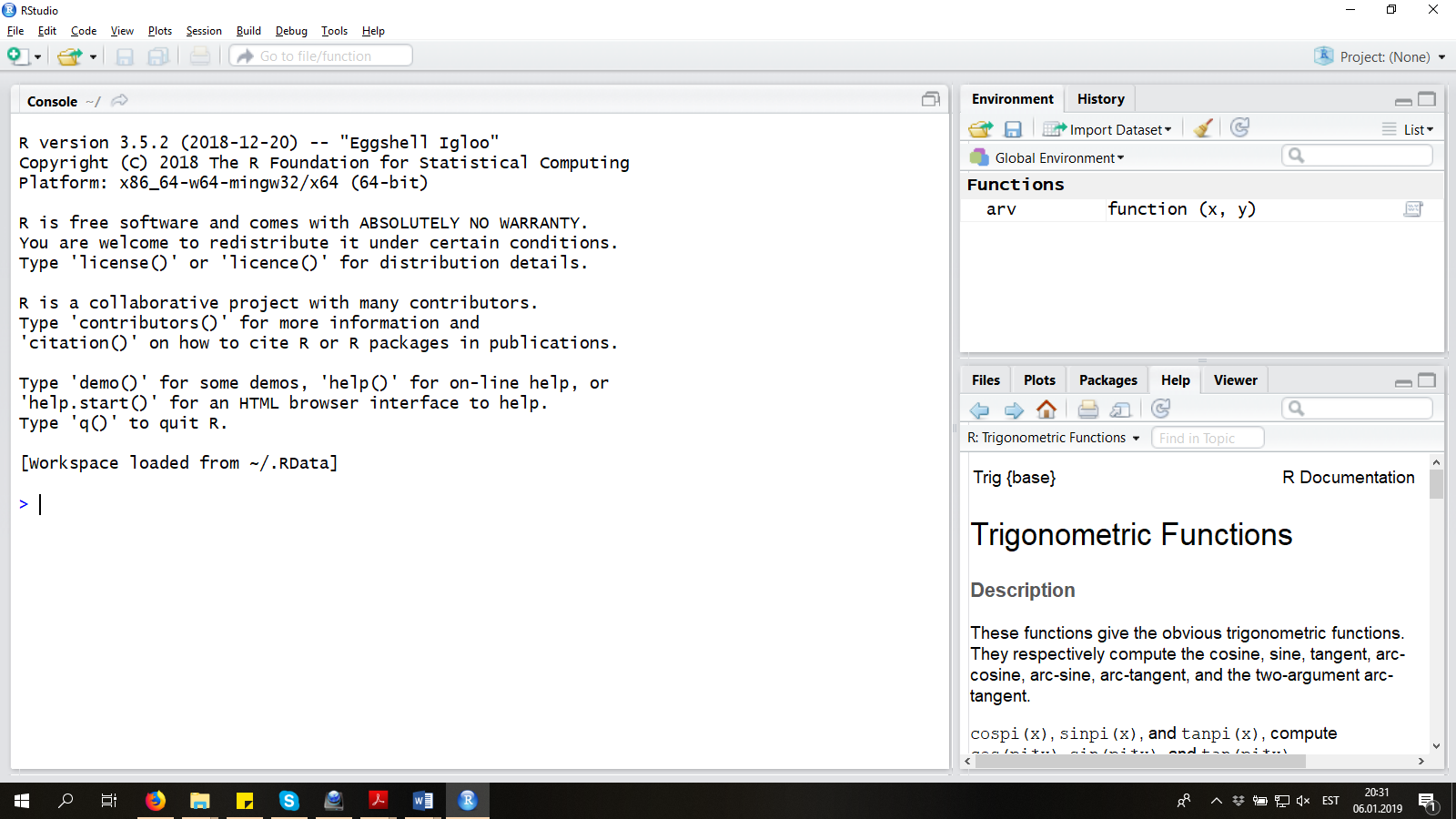
### RStudio paigaldamine

R ise näeb üsna askeetlik välja. Seetõttu on mõttekas kasutada täiendavat kasutajaliidest nagu RStudio. Selle saab paigaldada nii:

1. mine <https://www.rstudio.com/products/rstudio/download/>
2. laadi alla sobiv installer , e.g., [RStudio 1.1.463 - Windows Vista/7/8/10](https://download1.rstudio.org/RStudio-1.1.463.exe) (<https://download1.rstudio.org/RStudio-1.1.463.exe>)
3. käivita see

## R ja RStudio põhiasjad

Pärast R ja RStudio paigaldamist saab R’i kasutada läbi RStudio. Kui avad RStudio, peaks ekraanile ilmuma midagi sellist:



Vasakul poolel saad sisestada käske ja näha tulemusi.

Ülemises paremas nurgas näed nimekirja kõigist muutujatest, andmestikest ja muudest sinu loodud R objektidest.

Alumises paremas nurgas on praegu lahti abifail, aga seal võib olla muudki (vaata sakke).

### R’ga arvutamine. Funktsioonid

Klõpsa RStudio vasakpoolset akent ja veendu, et kursor vilgub > sümboli ees: R ootab käsku!

Järgmised käsud ei vaja pikalt seletamist:

(1+3)\*2

3/5

2\*\*3

sqrt(4)

log(10)

sum(1,2, 3)

log(10, base=10)

Ruutjuure, logaritmi ja summa leidmiseks tuleb asjakohased arvud sisestada sulgudesse. Selliseid asju nimetatakse **funktsioonideks**. Funktsiooni mõiste on analoogne matemaatikatunnis õpitule: funktsiooni nimele järgneb funktsiooni argument. Funktsioonil võib olla mitu argumenti, need on R’s sel puhul komadega eraldatud. Argumentidel võivad olla nimed, mispuhul tuleb argument sisestada kirjutades tema nime järele võrdusmärgi ja soovitud väärtuse. Argumentidel võivad olla ka vaikeväärtused, mispuhul pole nende sisestamine üldse vajalik. Pea kõik, mida me R’ga teeme, on saavutatav just selliste funktsioonide abil.

