R-kielen opas

Asiantuntijaluennot

Bragge Janne TTM23SAI

2025

Table of Contents

[Johdanto 6](#_Toc184373640)

[Mikä on R-kieli? 6](#_Toc184373641)

[Kenelle tämä opas on tarkoitettu? 6](#_Toc184373642)

[Miksi oppia R-kieltä? 6](#_Toc184373643)

[Mitä opas kattaa? 6](#_Toc184373644)

[Miten käyttää tätä opasta? 7](#_Toc184373645)

[Mitä opas ei kata? 7](#_Toc184373646)

[Tavoite 7](#_Toc184373647)

[R-kielen Asentaminen 8](#_Toc184373648)

[1. R-kieli 8](#_Toc184373649)

[1.1. Windows 8](#_Toc184373650)

[1.2. MacOS 8](#_Toc184373651)

[1.3. Linux 8](#_Toc184373652)

[2. RStudio-työkalun asentaminen 8](#_Toc184373653)

[3. Testaa asennus 9](#_Toc184373654)

[4. Lisäpakettien asentaminen 9](#_Toc184373655)

[5. Vinkkejä ja huomioitavaa 9](#_Toc184373656)

[Käytön aloittaminen 10](#_Toc184373657)

[Pseudo-datasetin luominen 10](#_Toc184373658)

[1. Avaa RStudio 10](#_Toc184373659)

[2. Luo uusi skriptitiedosto 10](#_Toc184373660)

[3. Tallenna tiedosto 10](#_Toc184373661)

[4. Aloita koodin kirjoittaminen 10](#_Toc184373662)

[5. Suorita koodi 10](#_Toc184373663)

[Pseudodatan visualisointi 11](#_Toc184373664)

[1. Luo pseudodata 11](#_Toc184373665)

[2. Perusvisualisointi: pistekaavio 11](#_Toc184373666)

[3. Histogrammi 12](#_Toc184373667)

[4. Laatikkokaavio (Boxplot) 13](#_Toc184373668)

[5. Viivakaavio (jos aikasarjadatan tyyppistä) 13](#_Toc184373669)

[Vinkkejä 14](#_Toc184373670)

[Muuttujat (Variables) 15](#_Toc184373671)

[Kappale 2: R-kielen muuttujat (Variables) 15](#_Toc184373672)

[Muuttujien luonti 15](#_Toc184373673)

[Muuttujien nimeäminen 16](#_Toc184373674)

[Muuttujatyypit 16](#_Toc184373675)

[Muuttujien muuttaminen 17](#_Toc184373676)

[Muuttujien käyttäminen laskutoimituksissa 17](#_Toc184373677)

[Erikoismuuttujat 17](#_Toc184373678)

[Muuttujien hallinta 18](#_Toc184373679)

[Näytä kaikki nykyiset muuttujat 18](#_Toc184373680)

[Poista muuttuja 18](#_Toc184373681)

[R-kielen loogiset operaatiot (Logical Operators) 19](#_Toc184373682)

[Mikä on looginen operaatio? 19](#_Toc184373683)

[Loogiset operaattorit 19](#_Toc184373684)

[Perusesimerkkejä loogisista operaattoreista 19](#_Toc184373685)

[1. Yhden ehdon tarkistus 19](#_Toc184373686)

[2. Usean ehdon yhdistäminen 19](#_Toc184373687)

[Loogiset operaatiot vektoreissa 20](#_Toc184373688)

[Loogisten ehtojen hyödyntäminen 21](#_Toc184373689)

[1. Ehtolauseet (if, else if, else) 21](#_Toc184373690)

[2. Tietojen suodattaminen 21](#_Toc184373691)

[Erikoistapaukset: NA ja NaN 22](#_Toc184373692)

[Print()-funktio R-kielessä 23](#_Toc184373693)

[Mikä on print()? 23](#_Toc184373694)

[print()-funktion käyttö 23](#_Toc184373695)

[Tulostuksen oletusmuoto 23](#_Toc184373696)

[Tulostaminen ilman print()-funktiota 24](#_Toc184373697)

[print()-funktion käyttö funktioiden sisällä 24](#_Toc184373698)

[Monimutkaisten rakenteiden tulostaminen 24](#_Toc184373699)

[Tulostuksen muotoiluvinkkejä 25](#_Toc184373700)

[Eroja print() ja muiden tulostusfunktioiden välillä 26](#_Toc184373701)

[Vektorit (Vectors) R-kielessä 28](#_Toc184373702)

[Mikä on vektori? 28](#_Toc184373703)

[Vektorin luominen 28](#_Toc184373704)

[Vektorin tietotyypit 28](#_Toc184373705)

[Vektorin käsittely 29](#_Toc184373706)

[1. Indeksointi 29](#_Toc184373707)

[2. Muokkaaminen 29](#_Toc184373708)

[3. Ehdollinen valinta 29](#_Toc184373709)

[Operaatiot vektoreilla 29](#_Toc184373710)

[1. Aritmeettiset operaatiot 29](#_Toc184373711)

[2. Loogiset operaatiot 30](#_Toc184373712)

[3. Aggregaattifunktiot 30](#_Toc184373713)

[Erityistoiminnot vektoreilla 30](#_Toc184373714)

[1. Vektorin luonti sarjoilla 30](#_Toc184373715)

[2. Vektorin lajittelu 30](#_Toc184373716)

[3. Vektorin yhdistäminen 31](#_Toc184373717)

[Vektorin yhteensopivuus 31](#_Toc184373718)

[Käytännön esimerkkejä 31](#_Toc184373719)

[Esimerkki 1: Lämpötilojen analyysi 31](#_Toc184373720)

[Esimerkki 2: Myyntidatan käsittely 31](#_Toc184373721)

[Listat (Lists) R-kielessä 33](#_Toc184373722)

[Mikä on lista? 33](#_Toc184373723)

[Listan luominen 33](#_Toc184373724)

[Nimetyt listan elementit 33](#_Toc184373725)

[Listan elementtien hakeminen 34](#_Toc184373726)

[Listan elementtien muokkaaminen 34](#_Toc184373727)

[Listan yhdistäminen 35](#_Toc184373728)

[Listan sisäkkäisyys 36](#_Toc184373729)

[Listan muunnokset 37](#_Toc184373730)

[Listan tutkiminen 37](#_Toc184373731)

[Käytännön esimerkkejä 37](#_Toc184373732)

[Esimerkki 1: Opiskelijadata listassa 37](#_Toc184373733)

[Esimerkki 2: Sisäkkäiset listat 38](#_Toc184373734)

[Matriisit (Matrices) R-kielessä 39](#_Toc184373735)

[Mikä on matriisi? 39](#_Toc184373736)

[Matriisin luominen 39](#_Toc184373737)

[Perusmuoto 39](#_Toc184373738)

[Esimerkkejä 39](#_Toc184373739)

[1. Matriisi numeerisilla arvoilla 39](#_Toc184373740)

[2. Matriisi rivien mukaan täytettynä 39](#_Toc184373741)

[3. Merkkijonomatriisi 40](#_Toc184373742)

[Matriisin ominaisuudet 40](#_Toc184373743)

[Esimerkki 40](#_Toc184373744)

[Matriisin elementtien käsittely 40](#_Toc184373745)

[1. Hakeminen 40](#_Toc184373746)

[2. Elementtien muokkaaminen 41](#_Toc184373747)

[Matriisioperaatiot 41](#_Toc184373748)

[1. Perusaritmetiikka 41](#_Toc184373749)

[2. Matriisikertolasku 41](#_Toc184373750)

[3. Transponointi 42](#_Toc184373751)

[4. Rivien ja sarakkeiden summa tai keskiarvo 42](#_Toc184373752)

[Matriisin yhdistäminen 42](#_Toc184373753)

[1. Rivien lisääminen 42](#_Toc184373754)

[2. Sarakkeiden lisääminen 43](#_Toc184373755)

[Käytännön esimerkkejä 43](#_Toc184373756)

[Esimerkki 1: Taulukon analyysi 43](#_Toc184373757)

[Esimerkki 2: Opiskelijoiden arvosanat 43](#_Toc184373758)

[Kuvaajat (Data Frames) R-kielessä 45](#_Toc184373759)

[Mikä on dataframe? 45](#_Toc184373760)

[Dataframen luominen 45](#_Toc184373761)

[1. Manuaalisesti 45](#_Toc184373762)

[2. CSV-tiedostosta 45](#_Toc184373763)

[3. Dataframen rakenne 46](#_Toc184373764)

[Ohjausrakenteet (Flow Control) R-kielessä 51](#_Toc184373765)

[Mikä on ohjausrakenne? 51](#_Toc184373766)

[1. Ehtolauseet: if, else if, else 51](#_Toc184373767)

[2. Lyhennetty ehtolause: ifelse() 52](#_Toc184373768)

[3. Toistorakenteet 52](#_Toc184373769)

[4. Keskeytys ja jatkaminen: break ja next 54](#_Toc184373770)

[5. Pesäkkäiset silmukat 54](#_Toc184373771)

[Käytännön esimerkkejä 55](#_Toc184373772)

[Funktiot R-kielessä 57](#_Toc184373773)

[Mikä on funktio? 57](#_Toc184373774)

[Funktioiden rakenne 57](#_Toc184373775)

[Funktion luominen ja käyttäminen 57](#_Toc184373776)

[Esimerkki 1: Yksinkertainen funktio 57](#_Toc184373777)

[Esimerkki 2: Laskufunktio 57](#_Toc184373778)

[Funktion parametrit 58](#_Toc184373779)

[1. Oletusarvot 58](#_Toc184373780)

[2. Argumenttien nimeäminen 58](#_Toc184373781)

[Funktion paluuarvo 58](#_Toc184373782)

[Lokaalit ja globaalit muuttujat 59](#_Toc184373783)

[Anonyymit funktiot 59](#_Toc184373784)

[Sisäänrakennetut funktiot 59](#_Toc184373785)

[Soveltaminen data.frameissä 60](#_Toc184373786)

[Käytännön esimerkkejä 60](#_Toc184373787)

[Esimerkki 1: Fahrenheit Celsius-asteiksi 60](#_Toc184373788)

[Esimerkki 2: Ryhmän keskiarvojen laskenta 60](#_Toc184373789)

[Hyviä käytäntöjä 61](#_Toc184373790)

[Datamanipulaatio R-kielessä 62](#_Toc184373791)

[Mikä on datamanipulaatio? 62](#_Toc184373792)

[1. Perustoiminnot base R:llä 62](#_Toc184373793)

[2. Datamanipulaatio dplyr-paketilla 63](#_Toc184373794)

[3. Yhdistely ja muotoilu 65](#_Toc184373795)

[4. Käytännön esimerkkejä 66](#_Toc184373796)

[Yhteenveto 67](#_Toc184373797)

[Datavisualisointi R-kielessä 68](#_Toc184373798)

[Mikä on datavisualisointi? 68](#_Toc184373799)

[1. Visualisointi Base R:llä 68](#_Toc184373800)

[1.1 Pylväsdiagrammi: barplot() 68](#_Toc184373801)

[1.2 Histogrammi: hist() 68](#_Toc184373802)

[1.3 Hajontakaavio: plot() 68](#_Toc184373803)

[1.4 Viivakaavio: lines() 69](#_Toc184373804)

[2. Visualisointi ggplot2:lla 69](#_Toc184373805)

[2.1 Peruskaavio: ggplot() 69](#_Toc184373806)

[2.2 Pylväsdiagrammi 70](#_Toc184373807)

[2.3 Histogrammi 70](#_Toc184373808)

[2.4 Boxplot 70](#_Toc184373809)

[2.5 Ryhmitelty visualisointi 71](#_Toc184373810)

[3. Interaktiiviset visualisoinnit 71](#_Toc184373811)

[4. Käytännön esimerkkejä 72](#_Toc184373812)

[Esimerkki 1: Myyntidata 72](#_Toc184373813)

[Esimerkki 2: Histogrammi ja tiheyskäyrä 72](#_Toc184373814)

# Johdanto

## Mikä on R-kieli?

R on avoimen lähdekoodin ohjelmointikieli, joka on erityisesti suunniteltu tilastolliseen laskentaan ja datan analysointiin. Se on suosittu erityisesti akateemisessa maailmassa, datatieteessä, tilastotieteessä ja koneoppimisessa. R:n vahvuudet ovat sen monipuoliset datan käsittely- ja visualisointityökalut sekä laaja yhteisö, joka kehittää jatkuvasti uusia paketteja ja resursseja.

