Kort introduksjon til R

Kjartan Kloster Osmundsen
Universitetet i Stavanger

5. april 2018

Innhold

1 Operatorer og variabler

R støtter alle standard aritmetiske operasjoner, og kan brukes på samme måte som en kalkulator:

```
2 + 2

2 * 2.4

(5.7 - 2)/2

2^3
```

En variabel kan tilordnes en numerisk verdi ved bruk av = operatoren, som i de fleste andre programmeringsspråk:

```
x = 2 + 2

y = 14.2

z = 2 * x + 3 * y
```

Det er hensiktsmessig å lagre numeriske verdier som variabler hvis disse verdiene skal brukes for videre beregninger, som illustrert i det siste eksempelet med `z`-variabelen.

I stedet for å tilordne verdier med = operatoren, er det også mulig å oppnå det samme ved å bruke <− operatoren:

```
x <- 5
```

Det er ofte nyttig å overskrive verdien til en eksisterende variabel, for eksempel i forbindelse med telling:

```
x = 1

x = x + 1

x
```

Den første linjen tilordner verdien 1 til `x`-variabelen. Den andre linjen tilordner en ny verdi til denne variabelen, som er beregnet fra variabelens opprinnelige verdi. Den siste linjen, der vi bare skriver navnet til variabelen, forteller R at variabelverdien skal vises på skjermen, i dette tilfellet tallet 2.

2 Bruke eksisterende funksjoner

Standardinstallasjonen av R inkluderer en stor mengde funksjoner, og det finnes i tillegg et globalt arkiv av nedlastbare pakker som videre utvider bruksområdet til R.

Hver funksjon har en tilhørende manual, som finnes ved å skrive et spørsmålstegn foran navnet på funksjonen: ?`mean`
Denne manualen beskriver hvordan funksjonen brukes, og inneholder detaljer om funksjonens *argumenter*. Du bruker en funksjon ved å skrive inn argumenter i en parentes som etterfølger funksjonsnavnet:

$$\texttt{v} = \texttt{seq}(\texttt{from} = 1, \texttt{to} = 9, \texttt{by} = 2)$$

$$\texttt{meanOfVec} = \texttt{mean}(\texttt{v})$$

Her brukes `seq`-funksjonen med tre argumenter, mens `mean`-funksjonen brukes med ett argument. Begge funksjonene har ytterligere tilleggsargumenter som kan brukes om ønskelig.

$$\texttt{seq}(\texttt{from} = 1, \texttt{to} = 1, \texttt{by} = ((\texttt{to} - \texttt{from})/(\texttt{length.out} - 1)),$$
$$\texttt{length.out} = \texttt{NULL}, \texttt{along.with} = \texttt{NULL}, ...)$$

Ovenfor vises definisjonen for `seq`-funksjonen, hentet fra manualen som finnes ved kommandoen ?`seq`. Manualen inneholder også mer detaljert informasjon om hvert enkelt argument. Argumentene som etterfølges av et likhetstegn har en standardverdi (den verdien som står etter likhetstegnet), slik at man bare trenger å oppgi disse argumentene hvis man ønsker å avvike fra standardverdien. For `seq`-funksjonen ser vi at dette gjelder alle argumentene, så man kan faktisk bruke denne funksjonen uten å oppgi et eneste argument. Kommandoen `seq`() brukes for å kjøre funksjonen med alle standardverdiene.
Legg merke til at rekkefølgen til de ulike elementene kan ha stor betydning. Kommandoen $\texttt{seq}(1, 9, 2)$ er det samme som $\texttt{seq}(\texttt{from} = 1, \texttt{to} = 9, \texttt{by} = 2)$. Rekkefølgen er derimot irrelevant hvis man eksplisitt skriver navnet på hvert argument: $\texttt{seq}(\texttt{by} = 2, \texttt{to} = 9, \texttt{from} = 1)$ er identisk med de to foregående kommandoene.

