Innhold

1 Om R

R er et programmeringsspråk som fokuserer på statistikk og statistisk analyse. Det så dagens lyd ved University of Auckland, New Zealand hvor Ross Ihaka og Robert Gentleman startet på språket originalt.

2 Installasjon

R kan lastes ned fra https://cran.uib.no/. Installasjonen er rett fram, og enn kan akseptere standard innstillingene hele veien. Du har da R installert som en kommandolinje applikasjon, og et enkelt grensesnitt. Det anbefales å installere RStudio og bruke dette som utviklingsmiljø, RStudio kan lastes ned fra https://www.rstudio.com/products/rstudio/download/. Dette dokumentet tar utgangspunkt i en Windows klient.

3 Bruk av RStudio

I RStudio så jobber vi gjerne med «prosjekter», med flere underscript. Eksempelvis ville det være naturlig å opprette et eget prosjekt for hvert sett med ukesoppgaver i MA-155.

3.1 Nytt prosjekt

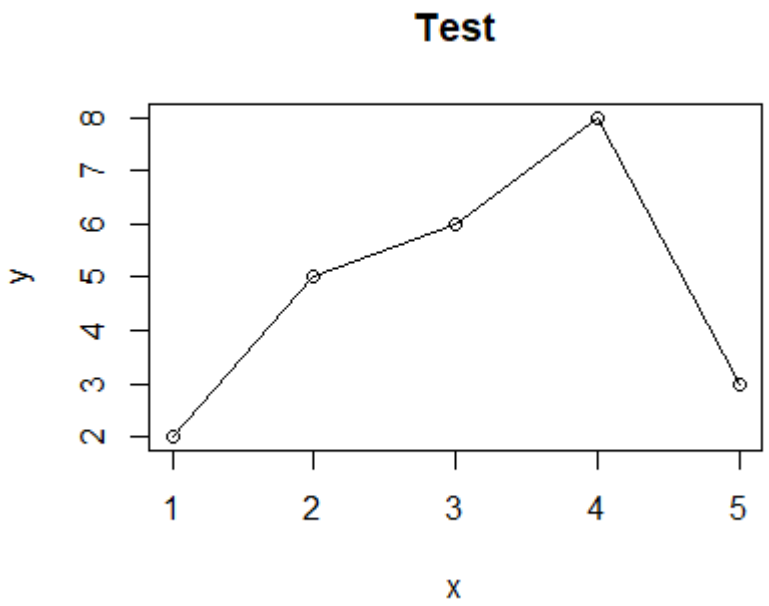
For å sette opp et nytt prosjekt velger man File > New Project > New Directory > Empty Project. Skriv så hva prosjektet skal hete i «Directory name» feltet, dette blir også navnet på mappen for prosjektet. Velg så stien til hvor du vil lagre det.

3.2 Test at RStudio fungerer

Opprett et nytt script ved å velge File > New file > R Script, og skriv inn

```
x <- (1:5)
y <- sample(1:10,5)
plot(x, y, main="Test")
lines(x, y)
```

marker teksten, og trykk CTRL+ENTER. Du skal nå se et enkelt linjediagram med 5 vilkårlige punkter i



y-aksen.

4 Grunnleggende R

Underveis vil det være to notasjonsformer for kode;

```
> x = 1
> x
[1] 1
```

, der > karakteren angir at dette er skrevet rett i kommandolinje og skal da unnlates for de som følger steg for steg. [1] referer til det som blir skrevet ut. Når en lenger sekvens med kode er angitt uten >, betegnes dette som et komplett script som enten kjøres i sin helhet eller som videre angitt.

4.1 Variabler

I R så kan vi tilordne verdier på to måter

```
x = 5
```

eller

```
x <- 5
```

Merk at vi ikke angir typen, dette ordner R for oss automatisk og en variabel kan da endre type underveis.

```
> x = 5
> x
[1] 5
> x = "abc"
> x
[1] "abc"
```

Alle vanlige aritmetiske operatører er med, addisjon (+), subtraksjon (−), multiplikasjon (∗), divisjon (/), potens (∗∗ eller ∧).

```
> a = 2
> b = 3
> a+b
[1] 5
> a-b
[1] -1
> a*b
[1] 6
> a/b
[1] 0.6666667
> a**b
[1] 8
> a^b
[1] 8
```

4.2 Vektorer

Vektorer fungerer på samme måte som arrays, lister og lignende fra andre språk med et unntak, i R så er vektorene 1 indeksert og ikke 0 indeksert. Det vil si første element i vektoren er på indeks 1.

```
> vec = c(4,3,2,1)
[1] 4 3 2 1
> vec[1]
[1] 4
```

