Tallinna Ülikool

Digitehnoloogiate Instituut

Informaatika

SISSEJUHATAV ÕPPEMATERJAL R-KEELDE

Bakalaureusetöö

Autor: Harry Kaarma

Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „ 2017

Juhendaja: ..... „ ..... „ 2017

Instituudi direktor: ..... „ ..... „ 2017

Tallinn 2017

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Harry Kaarma (sünnikuupäev: 16.03.1994)

1. Annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Sissejuhatav õppematerjal R-keelde” , mille juhendaja on Jaagup Kippar , säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

2. Olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____ , _____

(digitaalne) allkiri kuupäev

Sisukord

Sissejuhatus

Modernses ühiskonnas on väga tähtsal kohal statistika. Kõigis valdkondades, riigivalitsemisest majanduseni ja rahva arvamusest täppisteadusteni on väga tähtis hoomata läbivaid trende, tulevikku prognoosivaid tunnusmärke, ja eksistentsi olemust kirjeldavaid mustreid. Statistiliste andmete kogumiseks ja nende töötlemiseks on olemas mitemeid häid viise ning üheks üsna unikaalseks andmete töötlusviisiks ongi R programmeerimiskeel. Erinevalt teistest sama otstarbega tööriistadest on R palju paindlikuma kasutusmudeliga, kuid selle võrra ka tehnilisem ja raskemini hoomatavam. Selliste tehnilise olemusega tööriistade jaoks on eriti tähtis, et oleks olemas head õppematerjalid. Probleem seisneb selles, et R-keele jaoks ei ole veel olemas eestikeelseid algtaseme õppematerjale. Sellised õppematerjalid kuluks ära nii iseõppijatele, kui ka näiteks andmeanalüüsi kursuse kuulajatele. Seega seadsingi selle bakalaureusetöö eesmärgiks luua selline õppematerjal, mis demonstreeriks R-keele kasutust lihtsamate statistiliste ülesannete lahendamiseks, ja aitaks kasutajatel mõista R-keele kui tööriista olemust ning selle võimekust oma valdkonnas. See õppematerjal peaks olema seletava ja jutustava ülesehitusega ning mitte ülemäära spetsiifiline ega tehniline. Paljud inimesed, kes vajavad statistikaga tegelemiseks tööriistu, ei ole informaatika taustaga. Materjal peaks olema ka neile mõistetav, ning seega tuleks hoiduda liialt keerulisest lähenemisest. Sihtgrupiks võiks siis määratleda kõik ülikooli tudengid, olenemata erialast, kellel võib vaja minna statistiliste andmetega töötamiseks tööriistu. Loodav materjal peaks sisaldama jutustavaid seletusi, näiteid ning lihtsamaid ülesandeid, mille lahendamine aitaks lugejatel R-keele ehitust ja struktuuri enda jaoks kinnistada. Materjali kavatsen üles ehitada R-keele ametlikule dokumentatsioonile tuginedes (R Development Core Team, 2017). Loodetavasti on sellest õppematerjalist kasu kõigile, kellel on vaja paindlikul viisil statistikaga tegeleda.

1 R keele olemus

R on programmeerimiskeel ja keskkond, mis on mõeldud statistiliste arvutuste tegemiseks ja statistilise graafika loomiseks. R on GNU GPL litsentsi alusel jagatav tasuta tarkvara, ja seda võib pidada S-programmeerimiskeele järeltulijaks. R-keelt on võimalik kasutada mitmetes operatsioonisüsteemides, sealhulgas Windowsis, Linuxis, ja MacOSis. R-i statistikafookust iseloomustavad selle andmemassiividekeskne muutujatesüsteem, lai sisseehitatud andmemanipulatsioonifunktsioonide kogu, lai sisend/väljund süsteem, ning selle võimekas ja paindlik diagrammide koostamise liides. R on kasutaja poolt laiendatav: kasutajad saavad ise funktsioone ja pakette luues lisada keelele funktsionaalsust. R-keele avatud legaalne staatus teeb nende lisapakettide loomise ja jagamise kasutajate jaoks lihtsaks. (The R Foundation, kuupäev puudub)

2 Sihtgrupp ja metoodika

Nagu sissejuhatuses mainitud on loodava materjali sihtgrupiks kõik tudengid, kellel võiks R-keele statistikatööriistu vaja minna, olenemata nende erialast või eelnevast programmeerimiskogemusest. Seda arvestades otsustasin kirjutada materjali põhiosaks jutustavas laadis ohtrate näidetega lühikursuse. Eeldatav töövool oleks seega järgmine: kasutaja loeb läbi kirjeldava teksti, ning sisend/väljund formaadis koodinäite. Seejärel võib kasutaja koodinäite omas arvutis tööle panna, ning seda muutes funktsionaalsust katsetada. Sellise lähenemise tugevaks kohaks on loetavus: jutustavas laadis püstitatud probleemi lahendamist demonstreerivat materjali on lihtne järgida ka vähem tehnilise suunitlusega inimestel.

Üheks nõrgaks kohaks jääb ülevaatlikkus: jutustav laad ei võimalda tihti selgelt ja ladusalt määratleda kogu tõde mõne konstrukti või käsu olemuse kohta. Liiga palju tehnilise info ühte kohta surumine raskendab loetavust ja alistab nii jutustava laadi kasutamise mõtte. Selle murekoha vähemalt osaliseks lahendamiseks otsustasin õppematerjalile lisada teise poole spikkertabelitega (*quick-reference table*), mis sisaldavad ka spetsiifilisemaid tehnilisi andmeid, mida töö esimeses osas pikalt lahti ei seletata. Nendest võivad usinamad õpilased, kelle jaoks esimene pool üksi igavaks jääb, ammutada veidi selgema tehnilise pildi, kuid nende kasutamine ei tohiks olla esimese osa läbi töötamiseks vajalik.

Jutustava laadi teiseks nõrgaks kohaks võib saada teadmiste kinnistamine. Liiga lihtsate näidiste ja vabalt ümber kopeeritavate koodinäidete tõttu on reaalne oht, et õpilased teevad küll kaasa, kuid ei mõista tegelikult demonstreeritavate asjade olemust. Selle probleemi lahendamisel on abiks, kui õppematerjali lisada harjutusi. Need harjutused ei peaks olema liialt keerukad: nende lahendamiseks peaks piisama, kui kasutaja mõistab eelnevates näidetes demonstreeritud funktsioonide ja struktuuride toimimispõhimõtteid, ja oskab neid oma vajadustele vastavalt ümber töödelda. Harjutused ei ole mõeldud kontrolltööks või eksamiks, vaid lihtsalt selleks, et kontrollida, et õppematerjali kasutaja ikka kaasa mõtleb.

Nagu sissejuhatuses mainitud, oleks õppematerjali ülesehitamisel põhiliseks kasutatud kirjanduseks R-keele ametlik dokumentatsioon (R Development Core Team, 2017). Tasub veel mainida, et kasutajasõbralikkuse eesmärgil oleks materjali näidetes

kasutatud ka RStudio kasutajaliidest, mis teeb teatud operatsioonid lihtsamaks. Valmis õppematerjali leiate selle töö lisade alt (Lisa 1).

3 Sarnased materjalid

Vaatame võrdluseks ja kattuvuse analüüsimiseks lühidalt üle mõned sarnaste või seotud teemadega õppematerjalid.

3.1 Lingvistika analüüs R-keele abil

Magnus Kvelli poolt bakalaureusetöö „Lingvistika analüüs R-keele abil: õppematerjal" (Kvell, 2016) korras kirjutatud õppematerjal, mis demonstreerib R-keele kasutamist üsna spetsiifiliste lingvistiliste statistikaülesannete lahendamiseks. Teemade kattuvus käesoleva töö käigus loodava materjaliga peaks olema vaid üpris pinnapealne, kuna käesoleva töö käigus loodav materjal on kavandatult väga laiapõhjaline, seal kus M. Kvelli materjal esitleb teatud funktsioone väga kindlas kontekstis kindlate probleemide lahendamiseks (lingvistika analüüsimiseks). Tehniliselt peaks käesoleva töö käigus valmiv materjal M. Kvelli materjali komplementeerima, mitte seda otse asendama või sellel põhinema, niiet siinkohal pole suurt mõtet otseselt eeskuju võtta.

3.2 R Tutorial (tutorialspoint)

Inglise keelne R-i algtaseme õppematerjal, veebisaidilt tutorialspoint.com (R Tutorial, 2017). See on üks esimesi vasteid, kui otsingumootorist inglisekeelseid R-i õppematerjale otsida. Antud materjal on väga struktuuripõhise ehitusega: selles on palju loogilisi koodinäiteid ja andmetabeleid, aga jutustavaid/kirjeldavaid näiteid ei ole. Arvestatava eelneva programmeerimiskogemusega ning inglise keele oskusega inimeste jaoks võib selline lähenemine olla loogiline ja kasulik, kuid meie sihtrühma jaoks on selline ehitus liialt tehniline, ja raskesti hoomatav. Kui selle töö käigus valmiva materjali sihtrühmaks oleks ainult informaatikud, võiks selle inglise keelse materjali (või mõne teise selle sarnase) üks-ühele ümbertõlkimine olla praktiline lahendus. Töö tehniliste andmete pooles leiduvate tabelite loomisel kavatsen selle materjali tabelitest eeskuju võtta, kuid lühikursuse osa saab olema kardinaalselt erinev.

3.3 R Tutorial (cyclismo)

Veel üks inglise keelne R-i algtaseme õppematerjal, seekord veebisaidilt cyclismo.org (Black, 2015). Samuti esimeste otsingumootori vastete seas, kui otsida inglisekeelseid R-keele õppematerjale. Erinevalt eelmisest näitest on see materjal jutustavama laadiga. Tehniliselt võiks selle materjali tõlge olla täiesti adekvaatne eesti keelne ressurss, kuid minu arvates valmistab probleeme teatud näidisandmete ülemäärane keerukus, ja natuke veider struktuurne järjestus. Programmeerimise mõttelaadi viimisega nõuame kogemusteta kasutajatelt juba märgatavat keskendumist. Ei maksa neid lisaks sellele veel liialt keerukate siendandmetega kohutada. Üldisest ehitusest ja koodinäidete esitusest kavatsen lühikursuse osa jaoks eeskuju võtta, kuid sisendandmed loodaksin vähemalt alguses hoida lihtsamatena.

3.4 Statistikatarkvara R õpetus

Tartu Ülikooli aine „Rakendustarkvara: R” jaoks koostatud õppematerjal (Raag & Kolde, 2015). Muidu hästi seletatud, kena kujundusega ning käsitleb kõiki teemasid, mida veel tahta võiks, kuid on üles ehitatud eeldusel, et seda kasutatakse käsurealt, ilma igasuguste ilustusteta. Teemad kattuvad selle kursuse käigus loodava materjaliga märgatavalt, kuid Tartu Ülikooli materjal on ilmselgelt kirjutatud arvestatavama programmeerimiskogemusega kasutajaid silmas pidades. Samuti on antud materjal kohati isegi veidi liiga põhjalik. Rangelt sissejuhatavas materjalis on sellisel tasemel põhjalikkus ehk liigne, nii et kavatsen seda oma töös vältida. Samas, näidete kvaliteedi, ja üldise eesti keelse kirjutuslaadi osas tasub sellest materjalist igatahes lühikursuse osa jaoks eeskuju võtta.

4 Testimine ja tagasiside

Kahjuks ei avanenud mul võimalust materjali laias ringkonnas testida. Küll aga sain materjali intensiivsemalt katsetada sihtgruppi kuuluva isiku peal, (ilma informaatika taustata, vaid minimaalse eelneva programmeerimiskogemusega) kellelt sain ka vahetult tagasisidet.

Üldiselt pidas testija materjali arusaadavaks, ning ei vajanud üheski kohas kaasa tegemiseks vahetult abi. Küll aga jäid mõned keerulisema ehitusega käskude nüansid testija arvates ebaselgeks. Sellele kriitikale vastukajaks muutsin nende käskude sissekandeid töö teises pooles, et nende argumentide olemused selgemini lahti seletada.

Samuti juhtis testija tähelepanu seigale, et teksti sees oli käsunimede kujundus ebaühtlane. Vastukajaks muutsin käsunimede kujundust nii, et väljaspool koodinäiteid on käsunimed alati kaldkirjas, ja muust tekstist selgesti eristatavad.