## Kenelle tämä opas on tarkoitettu?

Tämä opas on suunniteltu kaikille, jotka haluavat oppia R-ohjelmointia ja käyttää sitä datan käsittelyyn, analyysiin ja visualisointiin. Se sopii erityisesti:

* **Aloittelijoille**, jotka eivät ole aiemmin ohjelmoineet.
* **Tilastotieteilijöille** ja **data-analyytikoille**, jotka haluavat hyödyntää R:ää työssään.
* **Data Scientists** -ammattilaisille, jotka haluavat hallita R:n vahvuuksia datan manipuloinnissa ja visualisoinnissa.

## Miksi oppia R-kieltä?

R tarjoaa lukuisia etuja datan käsittelyyn ja analysointiin:

1. **Monipuolisuus**  
   R tukee sekä yksinkertaisia laskelmia että monimutkaisia tilastollisia malleja ja koneoppimisalgoritmeja.
2. **Tehokkaat visualisointityökalut**  
   R:llä voi luoda korkealaatuisia kaavioita, jotka tekevät datasta helppotajuisempaa.
3. **Laaja pakettivalikoima**  
   CRAN-arkistossa on tuhansia paketteja, jotka laajentavat R:n ominaisuuksia eri tarkoituksiin.
4. **Yhteisön tuki**  
   R:n käyttäjillä on aktiivinen yhteisö ja runsaasti resursseja, kuten oppaita, foorumeita ja verkkokursseja.

## Mitä opas kattaa?

Tämä opas tarjoaa perusteellisen johdatuksen R-kieleen ja sen soveltamiseen datatieteessä. Opas etenee loogisessa järjestyksessä:

1. **Muuttujat ja tietotyypit**  
   Opi R:n perusrakenteet, kuten vektorit, listat, matriisit ja dataframet.
2. **Ohjausrakenteet ja funktiot**  
   Hallitse ohjelmoinnin perusperiaatteet ehtolauseilla, silmukoilla ja funktioilla.
3. **Datamanipulaatio**  
   Opi muokkaamaan ja järjestelemään dataa sekä hyödyntämään dplyr- ja tidyr-paketteja tehokkaaseen analyysiin.
4. **Datavisualisointi**  
   Sukella visuaalisuuden maailmaan ja opi luomaan kauniita kaavioita ggplot2-kirjastolla.
5. **Käytännön esimerkit**  
   Sovella opittuja taitoja todellisiin ongelmiin, kuten myyntianalyysiin ja tilastollisiin laskelmiin.

## Miten käyttää tätä opasta?

* Aloita perusteista ja etene järjestyksessä, erityisesti jos olet uusi ohjelmoinnissa.
* Kokeile kaikkia esimerkkejä itse. RStudio on suositeltu ympäristö harjoitteluun.
* Hyödynnä käytännön esimerkkejä ja sovella niitä omiin projekteihisi.
* Palaa oppaan osioihin aina tarvittaessa, sillä se toimii myös viitekirjana.

## Mitä opas ei kata?

Vaikka tämä opas tarjoaa kattavan perustan, se ei mene syvälle erityisaloihin, kuten koneoppimiseen tai biostatistiikkaan. Näille aiheille suositellaan erikoistuneita oppaita.

## Tavoite

Oppaan tavoitteena on antaa sinulle käytännön taidot R-kielen hyödyntämiseen datan analysoinnissa ja visualisoinnissa. Opit soveltamaan R:ää tehokkaasti työssäsi ja projekteissasi.

**Tervetuloa oppimaan R-ohjelmointia!**

# R-kielen Asentaminen

Tässä raportissa kuvataan vaiheittain, kuinka asentaa R-kieli ja siihen liittyvät työkalut (RStudio) Windows-, MacOS- ja Linux-käyttöjärjestelmissä.

## 1. R-kieli

R on avoimen lähdekoodin ohjelmointikieli, joka on erityisen suosittu data-analyysissä ja tilastollisessa laskennassa.

### 1.1. Windows

1. Siirry R:n viralliselle verkkosivustolle: [CRAN R Project](https://cran.r-project.org/).
2. Valitse “Download R for Windows”.
3. Klikkaa "base" ja lataa viimeisin versio ("Download R X.X.X for Windows").
4. Suorita ladattu asennustiedosto.
   * Valitse asennuksen aikana oletusasetukset, ellet tiedä tarvitsevasi erityisiä määrittelyjä.
5. Kun asennus on valmis, R voidaan avata joko komentoriviltä tai RStudion kautta.

### 1.2. MacOS

1. Siirry CRAN-sivustolle: [CRAN R Project](https://cran.r-project.org/).
2. Valitse “Download R for macOS”.
3. Lataa sopiva versio (yleensä "arm64.pkg" tai "x86\_64.pkg" riippuen prosessoristasi).
4. Avaa ladattu tiedosto ja noudata ohjeita asentaaksesi R:n.
5. R on nyt käytettävissä. Asenna RStudio lisämukavuuden saavuttamiseksi.

### 1.3. Linux

1. Avaa komentorivi.
2. Lisää CRAN:n pakettivarasto järjestelmääsi. Esimerkiksi Ubuntu/Debian-järjestelmissä:
3. sudo apt update
4. sudo apt install r-base
5. Asenna tarvittaessa ylimääräiset paketit komennolla install.packages() R-konsolissa.
6. Käynnistä R kirjoittamalla komentoriville R.

## 2. RStudio-työkalun asentaminen

RStudio on R:n kehitysympäristö, joka tarjoaa helppokäyttöisen käyttöliittymän ja monipuolisia ominaisuuksia.

1. Siirry RStudion verkkosivustolle: [RStudio Download](https://posit.co/download/rstudio-desktop/).
2. Lataa ilmainen versio, joka on saatavilla kaikille käyttöjärjestelmille.
3. Asenna ohjelma ja avaa se. Varmista, että R on asennettu ennen RStudion käyttämistä.

## 3. Testaa asennus

1. Avaa R tai RStudio.
2. Kirjoita konsoliin:
3. print("R toimii oikein!")

Jos saat tulosteena tekstin "R toimii oikein!", asennus on onnistunut.

## 4. Lisäpakettien asentaminen

Voit lisätä R:ään laajennuksia (paketteja), jotka tarjoavat uusia toimintoja:

install.packages("dplyr")

library(dplyr)

Tämä komento asentaa ja lataa dplyr-paketin käyttöön.

## 5. Vinkkejä ja huomioitavaa

* **Päivitykset:** Pidä R ja RStudio ajan tasalla uusien ominaisuuksien ja parannusten vuoksi.
* **RMarkdown:** Asenna rmarkdown-paketti raporttien luomiseen:
* install.packages("rmarkdown")
* **Dokumentaatio:** Hyödynnä R:n laajaa verkkodokumentaatiota ja yhteisön resursseja.

Raportin avulla voit suorittaa R:n ja siihen liittyvien työkalujen asennuksen onnistuneesti eri käyttöjärjestelmissä.

# Käytön aloittaminen

## Pseudo-datasetin luominen

### 1. Avaa RStudio

* Käynnistä RStudio tietokoneellasi.

### 2. Luo uusi skriptitiedosto

1. Valitse ylävalikosta **File > New File > R Script**.
2. Uusi tyhjä R-skriptitiedosto avautuu editoriin.

### 3. Tallenna tiedosto

1. Klikkaa ylävalikosta **File > Save** tai paina **Ctrl + S** (Windows) tai **Cmd + S** (Mac).
2. Anna tiedostolle nimi, esimerkiksi:  
   **pseudo\_dataset.R**
3. Valitse tallennuspaikka ja paina **Save**.

### 4. Aloita koodin kirjoittaminen

Esimerkki pseudodatasetin luomisesta:

# Tämä on uusi R-tiedosto

set.seed(123)

pseudo\_data <- data.frame(

ID = 1:50,

Value = rnorm(50, mean = 100, sd = 15)

)

print(head(pseudo\_data))

View(pseudo\_data)

### 5. Suorita koodi

1. Valitse haluamasi koodirivi tai lohko.
2. Paina **Ctrl + Enter** (Windows) tai **Cmd + Enter** (Mac) suorittaaksesi sen.

## Pseudodatan visualisointi

RStudiossa pseudodatan visualisointi onnistuu helposti esimerkiksi **ggplot2**-kirjastolla, joka on yksi suosituimmista R:n visualisointikirjastoista. Tässä on vaiheittainen ohje, kuinka voit visualisoida pseudodatan.

**Esimerkki pseudodatan luomisesta ja visualisoinnista**

### 1. Luo pseudodata

Aloitetaan yksinkertaisella pseudodatasetillä:

# Lataa ggplot2-kirjasto (asennus: install.packages("ggplot2"))

library(ggplot2)

# Luo pseudodata

set.seed(123)

pseudo\_data <- data.frame(

ID = 1:100,

Age = sample(18:65, 100, replace = TRUE),

Income = round(rnorm(100, mean = 50000, sd = 10000), 0),

Gender = sample(c("Male", "Female"), 100, replace = TRUE)

)

# Tarkastele dataa

head(pseudo\_data)