Aritmetiske operasjoner fungerer på vektorer, eksempelvis

```
> vec**2
[1] 16 9 4 1
```

4.3 Matriser

Matriser kan angis så lenge alle elementene er av samme type på følgende måte

```
> A = matrix(nrow=3, ncol=2, c(1,2,3,4,5,6))
> A
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

her kunne enn unnlatt å skrive nrow eller ncol, så lenge en av disse er angitt vil R fordele verdiene så matrisen stemmer, med andre ord

```
A = matrix(nrow=3, c(1,2,3,4,5,6))
```

ville gitt samme resultat.
På lik linje med vektorer, er rader og kolonner 1 indekserte, vi kan så hente ut en rad med

```
> A[1,]
[1] 1 4
```

eller en kolonne med

```
> A[,1]
[1] 1 2 3
```

Vi kan transponere en matrise med

```
> t(A)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

og således kan vi da gjennomføre matrise multiplikasjon med

```
> A %*% t(A)
     [,1] [,2] [,3]
[1,]   17   22   27
[2,]   22   29   36
[3,]   27   36   45
```

4.4 Data frame

En siste type er «data frames», dette kan anses på som navngitte tabeller som enn mer normalt er vant med. Dette gjør det også lettere å jobbe med, ved at vi kan referere til navngitte kolonner istedenfor å huske på indekser.

```
> movie = data.frame(c("A New Hope","The Empire Strikes Back","Return of the Jedi","The
↪  Phantom Menace","Attack of the Clones"),
                   c(1977,1980,1983,1999,2002),c(786,534,572,1027,656))
> names(movie)=c("Navn","År", "WBOM")
                     Navn   År WBOM
1              A New Hope 1977  786
2 The Empire Strikes Back 1980  534
3      Return of the Jedi 1983  572
4      The Phantom Menace 1999 1027
5    Attack of the Clones 2002  656
```

Ønsker vi da å hente ut «År» kolonnen gjør vi det på følgende måte

```
> movie$År
[1] 1977 1980 1983 1999 2002
```

4.5 Kommentering

Det kan være fornuftig å kommentere koden sin så enn husker hva enn har gjort og tenkt når enn kommer tilbake en måned senere. I R så kommenterer vi en enkelt linje med #,

```
> a = 5 # Setter variabelen a, til 5
```

4.6 Funksjoner

Funksjoner fungerer som i matematikken, de har et navn og tar én eller flere variabler. Følgende funksjon «add» tar inn 2 variabler, x og y, og adderer disse sammen og returnerer summen

```
add = function(x,y) {
    return x+y
}
```

enn kaller så funksjonen ved å angi navnet og parameterne

```
> add(1,2)
[1] 3
```

4.7 Pakker

Der eksisterer allerede mange ferdiglagde funksjoner, og disse kommer i «pakker». Et eksempel er «ggplot2», som er en veldig pakke for å plotte grafer. Først må vi laste ned pakkene for så å spesifisere at disse er påkrevd for at de skal bli lastet inn.

```
install.packages("ggplot2") #Installerer ggplot2, merk ""
library(ggplot2) #Spesifiserer at vi krever ggplot2, merk at vi ikke har "" her!
```

Det anbefales at så lenge du har behov for en pakke i et script, så inkluder «install.packages» med pakkene i starten, hvis de allerede er installert så blir de ikke installert på nytt ved senere kjøringer.

5 Lineær regresjon

R har en innebygd funksjon `lm()` som beregner det vi måtte ønske, bruken er ganske rett fram, gitt punkter (x,y) fordelt på vektorene x og y kan vi da beregne data på følgende måte

```
x = c(85,103,98,94,90,80,75,102,107,102)
y = c(221.5,146.1,262.6,139.8,189.0,126.0,146.5,221.4,252.9,121.4)
fit = lm(y~x)
```

Vi har da resultatet lagret i variabelen `fit`, vi kan da bruke `summary(fit)` for å få et generalt overblikk av beregningene

```
> summary(fit)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-76.005 -40.951   4.435  41.065  72.188

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   19.092    154.777   0.123    0.905
x              1.748      1.644   1.063    0.319

Residual standard error: 53.17 on 8 degrees of freedom
Multiple R-squared:  0.1239,        Adjusted R-squared:  0.01435
F-statistic: 1.131 on 1 and 8 DF,  p-value: 0.3186
```