Kokkuvõte

Selle töö kirjutamine võttis omajagu tegemist. Kõige suuremaks peavaluks osutus sobiva kirjutustaseme saavutamine: esimesed versioonid olid liiga range ülesehitusega ja näidete-harjutuste struktureerimine muutus väga segaseks. Töö struktuuri sain korralikult paika, kui otsustasin lõplikult, et jagan töö vastavalt „lühikursuse" ja „tehniliste tabelite" osadeks. Kartsin isegi siis, et õppematerjal võib olla paljude sihtrühma esindajate jaoks kohati liiga tehniline ja raskesti mõistetav. Testimise käigus aga selgus, et probleem ei olnud üldse nii suur. Suures osas oli see tänu RStudio kasutajaliidesele, mis muutis koodinäidete haldamise ja käivitamise väga mugavaks ning hoidis ära kasutajaliidese puudusest tulenevad lisaprobleemid. Kahtlemata heaks ideeks osutus õppematerjali alguses kasutajaliidese üles seadmisele terve peatüki pühendamine ning seal ohtralt piltide kasutamine. Töö nõrkadeks kohtadeks loeksin eelkõige seda, et teatud teemasid ei õnnestunud mugavalt sisse mahutada. Näiteks ei jõudnud töösse head näidet karpdiagrammide kasutamisest, ning statistiliste arvnäitajate osa jäi kohati liiga pinnapealseks. Samuti oleks olnud parem, kui oleks saanud valmis materjali põhjalikult testida. Kõige märgatavamad probleemid said ehk kõrvaldatud, kuid kindlasti leiavad tulevased kasutajad neid veel. Lõppkokkuvõttes aga sai materjal siiski valmis, ning kõige põhilisemad teemad said selles adekvaatselt käsitletud. Ruumi edasiarenguks kindlasti on, kuid pean oma praegust tulemust suuremalt osalt eesmärgile vastavaks.

Summary in english

Title: „Introductory Learning Material for the R Programming Language„

The goal of this bachelor's thesis was to produce an introductory learning material for the R programming language that requires no prior coding experience or computer science background in order to be understood by the user. The intended users could be, for example, attendants of the „Data Analysis" course, or independent learners who have need of statistics tools the likes of which R offers. Since none of the available introductory materials for R were structured in a way as to be simple enough, the author decided to write the material from scratch, instead of translating an existing material from english.

Over the course of this thesis, a small description of the R programming language was provided (chapter 1), the need for the material, and the intended methods for it's creation were established (chapter 2), some similar materials were briefly analyzed (chapter 3), the material was created, briefly tested, and small changes were made based on test feedback (chapter 4).

The author believes that the end product mostly fits the set goal, but could have used more testing.

Kasutatud kirjandus

The R Foundation, (kuupäev puudub). *What is R?* Loetud aadressil https://www.r-project.org/about.html

R Development Core Team, (2017). *The R Manuals.* Loetud aadressil https://cran.r-project.org/manuals.html

Kvell, M. (2016). *Lingvistika analüüs R-keele abil: õppematerjal* (bakalaureusetöö). Loetud aadressil http://www.cs.tlu.ee/teemaderegister/

Tutorialspoint, (2017). *R Tutorial.* Loetud aadressil https://www.tutorialspoint.com/r

Black, K. (2017). *R Tutorial.* Loetud aadressil http://www.cyclismo.org/tutorial/R/index.html

Raag, M. & Kolde, R. (2015). *Statistikatarkvara R õpetus.* Loetud aadressil http://andmeteadus.github.io/2015/rakendustarkvara_R/

LISAD

Lisa 1. Sissejuhatav õppematerjal R-keelde

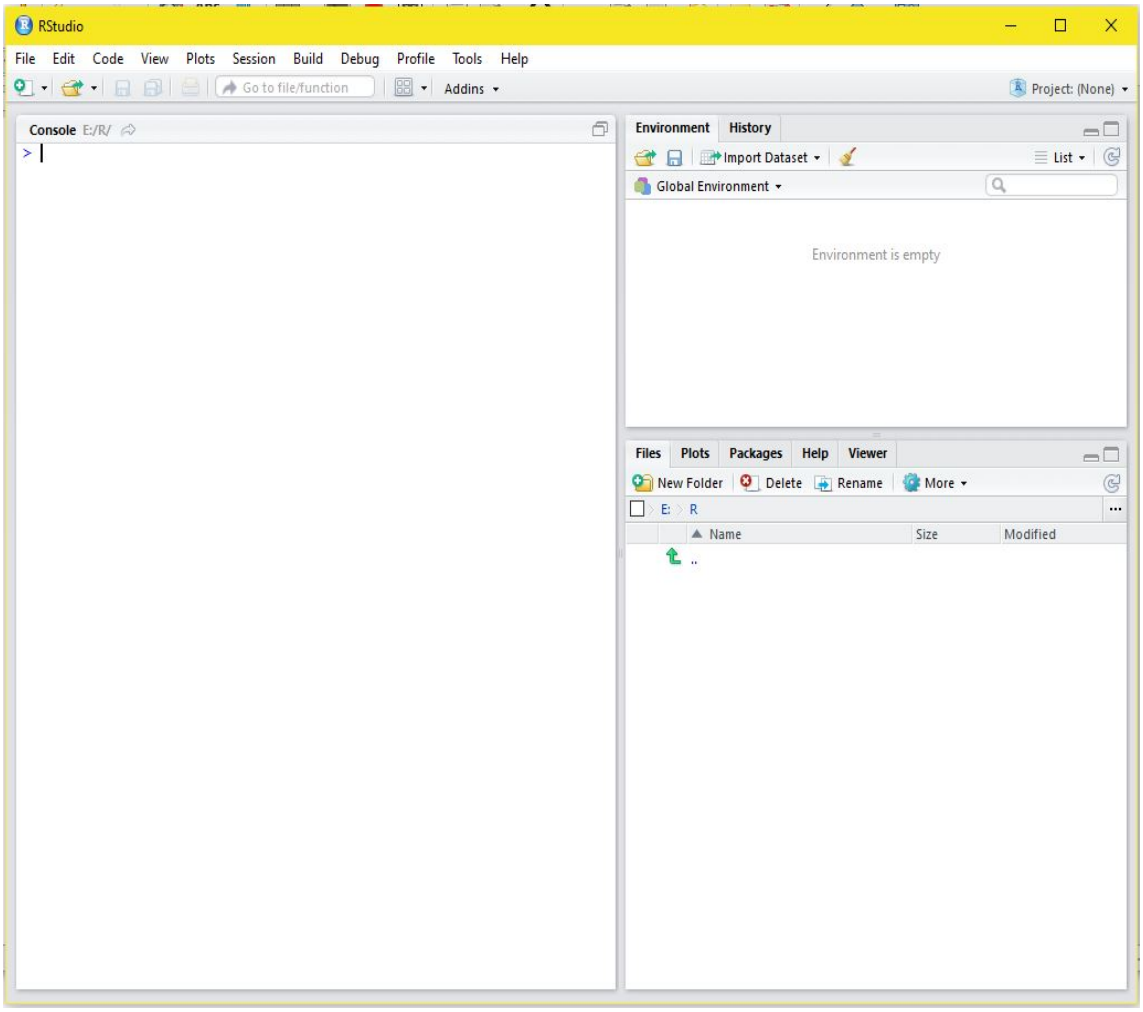
Õppematerjal on siinkohal esitatud ilma tiitellehe, sisukorra ja eessõnata. Vorming võib olla teistsuguse leheküljelise ehituse tõttu eraldiseisva versiooni omast märkimisväärselt erinev.

1 R-lühikursus

Järgnevad peatükid demonstreerivad R-keele lihtsamaid funktsioone, ja nende kasutamist lihtsate statistikaülesannete lahendamiseks. Kui teie arvutis ei ole veel R-keele põhitarkvara, või RStudio laiendust, saate need alla laadida vastavalt lehekülgedelt http://ftp.eenet.ee/pub/cran/index.html ja https://www.rstudio.com/products/rstudio/download2/ .

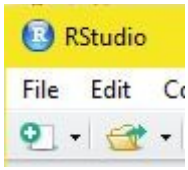
1.1 Kasutajaliides ja lihtsamad toimingud

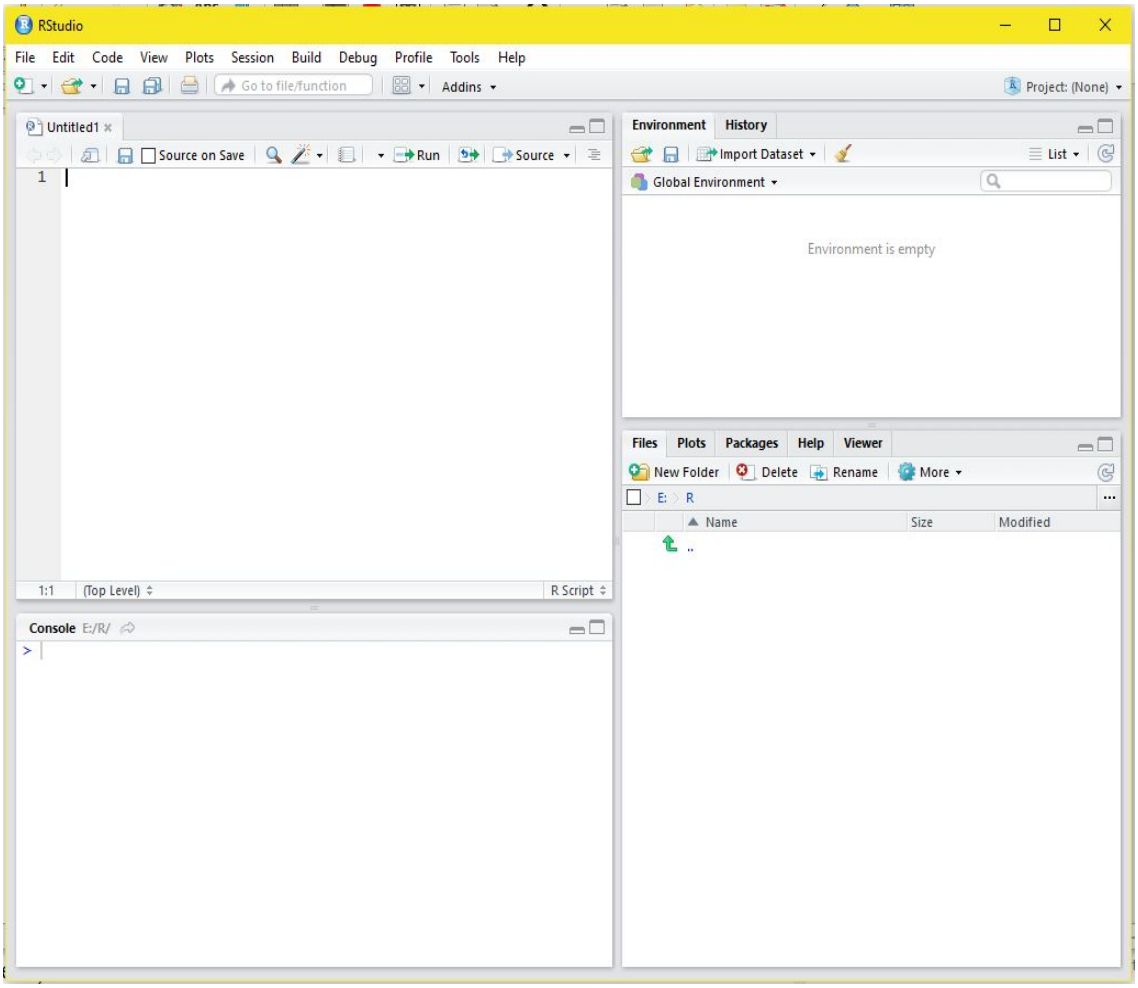
R-keelt saab kasutada nii käsurealt, kui ka läbi sellega kaasasoleva kasutajaliidese. Laienduspakett „RStudio" teeb aga mõned toimingud märgatavalt lihtsamaks, nii et selle lühikursuse näidetes kasutatakse seda. RStudio kasutajaliides näeb välja selline.



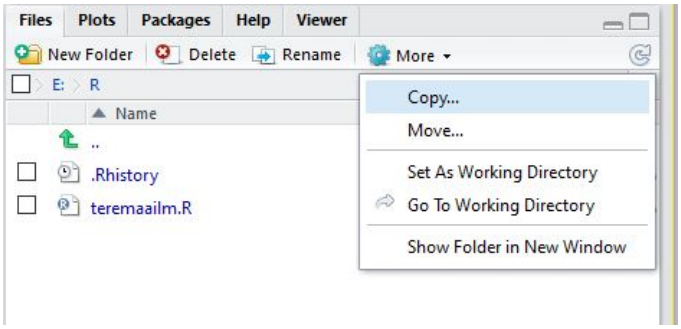
Vasakul asub R-keele konsool. Tehniliselt saab kõik vajaliku ära teha lihtsalt sinna

käsitsi üksteise järel käske sisestades. Praktiliselt aga on tihti mugavam käsud eraldi tekstifaili üles kirjutada ning siis sealt käivitada, ehk teisisõnu: kirjutada programm. Nii on lihtne käskude jada korduvalt kasutada või ümber töödelda. RStudios saab uue programmifaili luua, kui vajutada üleval vasakus nurgas rohelise plussmärgiga nupule ja valida sealt „R Script”.