### 2. Perusvisualisointi: pistekaavio

Visualisoidaan esimerkiksi **ikä ja tulot sukupuolen mukaan**:

ggplot(pseudo\_data, aes(x = Age, y = Income, color = Gender)) +

geom\_point(size = 3, alpha = 0.7) + # Lisää pisteet

labs(

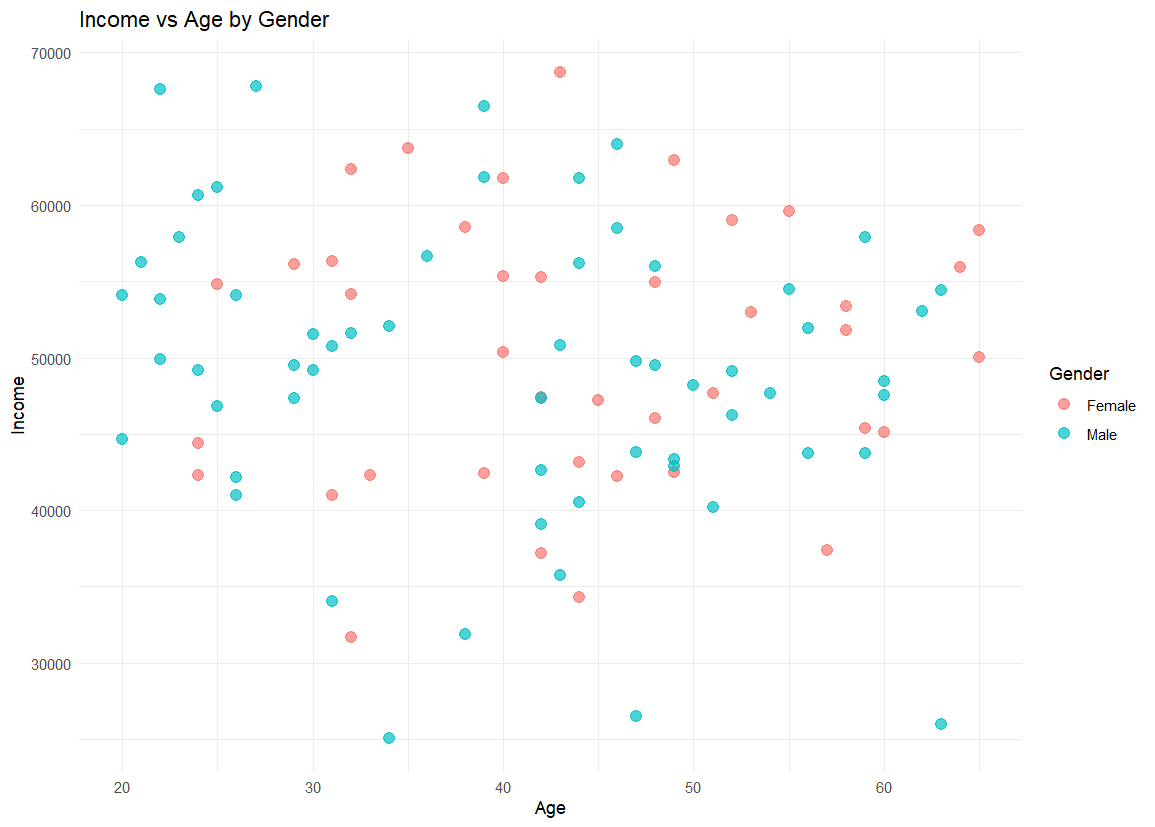
title = "Income vs Age by Gender",

x = "Age",

y = "Income"

) +

theme\_minimal() # Käytä minimalistista teemaa



### 3. Histogrammi

Histogrammin avulla voit tarkastella esimerkiksi ikäjakaumaa:

ggplot(pseudo\_data, aes(x = Age, fill = Gender)) +

geom\_histogram(binwidth = 5, alpha = 0.7, position = "dodge") + # Histogrammi sukupuolen mukaan

labs(

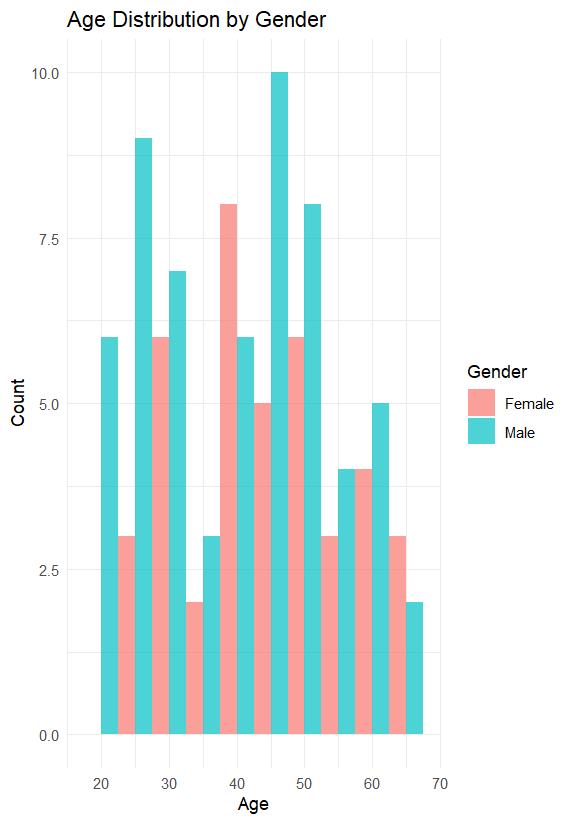
title = "Age Distribution by Gender",

x = "Age",

y = "Count"

) +

theme\_minimal()



### 4. Laatikkokaavio (Boxplot)

Laatikkokaavio sopii hyvin tulojen jakauman tarkasteluun sukupuolen mukaan:

ggplot(pseudo\_data, aes(x = Gender, y = Income, fill = Gender)) +

geom\_boxplot() +

labs(

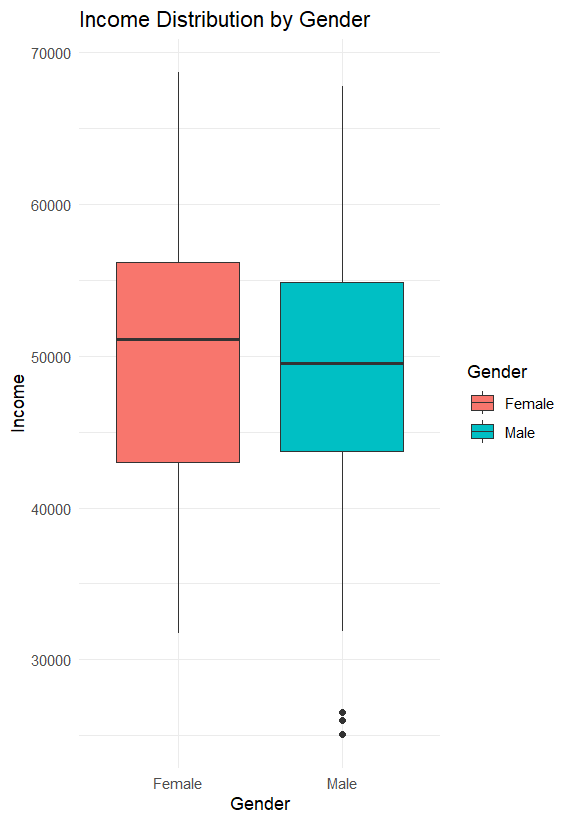
title = "Income Distribution by Gender",

x = "Gender",

y = "Income"

) +

theme\_minimal()



### 5. Viivakaavio (jos aikasarjadatan tyyppistä)

Jos sinulla olisi aikasarjaa, voit visualisoida sen näin:

pseudo\_data$Time <- seq.Date(from = as.Date("2023-01-01"), by = "day", length.out = 100)

ggplot(pseudo\_data, aes(x = Time, y = Income, color = Gender)) +

geom\_line() +

labs(

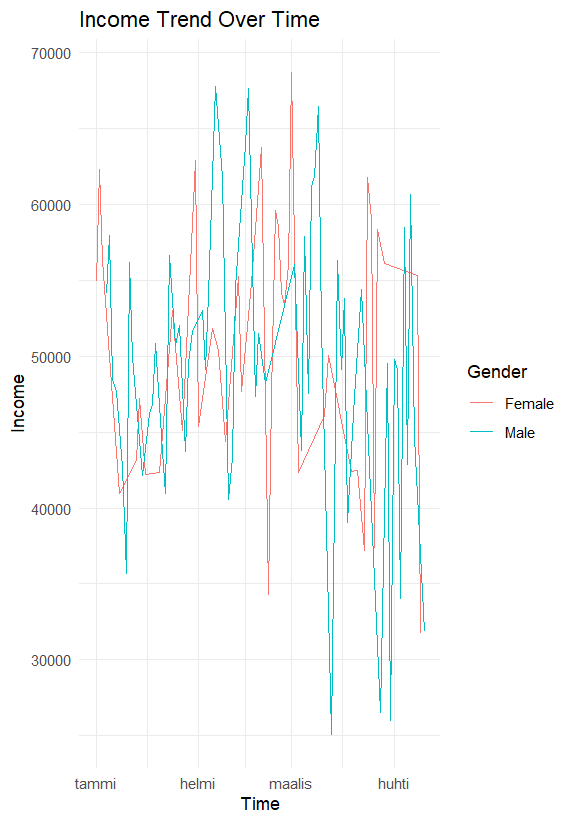
title = "Income Trend Over Time",

x = "Time",

y = "Income"

) +

theme\_minimal()



### Vinkkejä

1. **Mukauta visualisointeja**:  
   Lisää teemaa, värejä tai fontteja oman tarpeesi mukaan käyttämällä theme()-funktiota.
2. **Tallentaminen**:  
   Tallenna visualisointi tiedostoksi:
3. g <- ggplot(pseudo\_data, aes(x = Age, y = Income, color = Gender)) +
4. geom\_point(size = 3, alpha = 0.7) +
5. labs(title = "Income vs Age by Gender", x = "Age", y = "Income") +
6. theme\_minimal()
7. ggsave("income\_vs\_age.png", plot = g, width = 8, height = 6)

# Muuttujat (Variables)

## Kappale 2: R-kielen muuttujat (Variables)

**Mikä on muuttuja?**

Muuttuja on nimetty säilö, joka tallentaa arvoja, kuten numeroita, tekstiä, loogisia arvoja tai muita tietotyyppejä. R-kielessä muuttujien avulla voimme tallentaa, muokata ja käyttää tietoja ohjelmoinnissa.

## Muuttujien luonti

Muuttuja luodaan käyttämällä **<-**-operaattoria tai yhtälailla **=**-operaattoria.

**Esimerkkejä:**

# Muuttujan luonti

x <- 10 # Luvun tallentaminen muuttujaan x

y = "Hello" # Merkkijonon tallentaminen muuttujaan y

z <- TRUE # Loogisen arvon tallentaminen muuttujaan z

Voit myös tarkistaa muuttujan arvon kirjoittamalla sen nimen:

x # Tulostaa: 10

y # Tulostaa: "Hello"

z # Tulostaa: TRUE

A screenshot of a computer

Description automatically generated

## Muuttujien nimeäminen

R-kielessä muuttujien nimet voivat sisältää kirjaimia, numeroita, alaviivoja (\_) ja pisteitä (.), mutta niiden on aloitettava kirjaimella tai pisteellä, joka ei seuraa numeroa.

**Hyviä käytäntöjä:**

* Käytä kuvaavia nimiä, kuten age tai total\_sales.
* Noudatetaan **snake\_case** tai **camelCase**-tyylejä:
  + **snake\_case**: total\_sales
  + **camelCase**: totalSales

**Esimerkkejä:**

valid\_variable <- 5 # Sallittu

variable.name <- "text" # Sallittu

total\_sales <- 200 # Sallittu

**Virheelliset nimet:**

2variable <- 5 # Ei sallittu (ei voi alkaa numerolla)

variable-name <- 5 # Ei sallittu (viiva ei ole sallittu merkki)

## Muuttujatyypit

Muuttuja voi sisältää monentyyppisiä tietoja. Tässä yleisimmät tietotyypit:

1. **Numerot (numeric)**  
   Esimerkiksi desimaaliluvut tai kokonaisluvut.
2. a <- 42 # Kokonaisluku
3. b <- 3.14 # Desimaaliluku
4. **Merkkijonot (character)**  
   Tekstiä, jonka ympärillä käytetään lainausmerkkejä.
5. text <- "Hello, R!" # Merkkijono
6. **Loogiset arvot (logical)**  
   Arvot **TRUE** tai **FALSE**.
7. is\_active <- TRUE
8. **Vektorit (vector)**  
   Useiden arvojen lista, kuten [1, 2, 3].
9. numbers <- c(1, 2, 3, 4, 5)

**huom!**

A computer code with black text

Description automatically generated with medium confidence

## Muuttujien muuttaminen

Muuttuja voidaan päivittää helposti antamalla sille uusi arvo:

x <- 10 # Alkuperäinen arvo

x <- x + 5 # Päivitetään arvo (x nyt 15)