Kui tahad miskist funktsioonist rohkem teada (nt argumentide esitamise formaati), võid küsida abi:

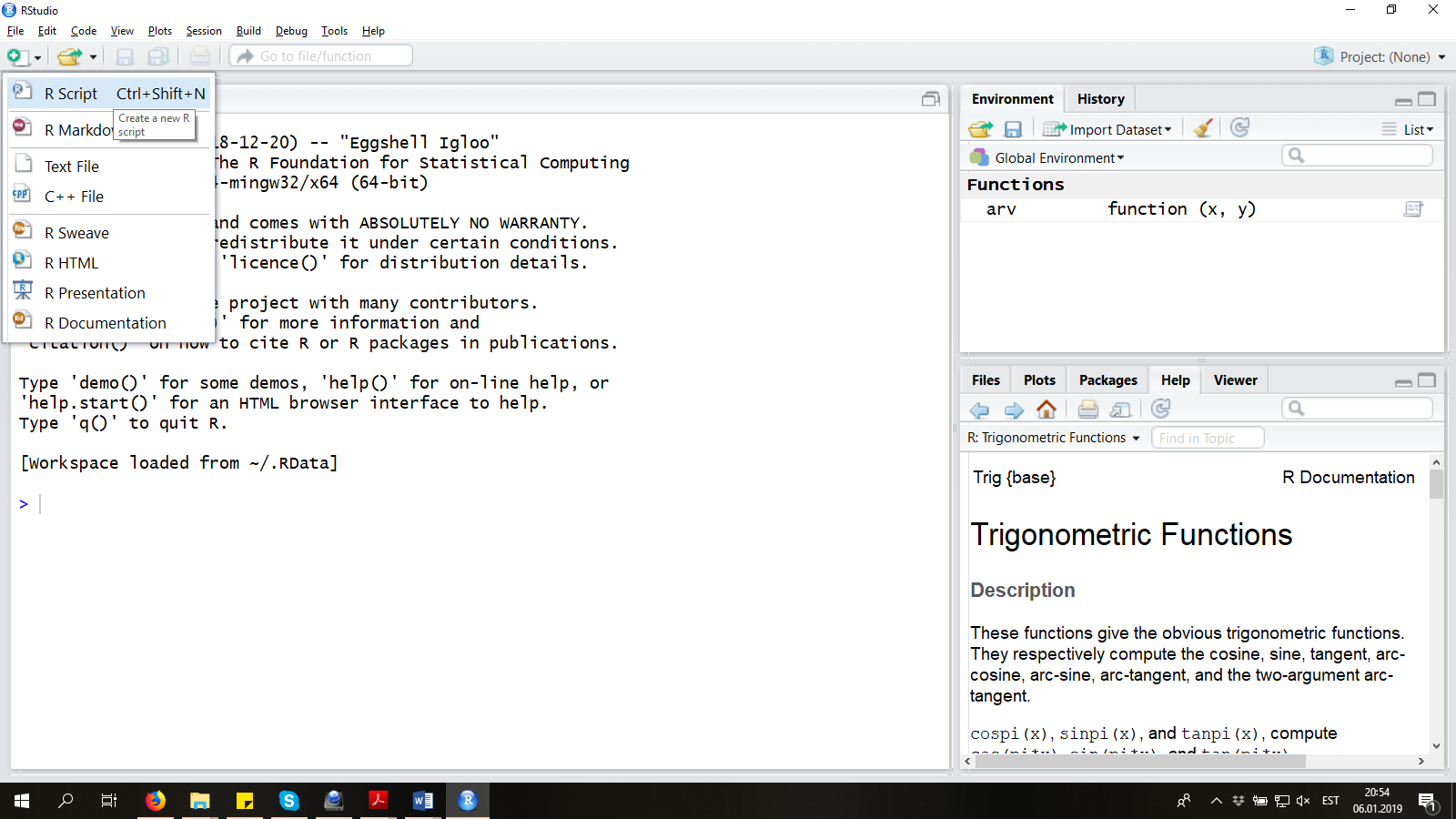
?log

Abifail avaneb akna vasakus paremas nurgas.

