

2. Grunnleggende konsepter i R

October 6, 2025

```
[ ]: renv::autoload()
```

0.1 Grunnleggende konsepter i R

0.1.1 Objekter og objekttyper

I R kan nesten alt betraktes som et objekt. Objekter kan inneholde ulike typer data og de vanligste objekttypene inkluderer:

- **Numeriske verdier:** representerer tall, enten heltall (**integer**) eller flyttall (**double**). F.eks. 4 og -2.5
- **Karakterverdier:** representerer tekststrenger (**character**). Disse er omgitt av enten enkle eller doble anførselstegn. F.eks. "Dette er en tekst" og 'R er nyttig'.
- **Boolske verdier:** representerer sannhetsverdier (**logical**). Det er kun to mulige verdier: TRUE eller FALSE.

Et objekt opprettes ved å gi det et navn (f.eks. `numerisk_objekt`) etterfulgt av en pil (`<-`). Innholdet man ønsker å lagre i objektet kommer etter pilen. Det er imidlertid noen regler og forbehold for navngivning av objekter i R:

- Kan kun inneholde bokstaver, tall, understrek og punktum
- Må starte med en bokstav eller et punktum
- Skiller mellom store og små bokstaver
- Unngå reserverte ord (f.eks. TRUE/FALSE/NA og navn på eksisterende funksjoner)

Numeriske verdier R fungerer som en kalkulator der det er mulig å gjøre beregninger:

```
[ ]: 2 + 2
```

```
[ ]: 5 - 2
```

```
[ ]: 25 * 4
```

```
[ ]: 100 / 5
```

```
[ ]: 5 ^ 2
```

Parenteser i numeriske regnestykker fungerer på samme måte som i matematiske formler.

```
[ ]: 5 + 3 * 10 / 2 - 4 ^ 2
```

```
[ ]: (5 + 3) * (10 / 2) - (4 ^ 2)
```

Alt som kan printes i konsollen kan lagres i et objekt og kalles på senere:

```
[ ]: numerisk_objekt <- 5
numerisk_objekt
class(numerisk_objekt)
```

```
[ ]: tall_1 <- 1
tall_2 <- 2
tall_1+tall_2
```

```
[ ]: teller <- 10
nevner <- 100
andel <- teller/nevner*100
andel
```

Funksjonen `round()` brukes til å avrunde desimaltall. Desimaltegnet i R er `.` (punktum) og ikke `,` (komma)

```
[ ]: tall_med_desimaler <- 10/3
tall_med_desimaler
```

```
[ ]: round(tall_med_desimaler, digits = 0)
```

```
[ ]: # OBS: runder "av", ikke "opp"
round(0.5, digits = 0)
round(1.5, digits = 0)
round(2.5, digits = 0)
round(3.5, digits = 0)
```

Karakterverdier

```
[ ]: karakter_objekt <- "Kongsvinger"
karakter_objekt

class(karakter_objekt)
```

Boolske verdier (TRUE/FALSE) En boolsk verdi kan være en av to mulige tilstander: enten TRUE eller FALSE. Disse verdiene brukes for å representere sannhetsverdier i logiske operasjoner og betingelser.

Vanlige betingelser i R er:

- `==` - er det lik?
- `!=` - er det ikke lik?
- `>` - er det større enn?
- `>=` - er det større enn eller lik?
- `<` - er det mindre enn?

- `<=` - er det mindre enn eller lik?
- `%in%` - er det lik én av flere verdier?
- `is.na()` - er det missing (NA)?
- `!is.na()` - er det ikke missing (NA)?

```
[ ]: "Kongsvinger" == "Kongsvinger" # test for likhet
```

```
[ ]: "Kongsvinger" == "kongsvinger" # test for likhet
```

```
[ ]: !("Kongsvinger" == "Oslo") # test for motsetning
```

```
[ ]: boolsk_objekt <- "Kongsvinger" != "Oslo" # test for motsetning
boolsk_objekt
class(boolsk_objekt)
```

0.1.2 Endre objekttype

Det er mulig å endre objekttype fra karakter til numerisk (eller motsatt) med funksjonene `as.character()`, `as.numeric()`, `as.integer()`

- `as.character()`
- `as.numeric()`
- `as.integer()`

```
[ ]: as.character(2024)
```

```
[ ]: as.numeric(substr(2024, 3, 4))
```

```
[ ]: as.integer(1.5) # OBS: avrunder ikke, beholder kun sifrene før desimaltegnet
```