Vi kan også hente ut enkeltverier

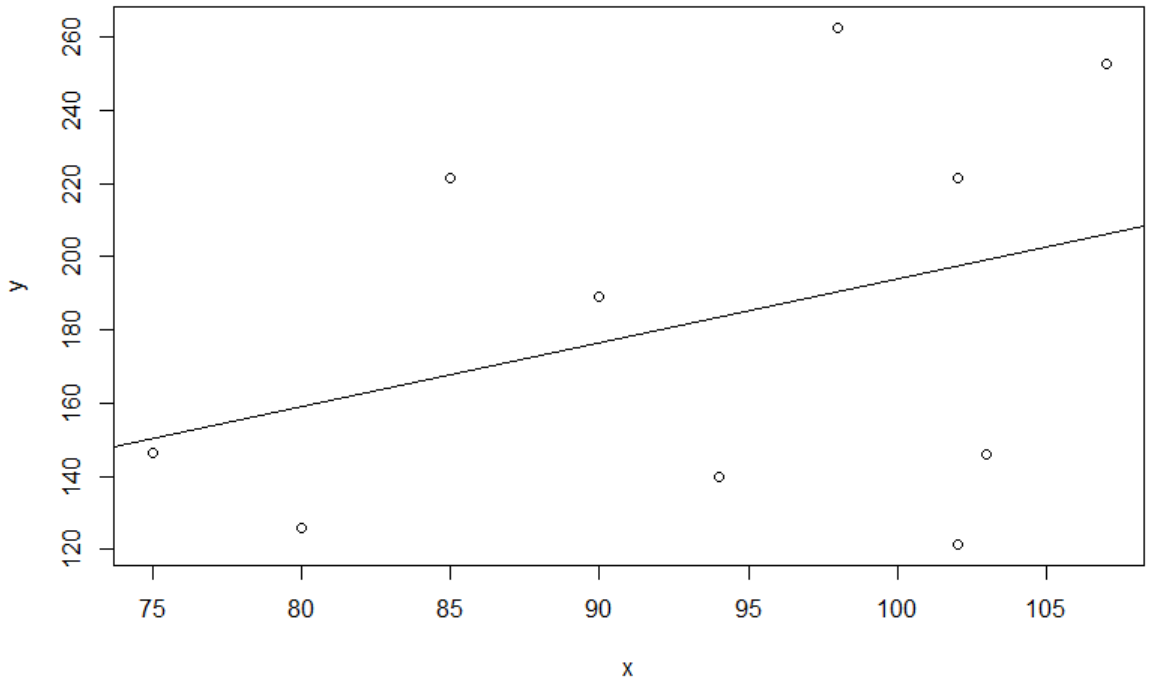
```
> coefficients(fit)
(Intercept)           x
  19.091743    1.748165
```

Her kan vi lese av at $\alpha = 19.091743$, og $\beta = 1.748165$.
Vi kan også få residualene mellom hvert punkt og regresjonslinjen vår.

```
> residuals(fit)
        1         2         3         4         5         6         7         8         9
 53.814220 -53.052752  72.188073 -43.619266  12.573394 -32.944954  -3.704128  23.995413  46.754587 -76.
```

Vi kan da plotte en enkel graf med

```
> plot(x,y)
> abline(fit)
```



5.1 Kovarians, korrelasjon, standardfeil

Vi kan finne kovariansen med `cov(x,y)`, og korrelasjonen med `cor(x,y)`, det er viktig å merke seg at disse regner utvalg og ikke populasjon.

```
> cov(x,y)
[1] 203.2533
> cor(x,y)
[1] 0.3519421
```

Standard kvadratisk feil s_e^2 er det ingen innebygd funksjon for, den enkleste måten å regne denne på rett og slett å bruke formel,

```
> fit = lm(y~x)
> sum(residuals(fit)**2)/df.residual(fit)
[1] 2827.497
```

6 Mengdelære

R kan gjennomføre de enkleste sett operasjonene på vektorer. Vi setter opp følgende for bruk i resten av kapitellet

```
A = c(1,2,3,4)
B = c(3,4,5,6)
```

6.1 Element i, ikke element i

Vi kan sjekke om x er et element i A, $x \in A$, ved `is.element(x, A)`, vi får da tilbake SANT/USANT,

```
> is.element(3, A)
[1] TRUE
> is.element(5, A)
[1] FALSE
```

Vi kan da si det motsatte, $x \notin A$, ved `!is.element(x,A)`.