Muuttujan tyyppi voi myös vaihtua:

x <- "text" # x oli aiemmin numero, nyt se on merkkijono

## Muuttujien käyttäminen laskutoimituksissa

Muuttujia voidaan käyttää laskutoimituksissa ja muissa operaatioissa:

a <- 20

b <- 5

# Yhteenlasku

sum <- a + b # 25

# Kertolasku

product <- a \* b # 100

# Jakolasku

division <- a / b # 4

## Erikoismuuttujat

R sisältää kolme erikoisarvoa:

1. **NA**: Tarkoittaa "ei saatavilla" (missing value).
2. value <- NA
3. is.na(value) # Tarkistaa, onko arvo NA
4. **NULL**: Tarkoittaa "tyhjää arvoa" (null value).
5. empty <- NULL
6. **Inf ja -Inf**: Käytetään äärettömyyden ilmaisemiseen.
7. infinity <- 1 / 0 # Inf
8. **NaN**: Tarkoittaa "ei numero" (not a number).
9. not\_number <- 0 / 0 # NaN

## Muuttujien hallinta

### Näytä kaikki nykyiset muuttujat

ls() # Tulostaa kaikki ympäristössä olevat muuttujat

### Poista muuttuja

rm(x) # Poistaa muuttujan x

#### Käytännön esimerkki

# Myynnin seuranta

products\_sold <- 120 # Kuinka monta tuotetta myytiin

price\_per\_unit <- 25 # Tuotteen hinta

# Laske kokonaistulot

total\_revenue <- products\_sold \* price\_per\_unit

print(total\_revenue) # Tulostaa: 3000

# R-kielen loogiset operaatiot (Logical Operators)

## Mikä on looginen operaatio?

Loogiset operaatiot vertaavat arvoja ja palauttavat **TRUE** tai **FALSE**. Niitä käytetään yleisesti ehtolauseissa, tietojen suodattamisessa ja päätöksenteossa ohjelmoinnin aikana.

### Loogiset operaattorit

R-kielessä loogisia operaattoreita ovat:

| **Operaattori** | **Tarkoitus** | **Esimerkki** | **Tulos** |
| --- | --- | --- | --- |
| == | Onko yhtä suuri? | 5 == 5 | TRUE |
| != | Onko erisuuri? | 5 != 3 | TRUE |
| < | Onko pienempi? | 3 < 5 | TRUE |
| > | Onko suurempi? | 5 > 3 | TRUE |
| <= | Onko pienempi tai yhtä suuri? | 3 <= 3 | TRUE |
| >= | Onko suurempi tai yhtä suuri? | 5 >= 3 | TRUE |
| & | Ja (AND) | (5 > 3) & (2 < 4) | TRUE |
| ` | ` | Tai (OR) | `(5 > 3) |
| ! | Ei (NOT) | !(5 > 3) | FALSE |

## Perusesimerkkejä loogisista operaattoreista

### 1. Yhden ehdon tarkistus

x <- 10

x == 10 # Tarkistaa, onko x yhtä suuri kuin 10 (TRUE)

x > 5 # Tarkistaa, onko x suurempi kuin 5 (TRUE)

x < 3 # Tarkistaa, onko x pienempi kuin 3 (FALSE)

### 2. Usean ehdon yhdistäminen

x <- 10

y <- 20

(x > 5) & (y < 30) # Molemmat ehdot totta (TRUE)

(x < 5) | (y > 15) # Ainakin yksi ehto totta (TRUE)

!(x == 10) # Käänteinen arvo (FALSE)

## Loogiset operaatiot vektoreissa

Loogiset operaatiot voidaan soveltaa myös vektoreihin, ja tulos on vektori, jossa on loogisia arvoja (**TRUE** tai **FALSE**).

**Esimerkki:**

# Vektorien luonti

a <- c(10, 20, 30)

b <- c(15, 20, 25)

# Yhtäsuuruus

a == b # FALSE TRUE FALSE

# Suuruusvertailu

a > 15 # FALSE TRUE TRUE

# Looginen JA (AND)

(a > 15) & (b < 30) # FALSE TRUE TRUE

# Looginen TAI (OR)

(a > 15) | (b < 30) # TRUE TRUE TRUE

**Kaavojen lyhennetyt muodot**

R tarjoaa myös lyhennettyjä loogisia operaattoreita erityisesti, kun ehtoja tarkastellaan **kokonaisessa vektorissa**:

| **Operaattori** | **Tarkoitus** | **Esimerkki** | **Tulos** |
| --- | --- | --- | --- |
| && | Lyhennetty AND | (5 > 3) && (2 < 4) | TRUE |
| ` |  | ` | Lyhennetty OR |

Erona on, että && ja || tarkistavat vain ensimmäiset elementit, kun taas & ja | käsittelevät koko vektorin.

**Esimerkki:**

# AND ja OR ensimmäiselle elementille

a <- c(TRUE, FALSE, TRUE)

b <- c(FALSE, TRUE, TRUE)

a && b # Tarkastaa vain ensimmäisen elementin (FALSE)

a || b # Tarkastaa vain ensimmäisen elementin (TRUE)

## Loogisten ehtojen hyödyntäminen

### 1. Ehtolauseet (if, else if, else)

Loogisia operaattoreita käytetään ehtolauseiden kanssa:

x <- 10

if (x > 5) {

print("x on suurempi kuin 5")

} else if (x == 5) {

print("x on yhtä suuri kuin 5")

} else {

print("x on pienempi kuin 5")

}

### 2. Tietojen suodattaminen

Voit käyttää loogisia ehtoja tietojen suodattamiseen vektoreista tai data.frameistä:

# Vektorin suodatus

numbers <- c(1, 2, 3, 4, 5, 6)

numbers[numbers > 3] # Palauttaa arvot: 4, 5, 6

# data.frame-suodatus

df <- data.frame(Name = c("A", "B", "C"), Age = c(20, 25, 30))

subset(df, Age > 20) # Palauttaa rivit, joissa ikä > 20

### Erikoistapaukset: NA ja NaN

Loogisissa vertailuissa **NA** (puuttuva arvo) voi tuottaa erityisiä tuloksia:

**Esimerkki:**

x <- NA

x == 10 # Palauttaa: NA

is.na(x) # Tarkistaa, onko x puuttuva arvo (TRUE)

**Käytännön esimerkki**

# Oletetaan lista arvosanoista

grades <- c(85, 92, 78, 90, 66, 58, 89)

# Tarkistetaan, ketkä saivat arvosanan yli 80

high\_grades <- grades[grades > 80]

print(high\_grades) # Tulostaa: 85, 92, 90, 89

# Ketkä saivat hylätyn (alle 60)?

failed <- grades[grades < 60]

print(failed) # Tulostaa: 58

# Print()-funktio R-kielessä

## Mikä on print()?

R-kielen **print()**-funktio on yksinkertainen ja tehokas tapa tulostaa tietoa konsoliin. Se on yksi yleisimmin käytetyistä funktioista ohjelmoinnissa, koska se auttaa tarkistamaan muuttujien arvot, debuggaamaan koodia ja näyttämään tuloksia.

## print()-funktion käyttö

print()-funktio ottaa yhden argumentin (tulostettavan objektin) ja näyttää sen arvon konsolissa. Se toimii kaikenlaisten objektien kanssa, kuten:

* Numerot
* Merkkijonot
* Vektorit
* Matriisit
* Data.frame-rakenteet

**Esimerkki:**

# Yksinkertaisia tulosteita

print(42) # Tulostaa: [1] 42

print("Hello, R!") # Tulostaa: [1] "Hello, R!"

# Muuttujan tulostus

x <- 100

print(x) # Tulostaa: [1] 100

## Tulostuksen oletusmuoto

Tulosteessa näkyvä **[1]** kertoo, että ensimmäinen arvo sijaitsee ensimmäisessä indeksissä (indeksointi alkaa aina yhdestä).

**Esimerkki:**

x <- c(10, 20, 30)

print(x)

# Tulostaa:

# [1] 10 20 30

## Tulostaminen ilman print()-funktiota

R tulostaa muuttujan arvon automaattisesti, jos kirjoitat vain muuttujan nimen:

x <- 50

x # Tulostaa: [1] 50

Mutta **print()** on hyödyllinen erityisesti funktioiden sisällä, koska funktiot eivät oletuksena tulosta arvoja ilman erillistä käskyä.

## print()-funktion käyttö funktioiden sisällä

Kun kirjoitat omia funktioita, voit käyttää **print()**-funktiota näyttääksesi välituloksia debuggausta tai tarkastelua varten.

**Esimerkki:**

my\_function <- function(x) {

print(paste("Annettu arvo on:", x))

return(x \* 2)

}

my\_function(10)

# Tulostaa:

# [1] "Annettu arvo on: 10"

# Palauttaa: 20

## Monimutkaisten rakenteiden tulostaminen

print() toimii myös monimutkaisempien objektien, kuten data.frame- tai listarakenteiden kanssa.

**Esimerkkejä:**

# Data.frame-tulostus

df <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Age = c(25, 30, 35)

)

print(df)

# Tulostaa:

# Name Age

# 1 Alice 25

# 2 Bob 30

# 3 Charlie 35

# Listan tulostus

my\_list <- list(a = 1:5, b = "Hello", c = TRUE)

print(my\_list)

# Tulostaa:

# $a

# [1] 1 2 3 4 5

#

# $b

# [1] "Hello"

#

# $c

# [1] TRUE

## Tulostuksen muotoiluvinkkejä

print()-funktio näyttää arvot sellaisenaan, mutta joskus tulostus halutaan muotoilla paremmin. Käytä esimerkiksi **paste()**- tai **sprintf()**-funktiota yhdessä print()-funktion kanssa.

**Esimerkkejä:**

# Käytä paste() yhdistämään tekstiä ja muuttujia

x <- 42

print(paste("Luvun arvo on:", x))

# Tulostaa: [1] "Luvun arvo on: 42"

# Käytä sprintf() tarkempaan muotoiluun

y <- 3.14159

print(sprintf("Pi-arvo on noin %.2f", y))

# Tulostaa: [1] "Pi-arvo on noin 3.14"

## Eroja print() ja muiden tulostusfunktioiden välillä

Vaikka **print()** on yleisin tapa tulostaa, R tarjoaa myös muita funktioita erityisiin käyttötapauksiin:

1. **cat()**
   * Yhdistää ja tulostaa tekstiä ilman [1]-merkintää.
   * Sopii yksinkertaisiin tekstitulosteisiin.
2. cat("Hello,", "world!", "\n")
3. # Tulostaa: Hello, world!
4. **message()**
   * Tulostaa viestejä, joita voidaan käyttää debuggaamiseen.
5. message("Tämä on viesti!")
6. # Tulostaa: Tämä on viesti!
7. **str()**
   * Näyttää objektin rakenteen tiivistetyssä muodossa.
8. str(df)
9. # Tulostaa:
10. # 'data.frame': 3 obs. of 2 variables:
11. # $ Name: chr "Alice" "Bob" "Charlie"
12. # $ Age : num 25 30 35

**Käytännön esimerkki**

**Esimerkki pseudodatan tulostamisesta:**

# Luo pseudodata

df <- data.frame(

Product = c("A", "B", "C"),

Sales = c(100, 200, 150)

)

# Tulosta koko taulukko

print(df)

# Tulosta vain yksittäinen sarake

print(df$Sales)

# Yhdistä teksti ja data

print(paste("Ensimmäisen tuotteen myynti:", df$Sales[1]))

# Vektorit (Vectors) R-kielessä

## Mikä on vektori?

Vektori on R:n perusdatatyyppi, joka on **samantyyppisten arvojen järjestetty kokoelma**. Vektorit voivat sisältää esimerkiksi numeroita, merkkijonoja tai loogisia arvoja, mutta eivät voi yhdistää eri tyyppisiä arvoja.