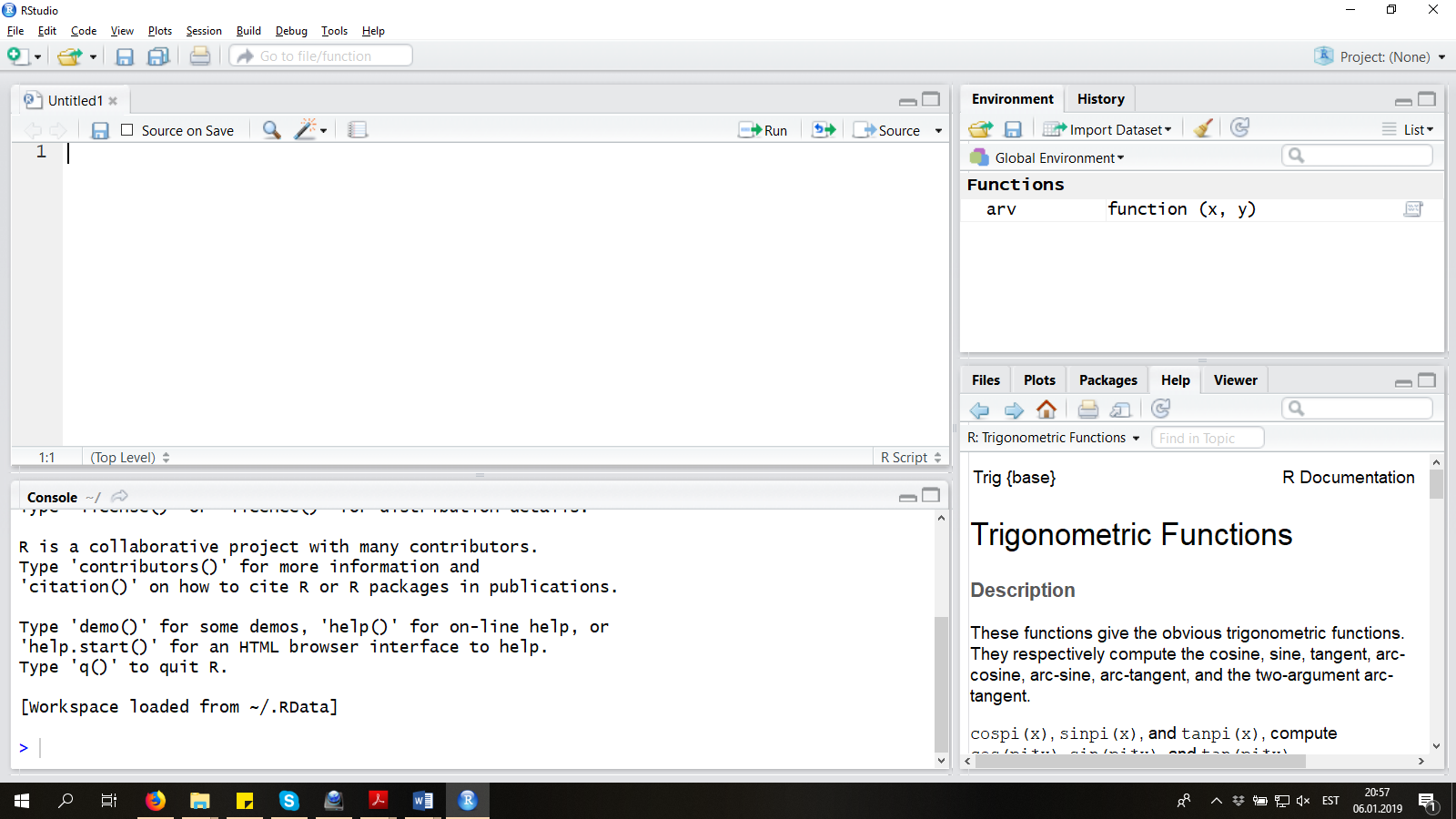
Pane tähele, RStudio’s (all vasakus alumises nurgas, kuid mitte programmiaknas) töötab *auto-complete*, st kirjutamise ajal kuvatakse nimekiri funktsioonidest ja muudest objektidest, mis algavad samamoodi kui juba sisestatud tekstijupp. Seega, kui vajutada sisestusklahvi olles kirjutanud ?log, võid kogemata saada abi hoopis funktsiooni logLik kohta.

### Skriptide salvestamine

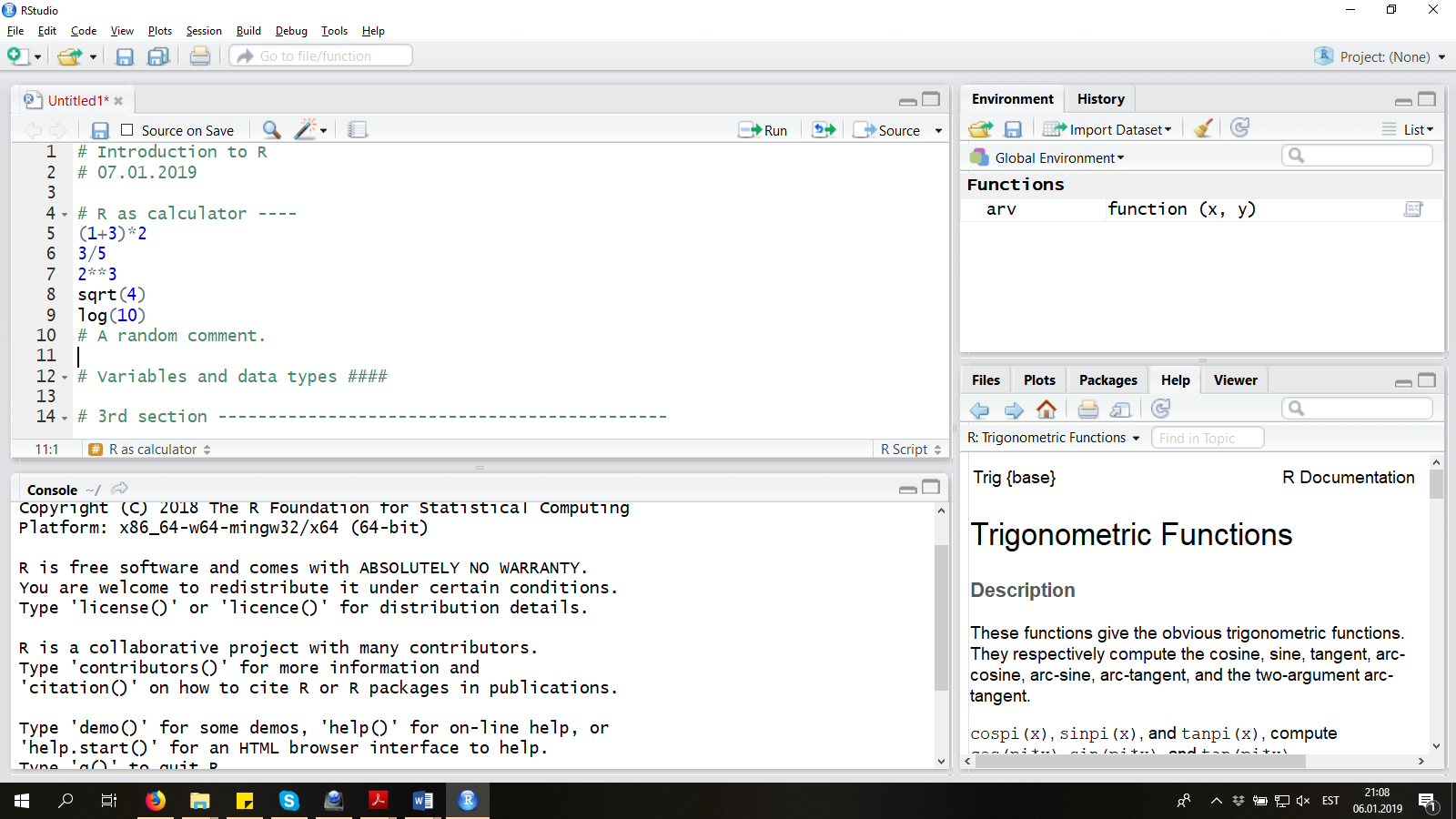
Oma tööd tasub ikka aeg-ajalt salvestada, seda ka R’s ja seal just eriti andmete teisendamise ja analüüside läbiviimise käske. Käsud saab salvestada tavalise tekstifailina (selguse huvides on neil siiski laiendiks .R, mitte .txt), mida kutsutakse skriptideks. Uue skripti saab luua valides New -> Script:



Selleks avaneb ekraanil neljas jaotis:



Skripti need osad, mida R ei pea lugema (kommentaarid sinu enda jaoks), tuleb lõpetada sümboliga #. R Studio tõlgendab kommentaare, mis lõppevad vähemalt nelja # või sidekriipsuga sektsioonide pealkirjadena ja sellistest teeb ta eraldi nimekirja programmiakna alla.