2.1 R-pakker

De fleste funksjoner som nevnes i dette dokumentet er inkludert i standardinstallasjonen av R. Man kan også skaffe seg ytterligere funksjoner fra *CRAN* (The Comprehensive R Archive Network) *repository*, ved å laste ned *R-pakker*. En pakke inneholder en eller flere funksjoner, som er tilgjengelig for bruk etter at pakken er nedlastet og installert. Det finnes over 10 000 R-pakker tilgjengelig for nedlasting, hvor man finner pakker spesialtilpasset for bortimot alle tenkelige statistiske prosedyrer/modeller. Hvis du ikke finner en pakke som oppfyller behovene dine, kan du alltids lage din egen pakke (og laste den opp til CRAN)!

3 Datatyper

Utover skalarer, er det vektorer og matriser som er de mest nyttige datatypene.

3.1 Vektorer

En vektor er en indeksert samling av skalarer, og opprettes vanligvis ved hjelp av c()-funksjonen:

$\texttt{vec1} = \texttt{c}(1, 2, 3, 4, 5)$

$\texttt{myVec} = \texttt{c}(4, 7, 3.5, 0.1)$

Hvis vektorelementene har påfølgende verdier, som i det første eksempelet over, kan man også bruke følgende notasjon for å opprette vektoren:

$\texttt{v} = 1 : 5$

Det er også mulig å opprette en vektor med den største verdien først, slik at inkrementene blir negative:

$\texttt{v} = 5 : 1$

For andre inkrementer enn 1 og -1, kan man bruke funksjonen `seq`. Følgende kommando oppretter vektoren (1, 3, 5, 7, 9):

$\texttt{v} = \texttt{seq}(\texttt{from} = 1, \texttt{to} = 9, \texttt{by} = 2)$

Hvis du ønsker å opprette en vektor der alle elementene er identiske, er funksjonen `rep` praktisk:

$\texttt{v} = \texttt{rep}(\texttt{x} = 4, \texttt{times} = 5)$

Vektorelementer og delvektorer hentes ut ved hjelp av klammeparentes og indekser. $\texttt{v}[1]$ gir det første elementet av v, mens $\texttt{v}[1:3]$ er en delvektor av v bestående av de tre første elementene. Det er også mulig å hente ut en delvektor ved å spesifisere hvilke elementer som *ikke* skal inkluderes: $\texttt{v}[-1]$ ekskluderer det første elementet, mens $\texttt{v}[-\texttt{c}(1, 3)]$ ekskluderer det første og det tredje elementet.
Vektorer kan også brukes i aritmetiske operasjoner, ved at operasjonene utfører elementvis:

$\texttt{v} = 1 : 5$

$\texttt{v2} = 2 : 6$

$\texttt{v3} = \texttt{v} + \texttt{v2}$

$\texttt{v4} = (2 * \texttt{v})/(3 * \texttt{v2})$

Koden nedenfor øker hvert element av v med 2.5 og lagrer resultatet i en ny vektor:

$\texttt{newV} = \texttt{v} + 2.5$

Slike elementvise operasjoner kan også inkludere subtraksjon, multiplikasjon og divisjon.

3.2 Matriser

En matrise er en todimensjonal, indeksert samling av skalarer. Opprettes ved å bruke `matrix`-funksjonen:

$$\texttt{m} = \texttt{matrix}(1:9, \texttt{nrow} = 3, \texttt{ncol} = 3)$$

$$\texttt{m2} = \texttt{matrix}(\texttt{c}(4, 8, 2.5, 7), \texttt{nrow} = 2, \texttt{ncol} = 2)$$

Det er strengt tatt tilstrekkelig å spesifisere kun antall rader (`nrow`) eller kun antall kolonner (`ncol`), ettersom R kan bruke antall elementer i matrisen til å beregne den uspesifiserte dimensjonen.

Matriseelementer hentes ut ved hjelp av klammeparenteser som inneholder radindeks og kolonneindeks. $\texttt{m}[1, 1]$ gir verdien til elementet øverst til venstre i matrisen `m`, mens $\texttt{m}[3, 3]$ er verdien nederst til høyre. $\texttt{m}[1, 2]$ gir verdien til det første elementet i den andre kolonnen. Det er også mulig å hente ut submatriser:

$$\texttt{m3} = \texttt{m}[1:2, 1:2]$$

Man kan også benytte seg av kun én indeks, som er ekvivalent med å benytte seg av alle verdiene for den andre indeksen. Begge følgende kommandoer returnerer første raden av `m`:

$$\texttt{m}[1,]$$

$$\texttt{m}[1, 1:3]$$

Aritmetiske operatorer fungerer elementvis, som for vektorer:

$$\texttt{m4} = \texttt{m2} + \texttt{m3}$$

$$\texttt{m5} = (3 * \texttt{m3}) + 2.5$$

3.3 Data frames

Data frames har store likheter med matriser, ettersom de er navngitte lister av vektorer. Det er lett å konvertere en matrise til en data frame, og omvendt. Fordelen med data frame er at du kan referere til hver vektor med navn, i stedet for å spesifisere enn kolonneindeks. Elementene kan likevel spesifiseres med rad- og kolonneindeks, som for matriser. Vektorene som en data frame består av refereres til ved hjelp av et dollartegn:

$$\texttt{myDf} = \texttt{data.frame}(\texttt{apples} = \texttt{c}(1, 2, 3, 4), \texttt{oranges} = \texttt{c}(5, 6, 7, 8))$$

$$\texttt{myApples} = \texttt{myDf\$apples}$$

Data frames er praktiske for å lagre data, mens matriser bør bruker for beregninger (matrisealgebra fungerer ikke med data frames).

4 Logiske operatorer

Utrykk med logiske operatorer returnerer `TRUE` eller `FALSE`:

$5 < 3$ (less) (returns `FALSE`)

$7 >= -2$ (greater or equal) (returns `TRUE`)

$3 == 2$ (equal) (returns `FALSE`)

$4 \,!= 4.2$ (not equal) (returns `TRUE`)

Logiske utrykk kan kombineres med or-operatoren (|) og and-operatoren (&). Hvis utrykket brukes til videre beregninger, behandles TRUE som verdien 1 og FALSE som verdien 0:

$2 * (5 < 3)$ (returns 0)

$1 + (4 \,!= 4.2)$ (returns 2)

$2 + \,!(4 >= 2)$ (returns 2)

$1 - [\{(4 < 3)|(3 > 2)\} \& (2 + 2 == 4)]$ (returns 0)

Logiske operatorer kan også brukes på vektorer/matriser, og resulterer i logiske vektorer/matriser hvor hvert element er enten TRUE eller FALSE (1 eller 0 hvis brukt for videre beregninger):

$\mathtt{v} = \mathtt{c}(1, 4, 7)$

$\mathtt{v2} = \mathtt{c}(0, 5, 2)$

$\mathtt{v} > \mathtt{v2}$

$\mathtt{m} = \mathtt{matrix}(1 : 9, \mathtt{nrow} = 3)$

$\mathtt{myCount} = \mathtt{sum}(\mathtt{m} >= 4)$

5 If/else

If/else bruker logiske uttrykk til å bestemme hvilke kommandoer som utføres av R:

```
a = runif(1)
if(a < 0.5)
{
   b = 2
   print("yes")
}
```

```
a = runif(1)
if(a < 0.5)
{
   b = 2
   print("yes")
} else
{
   b = 1
   print(no")
}
```

```
a = runif(1)
if(a < 0.5)
{
   b = 2
   print(small")
} elseif(a > 1.5)
{
   b = 0
   print(large")
} else
{
   b = 1
   print("medium")
}
```

Kommandoene som er omgitt av krøllparenteser blir bare utført hvis det logiske if-uttrykket evalueres til `TRUE`. En *if* kan også kombineres med en *else*, som inneholder kommandoer som skal utføres hvis det logiske uttrykket evalueres til `FALSE`. Man kan også ha en eller flere *else if* mellom *if* og *else*, for tilfeller med mer enn to utfall.

Enkle if/else-setninger (en linje med kode per krøllparentes) kan skrives på én linje:

```
if(a < 0.5){b = 2}else if(a > 1.5){b = 0}else{b = 1}
```

```
if(a < 0.5 & a > 0){b = 15}
```