6.2 Union, snitt, differanse

Det er innebygde funksjoner for union, snitt og differanse, vi kan få union $A \cup B$ ved

```
> union(A, B)
[1] 1 2 3 4 5 6
```

, snitt $A \cap B$ er

```
> intersect(A, B)
[1] 3 4
```

, og differanse $A \setminus B$

```
> setdiff(A, B)
[1] 1 2
```

6.3 Komplement

Det er ingen innebygd komplement funksjon, så denne må vi selv bygge fra definisjon $A^c = \Omega - A$ [1, s. 68]. Her kan vi da enten definere en egen vektor for Ω, eller bruke union på alle sett, her vist ved sistnevnte

```
> setdiff(union(A,B), A)
[1] 5 6
```

6.4 Binomialkoeffisient og multinominalkoeffisient

Binomialkoeffisient kan vi finne gjennom `choose(n, k)`,

```
> choose(10,2)
[1] 45
```

Multinominal er ikke implementert, men vi kan bruke definisjonen og lage vår egen funksjon

```
multinomial = function(x){
  # Summer opp alle elementene
  n = sum(x)
  sum = 1
  # For hvert element, y, i vektoren x, gjør følgende
  for(y in x){
    # Beregn den nye delsummen
    sum = sum*choose(n,y)
    # Trekk så fra totale antallet
    n = n-y
  }
  return(sum)
}
```

Denne bruker vi da som vanlige funksjoner

```
> multinomial(c(1,2,3,4))
[1] 12600
```

7 Diskrete sannsynlighetsfordelinger

7.1 Binomisk sannsynlighetsfordeling

Vi har en D_{32}, og regner suksess som 8 eller mindre, altså $p = \frac{8}{32} = 0.25$. Hva er sannsynligheten for 3 suksess på 12 kast?
Her bruker vi den binomiske distribusjonen `dbinom(x,size=n,prob=p)`, der x er antall suksess, n er antall forsøk, og p er sannsynligheten for suksess.

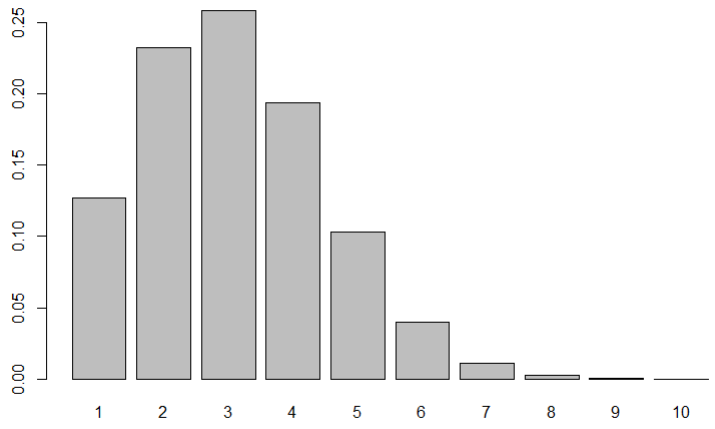
```
> dbinom(3,size=12,prob=0.25)
[1] 0.2581036
```

Hva er så sannsynligheten for minst 3 suksess på 12 kast?
Da bruker vi den kumulative `pbinom(x,size=n,prob=p)` for å finne sannsynligheten for $P(X \leq 2)$, og trekker den fra den totale sannsynligheten.

```
> 1-pbinom(2,12,0.25)
[1] 0.609325
```

Vi kan plotte distribusjonen som et stolpediagram på følgende måte

```
x = seq(1:10)                      # Lag en sekvens med nummerene 1 til og med 10
hx = dbinom(x,size=12,prob=0.25) # Beregn sannsynligheten for et gitt antall suksess
barplot(hx, names.arg=x, width = 1)
```



7.2 Hypergeometrisk

Vi har en urne med 30 baller, 10 blå og 20 gule. Vi trekker 15 tilfeldige uten å legge tilbake, hva er sannsynligheten for 8 blå?
Vi skal da regne $P(X = 8)$, da bruker vi `dhyper(x,S,N-S,n)`, der S er antall gunstige, N er totalt mulige, og n er antall trekk som, gir oss

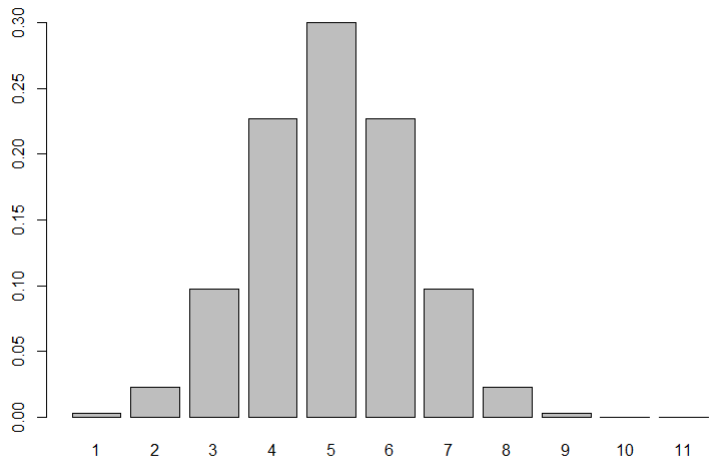
```
> dhyper(8,10,30-10,15)
[1] 0.02248876
```