Kirjutatud skript ei ole veel salvestatud. Oodatult saab skripti salvestada kasutades sümbolit Save ülemises menüüs või siis käsuga Ctrl+S. Mõned arvutid ei lisa laiendit .R automaatselt ja see tuleb ise kirjutada.

Viimane ülesanne on sundida R kirjutatud käsud täide viima, et me soovitud tulemuse ikka kätte saaksime. Selleks on mitmeid võimalusi:

* ühe käsurea käivitamiseks tuleb kursor asetada suvalisse kohta real ja
  + vajutada Ctrl+R (või mõnedes arvutites Ctrl+Enter) või
  + vajutada Run nuppu programmiakna paremas ülemises nurgas
  + mitmest käsureast koosneva programmijupi käivitamiseks tuleb see selekteerida ja teha ikka samamoodi, nagu eelmistes punktides kirjas.

Peab meeles pidama, et pelgalt programmi kirjutamine ülal vasakul asetsevase aknasse ei käivita seda. R’le tuleb anda konkreetne käsk (ülal), et ta asjast aru saaks ja töö ära teeks.

### Muutujate ja andmete tüübid

R oskab meelde jätta erinevaid objekte hilisemaks kasutamiseks, näiteks

x = -2.5

loob muutuja x ja omistab sellele väärtuse -2.5 (punkt, mitte koma). Pane tähele, et x koos oma väärtusega -2.5 ilmub ka ülemisse paremasse aknasse (Environment).

tekst = "Kassid on siledad."

loob muutuja “tekst” ja omistab talle väärtuseks jutumärkidesse kirjutatud jutu.

Tekstmuutuja väärtusi võib sisestada nii ülakomade ‘ kui ka jutumärkide “ vahel.

Muutujate loomiseks (ja ülekirjutamiseks) võib võrdusmärgi asemel võib kasutada ka kombinatsiooni

<- (meenutab noolt), mis annab tugevama visuaalse signaali toimuva kohta.

Muutujate nimed ei saa alata numbriga, kuid sobib pea iga muu sümbolite kombinatsioon.

**Objektide nimed R’s on üldiselt tõusutundlikud (*case-sensitive*)** nt

sqrt(4)

töötab, aga

Sqrt(4)

SQRT(4)

annab veateate, sest R jaoks on sqrt, Sqrt ja SQRT erinevad objektid ja vaid esimesel on hetkel R jaoks tähendus.

#### Vektor

Vektor on lihtsalt andmepunktide jada, mis võivad olla nii arvulised kui tekstilised. Vektoreid saab luua funktsiooniga c (lühend sõnadest “concatenate” või “combine”, esimest saab R’s ka niisugusena kasutada):

z <- c(1.5,2,3,4)

w <- c("kass", "koer", "kala", "kala")

Vektoritega saab sooritada erinevaid operatsioone, mitmed neist on üsna ilmsed:

sum(z)

mean(z)

sd(z) # standardhälve

var(z) # dispersioon

table(w)

length(w)

Üldiselt tahab R siiski teostada operatsioone elementhaaval:

2\*z

z\*\*2

log(z)

z+z

#### Matrix

Maatriks on maatriks nagu matemaatikaski. Maatrikseid saab luua funktsiooniga matrix.

#### Data frame

*data frame* on lihtsalt andmetabel. Data frames saab luua liites vektoreid nimega tulpadeks funktsiooni data.frame abil:

a <- data.frame(animal=w, smoothness=z)

## Andmestiku määratud osade kuvamine

Vektorite ja maatriksite soovitud elemente saab ühe- või mitmekaupa vaadata viidates nende järjekorranumbrile nurksulgude sees. Selleks on mitmeid võimalusi:

z[1]

z[c(1,3)]

z[1:3] # uuri ka seda, mida teeb pelk 1:3

z[z < 3] # uuri ka käsku z < 3

z[w == "kala"] # uuri ka käsku w == "kala"

Elemente võib kuvatavate hulgast välja arvata miinusmärgi abil:

z[-c(1,3)]

Maatriksite puhul tuleb mõistagi elemendi määramiseks kasutada kahte indeksit, nende vahele käib koma: enne rea number, siis tulba number.

zmat <- matrix(z, nrow=2)

zmat

zmat[1,2]

zmat[1, ] # esitab kogu esimese rea

*Data frame’*de jaoks on kaks võimalust. Kasuta dollarimärki ühe tulba vaatamiseks vektorina

a$smoothness

Kui nii tehtud, saab valitud tulbaga teha kõike seda, mida vektorigagi:

a$smoothness[a$animal == "kala"]

*data frame*’t saab siiski ka otse nurksulgudesse kirjutatud täpsustuste abil osi välja kuvada, põhimõtteliselt samamoodi nagu maatriksitegi puhul:

a[1, 2]

a[a$animal == "kass", ]

kuid *data frame* korral saab tulpadele viidata ka nende nimesid kasutades:

a[, c("smoothness")]

*data frame* on teie enim kasutatud andmevorm, millest saate teha valikuid ka otseste kriteeriumite alusel

newdata <- subset(mydata, sex=="m" & age > 25)

b <- subset(a, animal == "kass")