6 For-løkke

En for-løkke brukes for å utføre lignende operasjoner gjentatte ganger. Hver *iterasjon* av for-løkken har en unik indeksverdi, noe som er nyttig for å kunne hente ut elementer fra vektorer. Navnet på samlingen av indeksverdier, samt selve indeksverdiene (en vektor), spesifiseres i parentesen som følger for-kommandoen. Indeksverdiene skrives oftest inn direkte, men man kan også benytte seg av en eksisterende vektor.

```
v = c(3, 8, 7, 5, 2)
sum = 0
for(i in 1 : 5)
{
   sum = sum + v[i]
}
```

```
v = c(3, 8, 7, 5, 2)
for(i in v)
{
   print(i)
}
```

```
m = matrix(1 : 9, nrow = 3)
for(i in 1 : 5)
{
   for(j in 1 : 5)
   {
      m[i, j] = 0.5 * m[i, j]
   }
}
```

En for-løkke kan inneholde en eller flere for-løkker, dette kalles en *nøstet* for-løkke. Hver for-løkke har sitt eget sett med indeksverdier, så det er viktig å gi dem unike navn. En nøstet for-løkke gjør det mulig å iterere over alle elementene i en matrise. Den *ytre* løkken kan iterere over radene, mens den *indre* løkken itererer over elementene i hver rad (eller omvendt).

7 Plotting

Et standard spredningsplott lages i R ved hjelp av funksjonen `plot`. Samme funksjon kan også produsere linjeplott, see funksjonens manual for mer detaljer.

```
a = 1 : 5
b = c(2, 4.7, 3, 8, 1.07)
plot(a, b)
```

```
x = seq(−3, 3, 0.05)
y = −3 + x^2
plot(x, y)
```

```
a = runif(100)
plot(a)
```

Det finnes også mange funksjoner som er laget for å produsere ulike typer plott. Noen eksempler er `barplot` for stolpediagram, `hist` for histogram og `pie` for kakediagram. For å lage ekstremt tiltalende plott, bør man ta en titt på R-pakken `ggplot2`, som tilbyr uendelige muligheter for grafiske framstillinger i R.

8 Scriptfiler

Så lenge man skal kode mer enn et par linjer, er det alltid best å skrive koden i en scriptfil. Det gir deg muligheten til å lagre koden, samt at alle kodelinjene enkelt kan utføres sekvensielt.

Kommentarer i scriptfiler innledes med #:

```
# This is a script file

# Creating a vector

v = c(1, 2, 4, 7)

# Multiplying each vector element by 4

v = 4 * v
```

Hver operasjon er avskilt med et linjeskift (det finnes ingen tegn for å indikere slutten av en operasjon, slik det er for mange andre programmeringsspråk).

9 Lage egne funksjoner

Man kan enkelt lage sine egne funksjoner i R:

```
myFunctionName = function(arg1, arg2, ..., argN)
{

    ⋮

    ⋮

    value = ...

    return(value)

}
```

10 Liste over vanlige funksjoner

Generelle funksjoner

Standard matematiske funksjoner: `sin`,`cos`,`tan`,`log`,`log10`,`exp`,`sqrt`

mean	Gjennomsnitt av en vektor/matrise
sd	Utvalgsstandardavvik av en vektor/matrise
median	Utvalgsmedian av en vektor/matrise
min/max	Minimumsverdi/maksimumsverdi av en vektor/matrise
quantile	Utvalgskvantil for gitte sannsynligheter
pmin/pmax	Elementvise minimums/maksimumsverdier av to eller flere vektorer/matriser av lik størrelse
var/cov/cor	Varians, kovarians og korrelasjon
summary	Beskrivende statistikk for et gitt datasett (min, median, mean, max, 1st and 3rd quantile)
sum	Summen av all elementer i en vektor/matrise
diff	Returnerer en vektor bestående av differansen mellom hvert element i en gitt vektor
length	Lengden av en vektor
nrow/ncol	Antall rader/kolonner i en matrise
sort	Sorterer vektorelementene i stigende/synkende rekkefølge
round	Avrunder til et gitt antall desimaler
abs	Absoluttverdi
table	Frekvenstabell
paste	Kombinerer tekst og tall for utskrift
print	Printer ut (viser) spesifiser variabel/objekt/tekst
library	Laster inn spesifisert R-pakke
lm	Lineær regresjon
anova	Variansanalyse
t.test	T-test
z.test	Z-test [krever R-pakken `BSDA`]