Gitt samme urne (med alle baller lagt tilbake), hva er sannsynligheten for å trekke 8 eller flere blå uten tilbakelegg?
Vi regner da $1 - P(X \leq 7)$, med andre ord bruker vi `phyper(x,S,N-s,n)` som gir oss

```
> 1-phyper(7,10,30-10,15)
[1] 0.02508746
```

Vi kan plotte distribusjonen som et stolpediagra på følgende måte

```
x = seq(0:12)
hx = dhyper(x,10,20,15)
barplot(hx,names.arg=x,width = 1)
```



7.3 Poisson sannsynlighetsfordeling

Antall ugler du ser i løpet av en times tur i Grimstad-marka etter 12, er Poisson-fordelt med rate-parameter $\lambda = 2$. Hva er sannsynligheten for at du ser 3 ugler?
Vi skal da regne $P(\mathrm{X} = 3$, da bruker vi `dpois(x, lambda)` som gir oss

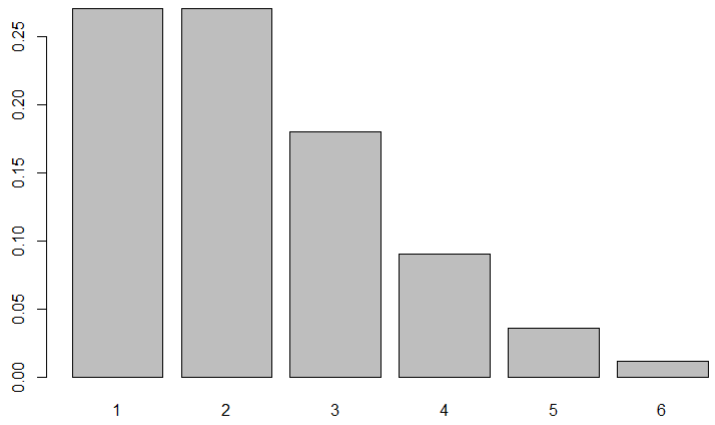
```
> dpois(3,2)
[1] 0.180447
```

Hva er sannsynligheten for at du ser max 5 ugler?
Vi skal da regne $P(\mathrm{X} \leq 5$, da bruker vi `ppois(x,lambda)` som gir oss

```
> ppois(5,2)
[1] 0.9834364
```

Vi kan plotte stolpediagram på samme måte som binomisk,

```
x = seq(0:5)
hx = dpois(x,2)
barplot(hx, names.arg=x, width=1)
```



8 Kontinuerlige fordelinger

8.1 Normalfordeling

Normalfordelingen er en velkjent symmetrisk fordeling, som brer seg ut fra forventet verdi μ. Notasjonen $\Phi_{\mu,\sigma}(x)$ oversettes til `pnorm(x,mean=`μ`,sd=`σ`)` .

Du er ute å fisker hval, og etter det du vet er vekten til de hvalene du fisker normalfordelt med forventning $\mu = 20$tonn, og standardavvik $\sigma = 3$tonn. Hva er sanns. for at hvalen du fanger er under 21 tonn?
Dette kan vi finne ved å regne $P(X \leq 21) = \Phi_{20,3}(21) =$

```
> pnorm(21,20,3)
[1] 0.6305587
```

Hva er sanns. for at hvalen du fanger er mellom 21 og 24 tonn?
Dette kan vi finne ved å regne $P(X \in [21, 24]) = P(X \leq 24) - P(X \leq 21) =$

```
pnorm(24,20,3)-pnorm(21,20,3)
[1] 0.2782301
```

8.2 Invers normalfordeling

Forts. av eksempel, for hvilken vekt v er sannsynligheten for at $X \leq v$ lik 80%?
Da benytter vi invers! $v = \Phi^{-1}_{20,3}(0.8) =$

```
> qnorm(0.8,20,3)
[1] 22.52486
```

Referanser

[1] S. O. Nyberg, *Statistikk - en bayesiansk tilnærming. 1. utgave.* Universitetsforlaget, 2016.

[2] "Learn x in y minutes, where x=r." [Online]. Available: https://learnxinyminutes.com/docs/r/