Bruksanvisning for statistikkprogrammet og programmeringsspråket R

Department of Biology, University of Oslo, NORWAY.

Innholdsfortegnelse

R-et objektorientert programmeringsspråk

Statistikkprogrammet SPLUS baserer seg på programmeringsspråket S utviklet Rick Becker, John Chambers og Allan Wilks ved AT&T Bell Laboratories. Ross Ihaka og Robert Gentleman fra New Zealand startet et altruistisk prosjekt på 1990-tallet med å skrive om S og utvikle det objektorienterte dataspråket R, en dialekt av S, og fra 1997 er dette R-prosjektet blitt organisert av R Development Core Team. R har åpen kildekode og er en del av GNU-prosjektet som startet i 1994 (Free Software Foundation). Fri programvare betyr nødvendigvis ikke at den er ikke-kommersiell, men at kildekoden er tilgjengelig for alle. R er et kraftfullt og effektivt språk, har millioner av brukere ved universiteter og høyskoler. R er i rask utvikling og stadig kommer nye oppdateringer og tilleggspakker til. R virker som en interpreter og tolker hver kommandolinje. Derfor er R ikke like raskt som Fortran og C som først kompileres til maskinkode før programmet blir kjørt. R krever mye datahukommelse og du kan få problemer med svært store datasett. R kan kjøres under de tre hovedplatformene Windows, MacOS og Linux. Det foreslås her at R bør tas i bruk også i norsk videregående skole.
Informasjon om nedlasting og installering av programmet programmet finnes på hjemmesiden:
http://www.r-project.org
eller
http://cran.r-project.org/

Alltid når du bruker R bruk følgende sitering:

R Development Core Team (2007). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org.

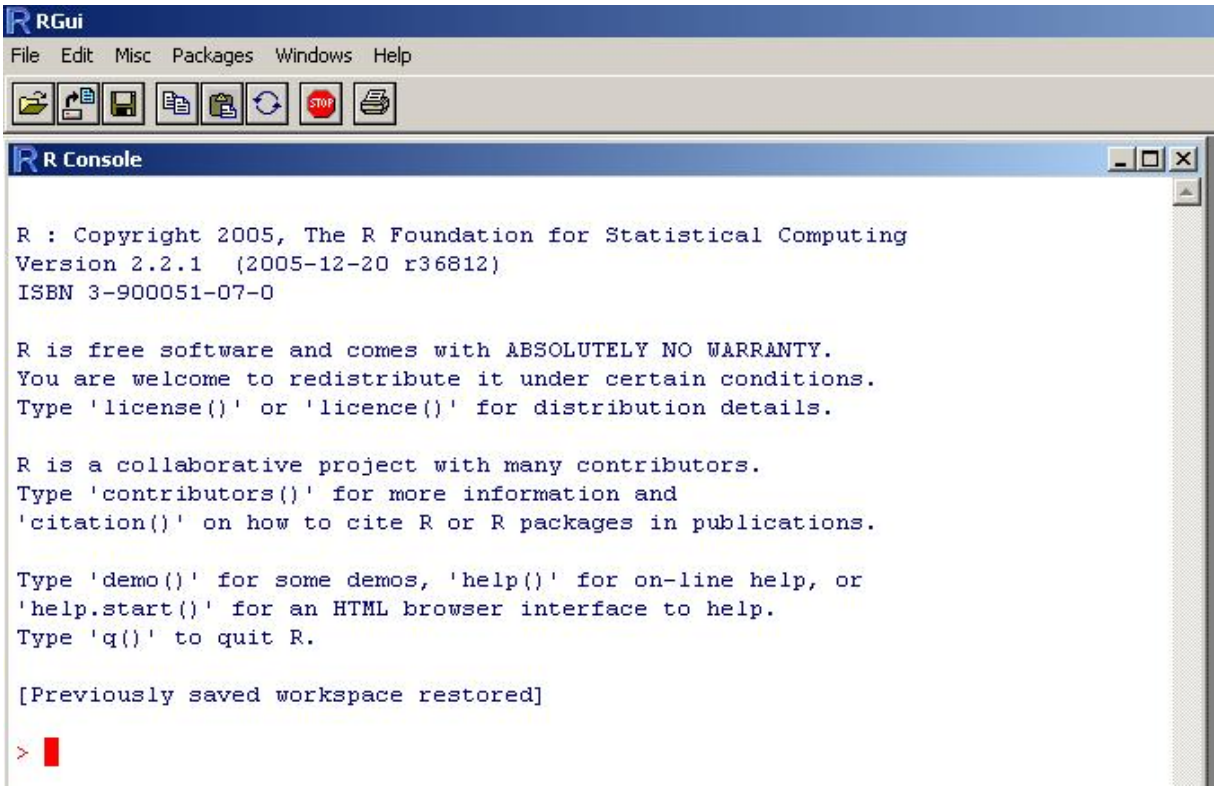
Se følgende:

citation()