## Andmete sisselugemine R’i

Saamaks muus formaadis olevaid andmeid R’i, tuleb eelkõige kasutada funktsiooni read.table, mis loeb andmeid süsteemivälisest tekstifailist ja loob neist *data frame*. Mängime asja läbi:

### Mängunäide

Võta Excel lahti. Kirjuta sinna järgmine tekst ja salvesta see faili testdata (ka viimane tühi rida). Jäta hästi meelde, kuhu sa faili salvestasid!

nimi,sugu,kaal

Mark,mees,85

Mati,mees,100

Kati,naine,60

Mari,naine,75

Ava R(Studio). Ütle programmile, kust sisseloetavaid faile otsida ja kuhu salvestada loodavad failid ehk siis teisisõnu defineeri **töökataloog** (working directory**).**  See peab mõistagi olema see, kuhu sa andmed salvestasid ega pruugi olla sama nimega kui siin näites:

setwd("C:/Users/Kasutaja/Documents")

Selle aadressi saad klõpsates paremat klahvi mis iganes failil, mis seal kataloogis juba on, seejärel vali „Atribuudid“ (Properties) ja kopeeri ilmuv „Asukoht“ (Location) vajalikku kohta.

R’ile ei meeldi sedapidi \ kaldkriipsud, seega asenda kõik \-märgid /-märkidega või siis topelt tagurpidi kaldkriipsudega (\\).

Mac’i kasutajad ei vaja C: ‘d.

Loeme nüüd andmed sisse. Sisse loetav *data frame* tuleb salvestada objektina ja anda sellele nimi, olgu siis nimi praegu testdata. Funktsioon read.table tahab argumentidena teada sisse loetava faili nime, andmevälju eraldavat märki ja lisaks seda asjaolu, kas esimeses reas on ära toodud ka muutujate nimed:

testdata <- read.table("testdata.txt", header=TRUE, sep=",")

Sisastatud andmete kuvamiseks arvutiekraanil sisesta käsuna andmestiku nimi:

testdata

Andmete sisselugemisel saab täpsustada paljut muudki (käsk ?read.table on seega väga hea mõte).

### Andmete sisselugemine excelist

Andmete sisselugemine R’i otse Exceli failist ei ole paraku väga lihtne, kuigi internetist muidugi leiab asjakohaseid õpetusi. Töökindel viis on salvestada andmed Excelist miskisse lihtsamasse formaati ja edasi minna read.table funktsiooniga nagu ülal seletatud.

Ava Exceli fail korduvmõõtmiseülesannetedata19.xls kursuse võrgulehelt:

<http://lepo.it.da.ut.ee/~tammarut/repedata18eng.xls>

Klõpsa File -> Save As… ja vali Save as type: Text (Tab delimited). Klõpsa OK.

Sule Excel. Klõpsa Yes/OK/Save nii mitu korda kui nõutud (minul neli korda). Pane jälle tähele, kuhu sa faili salvestasid. Lihtsuse mõttes salvesta see samasse kausta, kus on testdata.txt.

Andmeid saad vaadata Notepadis. Andmed peavad olema failis repedata18eng.txt ja andmeväljad peavad olema eraldatud tabulaatoritega.

Loe andmestik sisse järgmise käsuga:

repedata <- read.table("repedata18eng.txt", header=T, sep="\t")

Pane tähele, et loogiline väärtus TRUE võib olla kirjutatud lihtsalt T (analoogselt, FALSE on sama mis F),

ja sep="\t" tähendab tabulaatoritega eraldamist.

### Sagedasi probleeme

1. Andmetes sisaldub välju eraldav sümbol (nt koma). Kui nii, siis andmete sisselugemine äpardub täiega:

father,mother,children

Martin,Mari,Uku,Kalle

Kui nii, siis tuleb kas oma andmetes kirjavehemärke muuta või lasta R’l kasutada muud andmeväljade eraldamise sümbolit.

2. Andmetes sisaldub jutumärke või muud, mida R tsitaati tähistavana tõlgendab:

Nii võib saada hoiatuse (nt “incomplete final line found” vms) sellisel puhul:

sender,subject,time

markgimbutas@gmail.com,Mark's homework,2019-01-01 16:45

või veateate sellisel puhul:

nimi,sugu,kaal

Mati',mees',100

Kati,naine,60

või võib ka nii minna, et veateadet ei kuvata, kuid andmed loetakse valesti sisse:

nimi,sugu,kaal

Mati',mees,100

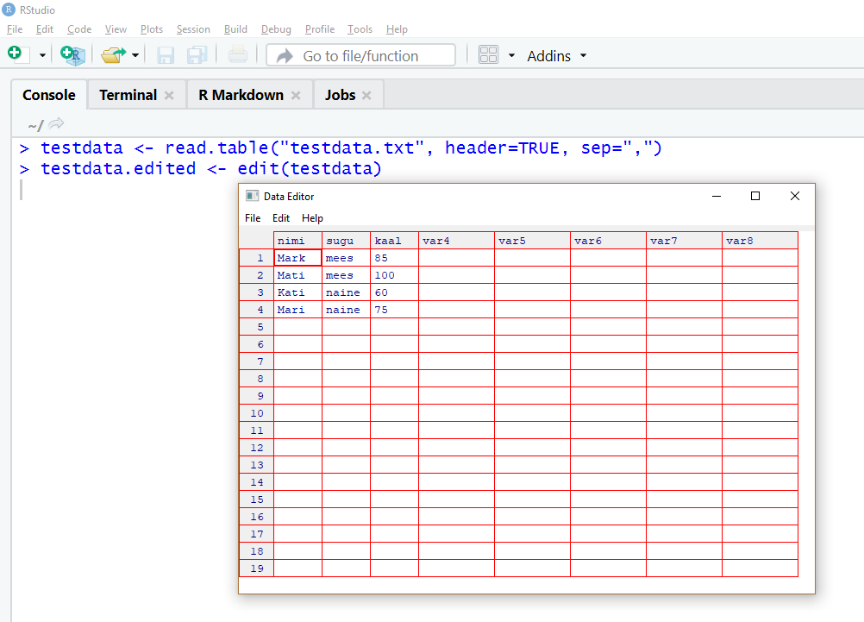
Kati',naine,60

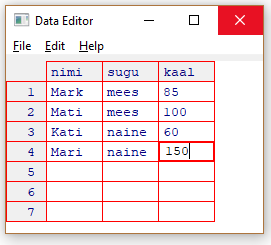
Lahendus: lisa quote = "\"" funktsiooni argumentide hulka.