### Vektorin luominen

Vektori luodaan **c()**-funktiolla (combine).

**Esimerkkejä:**

# Numerovektori

numbers <- c(10, 20, 30, 40)

# Merkkijonovektori

names <- c("Alice", "Bob", "Charlie")

# Looginen vektori

logical\_values <- c(TRUE, FALSE, TRUE)

Tulosta vektorin sisältö:

print(numbers) # Tulostaa: [1] 10 20 30 40

### Vektorin tietotyypit

R tukee useita tietotyyppejä vektoreissa:

| **Tietotyyppi** | **Kuvaus** | **Esimerkki** |
| --- | --- | --- |
| numeric | Desimaaliluvut ja kokonaisluvut | c(1, 2.5, 3.14) |
| character | Teksti | c("apple", "banana") |
| logical | Loogiset arvot | c(TRUE, FALSE, TRUE) |
| integer | Kokonaisluvut | c(1L, 2L, 3L) (lisää L) |

**Sekatyyppiset arvot muunnetaan automaattisesti:**

mixed <- c(1, "two", TRUE)

print(mixed) # Tulostaa: [1] "1" "two" "TRUE" (kaikki merkkijonoina)

## Vektorin käsittely

### 1. Indeksointi

Vektorin tiettyyn elementtiin pääsee käsiksi hakasulkeilla **[]**.

# Esimerkkivektori

numbers <- c(10, 20, 30, 40)

# Pääsy tiettyihin elementteihin

numbers[1] # Ensimmäinen arvo: 10

numbers[2:3] # Toinen ja kolmas arvo: 20, 30

numbers[c(1, 4)] # Ensimmäinen ja neljäs arvo: 10, 40

### 2. Muokkaaminen

Voit päivittää vektorin elementtejä samalla tavalla.

numbers[2] <- 25 # Päivittää toisen arvon 25:ksi

print(numbers) # Tulostaa: [1] 10 25 30 40

### 3. Ehdollinen valinta

Voit valita elementtejä ehdon perusteella.

numbers <- c(10, 20, 30, 40)

# Valitse luvut, jotka ovat suurempia kuin 20

filtered <- numbers[numbers > 20]

print(filtered) # Tulostaa: [1] 30 40

## Operaatiot vektoreilla

R tukee monenlaisia laskutoimituksia vektoreilla.

### 1. Aritmeettiset operaatiot

Operaatiot suoritetaan elementtikohtaisesti:

a <- c(1, 2, 3)

b <- c(4, 5, 6)

sum <- a + b # [1] 5 7 9

diff <- a - b # [1] -3 -3 -3

prod <- a \* b # [1] 4 10 18

div <- b / a # [1] 4.0 2.5 2.0

### 2. Loogiset operaatiot

Vektorin arvoja voidaan vertailla loogisilla operaattoreilla:

x <- c(10, 20, 30)

x > 15 # [1] FALSE TRUE TRUE

x == 20 # [1] FALSE TRUE FALSE

x != 10 # [1] FALSE TRUE TRUE

### 3. Aggregaattifunktiot

x <- c(5, 10, 15, 20)

sum(x) # Summa: 50

mean(x) # Keskiarvo: 12.5

min(x) # Pienin arvo: 5

max(x) # Suurin arvo: 20

length(x) # Alkioiden lukumäärä: 4

## Erityistoiminnot vektoreilla

### 1. Vektorin luonti sarjoilla

Sarjojen avulla voit luoda vektoreita helposti:

# Sarjat

x <- 1:10 # [1] 1 2 3 4 5 6 7 8 9 10

y <- seq(1, 10, by = 2) # [1] 1 3 5 7 9

z <- rep(5, times = 4) # [1] 5 5 5 5

### 2. Vektorin lajittelu

x <- c(30, 10, 50, 20)

# Nouseva järjestys

sorted <- sort(x) # [1] 10 20 30 50

# Laskeva järjestys

desc\_sorted <- sort(x, decreasing = TRUE) # [1] 50 30 20 10

### 3. Vektorin yhdistäminen

a <- c(1, 2, 3)

b <- c(4, 5, 6)

combined <- c(a, b) # [1] 1 2 3 4 5 6

## Vektorin yhteensopivuus

R pyrkii automaattisesti sovittamaan eri pituiset vektorit yhteen laskutoimituksissa:

a <- c(1, 2, 3)

b <- c(10, 20)

# Lyhyempi vektori toistetaan automaattisesti

result <- a + b # [1] 11 22 13 (b toistettiin: [10 20 10])

## Käytännön esimerkkejä

### Esimerkki 1: Lämpötilojen analyysi

temps <- c(20, 22, 19, 25, 30)

# Laske keskiarvo

average\_temp <- mean(temps)

print(average\_temp) # Tulostaa: 23.2

# Valitse lämpötilat, jotka ovat yli 22

hot\_days <- temps[temps > 22]

print(hot\_days) # Tulostaa: [1] 25 30

### Esimerkki 2: Myyntidatan käsittely

sales <- c(100, 200, 150, 250)

# Yhteensä myyntiä

total\_sales <- sum(sales)

print(total\_sales) # Tulostaa: 700

# Tuotteet, joiden myynti oli yli 150

high\_sales <- sales[sales > 150]

print(high\_sales) # Tulostaa: [1] 200 250

# Listat (Lists) R-kielessä

## Mikä on lista?

Lista on R-kielessä monipuolinen tietorakenne, joka voi sisältää **erityyppisiä ja erikokoisia arvoja**. Toisin kuin vektorit, joissa kaikki arvot ovat saman tyyppisiä, lista voi yhdistää numeroita, merkkijonoja, loogisia arvoja, vektoreita, data.frame-objekteja tai muita listoja.

## Listan luominen

Lista luodaan **list()**-funktiolla.

**Esimerkki:**

# Yksinkertainen lista

my\_list <- list(10, "text", TRUE)

print(my\_list)

# Tulostaa:

# [[1]]

# [1] 10

#

# [[2]]

# [1] "text"

#

# [[3]]

# [1] TRUE

## Nimetyt listan elementit

Listaelementtejä voidaan nimetä, jotta niihin on helpompi viitata.

**Esimerkki:**

# Nimetty lista

person <- list(

name = "Alice",

age = 30,

is\_employed = TRUE

)

print(person)

# Tulostaa:

# $name

# [1] "Alice"

#

# $age

# [1] 30

#

# $is\_employed

# [1] TRUE

## Listan elementtien hakeminen

1. **Hakeminen indekseillä**  
   Käytä hakasulkeita **[[ ]]** tai **[ ]** hakemiseen.
2. my\_list <- list(10, "text", TRUE)
3. my\_list[[1]] # Hakee ensimmäisen elementin (10)
4. my\_list[[2]] # Hakee toisen elementin ("text")
5. **Hakeminen nimillä**  
   Jos listaelementeillä on nimet, niitä voidaan hakea $-operaattorilla.
6. person$name # Hakee arvon "Alice"
7. person$is\_employed # Hakee arvon TRUE
8. **Hakeminen useista elementeistä**  
   Useamman elementin hakemiseen käytetään yksinkertaisia hakasulkeita [ ], jolloin tulos on alilista.
9. sublist <- my\_list[1:2]
10. print(sublist) # Tulostaa alilistan, joka sisältää ensimmäisen ja toisen elementin

## Listan elementtien muokkaaminen

1. **Muokkaa olemassa olevaa elementtiä**:
2. person$age <- 35 # Päivittää iän arvoksi 35
3. person[[1]] <- "Bob" # Päivittää nimen arvoksi "Bob"
4. **Lisää uusi elementti**:
5. person$city <- "Helsinki" # Lisää uuden elementin 'city'
6. **Poista elementti**:
7. person$age <- NULL # Poistaa 'age'-elementin

## Listan yhdistäminen

Listoja voidaan yhdistää **c()**-funktiolla.

list1 <- list(1, 2, 3)

list2 <- list("a", "b", "c")

combined <- c(list1, list2)

print(combined)

# Tulostaa:

# [[1]]

# [1] 1

#

# [[2]]

# [1] 2

#

# [[3]]

# [1] 3

#

# [[4]]

# [1] "a"

#

# [[5]]

# [1] "b"

#

# [[6]]

# [1] "c"

## Listan sisäkkäisyys

Listan elementit voivat itsessään olla listoja, jolloin saadaan hierarkkinen rakenne.

**Esimerkki:**

nested\_list <- list(

numbers = list(1, 2, 3),

texts = list("a", "b", "c")

)

print(nested\_list)

# Tulostaa:

# $numbers

# $numbers[[1]]

# [1] 1

#

# $numbers[[2]]

# [1] 2

#

# $numbers[[3]]

# [1] 3

#

# $texts

# $texts[[1]]

# [1] "a"

#

# $texts[[2]]

# [1] "b"

#

# $texts[[3]]

# [1] "c"

Hakeminen sisäkkäisestä listasta:

nested\_list$numbers[[2]] # Hakee arvon 2

nested\_list$texts[[3]] # Hakee arvon "c"

## Listan muunnokset

1. **Muuta lista vektoriksi**:
2. my\_list <- list(1, 2, 3)
3. vector <- unlist(my\_list)
4. print(vector) # Tulostaa: [1] 1 2 3
5. **Muuta lista data.frame-muotoon**:
6. my\_list <- list(name = c("Alice", "Bob"), age = c(30, 25))
7. df <- as.data.frame(my\_list)
8. print(df)
9. # Tulostaa:
10. # name age
11. # 1 Alice 30
12. # 2 Bob 25

## Listan tutkiminen

1. **Näytä listan rakenne**:
2. str(person)
3. # Tulostaa:
4. # List of 3
5. # $ name : chr "Alice"
6. # $ age : num 30
7. # $ is\_employed: logi TRUE
8. **Listan pituus**:
9. length(my\_list) # Palauttaa elementtien lukumäärän

## Käytännön esimerkkejä

### Esimerkki 1: Opiskelijadata listassa

student <- list(

name = "Anna",

grades = c(90, 85, 92),

passed = TRUE

)

# Tulosta nimi

print(student$name)

# Laske keskiarvo arvosanoista

average\_grade <- mean(student$grades)

print(average\_grade) # Tulostaa: 89

### Esimerkki 2: Sisäkkäiset listat

# Tuoteryhmien lista

products <- list(

electronics = list("phone", "laptop"),

groceries = list("bread", "milk", "cheese")

)

# Tulosta kaikki elektroniikkatuotteet

print(products$electronics)

# Lisää uusi tuote

products$electronics <- c(products$electronics, "tablet")

print(products$electronics)

# Matriisit (Matrices) R-kielessä

## Mikä on matriisi?

Matriisi on R-kielessä kaksiulotteinen tietorakenne, jossa **kaikki arvot ovat saman tyyppisiä** (esim. numerot, merkkijonot tai loogiset arvot). Se voidaan ajatella taulukkona, jossa on rivejä ja sarakkeita.

## Matriisin luominen

Matriisi luodaan **matrix()**-funktiolla.

### Perusmuoto

matrix(data, nrow, ncol, byrow = FALSE)

* **data**: Arvot, jotka täyttävät matriisin.
* **nrow**: Rivien määrä.
* **ncol**: Sarakkeiden määrä.
* **byrow**: Jos **TRUE**, arvot täytetään rivi kerrallaan; muuten sarake kerrallaan.