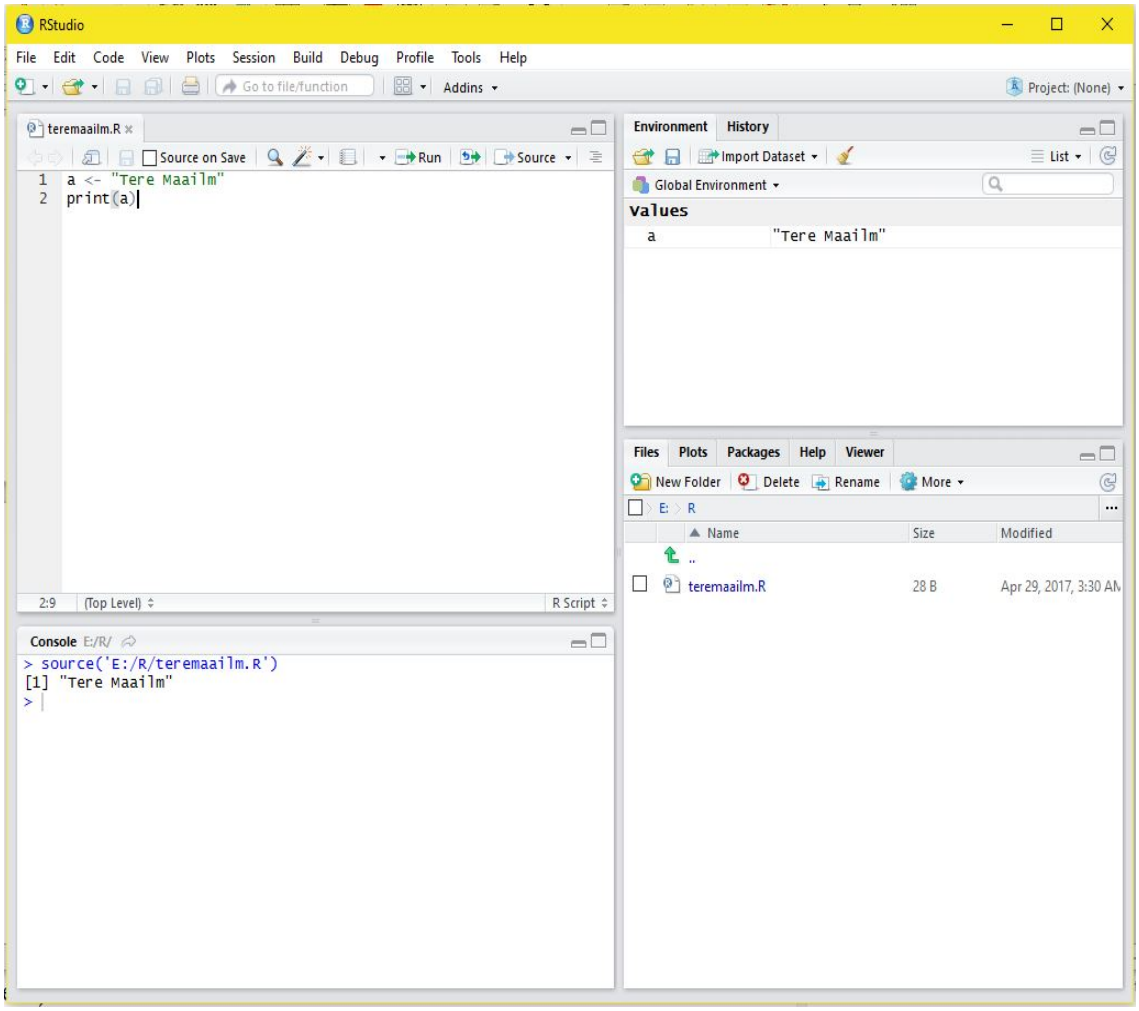
Uus programmifail salvestatakse hetkel kasutuselolevasse töökausta. Seda kausta saab all paremal aknas näha, kui vajutada „More” ja sealt „Go to working directory”. Töökausta saab vahetada, kui vajutada samas aknas „...” nupule, navigeerida avanevas menüüs soovitud kausta, seejärel vajutada „More” ning „Set as working directory”.



Alustame lihtsa „Tere Maailm” programmiga. Antud programm annab muutujale a väärtuseks teksti „Tere Maailm”, ning seejärel trükib selle muutuja väärtuse konsoolis välja.

```
a <- "Tere Maailm"
print(a)
```

R-keele konsoolis saab programmifaili käivitada käsuga *source(„faili nimi”)*. RStudio võimaldab selle automaatselt genereerida nupuvajutusega antud programmifaili juures. Kopeerige üleval antud koodinäide programmifaili, salvestage muudatused disketiikoonile vajutades, ning käivitage programm vajutades nupule „Source”. Konsooli peaks ilmuma automaatselt genereeritud *source()* käsk, ning tekst „Tere Maailm”.



Nagu ehk märkasite, peab üleval paremal asuv aken arvestust kasutusel olevatest muutujatest.

Veel üks lihtne näidisprogramm: Teeme mõned lihtsamad arvutused. Selles programmis on sees ka kommentaarid: #-märgiga algavad read. Neid võib selguse huvides programmifailidesse lisada. (Arvuti ignoreerib neid #-märgiga algavaid tekstiridu täielikult.)

Sisend:

```
a <- 4
b <- 6
c <- 2
#Trükib välja a, b ja c summa
print(a+b+c)
#Trükib välja a ja b korrutise
print(a*b)
#Trükib välja b ja c jagatise
print(b/c)
```

Väljund:

```
[1] 12
[1] 24
[1] 3
```

1.2 Andmete sisestamine

Selleks, et teha tähenduslikke statistilisi arvutusi, on kõigepealt vaja andmeid. Kasutame siinkohal näidete jaoks väljamõeldud spordivõistluste tulemuste tabelit.

Number	Nimi	Sugu	Kõrgushüpe	Kaugushüpe
1	Mart	M	2	7,01
2	Albert	M	1,89	4,32
3	Tiina	N	1,8	5,7
4	Maali	N	1,52	6,8
5	Joosep	M	1,4	4,4
6	Anna	N	1,66	3,97

Eelmises punktis tegime juba tutvust muutujatega. Kuna R-keel on otseselt mõeldud statistika jaoks, on iga muutuja tegelikult nimekiri, kuhu saab kirjutada sisse mitu sama tüüpi väärtust (Numbrit, või teksti). R-is nimetatakse sellist muutujat Vektoriks. Vektorisse saab mitu väärtust kirjutada käsuga kujul *muutuja nimi* <- *c(väärtus1,väärtus2,...)*. Nii võiksime tabeli andmed töötluseks vektoritesse ümber kirjutada näiteks järgmiste käskudega:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
```

Pange tähele, et komakohtadega arvude sisestamisel kasutatakse ameerikapärast kirjutusviisi, kus komakohta märgib punkt, mitte koma. Samuti on muutujanimede puhul parem hoiduda täpitähtede kasutamisest, et vältida võimalikke tekstikodeeringust tulenevaid vigu.

Selleks, et statistilisi funktsioone rakendada, seame loodud vektorid andmeraamistikku (inglise k. *Data frame*). Andmeraamistiku muutuja saab luua käsuga kujul *muutuja nimi <- data.frame(vektor1,vektor2,...)*. Lisame koodi vastava käsu, ja trükime loodud andmeraamistiku konsoolis välja.

Sisend:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
tulemused <-
data.frame(number,nimi,sugu,korgushupe,kaugushupe)
print(tulemused)
```

Väljund:

```
  number   nimi sugu korgushupe kaugushupe
1      1   Mart    M       2.00       7.01
2      2 Albert    M       1.89       4.32
3      3  Tiina    N       1.80       5.70
4      4  Maali    N       1.52       6.80
5      5 Joosep    M       1.40       4.40
6      6   Anna    N       1.66       3.97
```

Nagu näha, on meil nüüd sama tabel R-keele süsteemis olemas. Teeme sellega paar lihtsamat statistilist operatsiooni. Kuvame välja suurima kõrgushüppe tulemuse, ning mediaankeskmise kaugushüppe tulemuse. Selleks kasutame vastavalt käske kujul *muutuja nimi <- max(andmeraamistiku nimi$vektori nimi)*, ning *muutuja nimi <- median(andmeraamistiku nimi$vektori nimi)*.

Sisend:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
tulemused <-
data.frame(number,nimi,sugu,korgushupe,kaugushupe)
korgusmax <- max(tulemused$korgushupe)
```

```
kaugusmed <- median(tulemused$kaugushupe)
print("Parim kõrgushüppe tulemus oli")
print(korgusmax)
print("ning kaugushüppe mediaankeskmine tulemus oli")
print(kaugusmed)
```

Väljund:

```
[1] "Parim kõrgushüppe tulemus oli"
[1] 2
[1] "ning kaugushüppe mediaankeskmine tulemus oli"
[1] 5.05
```

Võitja nime teadasaamiseks peame tabelist võitja sissekande välja filtreerima. Saame seda teha *subset()* käsuga, mis loob filtri järgi vana andmeraami tingimustele vastavatest sissekannetest uue andmeraami. Käsku kasutame kujul *uue andmeraami nimi <- subset(originaalse andmeraami nimi, tingimused).*

Sisend:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
tulemused <-
data.frame(number,nimi,sugu,korgushupe,kaugushupe)
korgusmax <- max(tulemused$korgushupe)
kaugusmed <- median(tulemused$kaugushupe)
korgusvoitja <- subset(tulemused, korgushupe ==
korgusmax)
print("Kõrgushüppes võitis")
print(korgusvoitja$nimi)
print("tulemusega")
print(korgusmax)
```

Väljund:

```
[1] "Kõrgushüppes võitis"
[1] Mart
[1] "tulemusega"
[1] 2
```

Subset() käsu abil saab teha ka palju muud kasulikku. Näiteks võime tabeli jagada eraldi naiste ja meeste tulemusteks, kui filtreerime soo järgi. Arvutame meeste ja naiste kaugushüppe kalkuleeritud keskmised tulemused *mean()* käsuga. Saame teada

võistelnud naiste ja meeste arvu *nrow()* käsuga (mis loeb kokku andmeraamis olevate ridade arvu).

Sisend:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
tulemused <-
data.frame(number,nimi,sugu,korgushupe,kaugushupe)
naised <- subset(tulemused, sugu == "N")
mehed <- subset(tulemused, sugu == "M")
naistearv <- nrow(naised)
meestearv <- nrow(mehed)
meestekaugus <- mean(mehed$kaugushupe)
naistekaugus <- mean(naised$kaugushupe)
print("Osa võttis")
print(naistearv)
print("naist, ja")
print(meestearv)
print("meest")
print("Meeste kaugushüppe keskmine tulemus oli")
print(meestekaugus)
print("Naiste kaugushüppe keskmine tulemus oli")
print(naistekaugus)
```

Väljund:

```
[1] "Osa võttis"
[1] 3
[1] "naist, ja"
[1] 3
[1] "meest"
[1] "Meeste kaugushüppe keskmine tulemus oli"
[1] 5.243333
[1] "Naiste kaugushüppe keskmine tulemus oli"
[1] 5.49
```

Veel ühe näitena võime filtreerida välja kõik sportlased, kelle kõrgushüppe tulemus oli vähemalt 1.7m.

Sisend:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
tulemused <-
data.frame(number,nimi,sugu,korgushupe,kaugushupe)
korgusparimad <- subset(tulemused, korgushupe >= 1.7)
print(korgusparimad)
```

Väljund:

```
  number   nimi sugu korgushupe kaugushupe
1      1   Mart    M       2.00       7.01
2      2 Albert    M       1.89       4.32
3      3  Tiina    N       1.80       5.70
```

Subset() käsule saab anda ka mitu tingimust, kui need eraldada & sümbolitega. Loome tabeli kõigist naissportlastest, kelle kõrgushüppe tulemus oli vähemalt 1.7m.