Võid ise järgi uurida, mida täpselt selline argument ikkagi teeb (vihje ?read.table)

## Andmete muutmine R’s

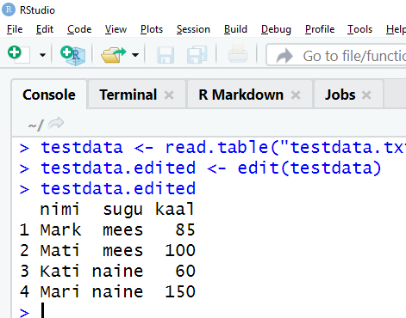
Üldiselt ei ole R eriti ette nähtud andmete sisestamiseks, toimetamiseks vms. See on siiski võimalik funktsiooniga edit. Nii näiteks avavad käsud

testdata.edited <- edit(testdata) andmetabeli eraldi aknas.



Väärtuse muutmiseks tee kaks klõpsu asjaomases lahtris. Ka uute väärtuste lisamine on võimalik täites tühje lahtreid. Kui andmed muudetud, pane aken kinni.

Akna sulgemisel salvestatakse andmed sellise nimega objektiks nagu kästud (testdata.edited). Vana andmetabeli võib muidugi uuega üle kirjutada (kirjuta testdata testdata.edited asemel), kui oled päris kindel, et midagi sassi ei ajanud.



Andmete muutmine R’s ei muuda algseid andmefaile Excelis või misiganes formaadis nad on, seega uusi andmeid kuskile faili ei salvestata ilma, et teeksid nii, nagu kohe allpool räägitud.

## Andmete eksportimine R’st

Sageli võib vaja minna R’s töödeldud andmestike kasutamist muus keskkonnas. Andmete eksportimiseks on R’s read.table’le analoogne funktsioon nimega write.table. Tee läbi järgmine näide:

write.table(testdata, "testdata.csv", sep=";", row.names = F)

Fail salvestatakse töökataloogi. Kui ei mäleta, kus see töökataloog on, siis sedagi saab R’lt küsida

getwd()

## Paketid

Lisaks ülaltutvustatud põhifunktsioonidele on R’s loodud ja luuakse juurde hulgaliselt spetsiifiliste ülesannete (n.t. elumusanalüüs, lineaarsed segamudelid, üldistatud lineaarsed mudelid, graafika, tekstianalüüs jne) lahendamiseks sobivaid funktsioonide **pakette (*packages*)**. Sellised (lisa)paketid ei tule kaasa R’i tavalise installeerimise käigus või vähemalt ei ole nad kasutamiseks valmis. Üheks selliseks on pakett ggplot2, milles on palju kasulikke funktsioone graafikute joonistamiseks. Võimaks noid kasutada tuleb siis asjaomane pakett installeerida:

install.packages("ggplot2")

Nüüd on pakett küll arvutisse laetud (sellist installeerimiskäsku ei ole vaja samas arvutis kunagi enam korrata), kuid ta pole veel kasutamiseks valmis. Viimaseks tuleb anda täiendav käsk

library(ggplot2)

ja seda juba iga kord, kui R’i käivitad. Kui nii tehtud, saab kasutada selliseid funktsioone nagu

qplot(data=repedata, x=time, y=shell, color=treatment, group=individual, geom="line")+ theme\_bw()

## Joonised ja kuidas neid salvestada

R pakub laialdasi võimalusi andmete ja tulemuste graafiliseks esitamiseks.

Jooniseid saab eraldi kaskude abil salvestada mitmesugustes erinevates formaatides. Kui joonis avaneb eraldi aknas kus muud polegi, tuleb lihtsalt valida File -> Save ja määrata soovitud formaat. Kui joonis avaneb alumises parempoolses aknas, siis tuleb valida Export -> Save as Image… ja jätkata asjakohasel viisil.

Jooniseid saab salvestada ka automaatselt (nt kui on vaja tekitada hulgakaupa samatüüpi jooniseid ). Näiteks nii:

png("testdataplot.png")

qplot(data=testdata, fill=sugu, x=sugu, geom="bar")

dev.off()

## Andmete ülevaade R’s

str(a)

'data.frame': 4 obs. of 2 variables:

$ animal : Factor w/ 3 levels "kala","kass",..: 2 3 1 1

$ smoothness: num 1.5 2 3 4

Uurige antud käske:

head(a, 2)

tail(a,3)

class(a)

class(a $ animal)

is.factor(a $ animal)

Saame ka muuta andmetüüpe:

a $ animal <- as.numeric(a $ animal)

# Põhilised andmetöötlusvõtted

Andmete sisselugemisest ei tarvitse olla küllalt, et analüüsima hakata. Andmed võivad olla ebasobival kujul. Näiteks võivad olla isendite andmed ühes, aga keskkonna-andmed teises andmetabelis. Või siis võivad sama isendi korduvad mõõtmised olla ühe rea erinevatel veergudel, aga tarvis oleks kõik mõõtmised saada ühte tulpa.

R on vabavara, kuhu saavad kõik soovijad funktsioone juurde kirjutada (ja need siis oma pakettides teistelegi kättesaadavaks teha). Seetõttu on ka andmete formaadi muutmise jaoks rohkesti erinevaid funktsioonide komplekte: vastavate autorite „visioon“ mugavatest andme-teisendustest on kajastatud R-i „baaspaketis“ ja lisapakettides reshape2, plyr, dplyr. Käesolevas materjalis on autori harjumuse tõttu kasutatud pigem baaspakettide funktsioone, kuigi kõiki järgnevaid asju oleks kindlasti võimalik (ilmselt ka kellegi arust mugavamalt ja kiiremini) ära teha ka eelmainitud pakettide vahendeid kasutades.