## Esimerkkejä

### 1. Matriisi numeerisilla arvoilla

# Matriisi, jossa arvot täytetään sarake kerrallaan

m <- matrix(1:6, nrow = 2, ncol = 3)

print(m)

# Tulostaa:

# [,1] [,2] [,3]

# [1,] 1 3 5

# [2,] 2 4 6

### 2. Matriisi rivien mukaan täytettynä

m\_by\_row <- matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)

print(m\_by\_row)

# Tulostaa:

# [,1] [,2] [,3]

# [1,] 1 2 3

# [2,] 4 5 6

### 3. Merkkijonomatriisi

m\_text <- matrix(c("a", "b", "c", "d"), nrow = 2, ncol = 2)

print(m\_text)

# Tulostaa:

# [,1] [,2]

# [1,] "a" "c"

# [2,] "b" "d"

## Matriisin ominaisuudet

Voit tutkia matriisin rakennetta ja ominaisuuksia seuraavasti:

* **Rivien määrä**: nrow(m)
* **Sarakkeiden määrä**: ncol(m)
* **Matriisin koko (elementtien määrä)**: length(m)
* **Dimensiot (rivien ja sarakkeiden määrä)**: dim(m)

### Esimerkki

nrow(m) # 2

ncol(m) # 3

dim(m) # c(2, 3)

length(m) # 6

## Matriisin elementtien käsittely

### 1. Hakeminen

Voit käyttää hakasulkeita [row, column] päästäksesi yksittäisiin elementteihin, riveihin tai sarakkeisiin.

# Elementti riviltä 1, sarakkeesta 2

m[1, 2] # 3

# Kokonainen rivi

m[1, ] # c(1, 3, 5)

# Kokonainen sarake

m[, 2] # c(3, 4)

### 2. Elementtien muokkaaminen

# Päivitä arvo rivillä 2, sarakkeessa 3

m[2, 3] <- 10

print(m)

# Tulostaa:

# [,1] [,2] [,3]

# [1,] 1 3 5

# [2,] 2 4 10

## Matriisioperaatiot

R tukee monenlaisia matriiseihin liittyviä operaatioita, kuten aritmeettisia laskutoimituksia, transponointia ja matriisikertolaskuja.

### 1. Perusaritmetiikka

Aritmeettiset operaatiot toimivat elementtikohtaisesti:

# Matriisin luonti

m <- matrix(1:6, nrow = 2, ncol = 3)

# Yhteenlasku

m + 2

# [,1] [,2] [,3]

# [1,] 3 5 7

# [2,] 4 6 8

# Kertolasku

m \* 2

# [,1] [,2] [,3]

# [1,] 2 6 10

# [2,] 4 8 12

### 2. Matriisikertolasku

Käytä **%\*%**-operaattoria matriisikertolaskuun.

m1 <- matrix(1:4, nrow = 2)

m2 <- matrix(5:8, nrow = 2)

result <- m1 %\*% t(m2) # Huom! Käytetään transponointia

print(result)

# Tulostaa:

# [,1] [,2]

# [1,] 23 53

# [2,] 31 71

### 3. Transponointi

Matriisin kääntäminen (rivien ja sarakkeiden vaihto):

t(m)

# Tulostaa:

# [,1] [,2]

# [1,] 1 2

# [2,] 3 4

# [3,] 5 6

### 4. Rivien ja sarakkeiden summa tai keskiarvo

rowSums(m) # Rivien summat

colSums(m) # Sarakkeiden summat

rowMeans(m) # Rivien keskiarvot

colMeans(m) # Sarakkeiden keskiarvot

## Matriisin yhdistäminen

### 1. Rivien lisääminen

m <- matrix(1:6, nrow = 2)

new\_row <- c(7, 8, 9)

m\_combined <- rbind(m, new\_row)

print(m\_combined)

# Tulostaa:

# [,1] [,2] [,3]

# [1,] 1 3 5

# [2,] 2 4 6

# [3,] 7 8 9

### 2. Sarakkeiden lisääminen

new\_col <- c(10, 11)

m\_combined <- cbind(m, new\_col)

print(m\_combined)

# Tulostaa:

# [,1] [,2] [,3] [,4]

# [1,] 1 3 5 10

# [2,] 2 4 6 11

## Käytännön esimerkkejä

### Esimerkki 1: Taulukon analyysi

# Myyntidataa riveittäin: tuotteet, sarakkeina kuukaudet

sales <- matrix(c(100, 200, 150, 300, 400, 350), nrow = 2, byrow = TRUE)

rownames(sales) <- c("Product\_A", "Product\_B")

colnames(sales) <- c("Jan", "Feb", "Mar")

print(sales)

# Tulostaa:

# Jan Feb Mar

# Product\_A 100 200 150

# Product\_B 300 400 350

# Kuukausien summat

print(colSums(sales)) # Jan: 400, Feb: 600, Mar: 500

### Esimerkki 2: Opiskelijoiden arvosanat

grades <- matrix(c(85, 90, 78, 88, 92, 80), nrow = 2)

rownames(grades) <- c("Student\_1", "Student\_2")

colnames(grades) <- c("Math", "Science", "History")

print(grades)

# Tulostaa:

# Math Science History

# Student\_1 85 88 92

# Student\_2 90 78 80

# Keskiarvo opiskelijaa kohden

student\_means <- rowMeans(grades)

print(student\_means)

# Tulostaa:

# Student\_1 Student\_2

# 88 82.7

# Kuvaajat (Data Frames) R-kielessä

## Mikä on dataframe?

Dataframe on R:n yleisin tietorakenne, jota käytetään taulukkomuotoisen datan käsittelyyn. Se muistuttaa Excel-taulukkoa, jossa:

* **Rivit** vastaavat havaintoja.
* **Sarakkeet** vastaavat muuttujia.
* Jokainen sarake voi sisältää erilaista tietotyyppiä (numeroita, merkkijonoja, loogisia arvoja jne.).

## Dataframen luominen

### 1. Manuaalisesti

Voit luoda dataframen **data.frame()**-funktiolla.

df <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Age = c(25, 30, 35),

Employed = c(TRUE, TRUE, FALSE)

)

print(df)

# Tulostaa:

# Name Age Employed

# 1 Alice 25 TRUE

# 2 Bob 30 TRUE

# 3 Charlie 35 FALSE

### 2. CSV-tiedostosta

Voit ladata dataframen ulkoisesta tiedostosta, kuten CSV-tiedostosta.

df <- read.csv("data.csv")

print(head(df)) # Tulostaa ensimmäiset 6 riviä

### 3. Dataframen rakenne

Voit tarkastella dataframen rakennetta seuraavilla komennoilla:

str(df) # Näyttää datan rakenteen

summary(df) # Näyttää yhteenvetotilastot

dim(df) # Palauttaa rivien ja sarakkeiden määrän

nrow(df) # Rivien määrä

ncol(df) # Sarakkeiden määrä

colnames(df) # Sarakkeiden nimet

rownames(df) # Rivien nimet

#### Dataframen käsittely

##### 1. Pääsy yksittäisiin elementteihin

Käytä hakasulkeita **[row, column]** tai $-operaattoria.

# Pääsy yksittäiseen elementtiin

df[1, 2] # Ensimmäisen rivin toinen sarake: 25

# Pääsy kokonaiseen sarakkeeseen

df$Name # Sarake "Name": c("Alice", "Bob", "Charlie")

# Pääsy kokonaiseen riviin

df[2, ] # Toinen rivi

##### 2. Suodattaminen ehdon perusteella

Voit suodattaa rivejä ehtojen avulla.

# Suodata henkilöt, joiden ikä on yli 30

subset\_df <- subset(df, Age > 30)

print(subset\_df)

# Tulostaa:

# Name Age Employed

# 3 Charlie 35 FALSE

##### 3. Rivien ja sarakkeiden lisääminen

###### Uuden sarakkeen lisääminen:

df$Salary <- c(50000, 60000, 45000)

print(df)

# Tulostaa:

# Name Age Employed Salary

# 1 Alice 25 TRUE 50000

# 2 Bob 30 TRUE 60000

# 3 Charlie 35 FALSE 45000

###### Uuden rivin lisääminen:

new\_row <- data.frame(Name = "David", Age = 40, Employed = TRUE, Salary = 70000)

df <- rbind(df, new\_row)

print(df)

# Tulostaa:

# Name Age Employed Salary

# 1 Alice 25 TRUE 50000

# 2 Bob 30 TRUE 60000

# 3 Charlie 35 FALSE 45000

# 4 David 40 TRUE 70000

##### 4. Rivien ja sarakkeiden poistaminen

# Poista sarake

df$Salary <- NULL

# Poista rivi

df <- df[-2, ] # Poistaa toisen rivin

#### Dataframen muokkaaminen

##### 1. Sarakkeiden uudelleennimeäminen

colnames(df) <- c("FullName", "AgeYears", "IsEmployed")

print(df)

# Tulostaa:

# FullName AgeYears IsEmployed

# 1 Alice 25 TRUE

# 2 Bob 30 TRUE

# 3 Charlie 35 FALSE

##### 2. Data-arvojen muokkaaminen

df$AgeYears <- df$AgeYears + 1 # Lisää yhden jokaisen ikään

#### Data-analyysi dataframeilla

##### 1. Ryhmitys ja yhteenvedot

Voit käyttää **aggregate()**-funktiota ryhmäkohtaisiin laskelmiin.

# Esimerkki: Laske keski-ikä työllisyyden perusteella

aggregate(Age ~ Employed, data = df, FUN = mean)

# Tulostaa:

# Employed Age

# 1 FALSE 35

# 2 TRUE 27.5

##### 2. Ryhmittely ja summat dplyr-kirjastolla

Käytä **dplyr**-pakettia tehokkaampaan data-analyysiin.

library(dplyr)

df %>%

group\_by(Employed) %>%

summarise(

AvgAge = mean(Age),

Count = n()

)

# Tulostaa:

# # A tibble: 2 × 3

# Employed AvgAge Count

# <lgl> <dbl> <int>

# 1 FALSE 35 1

# 2 TRUE 27.5 2

#### Tietojen tallentaminen

##### 1. Tallentaminen CSV-tiedostoksi

write.csv(df, "output.csv", row.names = FALSE)

##### 2. Lataaminen R:n omaan tiedostomuotoon

save(df, file = "data.RData")

load("data.RData") # Lataa takaisin

#### Käytännön esimerkkejä

##### Esimerkki 1: Opiskelijoiden arvosanat

students <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Math = c(90, 85, 88),

Science = c(92, 80, 85),

History = c(88, 87, 90)

)

# Laske keskiarvo jokaiselle opiskelijalle

students$Average <- rowMeans(students[, c("Math", "Science", "History")])

print(students)

# Tulostaa:

# Name Math Science History Average

# 1 Alice 90 92 88 90.00

# 2 Bob 85 80 87 84.00

# 3 Charlie 88 85 90 87.67

##### Esimerkki 2: Myyntidatan analyysi

sales <- data.frame(

Product = c("A", "B", "C", "A", "B", "C"),

Month = c("Jan", "Jan", "Jan", "Feb", "Feb", "Feb"),

Revenue = c(100, 200, 150, 120, 220, 180)

)

# Laske kuukausittaiset kokonaistulot

monthly\_sales <- aggregate(Revenue ~ Month, data = sales, sum)

print(monthly\_sales)

# Tulostaa:

# Month Revenue

# 1 Feb 520

# 2 Jan 450

# Ohjausrakenteet (Flow Control) R-kielessä

## Mikä on ohjausrakenne?

Ohjausrakenteet määrittelevät ohjelman suoritusjärjestyksen ja päätöksenteon. R-kielessä yleisimpiä ohjausrakenteita ovat:

1. **Ehtolauseet**: if, else if, else
2. **Toistorakenteet**: for, while, repeat
3. **Päätöksenteon lyhennetyt muodot**: ifelse()

### 1. Ehtolauseet: if, else if, else

Ehtolauseiden avulla ohjelma suorittaa erilaisia toimintoja riippuen tietyistä ehdoista.