Sisend:

```
number <- c(1,2,3,4,5,6)
nimi <-
c("Mart","Albert","Tiina","Maali","Joosep","Anna")
sugu <- c("M","M","N","N","M","N")
korgushupe <- c(2,1.89,1.8,1.52,1.4,1.66)
kaugushupe <- c(7.01,4.32,5.7,6.8,4.4,3.97)
tulemused <-
data.frame(number,nimi,sugu,korgushupe,kaugushupe)
korgus_parimad_naised <- subset(tulemused, korgushupe
>= 1.7 & sugu == "N")
print(korgus_parimad_naised)
```

Väljund:

```
  number  nimi sugu korgushupe kaugushupe
3      3 Tiina    N        1.8        5.7
```

Harjutused:

- Looge näidete põhjal programm, mis trükib välja kaugushüppe võitja nime
- Looge näidete põhjal programm, mis trükib välja kaugushüppes alla keskmise tulemuse saavutanud meeste tabeli

1.3 Andmete lugemine csv failist

Eelmises punktis kirjutasime andmetabeli R-keele süsteemi ümber käsitsi. Suurte või muutuvate andmehulkade puhul ei ole selline lähenemine praktiline. Parem on andmed sisse lugeda failist. Demonstreerime seda protsessi esmalt väga lihtsa struktuuriga andmefaili näitel: .csv failiga. Nagu nimi viitab, sisaldab csv, ehk *comma separated variables* (komaga eraldatud muutujad) fail andmeridu, mis on loogiliselt jaotatud komakohtade abil. Kasutame näiteks pikendatud spordivõistluste tulemuste nimekirja. Looge oma töökausta uus tekstidokument, kopeerige sinna all asuvad andmed, ning salvestage fail nimega andmed.csv .

```
number,nimi,sugu,korgushupe,kaugushupe
1,Mart,M,2.00,7.01
2,Albert,M,1.89,4.32
3,Tiina,N,1.80,5.70
4,Maali,N,1.52,6.80
5,Joosep,M,1.40,4.40
6,Anna,N,1.66,3.97
7,Taavi,M,1.54,5.00
8,Sulev,M,1.99,7.11
9,Mari,N,1.60,4.41
10,Annika,N,1.97,7.00
```

Sellise faili saab sisse lugeda *csv.read()* käsuga, kujul *andmeraamistiku nimi* <- *csv.read(„faili nimi”)*, eeldusel et loetav fail asub töökaustas.

Sisend:

```
tulemused <- read.csv("andmed.csv")
print(tulemused)
```

Väljund:

```
   number   nimi sugu korgushupe kaugushupe
1       1   Mart    M       2.00       7.01
2       2 Albert    M       1.89       4.32
3       3  Tiina    N       1.80       5.70
4       4  Maali    N       1.52       6.80
5       5 Joosep    M       1.40       4.40
6       6   Anna    N       1.66       3.97
7       7  Taavi    M       1.54       5.00
8       8  Sulev    M       1.99       7.11
9       9   Mari    N       1.60       4.41
10     10 Annika    N       1.97       7.00
```

Eeldusel, et csv failis olevad andmed on korrektselt vormistatud, saame nii üherealise käsu abil sisse lugeda terve hulga andmeid.

R on võimeline ka vastupidi andmeid faili kirjutama. Oletame näiteks, et soovime luua faili ainult kolme parima kaugushüppaja tulemustega. Appi tulevad *order()* ja *rev()* käsud. Kui neid vektoritele rakendada, vastvalt kujul *uus muutuja* <- *order(vektori nimi)* ning *uus muutuja* <- *rev(vektori nimi)*, loovad nad vastavalt vähimast-suurimani järjestatud vektori, ning ümberpööratud järjestusega vektori. Neid üksteise järel kasutades saame siis suurimast-vähimani järjestatud vektori. Andmeraamistikule saame neid rakendada järgmisel kujul:

Sisend:

```
tulemused <- read.csv("andmed.csv")
order_tulemused <-
tulemused[rev(order(tulemused$kaugushupe)),]
print(order_tulemused)
```

Väljund:

```
   number   nimi sugu korgushupe kaugushupe
8       8  Sulev    M       1.99       7.11
1       1   Mart    M       2.00       7.01
10     10 Annika    N       1.97       7.00
4       4  Maali    N       1.52       6.80
3       3  Tiina    N       1.80       5.70
7       7  Taavi    M       1.54       5.00
9       9   Mari    N       1.60       4.41
5       5 Joosep    M       1.40       4.40
2       2 Albert    M       1.89       4.32
6       6   Anna    N       1.66       3.97
```

Kolm parimat saame eraldada lihtsa ümberkirjutuse abil, mis avaldub kujul *uue andmeraamistiku nimi* <- *originaalse andmeraamistiku nimi[soovitud read, soovitud tulbad]*. Kuna soovime esimest kolme rida, kirjutame soovitud ridade kohale 1:3 (1 kuni 3), ning kuna soovime kõik tulbad alles jätta, jätame tulpade koha tühjaks.

Sisend:

```
tulemused <- read.csv("andmed.csv")
order_tulemused <-
tulemused[rev(order(tulemused$kaugushupe)),]
top3 <- order_tulemused[1:3,]
print(top3)
```

Väljund:

```
   number   nimi sugu korgushupe kaugushupe
8       8  Sulev    M       1.99       7.11
1       1   Mart    M       2.00       7.01
10     10 Annika    N       1.97       7.00
```

Uue csv faili saame kirjutada lihtsa *write.csv(andmeraami nimi, „faili nimi”, sätted)* käsu abil. Sätete kohal on võimalik täpsustada teatud kirjutusnüansse. Näiteks, kui me ei soovi faili kirjutada R-i poolt automaatselt lisatud reanumbreid, võime sinna lisada argumendi *row.names = FALSE*.

Sisend:

```
tulemused <- read.csv("andmed.csv")
order_tulemused <-
tulemused[rev(order(tulemused$kaugushupe)),]
top3 <- order_tulemused[1:3,]
write.csv(top3,"kaugusparimad.csv",row.names = FALSE)
```

Loodud fail, kaugusparimad.csv:

```
"number","nimi","sugu","korgushupe","kaugushupe"
8,"Sulev","M",1.99,7.11
1,"Mart","M",2,7.01
10,"Annika","N",1.97,7
```

Harjutused:

- Looge näidete põhjal programm, mis salvestab naiste ja meeste tulemused ümber eraldi failidesse
- Looge näidete põhjal programm, mis salvestab faili „korgusparimad.csv” kolme parima kõrgushüppe tulemusega inimese andmed

1.4 Andmete lugemine Exceli failist

Microsoft Exceli tabelid (vorming .xlsx) on väga populaarne viis kõiksugu struktuursete andmete hoiustamiseks. R-i põhiversioonis ei ole nendega ümber käimiseks tööriistu, küll aga on olemas mitu seleks mõeldud lisapaketti. Kasutame siin näiteks „xlsx” paketti. Kui teie arvutil on olemas internetiühendus, on lisapakettide hankimine väga lihtne. Tuleb lihtsalt R-i konsooli sisestada käsk kujul *install.packages(„soovitud paketi nimi”)*. Kui selle nimega pakett on olemas, laetakse see (ja kõik selle kasutamiseks vajalikud teised paketid) automaatselt alla.

Sisend:

```
install.packages("xlsx")
```

Väljund:

```
also installing the dependencies ‘rJava’, ‘xlsxjars’
trying URL
'https://cran.rstudio.com/bin/windows/contrib/3.3/rJa
va_0.9-8.zip'
Content type 'application/zip' length 713967 bytes
(697 KB)
downloaded 697 KB


trying URL
'https://cran.rstudio.com/bin/windows/contrib/3.3/xls
xjars_0.6.1.zip'
Content type 'application/zip' length 9485184 bytes
(9.0 MB)
downloaded 9.0 MB
trying URL
'https://cran.rstudio.com/bin/windows/contrib/3.3/xls
x_0.5.7.zip'
Content type 'application/zip' length 401439 bytes
(392 KB)
downloaded 392 KB
package ‘rJava’ successfully unpacked and MD5 sums
checked
package ‘xlsxjars’ successfully unpacked and MD5 sums
checked
package ‘xlsx’ successfully unpacked and MD5 sums
checked
```

Kontrollime, et pakett töötab. Selleks aktiveerime paketi *library()* käsuga.

```
library("xlsx")
```

Siinkohal võite saada järgneva veateate:

```
Loading required package: rJava
Error : .onLoad failed in loadNamespace() for
'rJava', details:
  call: fun(libname, pkgname)
  error: JAVA_HOME cannot be determined from the
Registry
Error: package ‘rJava’ could not be loaded
```

See tähendab, et R-keele keskkond ei leidnud teie R-i versioonile vastavat Java raamistiku versiooni. Viga saab parandada, kui laadida alla Java uusim versioon leheküljelt https://www.java.com/en/download/manual.jsp . Olenevalt kasutusel olevast R-i versioonist on jätkamiseks vaja kas 32 või 64 bitist Java raamistikku. Kindluse mõttes võite paigaldada mõlemad.

Pärast paigalduse lõppu proovige uuesti *library()* käsku. Pakett on töökorras, kui saate järgneva vastuse ilma veateadeteta.

```
Loading required package: rJava
Loading required package: xlsxjars
```

Paneme valmis exceli faili, mille peal lugemist testime. Looge töökausta fail nimega autod.xlsx . Looge kaks lehekülge, pange esimesele lehele nimeks „Autod”, ning teisele „Toonid”. Kopeerige andmed vastavatele lehekülgedele.

Autod:

Number	Mudel	Mark	Tootmisaasta	Seisukord
000AAA	Escort	Ford	1994	Rahuldav
110BCX	Civic	Honda	2000	Hea
121ABR	Passat	Volkswagen	2003	Rahuldav
209SET	Escort	Ford	1993	Halb
101LOL	Legend	Honda	2001	Suurepärane
022ACG	Sierra	Ford	1990	Hea
200ACC	Focus	Ford	2006	Suurepärane
061LNI	FRV	Honda	2006	Rahuldav
245VVE	Wrangler	Jeep	2001	Hea
114ABI	Logan	Dacia	1998	Rahuldav
099GBT	Impreza	Subaru	2004	Hea
081SKT	Lancer	Mitsubishi	2008	Halb
565MET	CRV	Honda	2001	Hea
666RPG	Mustang	Ford	2010	Suurepärane

Toonid:

Number	Toon
000AAA	Sinine
110BCX	Lilla
121ABR	Roosa
209SET	Must
101LOL	Oraanž
022ACG	Sinine
200ACC	Must
061LNI	Kollane
245VVE	Roheline
114ABI	Sinine
099GBT	Must
081SKT	Punane
565MET	Punane
666RPG	Must

Pärast pakettide paigaldamist on exceli faili sisselugemine väga sarnane csv faili lugemisele. Kasutame *xlsx.read()* käsku, kujul *andmeraamistiku nimi <- xlsx.read(„faili nimi”, sheetIndex = lehe number)*.

Sisend:

```
library("xlsx")
autod <- read.xlsx("autod.xlsx", sheetIndex = 1)
toonid <- read.xlsx("autod.xlsx", sheetIndex = 2)
print(autod)
```

Väljund:

```
   Number    Mudel          Mark      Tootmisaasta
Seisukord
1  000AAA   Escort         Ford          1994
Rahuldav
2  110BCX    Civic       Honda          2000
Hea
3  121ABR   Passat Volkswagen          2003
Rahuldav
4  209SET   Escort         Ford          1993
Halb
5  101LOL   Legend       Honda          2001
Suurepärane
6  022ACG   Sierra         Ford          1990
Hea
7  200ACC    Focus         Ford          2006
Suurepärane
8  061LNI      FRV       Honda          2006
Rahuldav
9  245VVE Wrangler         Jeep          2001
Hea
10 114ABI    Logan       Dacia          1998
Rahuldav
11 099GBT  Impreza      Subaru          2004
Hea
12 081SKT   Lancer Mitsubishi          2008
Halb
13 565MET      CRV       Honda          2001
Hea
14 666RPG  Mustang         Ford          2010
Suurepärane
```

Exceli faili kirjutamine on samuti csv failide sarnane. Filtreerime näiteks välja fordide andmed, ja kirjutame need uude faili *write.xlsx()* käsuga, kujul *write.xlsx(andmeraami nimi, „faili nimi", sheetName = „lehekülje nimi", sätted)*.