Opprette datatyper

c	Oppretter en vektor bestående av spesifiserte elementer
seq	Oppretter en vektor bestående av en numerisk rekke
rep	Oppretter en vektor ved å repetere en skalar/vektor et gitt antall ganger
matrix	Oppretter en matrise bestående av spesifiserte elementer
cbind/rbind	Slår sammen matriser/vektorer ved kolonner/rader
sample	Trekker tilfeldige elementer fra en gitt vektor, med/uten tilbakelegging, med/uten gitte sannsynligheter for hvert utfall

Plotting

plot	Standard plottefunksjon, for spredningsplott og linjeplott
barplot/hist/pie/boxplot	Stolpediagram, histogram, kakediagram og boksdiagram
qqnorm	Plotter empiriske kvantiler mot teoretiske normalkvantiler
xlim=, ylim=	Argumenter som spesifiserer øvre og nedre grenser for aksene
xlab=, ylab=	Argumenter som definerer aksetitlene
points/lines	Legger til punkter/linjer til eksisterende plott
abline	Tegner en spesifikk linje på et eksisterende plott (ax+b)

Matriseoperasjoner

t	Transponerer matrisen
diag	Returnerer matrisens diagonal
%*%	Matrisemultiplikasjon (A%*%B)
solve	Løser A %*% x = B, for x. Returnerer matriseinversen av A hvis B ikke er spesifisert.
rowSums/colSums	Returnerer summen av radene/kolonnene
rowMeans/colMeans	Returnerer gjennomsnittet av radene/kolonnene

Filhåndtering

read.table	Importerer tekstfil til data frame
write.table	Eksporterer vektor/matrise/data frame til tekstfil
read.xlsx	Importerer Excel-fil til data frame [krever R-pakken `openxlsx`]
write.xlsx	Eksporterer vektor/matrise/data frame til Excel-fil [krever R-pakken `openxlsx`]

Sannsynlighetsfordelinger

Prefiksene d,p,q and r er for funksjoner som returnerer henholdsvis punktsannsynlighet/sannsynlighetstetthet, kumulative sannsynlighet, kvantilfunksjon og tilfeldig genererte tall, fra den gitte sannsynlighetsfordelingen.

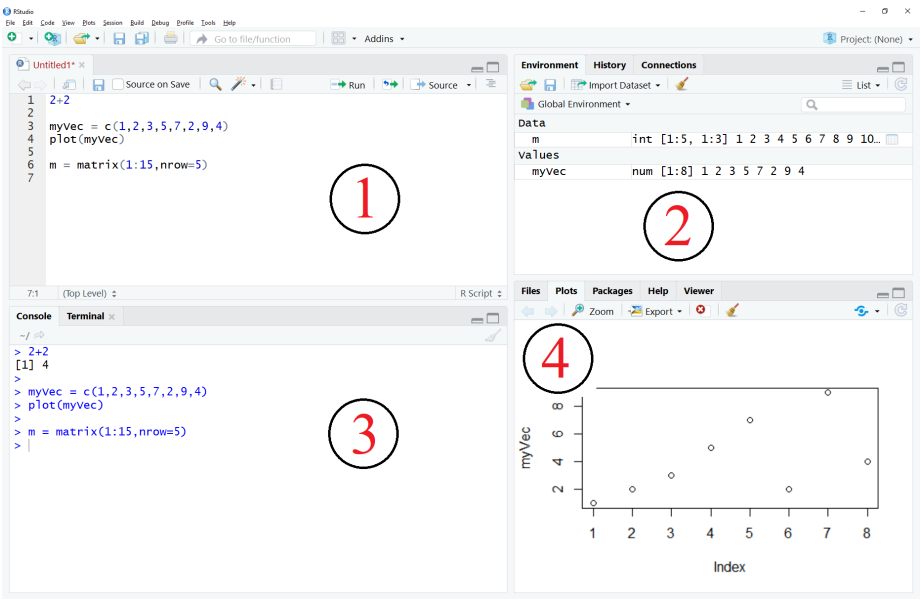
dbinom/pbinom/qbinom/rbinom	Binomisk fordeling
dhyper/phyper/qhyper/rhyper	Hypergeometrisk fordeling
dgeom/pgeom/qgeom/rgeom	Geometrisk fordeling
dpois/ppois/qpois/rpois	Poissonfordeling
dexp/pexp/qexp/rexp	Eksponentialfordeling
dnorm/pnorm/qnorm/rnorm	Normalfordeling
dt/pt/qt/rt	Student-t fordeling
dunif/punif/qunif/runif	Uniform fordeling