#### Yksinkertainen if-lause

x <- 10

if (x > 5) {

print("x on suurempi kuin 5")

}

# Tulostaa: "x on suurempi kuin 5"

#### if ja else

x <- 3

if (x > 5) {

print("x on suurempi kuin 5")

} else {

print("x on pienempi tai yhtä suuri kuin 5")

}

# Tulostaa: "x on pienempi tai yhtä suuri kuin 5"

#### Useita ehtoja: if, else if, else

x <- 7

if (x > 10) {

print("x on suurempi kuin 10")

} else if (x > 5) {

print("x on suurempi kuin 5 mutta pienempi tai yhtä suuri kuin 10")

} else {

print("x on 5 tai pienempi")

}

# Tulostaa: "x on suurempi kuin 5 mutta pienempi tai yhtä suuri kuin 10"

### 2. Lyhennetty ehtolause: ifelse()

ifelse()-funktiolla voit käyttää ehtolauseita vektorien kanssa.

**Esimerkki:**

numbers <- c(10, 3, 8, 6)

result <- ifelse(numbers > 5, "Suuri", "Pieni")

print(result)

# Tulostaa: [1] "Suuri" "Pieni" "Suuri" "Suuri"

### 3. Toistorakenteet

R tarjoaa kolme pääasiallista toistorakennetta: for, while, ja repeat.

#### 3.1 for-silmukka

for-silmukka toistaa toiminnon tietyn kokoelman läpi.

**Yksinkertainen for-silmukka**

# Tulosta luvut 1–5

for (i in 1:5) {

print(i)

}

# Tulostaa: 1, 2, 3, 4, 5

**for-silmukka vektoreiden kanssa**

fruits <- c("apple", "banana", "cherry")

for (fruit in fruits) {

print(paste("I like", fruit))

}

# Tulostaa:

# "I like apple"

# "I like banana"

# "I like cherry"

#### 3.2 while-silmukka

while-silmukka toistaa toiminnon niin kauan kuin ehto on **TRUE**.

**Esimerkki:**

x <- 1

while (x <= 5) {

print(x)

x <- x + 1

}

# Tulostaa: 1, 2, 3, 4, 5

#### 3.3 repeat-silmukka

repeat-silmukka jatkuu loputtomasti, kunnes se keskeytetään **break**-komennolla.

**Esimerkki:**

x <- 1

repeat {

print(x)

x <- x + 1

if (x > 5) {

break

}

}

# Tulostaa: 1, 2, 3, 4, 5

### 4. Keskeytys ja jatkaminen: break ja next

* **break**: Keskeyttää toiston.
* **next**: Ohittaa nykyisen iteraation ja siirtyy seuraavaan.

**Esimerkki break-komennosta:**

for (i in 1:10) {

if (i == 5) {

break

}

print(i)

}

# Tulostaa: 1, 2, 3, 4

**Esimerkki next-komennosta:**

for (i in 1:5) {

if (i == 3) {

next

}

print(i)

}

# Tulostaa: 1, 2, 4, 5 (ohittaa 3:n)

### 5. Pesäkkäiset silmukat

Toistorakenteita voidaan yhdistää toisiinsa.

for (i in 1:3) {

for (j in 1:2) {

print(paste("i =", i, ", j =", j))

}

}

# Tulostaa:

# "i = 1 , j = 1"

# "i = 1 , j = 2"

# "i = 2 , j = 1"

# "i = 2 , j = 2"

# "i = 3 , j = 1"

# "i = 3 , j = 2"

### Käytännön esimerkkejä

#### Esimerkki 1: Numeroiden luokittelu

numbers <- c(5, 10, 15, 20)

for (num in numbers) {

if (num < 10) {

print(paste(num, "on pieni luku"))

} else {

print(paste(num, "on suuri luku"))

}

}

# Tulostaa:

# "5 on pieni luku"

# "10 on suuri luku"

# "15 on suuri luku"

# "20 on suuri luku"

#### Esimerkki 2: Summa laskettuna while-silmukalla

sum <- 0

i <- 1

while (i <= 5) {

sum <- sum + i

i <- i + 1

}

print(sum) # Tulostaa: 15

#### Esimerkki 3: Ehtojen soveltaminen data.frameen

df <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Score = c(85, 50, 75)

)

df$Grade <- ifelse(df$Score >= 80, "Excellent",

ifelse(df$Score >= 60, "Good", "Fail"))

print(df)

# Tulostaa:

# Name Score Grade

# 1 Alice 85 Excellent

# 2 Bob 50 Fail

# 3 Charlie 75 Good

# Funktiot R-kielessä

## Mikä on funktio?

Funktio on R-kielessä koodin uudelleenkäytettävä yksikkö, joka suorittaa tietyn tehtävän. Funktiot tekevät koodista järjestelmällisempää ja helpottavat monimutkaisten laskelmien hallintaa.

## Funktioiden rakenne

Funktion perusrakenne R:ssä:

my\_function <- function(arg1, arg2, ...) {

# Koodilohko

result <- arg1 + arg2 # Esimerkki: kahden luvun yhteenlasku

return(result) # Palauttaa tuloksen

}

* **function**: Määrittää funktion.
* **arg1, arg2**: Parametrit, jotka otetaan syötteinä.
* **return**: Palauttaa funktion tuloksen.

## Funktion luominen ja käyttäminen

### Esimerkki 1: Yksinkertainen funktio

# Funktion määrittäminen

greet <- function(name) {

paste("Hello,", name)

}

# Funktion käyttäminen

greet("Alice") # Tulostaa: "Hello, Alice"

greet("Bob") # Tulostaa: "Hello, Bob"

### Esimerkki 2: Laskufunktio

# Funktion määrittäminen

add\_numbers <- function(a, b) {

sum <- a + b

return(sum)

}

# Funktion käyttäminen

result <- add\_numbers(5, 3)

print(result) # Tulostaa: 8

## Funktion parametrit

### 1. Oletusarvot

Funktiolle voidaan määrittää parametreille oletusarvot.

# Funktion määrittäminen

multiply <- function(a, b = 2) {

return(a \* b)

}

# Käyttäminen ilman b:tä

print(multiply(5)) # Tulostaa: 10

# Käyttäminen määrittämällä b

print(multiply(5, 3)) # Tulostaa: 15

### 2. Argumenttien nimeäminen

Parametreille voidaan antaa nimet funktiota kutsuttaessa.

# Argumenttien nimeäminen

print(multiply(b = 4, a = 3)) # Tulostaa: 12

## Funktion paluuarvo

Funktion tulos palautetaan yleensä **return()**-komennolla. Ilman return()-komentoa funktio palauttaa viimeisen suoritetun lausekkeen tuloksen.

**Esimerkki:**

# Ilman return

square <- function(x) {

x \* x # Viimeinen lauseke palautetaan

}

print(square(4)) # Tulostaa: 16

## Lokaalit ja globaalit muuttujat

Funktion sisällä määritellyt muuttujat ovat **lokaaleja**, eikä niitä voi käyttää funktion ulkopuolella.

**Esimerkki:**

add <- function(a, b) {

result <- a + b # Lokaali muuttuja

return(result)

}

add(3, 4) # Palauttaa: 7

print(result) # Virhe: 'result' ei ole olemassa globaalissa ympäristössä

## Anonyymit funktiot

R tukee myös anonyymejä (ilman nimeä määriteltyjä) funktioita, joita voidaan käyttää erityisesti sisäkkäisten laskentojen yhteydessä.

**Esimerkki:**

# Anonyymi funktio

numbers <- c(1, 2, 3, 4, 5)

# Kerrotaan jokainen luku kahdella

doubled <- sapply(numbers, function(x) x \* 2)

print(doubled) # Tulostaa: [1] 2 4 6 8 10

## Sisäänrakennetut funktiot

R sisältää monia valmiita funktioita, kuten:

* **Matematiikka**: mean(), sum(), prod(), sqrt()
* **Tekstinkäsittely**: paste(), substr(), toupper(), tolower()
* **Loogiset**: ifelse(), any(), all()
* **Data-analyysi**: aggregate(), apply(), sapply(), tapply()

## Soveltaminen data.frameissä

Funktiot ovat erityisen hyödyllisiä toistuvissa operaatioissa, kuten data-analyysissä.

**Esimerkki: Keskiarvon laskeminen riveittäin**

# Dataframe

df <- data.frame(

Math = c(90, 85, 88),

Science = c(92, 80, 85),

History = c(88, 87, 90)

)

# Sovella funktiota riveihin

row\_means <- apply(df, 1, mean)

print(row\_means)

# Tulostaa: [1] 90.00 84.00 87.67

## Käytännön esimerkkejä

### Esimerkki 1: Fahrenheit Celsius-asteiksi

# Fahrenheit to Celsius

fahrenheit\_to\_celsius <- function(f) {

c <- (f - 32) \* 5 / 9

return(c)

}

print(fahrenheit\_to\_celsius(100)) # Tulostaa: 37.77778

### Esimerkki 2: Ryhmän keskiarvojen laskenta

# Ryhmän keskiarvo

group\_means <- function(data, group\_col, value\_col) {

library(dplyr)

data %>%

group\_by(.data[[group\_col]]) %>%

summarise(mean\_value = mean(.data[[value\_col]], na.rm = TRUE))

}

# Esimerkkidata

df <- data.frame(

Group = c("A", "A", "B", "B"),

Value = c(10, 20, 30, 40)

)

group\_means(df, "Group", "Value")

# Tulostaa:

# # A tibble: 2 × 2

# Group mean\_value

# <chr> <dbl>

# 1 A 15

# 2 B 35

## Hyviä käytäntöjä

1. **Nimeä funktiot selkeästi**: Käytä kuvaavia nimiä, jotka kertovat, mitä funktio tekee.
   * Esimerkki: calculate\_total() on parempi kuin calc().
2. **Dokumentoi funktio**: Lisää kommentteja parametrien ja toiminnan selventämiseksi.
3. **Käytä oletusarvoja**: Aseta oletusarvot, kun se on mahdollista, käyttäjäystävällisyyden parantamiseksi.
4. **Testaa funktio**: Testaa funktiota erilaisilla syötteillä varmistaaksesi sen toiminnan.

# Datamanipulaatio R-kielessä

## Mikä on datamanipulaatio?

Datamanipulaatio tarkoittaa datan muokkaamista, suodattamista ja järjestämistä analyysia varten. R tarjoaa useita tehokkaita työkaluja ja paketteja, kuten **base R** ja **dplyr**, joiden avulla datan käsittely on suoraviivaista.