Sisend:

```
library("xlsx")
autod <- read.xlsx("autod.xlsx", sheetIndex = 1)
toonid <- read.xlsx("autod.xlsx", sheetIndex = 2)
fordid <- subset(autod, autod$Mark == "Ford")
write.xlsx(fordid, "fordid.xlsx", sheetName =
"autod", row.names = FALSE)
```

Väljund:

autod:

Number	Mudel	Mark	Tootmisaasta	Seisukord
000AAA	Escort	Ford	1994	Rahuldav
209SET	Escort	Ford	1993	Halb
022ACG	Sierra	Ford	1990	Hea
200ACC	Focus	Ford	2006	Suurepärane
666RPG	Mustang	Ford	2010	Suurepärane

Harjutused:

- Kirjutage programm, mis salvestab uude faili kõigi sinist värvi autode numbrimärgid.

1.5 Tabelid

Statistika ilmestamisel on võimsateks tööriistadeks risttabelid. Demonstreerime nende kasutamist poeketi siseküsitluse tulemuste peal. Salvestage antud andmed töökausta csv failis nimega uuring.csv .

uuring.csv:

```
20-30,Tallinn,Kassa,Ei,Jah
20-30,Tallinn,Kassa,Jah,Jah
20-30,Tallinn,Kassa,Ei,Ei
20-30,Tallinn,Ladu,Jah,Jah
31-40,Tallinn,Ladu,Ei,Jah
31-40,Tallinn,Ladu,Ei,Jah
41-50,Tallinn,Kassa,Ei,Jah
41-50,Tallinn,Ladu,Ei,Ei
31-40,Tallinn,Kontor,Jah,Ei
>50,Tallinn,Kontor,Ei,Jah
20-30,Tartu,Kassa,Ei,Ei
20-30,Tallinn,Kassa,Ei,Ei
20-30,Tallinn,Ladu,Jah,Jah
31-40,Paide,Ladu,Ei,Jah
31-40,Tallinn,Ladu,Ei,Jah
41-50,Tartu,Kassa,Ei,Jah
31-40,Tartu,Ladu,Ei,Ei
31-40,Tartu,Kontor,Jah,Ei
<20,Paide,Kassa,Ei,Jah
20-30,Paide,Kassa,Ei,Jah
31-40,Paide,Ladu,Jah,Jah
41-50,Paide,Kassa,Jah,Ei
21-30,Tartu,Kontor,Ei,Ei
>50,Paide,Kontor,Jah,Jah
41-50,Tartu,Kassa,Ei,Jah
31-40,Tallinn,Kassa,Ei,Ei
20-30,Tallinn,Ladu,Jah,Ei
<20,Tartu,Kassa,Ei,Jah
41-50,Tartu,Ladu,Ei,Ei
41-50,Paide,Kontor,Ei,Ei
20-30,Tallinn,Kontor,Ei,Jah
41-50,Tartu,Kassa,Ei,Jah
31-40,Tartu,Ladu,Ei,Jah
31-40,Tartu,Kontor,Ei,Ei
```

Anonüümses küsitluses küsiti poeketi töötajatelt nende vanusegruppi, töökohaks oleva esinduse asukohta, töösektorit (laotöötaja, kassapidaja, või kontoritöötaja), ning kas nad tarbivad tubaka ja/või alkoholitooteid. (Andmed on fiktiivsed)

Sellistes andmetes seoste märkamiseks on hea kasutada risttabelit. Risttabeli saame luua käsuga *table()*, kujul *tabeli muutuja nimi <- table(andmeraamistiku esimese tulba nimi, andmeraamistiku teise tulba nimi)*. Loome näiteks risttabelid tubaka ja alkoholi tarbimise vahelisest seosest, ning vanusegruppide ning tubaka tarbimise vahelisest seosest.

Sisend:

```
andmed <- read.csv("uuring.csv")
suits_alko <- table(andmed$tubakas,andmed$alkohol)
vanus_suits <- table(andmed$vanus,andmed$tubakas)
print(suits_alko)
print(vanus_suits)
```

Väljund:

```
     Ei Jah
Ei   10  15
Jah   4   5

       Ei Jah
<20     2   0
>50     1   1
20-30   6   4
21-30   1   0
31-40   8   3
41-50   7   1
```

Nagu näha, asetub esimene valitud tulp tabeli vertikaalsele, ning teine tabeli horisontaalsele teljele. Tabeli järgi on selgelt näha, et enamik vastanutest (15 inimest) tarbib alkoholi, aga mitte tubakat. Sellise risttabeli võib muuta protsendipõhiseks, kui kasutada käsku *prop.table()*, kujul *uue tabeli nimi <- prop.table(algse tabeli nimi)*.

Sisend:

```
andmed <- read.csv("uuring.csv")
suits_alko <- table(andmed$tubakas,andmed$alkohol)
suits_alko_prot <- prop.table(suits_alko)
print(suits_alko_prot)
```

Väljund:

```
            Ei       Jah
Ei  0.2941176 0.4411765
Jah 0.1176471 0.1470588
```

Selle tabeli järgi võime täpselt öelda, et 44% vastanutest tarbib alkoholi, aga mitte tubakatooteid.

Prop.table() käsuga võime luua ka rea või tulba järgi protsentarvestusega tabeleid, kui lisame pärast algse tabeli nime kas ühe(ridade järgi protsentide jaoks), või kahe(tulpade järgi protsentide jaoks), kujul *uue tabeli nimi <- prop.table(algse tabeli nimi, 1 / 2).* Teeme näiteks vanuse ja suitsetamise vahelise seose risttabelist ridade(vanusrühmade) järgi protsentarvestuse.

Sisend:

```
andmed <- read.csv("uuring.csv")
vanus_suits <- table(andmed$vanus,andmed$tubakas)
vanus_suits_prot <- prop.table(vanus_suits, 1)
print(vanus_suits_prot)
```

Väljund:

```
              Ei       Jah
<20   1.0000000 0.0000000
>50   0.5000000 0.5000000
20-30 0.6000000 0.4000000
21-30 1.0000000 0.0000000
31-40 0.7272727 0.2727273
41-50 0.8750000 0.1250000
```

Sellest tabelist on näiteks näha, et antud firmas on protsentuaalselt kõige rohkem suitsetajaid üle 50 aasta vanuste töötajate seas.

Sagedustabeli saab luua tavalise risttabeli põhjal, *margin.table()* käsuga, kujul *uue tabeli nimi <- margin.table(algse tabeli nimi, 1 / 2).* 1 puhul arvestatakse risttabeli vertikaalse telje elementide sagedus, 2 puhul horisontaalse telje elementide oma. (Sagedusprotsendi saab kätte, kui rakendada loodud tabelile omakorda *prop.table()* käsku.) Loome näiteks vanuse ja tubakatarbimise risttabeli põhjal töötajate vanuste sagedustabeli. Siiani on meie protsendipõhised tabelid olnud skaalas 100% = 1. Protsendipõhise tabeli saab panna tuttavamasse 100% = 100 skaalasse, kui korrutada tabeli muutuja sajaga.

Sisend:

```
andmed <- read.csv("uuring.csv")
vanus_suits <- table(andmed$vanus,andmed$tubakas)
vanus_suits_sag <- margin.table(vanus_suits, 1)
vanus_suits_sag_prot <-
prop.table(vanus_suits_sag)*100
print(vanus_suits_sag)
print(vanus_suits_sag_prot)
```

Väljund:

```
  <20   >50 20-30 21-30 31-40 41-50
    2     2    10     1    11     8

     <20       >50     20-30     21-30     31-40
41-50
 5.882353  5.882353 29.411765  2.941176 32.352941
23.529412
```

Loodud tabeli järgi on näiteks näha, et 32% protsenti vastanud töötajatest on vanusevahemikus 31-40.

Harjutused:

- Looge näidete põhjal programm, mis koostab tabeli, kust saab näha suitsetajate protsentuaalset osakaalu igas esinduses (Tallinnas, Tartus, Paides).
- Looge näidete põhjal programm, mis koostab tabeli, kust saab näha töötajate arvu igas töösektoris (Ladu, Kassa, Kontor).

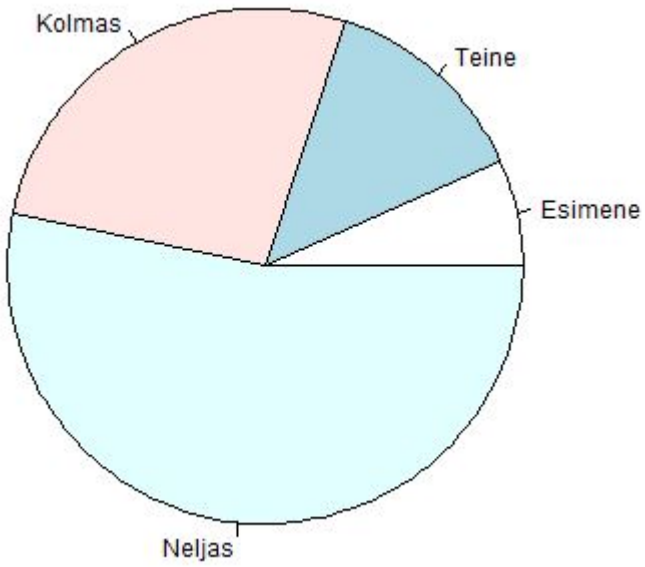
1.6 Diagrammid

Statistika visualiseerimiseks on väga kasulikud diagrammid. Diagramme ei trükita R konsoolis välja, vaid nad salvestatakse töökausta .png vormingus pildifailidena. Kõigepealt tuleb määrata piltide salvestamise kord. Selleks kasutame siin käsku *png()*, kujul *png(file = „pildifaili nimi.png”)*. Seejärel defineerime diagrammi enda. Näiteks sektordiagrammi saame luua käsuga *pie()*, kujul *pie(andmeid sisaldav vektor,sektorite nimesid sisaldav vektor)*. Sellele käsule saab lisada ka teisi sätteid, näiteks et muuta diagrammi kuju ja värve. Nende lisaargumentide kohta võite soovi korral lugeda töö teisest poolest. Kui muu on valmis, saab käsuga *dev.off()* valikud kinnitada, ning diagrammi ära salvestada. Koostame näiteks lihtsa sektordiagrammi.

Sisend:

```
andmed <- c(2,4,8,16)
sildid <- c("Esimene","Teine","Kolmas","Neljas")
png(file = "sektordiagramm.png")
pie(andmed,sildid)
dev.off()
```

Väljund:

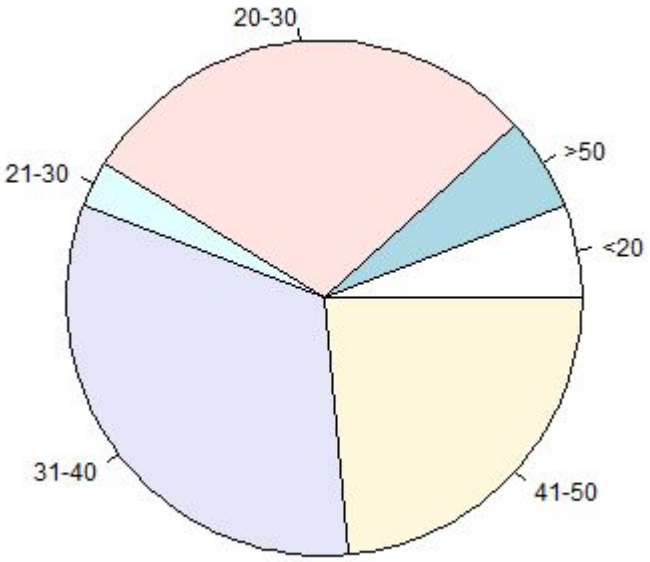


Praktilisema näitena võime teha sektordiagrammi eelmises punktis loodud vanuste sagedustabelist. Selleks lahutame tabeli read andmete ja siltide vektoriteks, *name()* (mis eraldab tabelist tulpade nimed) ja *as.vector()* (mis muudab tabeli rea arvuliseks vektoriks) käskude abil.