0.1.3 Vektorer

En vektor er en samling av elementer av samme objekttype og kan bl.a. være numerisk, karakter eller logisk. De er grunnleggende dataobjekter i R og kan opprettes ved hjelp av funksjonen `c()`. Elementene i vektoren ramses opp inne i parentesen og separeres med komma.

```
[ ]: numerisk_vektor <- c(28, 34, 22:25)
class(numerisk_vektor)
numerisk_vektor
```

```
[ ]: karakter_vektor <- c("Anna", "Bjørn", "Cecilie")
class(karakter_vektor)
karakter_vektor
```

```
[ ]: boolsk_vektor <- c(TRUE, FALSE, TRUE)
class(boolsk_vektor)
boolsk_vektor
```

Se hva som skjer når man kombinerer numerisk og karakter i samme vektor:

```
[ ]: c(28, "34", 22)
      c("Anna", 11, "Cecilie")
```

For å sjekke hvor mange elementer en vektor inneholder kan man bruke funksjonen `length()`. Fra en vektor er det mulig å hente ut et spesifikt element ut ifra posisjonen den har. Dersom man ønsker å hente ut det andre elementet i en vektor bruker man hakeparentes med tallet 2 i etter navnet på objektet.

```
[ ]: length(karakter_vektor)
```

```
[ ]: karakter_vektor[2]
```

Test for innhold i en vektor For å sjekke om en gitt verdi finnes i en vektor med flere elementer brukes tegnet `%in%`. Resultatet blir en boolsk verdi (`TRUE` eller `FALSE`) om verdien man har oppgitt finnes i vektoren.

```
[ ]: "Bjørn" %in% karakter_vektor
```

```
[ ]: "Anne" %in% karakter_vektor
```

0.1.4 Data Frames

En data frame er en tabellstruktur som kan inneholde kolonner med forskjellige typer data. Det er en av de mest brukte datastrukturene i R for å håndtere datasett. Hver kolonne i en data frame er en vektor og kan derfor kun inneholde data av samme objekttype. De ulike kolonnene kan derimot inneholde forskjellige objekttyper.

- `data.frame()`: funksjonen tar én eller flere argumenter som kan være vektorer, faktorer, eller andre data frames. Hvert argument blir en kolonne i data framen, og alle kolonnene må ha samme lengde.

```
[ ]: data <- data.frame(
      navn = c("Anna", "Anna", "Bjørn", "Cecilie"),
      alder = c(28, 30, 34, NA),
      er_student = c(TRUE, TRUE, FALSE, TRUE)
    )

class(data)
data
```

For å hente ut én enkelt kolonne fra en data frame som en vektor skriver man først navnet på datasettet etterfulgt av et dollartegn (`$`) og så navnet på kolonnen:

```
[ ]: data$navn
      data$alder
      data$er_student
```

Man kan også opprette nye kolonner i en eksisterende data frame:

```
[ ]: data$jaar <- 2024
data$jaar
```

```
[ ]: unique(data$jaar)
```

På samme måte om man kan hente ut enkelte elementer fra en vektor kan man hente ut valgte rader og kolonner fra en data frame med `[]`. Her oppgir nummer på rad og kolonne separert med komma, f.eks. rad 3 kolonne 1:

```
[ ]: data[3,1]
```

Om man ønsker å hente ut en enkelt rad oppgir man kun nummeret til raden etterfulgt av et komma, men lar kolonnennummeret stå blankt:

```
[ ]: data[2,]
```

Tilsvarende kan man kun hente ut en valgt kolonne ved å kun oppgi kolonnennummeret, men la radnummeret stå blankt:

```
[ ]: data[,1]
```

Få oversikt over dataene (deskriptiv statistikk)

- `summary()`: funksjon som gir en rask oversikt over viktige statistiske mål for ulike typer R-objekter, som data frames, vektorer, faktorer, og modeller.
- `glimpse()`: gir en horisontal visning av datasettet, som viser kolonnenavn, datatyper, og en rask forhåndsvisning av dataene (de første få verdiene). Denne visningen er mer kompakt og gjør det enkelt å få en rask oversikt over strukturen på datasettet, spesielt når du har mange kolonner.
- `describe()`: Gir en mer omfattende beskrivelse av variablene enn `summary()`. Den inkluderer flere statistiske mål, spesielt for numeriske og kategoriske data (viser frekvensen og prosentandelen av hver kategori). Viser også informasjon om hvor mange observasjoner som mangler (NA), noe som er viktig for datarensing og forståelse av datakvaliteten.

```
[ ]: summary(data)
summary(data$alder)
data$alder
```

```
[ ]: nrow(data)
ncol(data)
dim(data)
```

```
[ ]: colnames(data)
tibble::glimpse(data)
Hmisc::describe(data)
```