## Uue tunnuse lisamine andmestikku

Lähtume eelnevalt salvestatud andmestikust testdata.txt.

Meeldetuletuseks: kopeerige järgmised read (sealhulgas viimane, tühi rida!) Notepad-i ja salvestage see testdata.txt nime all.

nimi,sugu,kaal

Mark,mees,85

Mati,mees,100

Kati,naine,60

Mari,naine,75

Lugege see andmestik R-i sisse (juhis on leitav eelmise praktikumi materjalist) nime all testdata.

Lisame andmestikku nende inimeste pikkused (cm).

testdata$pikkus <- c(174, 180, 169, 165)

**---------------------------------------------------------------------------------------------------------------------------**

**Harjutus**

Arvuta välja ja salvesta kehamassiindeks selle andmetabeli tunnusesse nimega bmi.

## Kahe andmestiku ühendamine

Oletame, et mingis teises andmestikus on meil mõõdetud inimeste jäsemeid. Eesmärgiks on uurida jäsemete pikkuse ja kaalu vahelist seost. Tekitage järgnev andmestik (näiteks kopeerige Notepad-i, salvestage, lugege R-i sisse) nime all jäsemepikkused.

nimi,jäse,jäsemepikkus

Mark,käsi,73

Mari,käsi,70

Mark,jalg,100

Mari,jalg,100

Kati,jalg,95

Karl,jalg,101

Paneme tähele, et Matilt ei ole meil ühegi jäseme pikkust mõõdetud. Küll aga oleme suutnud mõõta Karli jala, kelle kaalu ja pikkust meil ei ole.

Kahe andmestiku ühendamiseks on vaja, et neis oleks mingid võtmetunnused, mille väärtuste kaudu saaks andmetabelite vastavaid ridu siduda. Tunnus nimi kõlbab selleks väga hästi: see on olemas mõlemas andmestikus ja selle läbi saab konkreetselt öelda, et missugused read tabelis testdata vastavad missugustele ridadele tabelis jäsemepikkused.

Andmestike ühendamise töö teeb meie eest ära funktsioon merge. Talle tuleb öelda, et missugused andmestikud on vaja ühendada. Lisaks on talle igaks juhuks hea üle täpsustada, et missuguse tunnuse järgi käib andmestike ühendamine. Tulemus on jällegi mõistlik ära salvestada.

yhendatud <- merge(testdata, jäsemepikkused, by="nimi")

Uurime nüüd korra tulemust.

> yhendatud

nimi sugu kaal pikkus jäse jäsemepikkus

1 Kati naine 60 169 jalg 95

2 Mari naine 75 165 käsi 70

3 Mari naine 75 165 jalg 100

4 Mark mees 85 174 käsi 73

5 Mark mees 85 174 jalg 100

Pange tähele, et vaikimisi jäetakse alles ainult need inimesed, kes olid olemas mõlemas ühendatavas tabelis. Kui me soovime näha kõiki vasakpoolses tabelis leidunud inimesi (näiteks et teada saada, kellel jäid kõik jäsemed mõõtmata), tuleks teha nii:

yhendatud2 <- merge(testdata, jäsemepikkused, by="nimi", **all.x=T**)

Tulemus on siis selline:

> yhendatud2

nimi sugu kaal pikkus jäse jäsemepikkus

1 Kati naine 60 169 jalg 95

2 Mari naine 75 165 käsi 70

3 Mari naine 75 165 jalg 100

4 Mark mees 85 174 käsi 73

5 Mark mees 85 174 jalg 100

**6 Mati mees 100 180 <NA> NA**

NA tähistab R-is puuduvaid väärtusi (Not Available).

Analoogia põhjal võite mõelda, mida võiks teha merge(testdata, jäsemepikkused, by="nimi", **all.y=T**).

## Kokkuvõtted tunnuse tasemete kaupa

Võime soovida näha näiteks keskmist kaalu sugude kaupa, või siis keskmist jala ja käe pikkust sugude kaupa. Selle jaoks sobib näiteks funktsioon tapply. See funktsioon tahab saada kolme argumenti:

1. arvuline tunnus, mille kohta kokkuvõtteid soovitakse (see peaks olema arvude vektor)
2. faktorid (üks või rohkem), mille tasemete (kombinatsioonide) kaupa soovitakse kokkuvõtteid saada (see peaks olema vektor või siis vektorite list, vaata näidet allpool)
3. funktsioon, mis annab meile soovitud kokkuvõtte

Näiteks keskmised kaalud sugude kaupa saame nii:

tapply(testdata$kaal, testdata$sugu, mean)

Keskmised jäsemete pikkused sugude kaupa saame aga nii:

tapply(yhendatud$jäsemepikkus, **list**(yhendatud$sugu, yhendatud$jäse), mean)

## for-tsükkel

R näeb välja nagu programmeerimiskeel. Nagu igale programmeerimiskeelele kohane, on ka R-is võimalik teha tsükleid. Jätame konkreetsed kasutus-näited hilisemaks ja toome siin ära ainult paar süntaksinäidet:

for (i in **1:nrow(yhendatud)**) {

print(i)

}

Vaadake, mida teeb eelneva programmikese esimesel real paksus kirjas toodud käsk eraldi võttes:

1:nrow(yhendatud)

Ilmselt ei ole üllatav, et funktsioon nrow(yhendatud) annab andmetabeli yhendatud ridade arvu. Üllatav võib olla see, et kooloni abil saab teha järjestikuste täisarvude vektoreid (alates : kuni).