Sisend:

```
andmed <- read.csv("uuring.csv")
vanus_suits <- table(andmed$vanus,andmed$tubakas)
vanus_suits_sag <- margin.table(vanus_suits, 1)
vanus_suits_sag_prot <- prop.table(vanus_suits_sag)
graafi_andmed = as.vector(vanus_suits_sag_prot)
graafi_sildid = names(vanus_suits_sag_prot)
png(file = "vanused.png")
pie(graafi_andmed,graafi_sildid)
dev.off()
```

Väljund:



Harjutus:

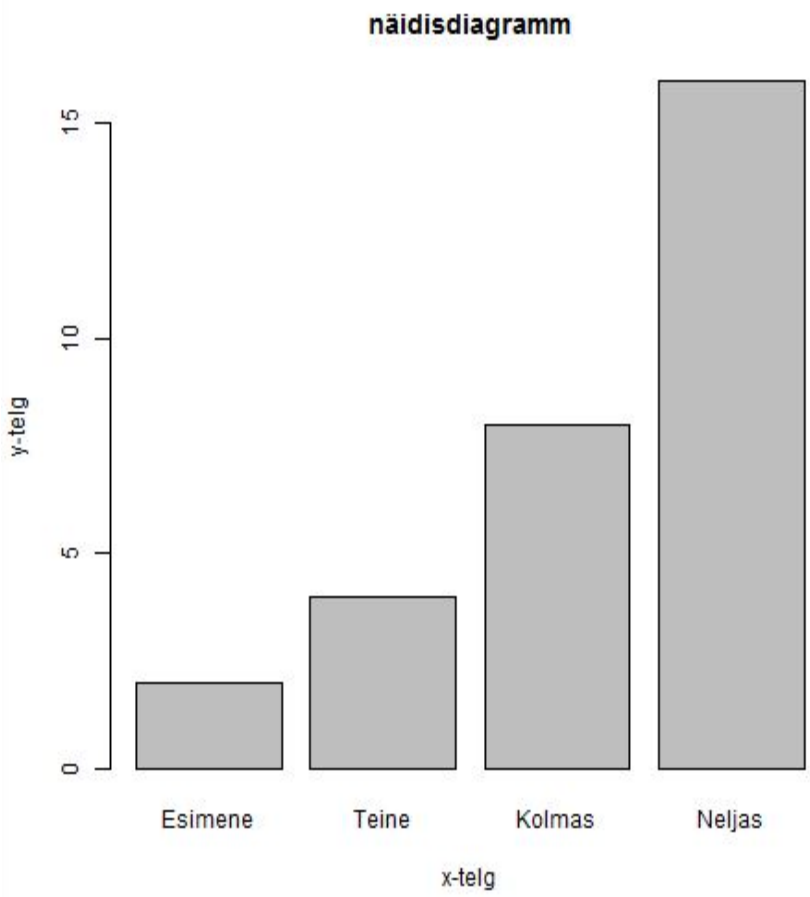
- *Leidke eelmiste näidetes kasutatud andmete seast veel mõni andmekogum, millest saaks hea sektordiagrammi, ning looge programm, mis selle väljastab,

Tulpdiagrammide loomine on üpris sarnane protsess. *Pie()* asemel kasutame aga *barplot()* käsku, kujul *barplot(andmeid sisaldav vektor, names.arg = tulpade nimesid sisaldav vektor, xlab = „x-telje nimi”,ylab= „y-telje nimi”, main= „diagrammi pealkiri”)*. Koostame lihtsa näite.

Sisend:

```
andmed <- c(2,4,8,16)
sildid <- c("Esimene","Teine","Kolmas","Neljas")
png(file = "tulpdiagramm.png")
barplot(andmed,names.arg=sildid,xlab="x-
telg",ylab="y-telg",main="näidisdiagramm")
dev.off()
```

Väljund:

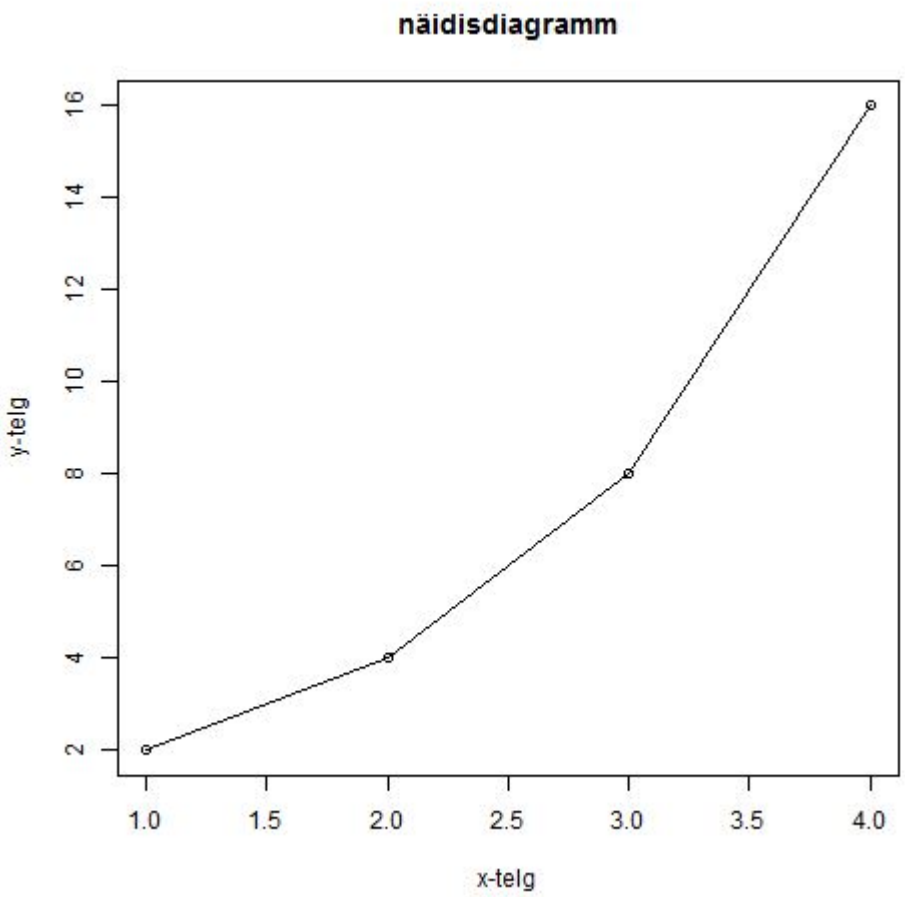


Joondiagrammi saab luua käsuga *plot()*, kujul *plot(andmeid sisaldav vektor, type = „p”/”l”/”o”, xlab = „x-telje nimi”,ylab= „y-telje nimi”, main= „diagrammi pealkiri”). Type* argument määrab ära joondiagrammi tüübi: p joonistab ainult punktid, l joonistab ainult jooned, o joonistab mõlemad. Järjekordne lihtne näide.

Sisend:

```
andmed <- c(2,4,8,16)
png(file = "joondiagramm.png")
plot(andmed,type="o",xlab="x-telg",ylab="y-
telg",main="näidisdiagramm")
dev.off()
```

Väljund:

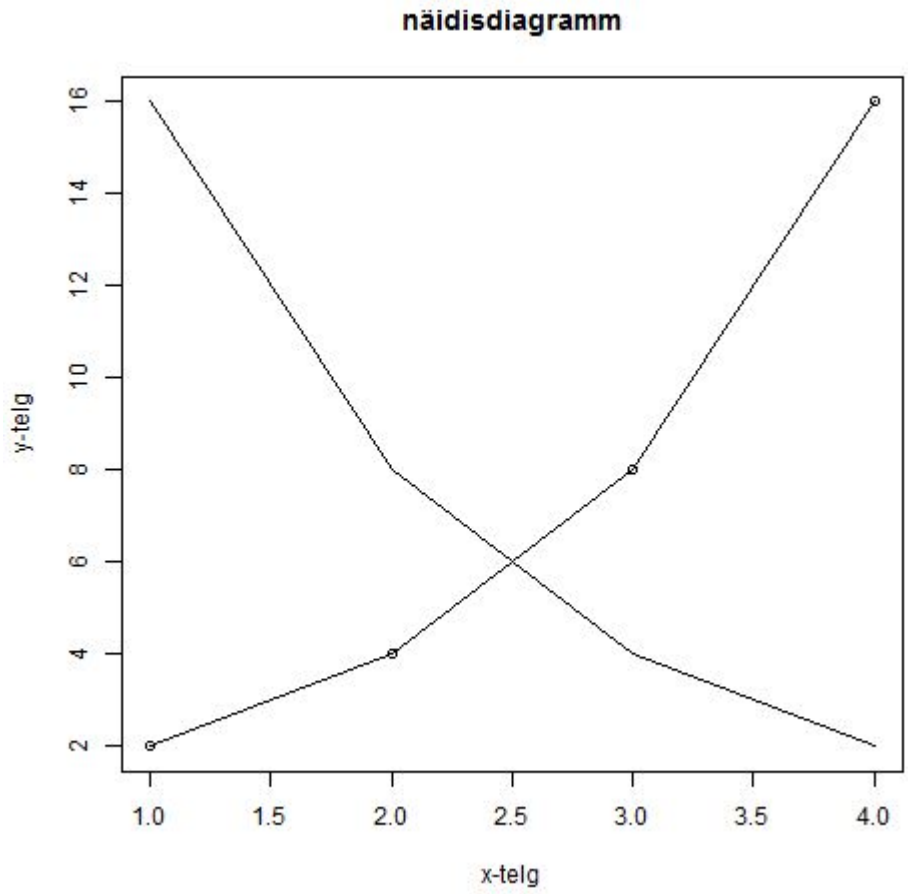


Joondiagrammile saab jooni lisada lines() käsuga, kujul *lines(andmeid sisaldav vektor, type = „p”/”l”/”o”).*

Sisend:

```
andmed <- c(2,4,8,16)
sildid <- c("Esimene","Teine","Kolmas","Neljas")
png(file = "joondiagramm2.png")
plot(andmed,type="o",xlab="x-telg",ylab="y-
telg",main="näidisdiagramm")
andmed2 <- c(16,8,4,2)
lines(andmed2,type="l")
dev.off()
```

Väljund:



Harjutused:

- *Lugege sisse punktis 1.4. näiteks olnud exceli tabel, ja koostage tulpdiagramm erinevat värvi autode arvu kohta.
- **Lugege töö teisest osast punktist 2.7. lisaks diagrammide värvimise kohta, ja värvige igat auto värvi kujutav tulp vastavat värvi.

1.7 Arvnäitajad

Vaatame üle, kuidas saab R-keeles arvutada teatud tavalisi statistilisi arvnäitajaid. Mõnedega neist oleme juba eelnevates näidetes kokku puutunud.

Keskmiste väärtuste seas on tavalisimad arvestused aritmeetiline keskmine, mediaan, ja mood. R-is saab aritmeetilise keskmise (kõigi väärtuste summa jagatud väärtuste arvuga) käsuga *mean()* ja mediaani (keskmisele kõige lähema arvujadas esineva arvu)

käsuga *median()*. Moodi arvutamiseks R-is funktsiooni ei ole, aga saame seda teha käsitsi, matemaatilise tehte abil.

Sisend:

```
a <-
c(1,1,1,3,2,4,2,3,6,7,8,6,7,8,24,8,2,21,2,3,7,5,5,4,3
,32)
keskmine <- mean(a)
mediaan <- median(a)
mood <- unique(a)
mood2 <- mood[which.max(tabulate(match(a, mood)))]
print(keskmine)
print(mediaan)
print(mood2)
```

Väljund:

```
[1] 6.730769
[1] 4.5
[1] 3
```

Arvudejada vähima arvu saab kätte käsuga *min()*, suurima arvu käsuga *max()*, ja vähima ning suurima arvu mõlemad käsuga *range()*. Kõigi arvude summa saab kätte käsuga *sum()*.

Sisend:

```
a <-
c(1,1,1,3,2,4,2,3,6,7,8,6,7,8,24,8,2,21,2,3,7,5,5,4,3
,32)
miinimum <- min(a)
maksimum <- max(a)
vahe <- range(a)
summa <- sum(a)
print(miinimum)
print(maksimum)
print(vahe)
print(summa)
```

Väljund:

```
[1] 1
[1] 32
[1]  1 32
[1] 175
```

Arvude standardhälbe saab arvutada käsuga *sd()*.

Sisend:

```
a <-
c(1,1,1,3,2,4,2,3,6,7,8,6,7,8,24,8,2,21,2,3,7,5,5,4,3
,32)
halve <- sd(a)
print(halve)
```

Väljund:

```
[1] 7.512963
```

Kvartiile saab arvutada käsuga *quantile()*. Kvartiilid on näitajad, kus 0% arvudest on väiksemad, 100% suuremad kui Q0 (tähistatud kui 0%), 25% arvudest on väiksemad, 75% suuremad kui Q1 (tähistatud kui 25%), 50% arvudest on väiksemad, 50% suuremad kui Q2 (tähistatud kui 50%), 25% arvudest on väiksemad, 75% suuremad kui Q3 (tähistatud kui 75%), ja 100% arvudest on väiksemad kui Q4 (tähistatud kui 100%).