11 Brukergrensesnitt: Rstudio

Rstudio er et populært grafisk brukergrensesnitt for R. Figur 1 viser et skjermbilde av arbeidsområdet i Rstudio, som er delt opp i (1) et scriptfil-vindu, (2) en liste over data og variabler, (3) selve R-konsollen og (4) et grafvindu. Den aktive fanen øverst i hvert vindu styrer hvilket innhold som vises. Det mest sentrale innholdet som ikke vises på bildet er *Help*- og *Packages*-fanen i vinduet nederst til høyre.

Vinduet øverst til venstre er skjult hvis det ikke er en aktiv scriptfil. En ny scriptfil opprettes ved å velge *File* → *New File* → *R Script* (Ctrl+Shift+N). Filen lagres ved å velge *File* → *Save File* (Ctrl+S). Velg *File* → *Open File* (Ctrl+O) for å åpne en eksisterende fil. For å utføre scriptfilens kommandoer, markerer man de aktuelle kodelinjene (Ctrl+A for å velge alle) og trykker på *Run*-knappen (Ctrl+Enter).

R-konsollen i vinduet nederst til venstre viser både utførte kommandoer og tilhørende utskrifter (resultater fra beregninger, feilkoder, osv.). Det er også mulig å skrive og utføre kommandoer direkte fra konsollvinduet.

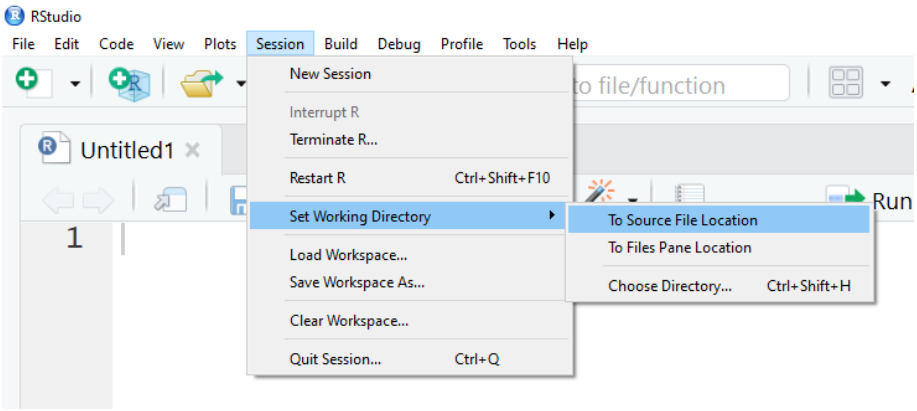


Figur 1: Arbeidsområdet i Rstudio.

11.1 Working directory

Hvis du ønsker å importere datafiler, er det viktig at Rstudio vet hvilken mappe datafilen ligger i. Det finnes to måter å informere Rstudio om dette, enten ved å eksplisitt skrive filstien, eller ved å definere mappen som *working directory*. Anta at vi har en datafil *datafile.txt* i mappen *C:/Rfiles/2018*. Følgende kommando importerer datafilen ved hjelp av den komplette filstien:

```
myData = read.table('C : /Rfiles/2018/datafile.txt')
```



Figur 2: Definering av working directory i Rstudio.

Alternativet er å definere mappen *C:/Rfiles/2018* som arbeidskatalog. Figur 2 viser hvordan man gjør dette fra menylinjen i Rstudio. Her kan man enten velge mappen der scriptfilen er lagret, eller velge mappe helt fritt ved hjelp av filutforskeren. Etter man har gjort dette, trenger man ikke spesifisere hvor datafilene ligger, så lenge de er plassert i mappen man har definert som arbeidskatalog:

```
myData = read.table('datafile.txt')
```

```
myData2 = read.table('datafile2.txt')
```

NB! Hvis man skal jobbe videre med en eksisterende scriptfil (og man ikke har åpnet Rstudio), kan man starte Rstudio ved å dobbeltklikke på filen. Da vil scriptfilens mappe automatisk defineres som workspace.