Veel üks näide mitmekesisusest:

for (kala in c("särg", "räim", "mättias")) {print(kala)}

## Uute funktsioonide kirjutamine

Soovime arvutada välja näiteks pikkuste variatsioonikordajat (*coefficient of variation*, standardhälbe ja keskmise suhe) sugude kaupa. Jõle mugav oleks kasutada sedasama tapply funktsiooni. Aga ei tule kohe meelde variatsioonikordaja funktsiooni nimi R-is… aga pole hullu, sest et me saame selle funktsiooni ise kirjutada. Skeem oleks selline:

*funktsiooni nimi* = function (*funktsiooni argumendid*) {

*mida see funktsioon teeb*

return(*mida see funktsioon tagastab*)

}

Näiteks variatsioonikordajat arvutav funktsioon võiks olla selline:

cv <- function(x) {

return( sd(x) / mean(x) )

}

Nüüd saame rakendada oma uut funktsiooni teise funktsiooni sees:

tapply(testdata$pikkus, testdata$sugu, cv)

## Andmetabeli teisendamine pikka ja laia formaati

Eelkõige näiteks korduvate mõõtmiste andmestike korral saab rääkida andmestiku pikast ja laiast kujust. Laiaks formaadiks nimetatakse sellist andmetabelit, kus iga mõõdetud looma kohta on üks rida. Näiteks käesoleva kursuse kodutöö korduvate mõõtmiste andmestik laias formaadis näeb välja selline:

individual treatment time1 time2 time3 time4

1 snail 1 5 4 3

2 snail 1 5 4 2

3 snail 2 4 4 2

4 nosnail 3 1 2 3

5 nosnail 3 1 1 3

6 nosnail 4 2 2 5

Lai formaat näeb välja kompaktsem ja inimesele paremini loetav / täidetav, aga arvutitele meeldib sageli statistilisi analüüse teha just pikas formaadis andmetega. Pikaks formaadiks nimetatakse sellist andmetabelit, kus iga mõõtmine on eraldi rida. Näiteks seesama korduvate mõõtmiste andmestik oma esialgsel kujul on pikas formaadis.

Andmete teisendamiseks pika ja laia kuju vahel on rohkesti funktsioone, üks neist on reshape.

### Pikk -> lai

Et saada andmed laia formaati, tuleks eelnevalt loodud jäsemepikkuste andmestikuga talitada nii:

jäsemed.lai <- reshape(

jäsemepikkused, # mis andmestikku teisendada

direction = "wide", # mis suunas teisendada

idvar = c("nimi"), # mis tunnus(ed) iseloomustavad indiviidi

timevar = "jäse" # mis tunnuse väärtused saavad tulbanimedeks

)

Harjutus

Vala eelmainitud korduvate mõõtmiste andmestik repedata18a.xls laia formaati.

### Lai -> pikk

Andmete „sulatamiseks“ pikka formaati kasutame funktsiooni melt lisapaketis nimega reshape2. Pakett tuleks siis kasutusele võtta käsuga

library(reshape2)

Eelnevalt loodud andmestikuga testdata tuleks siis talitada nii:

testdata.pikk <- melt(

testdata, # mis andmestikku teisendada

id.vars = c("nimi", "sugu"), # mis tunnus(ed) iseloomustavad indiviidi

variable.name = "tunnus", # endiste veerunimede veeru nimeks

value.name="väärtus" # väärtuste veeru nimeks

)

## Esimene suurem harjutus: liblikaloendus

Ühe suve jooksul loeti eelnevalt kindlaks määratud aladel kolmel korral liblikaid. Kõik need loendus-alad on kirjas tabelis alad.csv, kus on lisaks ala koodidele kirjas ka ala koordinaadid (antud juhul mitte pikkus- ja laiuskraadid, vaid sisuliselt meetrite arv kuskilt kokkuleppelisest lähtepunktist). Saadud liblikaloenduste tulemused (poole tunni jooksul alal kohatud liblikate arv) on failis loendused.csv. Et uurida liblikaliikide eelistusi keskkonnatingimuste suhtes, ongi alade tabelisse juba juurde otsitud mulla niiskus. Lisaks mulla niiskusele võivad liblikaid mõjutada igasugused asjad, miks mitte näiteks ümbritsevate metsade tüüp. Õnneks on Eestis olemas selline tore asi nagu metsaregister, kus on kõige muu kasuliku info seas olemas ka okaspuude osakaal iga väikese metsatükikese kohta. Neid metsatükikesi nimetatakse eraldisteks ja nende andmed on tabelis eraldised.csv. Targad onud arvavad, et eelkõige ala keskpunktist kuni 500 meetri kaugusel asuv mets võiks liblikate eluga kuidagi seotud olla.

Need kolm eraldiseisvat tabelit aga ei kõlba analüüsiks. Andmed tuleb saada sellisele kujule, kus tabeli iga rea peal on üks loendus-ala ja liblikaliigid on veergudes. Sealjuures iga liigi kohta peaks kirjas olema maksimaalne arvukus, mida antud alal suve jooksul kohati (kõiki alasid külastati ju kolm korda, et ära näha nii hiljem kui ka varem lendavad liigid).

Üks võimalik ettepanek tööde järjekorra suhtes:

1. Loo alade tabelisse uus tunnus, kus on iga ala jaoks kirjas nende eraldiste keskmine okaspuude osakaal, mis jäävad antud alast 500 meetri kaugusele (meenuta matemaatikatundi ja Pythagorase valemit). Kasuta for-tsüklit üle alade (for (i in 1:nrow(alad)) {…}).
2. Kasuta funktsiooni tapply, et leida iga liblikaliigi maksimaalne arvukus igal alal.
3. Veendu, et loendusandmete tabel oleks lõpuks laias formaadis (et alad oleksid ridades ja liblikaliigid veergudes).
4. Ühenda loendusandmete tabel ja alade tabel.

Kui andmestik on sobival kujul käes, võid proovida uurida erinevate liblikaliikide arvukuste omavahelisi seoseid, seoseid keskkonnatingimustega või lausa midagi mitmemõõtmelist.

Erilised huvilised võivad proovida ära arvata, kuidas (missugusest teile käesoleva kursuse raames teada olevast statistilisest jaotusest, millele vastavat analüüsimudelit te ka praktikumis nägite) ma genereerisin liblikate arvukused (käsitledes kõiki vaatlusi tabelis loendused.csv omavahel sõltumatutena, ärge arvutage mingeid suve maksimume vms) ja mis on selle mudeli parameetrid (nagu nad õigest analüüsimudelist peamõjude hinnangutena välja tulevad, arvestades et ainult eelmainitud kolmes tabelis leiduvad tunnused võivad seda liblikate arvukust mõjutada).