Sisend:

```
a <-
c(1,1,1,3,2,4,2,3,6,7,8,6,7,8,24,8,2,21,2,3,7,5,5,4,3
,32)
kvartiilid <- quantile(a)
print(kvartiilid)
```

Väljund:

```
   0%   25%   50%   75%  100%
 1.00  2.25  4.50  7.00 32.00
```

Harjutused:

- *Looge näidete põhjal programm, mis leiab punktis 1.3. kasutatud spordivõistluse tulemuste failist kõrgushüppe tulemuste kvartiilid.

1.8 Korrelatsioonianalüüs

Selleks, et saada teada, kui tugevad on seosed kahe erineva statistilise näitaja vahel, kasutatakse korrelatsioonianalüüse. Korrelatsioonianalüüsi rakendamine kahele andmejadale annab meile korrelatsioonikordaja, mille väärtus varieerub -1 kuni 1. Negatiivne korrelatsioonikordaja tähendab, et kahe tunnuse vahel on negatiivne side (üks suureneb, siis teine kahaneb), positiivne korrelatsioonikordaja tähendab, et kahe tunnuse vahel on positiivne side (üks kasvab, siis teine kasvab). Korrelatsioonikordaja väärtusega 0 näitab, et kahe tunnuse vahel ei ole mingit täheldatavat seost. Üldiselt öeldakse, et seos on tähenduslik, kui korrelatsioonikordaja absoluutväärtus on vähemalt 0,5 , kuid see piir oleneb kontekstist. Üldiselt, mida täpsem on teadus, seda kõrgem on tähenduslikkuse piir. Erinevate iseloomudega näitajate jaoks on kasutusel erinevad analüüsimeetodid.

Üks tavaliselt kasutatav korrelatsioonikordaja on lineaarne korrelatsioonikordaja, teise nimega Pearsoni korrelatsioonikordaja. See kordaja sobib hästi selliste tunnuste analüüsimiseks, milles ei esine väga suurt varieeruvust ega silmapaistvaid erandeid. Pearsoni kordaja saame arvutada, kui rakendame andmeraamistikule käsku *cor()*, kujul *korrelatsioonitabeli nimi <- cor(andmeraamistiku nimi, method=”pearson”)*.

Sisend:

```
rahvaarv <- c(12010,13020,15066,17111,15455,16820)
kuriteod <- c(121,130,151,159,149,153)
andmed <- data.frame(rahvaarv,kuriteod)
p <- cor(andmed, method="pearson")
print(p)
```

Väljund:

```
            rahvaarv  kuriteod
rahvaarv 1.0000000 0.9710207
kuriteod 0.9710207 1.0000000
```

Korrelatsioonikordaja 0.97 viitab, et antud näites on rahvaarvu ja kuritegude arvu vahel väga tugev positiivne korrelatiivne side.

Teine tihti kasutatav korrelatsioonikordaja on Spearmani korrelatsioonikordaja. See sobib hästi astmelise ehitusega andmete (näiteks meeldivushinnangute või jah/ei vastustega küsimuste) analüüsimiseks. Spearmani kordaja saame arvutada sama käsuga, lihtsalt vahetame meetodi sätet. *korrelatsioonitabeli nimi <- cor(andmeraamistiku nimi, method="spearman")*. Analüüsime näiteks tubakatoodete tarbimise ja kakao joomise vahelist seost.

```
tubakas <-
c("Jah","Jah","Ei","Jah","Ei","Ei","Jah","Ei","Jah","
Ei","Ei")
kakao <-
c("Ei","Jah","Ei","Ei","Jah","Ei","Ei","Ei","Jah","Ei
","Jah")
```

Meil on küsitluse tulemused tekstiliste jah/ei vastustena. Korellatsioonikordaja arvutamiseks on meil aga vaja numbreid. Antud vektorid saame numbrilisteks teisendada järgmiste käskudega:

```
tubakas[tubakas == "Jah"] <- 1
tubakas[tubakas == "Ei"] <- 0
tubakas <- as.numeric(tubakas)
kakao[kakao == "Jah"] <- 1
kakao[kakao == "Ei"] <- 0
kakao <- as.numeric(kakao)
```

Nendest saame luua andmeraami, millest annab arvutada korrelatsioonkordaja.

Sisend:

```
tubakas <-
c("Jah","Jah","Ei","Jah","Ei","Ei","Jah","Ei","Jah","
Ei","Ei")
kakao <-
c("Ei","Jah","Ei","Ei","Jah","Ei","Ei","Ei","Jah","Ei
","Jah")
tubakas[tubakas == "Jah"] <- 1
tubakas[tubakas == "Ei"] <- 0
tubakas <- as.numeric(tubakas)
kakao[kakao == "Jah"] <- 1
kakao[kakao == "Ei"] <- 0
kakao <- as.numeric(kakao)
andmed <- data.frame(tubakas,kakao)
p <- cor(andmed, method="spearman")
print(p)
```

Väljund:

```
             tubakas      kakao
tubakas 1.00000000 0.06900656
kakao   0.06900656 1.00000000
```

Nagu näha, on nende kahe tunnuse korrelatsioonikordaja alla 0,1, ning seega on antud uuringu põhjal seos kakao ja tubakatoodete tarbimise vahel põhimõtteliselt olematu.

Harjutused:

- Teostage korrelatsiooniuuring punktis 1.5. kasutatud poeketi siseküsitluse andmefailis avalduvatele tunnustele. Uurige välja, kui tähenduslik on nende andmete järgi seos suitsetamise ja alkoholitarbimise vahel.
- *Uurige välja, kui tähenduslik on nende andmete järgi seos Paides elamise ja alkoholitarbimise vahel.

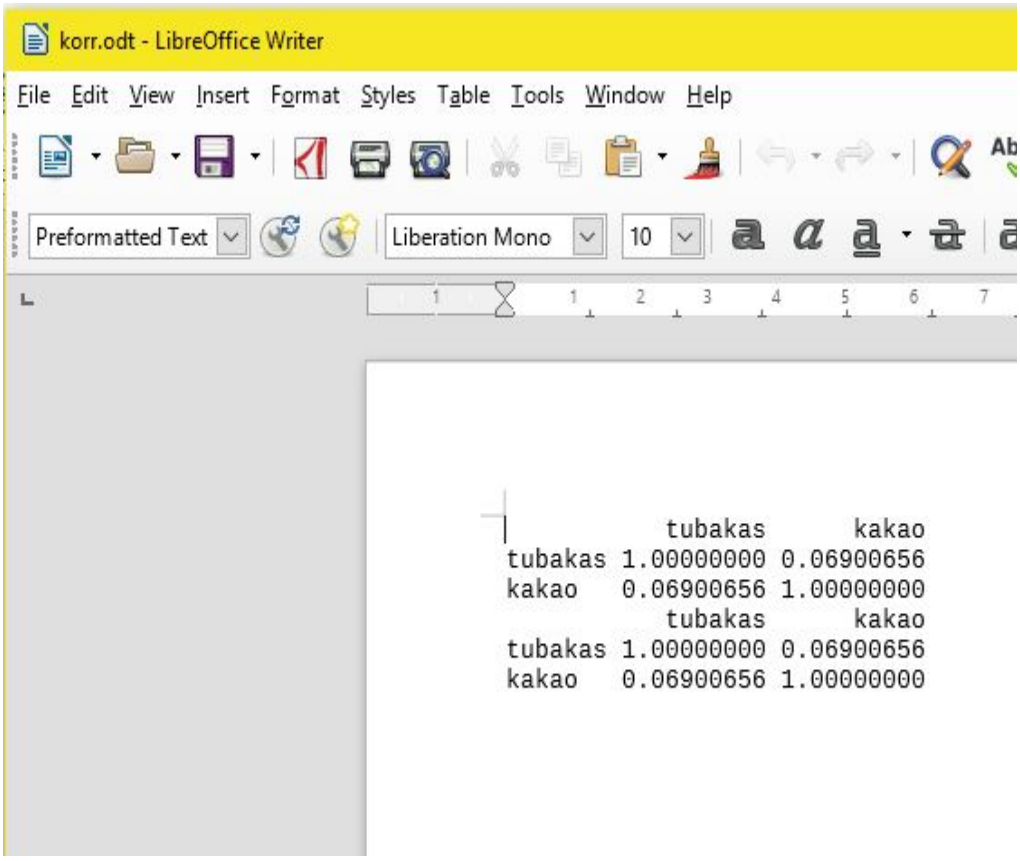
1.9 Tavalisse tekstifaili kirjutamine

Kui statistilised operatsioonid on tehtud, on hea, kui saab inimloetavad tulemused üles kirjutada. Siiani oleme loodud arvnäitajaid ja tabeleid R-i konsooli väljastanud *print()* käsu abil. Nende kirjutamiseks suvalisse tekstifaili kasutame *sink()* käsku, kujul *sink(„faili nimi”, argumendid)*. See käsk suunab väljundkäsud (nagu *print()*), mis muidu trükiksid andmed R-i konsooli, ümber soovitud faili. Argumentide kohal võime täpsustada, kas soovime faili infot lisada, või vana info uuega üle kirjutada. Kui argumente mitte anda, kirjutatakse vana info üle. Info saame juurde lisada, kui kirjutame käsku argumendi *append=TRUE* . Kui oleme soovitud kirjutustoimingud lõpetanud, anname käsu kujul *sink()*, et fail salvestada, ja väljund uuesti konsooli suunata. Demonstreerime näiteks eelmises punktis loodud korrelatsioonikordajate tabeli salvestamist .odt vormingus tekstifaili.

Sisend:

```
tubakas <-
c("Jah","Jah","Ei","Jah","Ei","Ei","Jah","Ei","Jah","
Ei","Ei")
kakao <-
c("Ei","Jah","Ei","Ei","Jah","Ei","Ei","Ei","Jah","Ei
","Jah")
tubakas[tubakas == "Jah"] <- 1
tubakas[tubakas == "Ei"] <- 0
tubakas <- as.numeric(tubakas)
kakao[kakao == "Jah"] <- 1
kakao[kakao == "Ei"] <- 0
kakao <- as.numeric(kakao)
andmed <- data.frame(tubakas,kakao)
p <- cor(andmed, method="spearman")
sink("korr.odt", append=TRUE)
print(p)
sink()
```

Väljund:



Antud pilt on tehtud pärast koodi kaks korda käivitamist. Kuna lisasime argumendi

append=TRUE, on nüüd failis kaks korda sama tabel.

Tasub mainida, et *sink()* käsuga saab tekstifaili suunata ainult teksti ja tabeleid. Väljundkäsud, mis salvestavad pilte (näiteks diagramme), salvestavad endiselt eraldi faile.

Harjutused:

- Võtke oma suva järgi mõni eelmistes peatükkides leiduv näidisprogramm, mis trükkis oma väljundit konsooli, ning pange ta *sink()* käsu abil oma väljundit tekstifaili salvestama.

2 Tehniline info

Sellest peatükist leiate tabelid, mis kirjeldavad käske, nende argumente, ja nendega seotud nüansse.

2.1 Muutujad ja andmetüübid

Muutuja tüüp	Olemus	Loomiskäsk
Vektor (Vector)	Võib sisaldada suvalises koguses sama tüüpi väärtuseid.	Muutuja nimi <- c(väärtus1,väärtus2,...)
Nimekiri (List)	Võib sisaldada suvalises koguses ükskõik mida, isegi teisi muutujaid.	Muutuja nimi <- list(element1,element2,...)
Faktor (Factor)	Vektori põhjal loodud konstrukt, mis sisaldab lisaks vektori andmetele ka statistilisi metaandmeid (*levels* näitajat).	Muutuja nimi <- factor(vektor)
Maatriks (Matrix)	Kahedimensiooniline tabel, mis sisaldab ainult ühte tüüpi väärtuseid. Moodustatakse vektori põhjal.	Muutuja nimi <- matrix(vektor, nrow = ridade arv, ncol = tulpade arv, byrow = TRUE/FALSE) (Byrow väärtus määrab, kas vektori elemendid jaotatakse maatriksisse ära ridade või tulpade kaupa)
Array	Mitut maatriksit sisaldav kogum, mis funktsionaalselt kujutab kasutaja soovitud hulga dimensioonidega maatriksit.	Muutuja nimi <- (andmete vektor, dimensioonide vektor) Dimensioonide vektor täpsustab soovitud dimensioonid. Näiteks vektor (2,2,3) loob array, kus on sees kolm 2x2 maatriksit.
Andmeraamistik (Data frame)	Paindlik andmetabel, mis võib sisaldada kõiksugu erinevaid väärtuseid eraldi tulpades. Moodustub vektorite põhjal.	Muutuja nimi <- data.frame(vektor1,vektor2, ...)