0.1.5 Strengbehandling

Lim objekter sammen til en karakterstreng

- `paste0()`: brukes til å lime sammen (konkatenerere) tekststrenger uten mellomrom mellom dem. Funksjonen `paste()` limer sammen karakterstrenger og legger automatisk til mellomrom. `paste0()` kan være nyttige for å f.eks. lage filstier som ikke har hardkodet hvilken årgang filen tilhører.
- `glue()`: kan også brukes til å lime sammen tekststrenger og ligner mer på måten dette gjøres i Python. Fordelen med `glue` er at du kan bruke `{}` for å referere til variabler direkte inne i strengen, noe som gjør koden mer lesbar og enklere å vedlikeholde enn hvis du brukte tradisjonell strengkonkatasjon.

```
[ ]: navn <- "Svenn"
alder <- 45

paste0("Svenn er ", 45, " år")
paste0(navn, " er ", alder, " år")
paste(navn, "er", alder, "år")

glue::glue("{navn} er {alder} år")

[ ]: aargang <- 2024
filsti <- paste0("C:/Data/Prosjekt/Aargang_", aargang, "/resultater_", aargang,
  ↪ ".csv")
filsti
```

Plukk ut deler av en streng

- `nchar()`: antall karakterer i en streng
- `substr()`: hent ut deler av en streng ut ifra posisjon (streng, fra-og-med, til-og-med)

```
[ ]: nchar("2024")
```

```
[ ]: substr("2024", start = 1, stop = 4)
substr(2024, 3, 4)
```

Legg til ledende null

- `str_pad()`: brukes til å fylle ut (padde) en streng med tegn slik at den får en spesifisert lengde. Denne funksjonen er nyttig når du ønsker at alle strenger skal ha samme lengde ved å legge til tegn på begynnelsen eller slutten av hver streng.

```
[ ]: kommunenummer <- 0301
kommunennummer

stringr::str_pad(kommunennummer, width = 4, "left", pad = "0")
```

Erstatt mønstre i en karakterstreng

- `gsub()`: lar deg søke etter et spesifikt mønster i en tekst og erstatte det med noe annet.

```
[ ]: tekst <- "Trøndelag - Trööndelaget"
ny_tekst <- gsub(" - Trööndelaget", "", tekst)
ny_tekst
```

```
[ ]: tekst <- "Min adresse er 1234 Eksempelgata"
ny_tekst <- gsub("[0-9]", "", tekst) # erstatter alle tall
ny_tekst
```

Finn og hent ut mønstre i en karakterstreng

- `str_extract()`: brukes til å trekke ut deler av en streng som matcher et gitt mønster (regulært uttrykk, eller regex). Returnerer kun den første matchen for det regulære uttrykket i hver streng.
- `str_extract_all()`: returnerer alle matchene for det regulære uttrykket i hver streng.
- `str_detect()`: gir TRUE/FALSE om mønsteret finnes i strengen

```
[ ]: tekst <- "Min adresse er 1234 Eksempelgata 123"
ny_tekst <- stringr::str_extract(tekst, "[0-9]+")
ny_tekst

stringr::str_extract_all(tekst, "[0-9]+")
```

```
[ ]: tekst <- "Min adresse er 1234 Eksempelgata 123"
ny_tekst <- stringr::str_detect(tekst, "[0-9]+")
ny_tekst
```

0.1.6 If-setninger

I R kan du bruke boolske verdier i betingelser som `if`-setninger, løkker, eller for å filtrere data. I eksempelet under kjøres kun koden dersom betingelsen for at årgang er lik 2023 oppfylles.

```
[ ]: aargang <- 2023

if (aargang == 2023){ # Betingelsen i parentes
  print("Kjør denne koden for 2023-årgangen") # Koden som kjøres dersom
  ↪betingelsen er TRUE (inne i krøllparentes)
}
```

En `if`-setning i R brukes til å utføre en bestemt kodeblokk basert på en betingelse. Hvis betingelsen er sann (TRUE), vil koden i `if`-blokken bli utført. Hvis betingelsen er usann (FALSE), kan du bruke en `else`-blokk til å utføre en alternativ kode.

```
[ ]: if (5 > 2){
  print("Fem er større enn to")
  print("Dette er innenfor if-setningen")
} else {
  print("Dette er utenfor if-setningen")
}
```

```
[ ]: if (5 < 2){  
    print("Fem er større enn to")  
    print("Dette er innenfor if-setningen")  
} else {  
    print("Dette er utenfor if-setningen")  
}
```