### 1. Perustoiminnot base R:llä

#### 1.1 Sarakkeiden ja rivien käsittely

**Sarakkeen valitseminen**

# Dataframe-esimerkki

df <- data.frame(

Name = c("Alice", "Bob", "Charlie"),

Age = c(25, 30, 35),

Score = c(80, 90, 85)

)

df$Name # Palauttaa sarakkeen "Name"

df[, "Age"] # Vaihtoehtoinen tapa valita sarake

**Rivien valitseminen**

df[1, ] # Palauttaa ensimmäisen rivin

df[2:3, ] # Palauttaa rivit 2 ja 3

**Rivien ja sarakkeiden poistaminen**

df <- df[-1, ] # Poistaa ensimmäisen rivin

df <- df[, -2] # Poistaa toisen sarakkeen

#### 1.2 Suodattaminen

# Valitse henkilöt, joiden ikä on yli 25

subset\_df <- df[df$Age > 25, ]

print(subset\_df)

#### 1.3 Sarakkeen lisääminen tai muokkaaminen

# Lisää uusi sarake

df$Category <- c("A", "B", "A")

# Muokkaa olemassa olevaa saraketta

df$Score <- df$Score + 5

#### 1.4 Tietojen järjestäminen

# Järjestä iän mukaan nousevasti

df\_sorted <- df[order(df$Age), ]

# Järjestä useamman sarakkeen mukaan

df\_sorted <- df[order(df$Category, -df$Score), ]

### 2. Datamanipulaatio dplyr-paketilla

**Mikä on dplyr?**

**dplyr** on suosittu paketti, joka tarjoaa intuitiivisen tavan datan manipulointiin. Tärkeimmät funktiot ovat:

* **filter()**: Rivien suodattaminen
* **select()**: Sarakkeiden valinta
* **mutate()**: Sarakkeiden lisääminen tai muokkaaminen
* **arrange()**: Rivien järjestäminen
* **summarise()** ja **group\_by()**: Yhteenvedot ryhmien mukaan

Asenna paketti, jos sitä ei ole vielä asennettu:

install.packages("dplyr")

library(dplyr)

#### 2.1 Rivien suodattaminen: filter()

# Valitse henkilöt, joiden ikä on yli 25

df\_filtered <- filter(df, Age > 25)

# Valitse usean ehdon perusteella

df\_filtered <- filter(df, Age > 25 & Score > 80)

#### 2.2 Sarakkeiden valinta: select()

# Valitse vain Name ja Score -sarakkeet

df\_selected <- select(df, Name, Score)

# Valitse kaikki paitsi Age

df\_selected <- select(df, -Age)

#### 2.3 Sarakkeiden lisääminen tai muokkaaminen: mutate()

# Lisää uusi sarake

df <- mutate(df, AdjustedScore = Score \* 1.1)

# Muokkaa olemassa olevaa saraketta

df <- mutate(df, Score = Score + 5)

#### 2.4 Rivien järjestäminen: arrange()

# Järjestä Score-sarakkeen mukaan nousevasti

df <- arrange(df, Score)

# Järjestä laskevasti

df <- arrange(df, desc(Score))

#### 2.5 Ryhmittely ja yhteenvedot: group\_by() ja summarise()

# Ryhmittele Categoryn mukaan ja laske keskiarvo Score:sta

df\_grouped <- df %>%

group\_by(Category) %>%

summarise(AverageScore = mean(Score))

# Laske useampia yhteenvedon mittareita

df\_grouped <- df %>%

group\_by(Category) %>%

summarise(

AverageScore = mean(Score),

TotalScore = sum(Score)

)

### 3. Yhdistely ja muotoilu

#### 3.1 Dataframien yhdistäminen

**Riveittäin yhdistäminen: rbind()**

df1 <- data.frame(Name = c("Alice", "Bob"), Age = c(25, 30))

df2 <- data.frame(Name = c("Charlie", "David"), Age = c(35, 40))

df\_combined <- rbind(df1, df2)

**Sarakkeittain yhdistäminen: cbind()**

df1 <- data.frame(Name = c("Alice", "Bob"))

df2 <- data.frame(Age = c(25, 30))

df\_combined <- cbind(df1, df2)

**Yhdistäminen avainarvon perusteella: merge()**

df1 <- data.frame(ID = c(1, 2), Name = c("Alice", "Bob"))

df2 <- data.frame(ID = c(1, 2), Score = c(80, 90))

df\_combined <- merge(df1, df2, by = "ID")

#### 3.2 Pitkä ja leveä muoto

**Muunna leveä muoto pitkäksi: pivot\_longer()**

library(tidyr)

df <- data.frame(

Name = c("Alice", "Bob"),

Math = c(90, 85),

Science = c(92, 80)

)

df\_long <- pivot\_longer(df, cols = c("Math", "Science"), names\_to = "Subject", values\_to = "Score")

**Muunna pitkä muoto leveäksi: pivot\_wider()**

df\_wide <- pivot\_wider(df\_long, names\_from = Subject, values\_from = Score)

### 4. Käytännön esimerkkejä

#### Esimerkki 1: Suodatus ja laskenta

# Suodata Score > 80 ja laske keskiarvo

filtered\_df <- filter(df, Score > 80)

mean\_score <- mean(filtered\_df$Score)

print(mean\_score)

#### Esimerkki 2: Ryhmittely ja järjestäminen

df <- data.frame(

Category = c("A", "A", "B", "B"),

Value = c(10, 20, 30, 40)

)

# Ryhmittele Categoryn mukaan ja järjestä tulokset

df\_summary <- df %>%

group\_by(Category) %>%

summarise(Total = sum(Value)) %>%

arrange(desc(Total))

print(df\_summary)

#### Esimerkki 3: Laaja manipulointi

# Luo data

df <- data.frame(

Product = c("A", "B", "C", "A", "B", "C"),

Month = c("Jan", "Jan", "Jan", "Feb", "Feb", "Feb"),

Sales = c(100, 200, 150, 120, 220, 180)

)

# Analyysi

result <- df %>%

group\_by(Product) %>%

summarise(

TotalSales = sum(Sales),

AvgSales = mean(Sales)

) %>%

arrange(desc(TotalSales))

print(result)

## Yhteenveto

Datamanipulaatio R:ssä on tehokasta, erityisesti base R:n ja **dplyr**-paketin avulla. Näitä työkaluja käytetään usein:

* **Base R**: Perustoiminnot, kuten suodatus, lajittelu ja laskenta.
* **dplyr**: Intuitiivinen syntaksi monimutkaisempaan datan käsittelyyn.
* **tidyr**: Tiedon muuntamiseen pitkän ja leveän muodon välillä.

# Datavisualisointi R-kielessä

## Mikä on datavisualisointi?

Datavisualisointi tarkoittaa datan esittämistä visuaalisessa muodossa, kuten kaavioina ja grafiikoina. Visualisoinnin tavoitteena on havainnollistaa tietoja ja tehdä niistä helposti ymmärrettäviä. R tarjoaa useita työkaluja datavisualisointiin, kuten:

1. **Base R**: Perusgrafiikat
2. **ggplot2**: Tehokas ja joustava visualisointikirjasto
3. **Lattice ja muut kirjastot**: Monimutkaisempia analyysejä varten

## 1. Visualisointi Base R:llä

R sisältää useita sisäänrakennettuja funktioita yksinkertaisten kaavioiden luomiseen.

### 1.1 Pylväsdiagrammi: barplot()

# Data

counts <- c(10, 20, 15)

categories <- c("A", "B", "C")

# Luo pylväsdiagrammi

barplot(counts, names.arg = categories, col = "blue",

main = "Pylväsdiagrammi", xlab = "Kategoria", ylab = "Määrä")

### 1.2 Histogrammi: hist()

# Data

data <- rnorm(1000, mean = 50, sd = 10)

# Luo histogrammi

hist(data, breaks = 20, col = "green",

main = "Histogrammi", xlab = "Arvot", ylab = "Tiheys")

### 1.3 Hajontakaavio: plot()

# Data

x <- 1:10

y <- x^2

# Luo hajontakaavio

plot(x, y, main = "Hajontakaavio", xlab = "x-akseli", ylab = "y-akseli",

col = "red", pch = 16)

### 1.4 Viivakaavio: lines()

# Data

x <- 1:10

y <- x^2

# Luo viivakaavio

plot(x, y, type = "o", col = "blue", main = "Viivakaavio",

xlab = "x-akseli", ylab = "y-akseli")

## 2. Visualisointi ggplot2:lla

**Mikä on ggplot2?**

**ggplot2** on yksi suosituimmista visualisointikirjastoista R:ssä. Se perustuu **"grammar of graphics"** -konseptiin, joka mahdollistaa joustavan ja kerroksittaisen lähestymistavan visualisointiin.

Asenna paketti tarvittaessa:

install.packages("ggplot2")

library(ggplot2)

### 2.1 Peruskaavio: ggplot()

#### Hajontakaavio

# Data

df <- data.frame(x = rnorm(100), y = rnorm(100))

# Luo hajontakaavio

ggplot(df, aes(x = x, y = y)) +

geom\_point(color = "blue") +

labs(title = "Hajontakaavio", x = "x-akseli", y = "y-akseli")

#### Viivakaavio

# Data

df <- data.frame(x = 1:10, y = cumsum(rnorm(10)))

# Luo viivakaavio

ggplot(df, aes(x = x, y = y)) +

geom\_line(color = "red") +

labs(title = "Viivakaavio", x = "x-akseli", y = "y-akseli")

### 2.2 Pylväsdiagrammi

# Data

df <- data.frame(Category = c("A", "B", "C"),

Count = c(10, 20, 15))

# Luo pylväsdiagrammi

ggplot(df, aes(x = Category, y = Count, fill = Category)) +

geom\_bar(stat = "identity") +

labs(title = "Pylväsdiagrammi", x = "Kategoria", y = "Määrä")

### 2.3 Histogrammi

# Data

data <- data.frame(Value = rnorm(1000))

# Luo histogrammi

ggplot(data, aes(x = Value)) +

geom\_histogram(binwidth = 5, fill = "green", color = "black") +

labs(title = "Histogrammi", x = "Arvot", y = "Frekvenssi")

### 2.4 Boxplot

# Data

df <- data.frame(

Category = rep(c("A", "B", "C"), each = 20),

Value = c(rnorm(20, mean = 5), rnorm(20, mean = 10), rnorm(20, mean = 15))

)

# Luo boxplot

ggplot(df, aes(x = Category, y = Value, fill = Category)) +

geom\_boxplot() +

labs(title = "Boxplot", x = "Kategoria", y = "Arvot")

### 2.5 Ryhmitelty visualisointi

# Data

df <- data.frame(

Group = rep(c("A", "B"), each = 10),

Category = rep(c("X", "Y"), times = 10),

Value = rnorm(20)

)

# Luo ryhmitelty pylväsdiagrammi

ggplot(df, aes(x = Category, y = Value, fill = Group)) +

geom\_bar(stat = "identity", position = "dodge") +

labs(title = "Ryhmitelty pylväsdiagrammi", x = "Kategoria", y = "Arvot")

## 3. Interaktiiviset visualisoinnit

**plotly**

Voit muuttaa ggplot2-grafiikat interaktiivisiksi käyttämällä **plotly**-pakettia.

install.packages("plotly")

library(plotly)

# Luo ggplot2-kaavio

p <- ggplot(df, aes(x = x, y = y)) +

geom\_point()

# Tee kaaviosta interaktiivinen

ggplotly(p)

## 4. Käytännön esimerkkejä

### Esimerkki 1: Myyntidata

# Data

sales <- data.frame(

Month = rep(c("Jan", "Feb", "Mar"), each = 3),

Product = rep(c("A", "B", "C"), times = 3),

Sales = c(100, 200, 150, 120, 220, 180, 130, 210, 170)

)

# Luo viivakaavio myynnistä

ggplot(sales, aes(x = Month, y = Sales, color = Product, group = Product)) +

geom\_line() +

geom\_point() +

labs(title = "Myynti kuukausittain", x = "Kuukausi", y = "Myynti")

### Esimerkki 2: Histogrammi ja tiheyskäyrä

# Data

data <- data.frame(Value = rnorm(1000))

# Luo histogrammi ja tiheyskäyrä

ggplot(data, aes(x = Value)) +

geom\_histogram(aes(y = ..density..), binwidth = 1, fill = "blue", alpha = 0.5) +

geom\_density(color = "red") +

labs(title = "Histogrammi ja tiheyskäyrä", x = "Arvot", y = "Tiheys")