NB: R-keeles saab muutujatele väärtusi omistada järgmiste käskudega. (Tulemuseks on siin iga kord täpselt sama nime ja väärtusega muutuja.)

```
A <- 1
A = 1
1 -> A
```

Andmetüüp	Olemus	Näide
Number (Numeric)	Suvaline ratsionaalarv	1, 2.34, -220
Täisarv (Integer)	Suvaline täisarv	1L, 25L, -219L
Kompleksarv (Complex)	Suvaline irratsionaalarv	1i, 6+2i
Loogiline (Logical)	Tõene või väär	TRUE, FALSE
Tähemärk (Character)	Jada tekstisümboleid (Võib sisaldada ka numbreid, kuid neid koheldakse samuti tähemärkidena)	„tere”, „Arbuus”, „2017AD”, „2”
16-süsteemi kood (Raw)	Tähemärkidele vastav 16-süsteemi kood.	A <- charToRaw(„lol”) A = 6c 6f 6c

2.2 Lihtsamad andmeoperatsioonid

Arvutused:

Operaator	Toime	Näide
+	Liidab kahe vektori vastavad elemendid	(1,2,3) + (4,5,6) = (5,7,9)
-	Lahutab kahe vektori vastavad elemendid	(1,2,3) - (4,5,6) = (-3,-3,-3)
*	Korrutab kahe vektori vastavad elemendid	(1,2,3) * (4,5,6) = (4,10,18)
/	Jagab esimese vektori vastavad elemendid teise vektori omadega	(8,15,3) / (4,5,6) = (2,3,0.5)
%%	Leiab esimese vektori vastavate elementide jagamisel teise omadega tekkivad jäägid	(9,15,3) %% (4,5,6) = (1,0,3)

NB: Kui teha operatsioone kahe erineva pikkusega vektoriga, hakatakse lühema vektori elemente algusest peale taaskasutama.

```
(1,2) + (1,2,3,4) = (2,4,4,6)
```

Võrdlused:

Operaator	Toime	Näide
<	Kontrollib, kas vasakpoolsed andmed on parempoolsetest väiksemad	(1,5,6) < (2,3,6) = (TRUE,FALSE,FALSE)
>	Kontrollib, kas vasakpoolsed andmed on parempoolsetest suuremad	(1,5,6) > (2,3,6) = (FALSE,TRUE,FALSE)
==	Kontrollib, kas vasakpoolsed andmed on parempoolsetega võrdsed	(1,5,6) == (2,3,6) = (FALSE,FALSE,TRUE)
<=	Kontrollib, kas vasakpoolsed andmed on parempoolsetest väiksemad või nendega võrdsed	(1,5,6) <= (2,3,6) = (TRUE,FALSE,TRUE)
>=	Kontrollib, kas vasakpoolsed andmed on parempoolsetest suuremad või nendega võrdsed	(1,5,6) >= (2,3,6) = (FALSE,TRUE,TRUE)
!=	Kontrollib, kas vasakpoolsed andmed ei ole parempoolsetega võrdsed	(1,5,6) != (2,3,6) = (TRUE,TRUE,FALSE)

Loogikatehted:

Operaator	Toime	Näide
&	Loogiline „ja”. Annab vastuseks TRUE, kui mõlemad võrreldavad elemendid on TRUE	TRUE & TRUE = TRUE FALSE & TRUE = FALSE FALSE & FALSE = FALSE
|	Loogiline „või”. Annab vastuseks TRUE, kui vähemalt üks võrreldavatest elementidest on TRUE	TRUE | TRUE = TRUE FALSE | TRUE = TRUE FALSE | FALSE = FALSE
!	Loogiline „mitte”. Annab vastuseks sisendi vastandi.	!TRUE = FALSE !FALSE = TRUE

Muu:

Operaator		
:	„Kuni” operaator. Aitab luua teatud arvuvahemikust vektori.	A <- 2:5 A = (2,3,4,5)
%in%	Ütleb, kas vasakpoolne element esineb parempoolses vektoris	3 %in% A = TRUE 10 %in% A = FALSE

2.3 Tingimuslaused

Tingimuskomponent	Olemus	Näide

if	Käivitab mingi loogelistes sulgudes antud koodiosa, kui harilikes sulgudes antud loogikatehte väärtuseks on TRUE	if(a>b){
		print(„Muutuja a on suurem, kui muutuja b”)
		}

else	Käivitab mingi koodiosa (või hindab uut if-lauset) kui eelmine if-lause ei käivitunud	if(a>b){
		print(„Muutuja a on suurem, kui muutuja b”)
		} else if (a<b) {
		print(„Muutuja a on väiksem, kui muutuja b”)
		} else {
		print(„Muutujad a ja b on võrdsed”)
		}

switch	Annab muutujale etteantud 1:n-arvu põhjal n-inda väärtuse	A <- switch(2,”Esimene”,”Teine”,”Kolmas”) A = „Teine”

2.4 Funktsioonid

Oma funktsiooni saab luua käsuga *funktsiooni nimi <- function(argument1,argument2,...){funktsiooni sisu}*.

Näide:

```
taishaalik <- function(a) {
  if(a %in% c("a","e","i","o","u","ö","ä","ü","õ")){
    print("See on täishäälik")
  } else {
    print("See ei ole täishäälik")
  }
}
taishaalik("a")
taishaalik("p")
```

Väljund:

```
[1] "See on täishäälik"
[1] "See ei ole täishäälik"
```

2.5 Andmeliidesed

CSV failid:

Liides lugemiseks/kirjutamiseks on algpaketis olemas.

Käsud:

```
#Loe
andmeraami_nimi <- read.csv("failinimi.csv")
#Kirjuta
write.csv(andmeraami_nimi,"failinimi.csv",lisaargumen
did)
#Lisaargumendid
row.names=TRUE/FALSE #Kas kirjutame sisse automaatsed
reanumbrid
```

Exceli failid:

Liidest lugemiseks/kirjutamiseks algpaketis ei ole. Sisaldub näiteks paketis xlsx.

Käsud:

```
#Paigaldus
install.packages("xlsx")
library("xlsx")
#Loe
andmeraami_nimi <- read.xlsx("failinimi.xlsx",
sheetIndex=lehekülje_number)
#Kirjuta
write.xlsx(andmeraami_nimi,"failinimi.xlsx",sheetName
=töölehe_nimi,lisaargumendid)
#Lisaargumendid
row.names=TRUE/FALSE #Kas kirjutame sisse automaatsed
reanumbrid
```

XML failid:

Liidest lugemiseks/kirjutamiseks algpaketis ei ole. Sisaldub näiteks paketis XML.

Käsud:

```
#Paigaldus
install.packages("XML")
library("XML")
#Loe
nimekirja_nimi <- xmlParse(file="failinimi.xml")
andmeraami_nimi <- xmlToDataFrame("failinimi.xml")
#Kirjuta
saveXML(andmestruktuuri_nimi,"failinimi.xml",lisaargu
mendid)
#Lisaargumendid
doctype=mingi_tähejada #Mida kirjutame XML faili
DOCTYPE väljale
```

2.6 Andmeteisendused

Vektorid:

Käsk	Olemus	Näide
rev()	Väljastab sisendiks antud vektori elemendid vastupidises järjekorras	A = (1,2,3) rev(A) = (3,2,1)
sort()	Sorteerib sisendiks antud vektori elemendid järjekorda vähimast-suurimani	A = (10,1,2,4) sort(A) = (1,2,4,10)
match()	Väljastab esimese sisseantud vektoriga sama pika vektori, kus on alles kõik teises vektoris leiduvad väärtused, aga teises vektoris mitteleiduvad väärtused on asendatud NA-väärtustega	A = (1,2,3,4) B = (1,2) match(A,B) = (1,2,NA,NA)

Andmeraamistikud:

Käsk	Olemus	Vorming
subset()	Eraldab olemasolevast andmeraamistikust tingimustele vastavad sissekanded uute andmeraamistikku	uus_raamistik <- subset(vana_raamistik, tingimused)
rbind()	Liidab samade tulpadega andmeraamistikud üheks andmeraamistikuks (lisab mõlema raamistiku andmeread ühte raamistikku)	Uus_raamistik <- rbind(raamistik1,raamistik2)
merge()	Liidab suvalise struktuuriga andmeraamistikud üheks andmeraamistikuks, sidudes andmed by-argumentidega määratud väljade järgi. Kui by argumente mitte anda, „keevitab” tulbad kokku lihtsalt järjekorra alusel	Uus_raamistik <- merge(raamistik1,raamistik2, by.raamistik1 = c(„tulbanimi1”,”tulbanimi2”,...), by.raamistik2 = c(„tulbanimi1”,”tulbanimi2”,...))

2.7 Tabelid ja diagrammid

Tabelid:

Käsk	Olemus	Vorming
table()	Seab sisendiks antud vektorite elemendid tabelisse	tabeli_nimi <- (vektor1,vektor2,...)
Prop.table()	Muudab sisendiks antud tabeli proportsionaalseks, ehk protsendipõhiseks	uus_tabel <- prop.table(vana_tabel)

Diagrammid:

Käsk	Olemus	Vorming
plot()	Toodab sisseantud vektori järgi joondiagrammi.	plot(andmed, lisaargumendid)
lines()	Lisab töös olevale joondiagrammile sisseantud vektori põhjal joone	lines(andmed, lisaargumendid)
hist()	Loob sisseantud vektori põhjal histogrammi	hist(andmed, lisaargumendid)
barplot()	Loob sisseantud vektori põhjal tulpdiagrammi	barplot(andmed, lisaargumendid)
boxplot()	Loob sisseantud vektori põhjal karpdiagrammi	boxplot(andmed, lisaargumendid)
pie()	Loob sisseantud vektori põhjal sektordiagrammi	pie(andmed,sektorite_nimed,lisaargumendid)

Diagrammide loomiseks kasutatavate käskude lõppu saab lisada soovi korral lisaargumente. Need tuleks üksteisest komadega eraldada, kuid neid võib käsu sisse lisada nii palju, kui vaja.

Lisaargumendid:

Argument	Otstarve
add=TRUE/FALSE	Kui TRUE, ja sama nimega diagramm on juba olemas, joonistatakse uue diagrammi andmed vana omade peale. Muidu kirjutatakse vana fail üle.
axes=TRUE/FALSE	Kui FALSE, siis diagrammil ei kuvata telgesid ega piirjoont.
type=”p”/”l”/”o”/”h”	Joondiagrammidele mõeldud käsk. Määrab andmete kuvamise viisi. P: ainult punktid, L: ainult jooned, O: nii punktid kui jooned, H: vertikaalsed jooned.
xlim=(arv1,arv2), ylim=(arv1,arv2)	Täpsustab telgede alumised ja ülemised piirid.
xlab=”tekst”,ylab=”tekst”	Määrab telgede juurde kirjutatud sildid.
main=”tekst”	Määrab diagrammi pealkirja.
sub=”tekst”	Määrab diagrammi alampealkirja.
Col.elemendi_nimi = ”värvi nimi” või „värvi RGB kood”	Määrab ära mingi sümboli või elemendi värvi. Näiteks *col.axis = „Red”* värvib teljed punaseks. Andmeid kujutavad jaotised saab ära värvida, kui lisada (ilma punktiga täpsustuseta) argument *col = c(„värv1”,”värv2”,...)*, kus vektoris on igale andmepunktile vastava värvi nimed. (Vt. näidet all)

Värvidega tulpdiagrammi kood:

```
andmed <- c(2,4,8,16)
sildid <- c("Esimene","Teine","Kolmas","Neljas")
varvid <- c("Red","Green","Blue","Yellow")
png(file = "värvitud_tulpdiagramm.png")
barplot(andmed,names.arg=sildid,xlab="x-
telg",ylab="y-telg",main="näidisdiagramm", col =
varvid)
dev.off()
```

2.8 Tulemuste eksportimine

Käsk	Olemus	Vorming
print()	Trükib mingi muutuja väärtuse konsooli.	print(muutuja)
sink()	Suunab print käsu ümber soovitud (teksti)faili. Tühjaks jäetuna salvestab väljundifaili muudatused, ning suunab print käsu tagasi konsooli. Append muutuja väärtus määrab, kas väljundfailis olev info kirjutatakse üle (FALSE), või kirjutatakse uus info lihtsalt faili lõppu (TRUE).	sink(„failinimi”,append=TRUE/FALSE)
png()	Määrab faili, kuhu salvestatakse visuaalne väljund (diagrammid).	png(„failinimi.png”)
Dev.off()	Salvestab visuaalse väljundi eelneva png() käsuga määratud pildifaili.	png(„failinimi.png”) **graafi loomise käsud** Dev.off()