Tegnet > (større enn) er signalet fra operativsystemet i R som forteller at R forventer en kommando fra brukeren. I R er det forskjell på små og store bokstaver. I det følgende er kommandoer gitt til R skrevet med rødfarget uthevet skrift, og resultatet fra R er med blå liten skrift. Foran kommandoen er det ofte skrevet >, men dette er bare prompten fra operativsystemet. Prompten > står der på forhånd og indikerer at programmet står og venter på å få beskjed om hva det skal gjøre. I scriptfiler uten prompt kjøres under *New script* som du finner under *File* på kommandovinduet. Hver enkelt linje i scriptfilen kan kjøres med Ctrl+r, men skal du kjøre hele scriptfilen ved *Run all* som du finner under *Edit* på kommandovinduet. Skrifttypen *Courier New* gjør at det ikke blir linjeforskyvninger i tabeller etc., og er velegnet når du skal kopiere kommandoer og resultater du får fra R og skrive det inn i en tekstbehandler. Når du begynner å bruke R lag din egen manual ved å kopiere kommandoer og resultater (Ctrl+c og Ctrl+v) inn i et tekstbehandlingsprogram *e.g.* Word og bruk *Courier New*.

Når du har startet programmet R får du fram vinduet nedenfor. R mangler et grafisk brukergrensesnitt som den kommersielle utgaven SPLUS. Det er derfor en liten brukerbarriære helt til du har fått dreisen på å skrive kommandolinjer.

Store deler av denne teksten i denne bruksanvisningen er basert på informasjon fra R-manualen som følger med programmet, skrevet av en rekke frivillige. Takk til dem. Det er også mulig å bruke en Tinn-R editor (Windows), og det finnes grafiske grensesnitt til R (SciViews).



Prompten **>** viser at programmet venter på at du skal gi en kommando og fortelle hva programmet skal gjøre. Dette kan kanskje virke frustrerende i forhold til programmer hvor man på menylinjer har fått oppgitt en rekke valgmuligheter, men her er det altså du som må ta initiativet. Det er ikke så vanskelig når man har skjønt prinsippene og lært seg noen kommandoer.
R opererer med navngitte datastrukturer, hvorav den enkleste er en vektor som består av en ordnet rekke med tall. R tar ikke hensyn til eventuelle mellomrom mellom objektnavn og regneoperatorer, bortsett fra i <- som betyr er lik (=).

Hjelp får det ved å skrive **help** etterfulgt av navnet på kommandoen du ønsker å vite mer om:
help(kommando)
eller
?kommando
help(help) gir oversikt over hjelpemuligheter som finnes.
Kommandoen **help.start()** starter et nettleservindu med hjelpemuligheter. ESC brukes for å stoppe utregninger, uten å stoppe R. Programmet stoppes ved **q()** eller ved å gå inn på kommandolinjen.
Kommandoer atskilles med semikolon (;) eller ny linje
Kommentarer kan settes inn ved å starte med en skigard #
Hvis en kommando ikke er fullført ved slutten av linjen gir R prompt + og venter på at du avslutter kommandoen riktig.

Vertikale piler på tastbordet kan brukes til å bevege seg gjennom tidligere kommandoer.
Gamle kommandoer kan redigeres med del-tasten og ved å skrive inn nye bokstaver i gamle kommandoer.
Arbeidsdirektoriet er "work". Oversikt over hvor arbeidsdirketoriet er får du med kommandoen **getwd()**.
Oversikt over objektene som R har brukt får du med:
objects()
Samlingen av objekter som er lagret kalles "workspace"
Objekter kan fjernes med kommandoen **rm**:
rm(navnpåobjektet, eventuelt, atskilt, med, komma, hvisflereobjekter)
Objekter kan lagres permanent, og du blir spurt om dette når du avslutter R.
Kommando som fjerner alt:
rm(list=ls())
Hvis du lagrer blir objektene skrevet inn i en fil .RData i direktoriet du befinner deg og kommandoene du har brukt blir lagret som .Rhistory

Kommandoen **search** gir en oversikt over objekter og pakker som er lastet opp. .GlobalEnv er arbeidsrommet "workspace" hvor R leter etter objekter i en søkesti. Kommandoen **attach()** plasserer objekter i søkestien, og de kan fjernes med **detach()**.

```
search()
 [1] ".GlobalEnv"        "oppg1"             "package:methods"
 [4] "package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "Autoloads"
[10] "package:base"
```

Hvis du er tilknyttet internett kan du finne hjelp via **RSiteSearch()**. Se **>help(RSiteSearch)**
Det finnes flere eksempler du kan se på. For eksempel funksjonen **image()** hvor du først lager et vindu som viser 4 figurer med kommandoen:

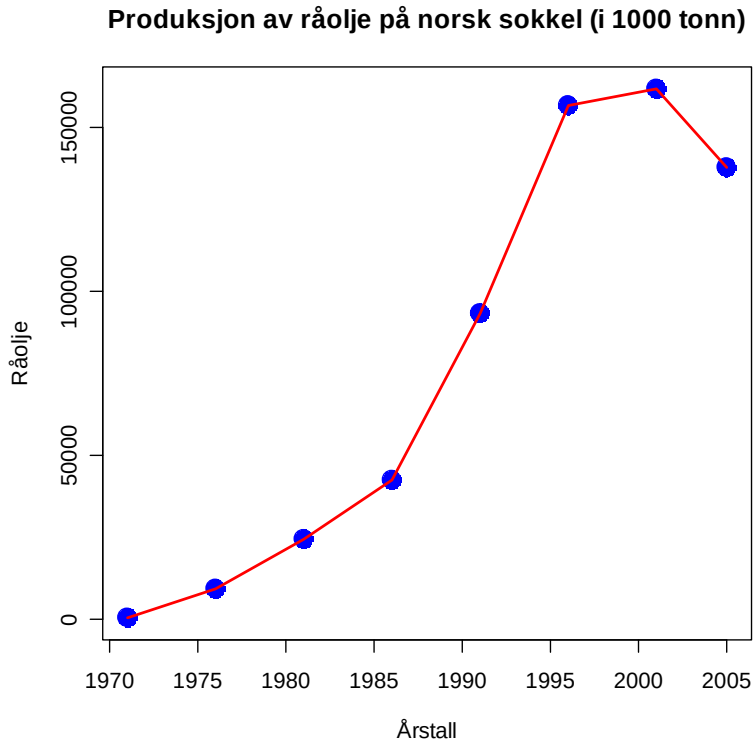
```
par(mfrow=c(2,2))
example(image)
```

Generelt vil kommandoen **par(mfrow=c(m,n))** dele opp plottearealet i *m* rader og *n* kolonner.

For å avslutte R skriver man **q()** eller du kan klikke på X på øverste høyre hjørnet på R-vindu. Når du avslutter blir du spurt om "Save the workspace image". Svarer du Yes lagres alle objektene i arbeidsdirektoriet som et bilde i en RDatafil.

Man kan skrive inn kommandoer fra en kommandolinje. Produksjon av råolje på norsk sokkel (SSB, Statistisk årbok 2006,387):

```
Årstall<-c(1971,1976,1981,1986,1991,1996,2001,2005)
Råolje<-c(301,9277,24409,42453,93346,156816,162099,138066)
plot(Råolje~Årstall,col="blue",pch=19, cex=2,main="Produksjon av råolje på
norsk sokkel (i 1000 tonn)")
lines(lowess (Råolje~Årstall, f = 0.1, iter = 3,),col="red",lwd=2)
```



Man kan også bruke kommandoen **data.frame** for å lage en dataramme:

```
produksjon<-data.frame(År=Årstall,Råolje=Råolje);produksjon
```

<- er tegnet som tilegner eller knytter sammen en variabel og en vektor. Det går an å bruke et vanlig likhetstegn (=) i stedet for <-. Variabelen har et variabelnavn som er følsom for små og store bokstaver og et variabelnavn kan aldri begynne med et tall. Variabelnavn kan ikke ha mellomrom, men man kan bruke _ eller . for å koble sammen to navn for eksempel klasse1B_lengde eller modell.fit. Variable kan inneholde numeriske (kvantitative) data som ved måling av lengde, masse og tid. Kategoriske (kvalitative) data er delt i kategorier.

R som kalkulator

Regneoperasjoner og logiske operasjoner i R

Symbol	Betydning
*	Multiplikasjon
/	Divisjon
+	Addisjon
-	Subtraksjon
^	Potens, opphøyd i
<	Mindre enn
>	Større enn
<=	Mindre enn eller lik
>=	Større enn eller lik
==	Logisk er lik
%%	Rest etter deling, modulo
%/%	Heltalls delen

```
2+10
[1] 12
3-4*3
[1] -9
(15*0.2)^2
[1] 9
```

I matematikk og statistikk brukes mange **funksjoner** og disse er vanlig kjent fra en lommekalkulator. Det går an å sette inn kommentarer til en kommando i R etter en **#**.

```
sqrt(12) # Kvadratrot sqrt
[1] 3.464102
sin(60) #Sinus-funksjon
[1] -0.3048106
```

Naturlige logaritmer med grunntall e har kommandoen **log(x)**, men skal man ha grunntall 10 må man bruke kommandoen **log10(x)**:

```
log(1)
[1] 0
log(10)
[1] 2.302585
```

Logaritmer med 10 som grunntall kan uttrykkes spesifikt som:

```
log(100,10)
[1] 2
```

Eller:

```
log(100, base=10)
[1] 2
```

Eller:

```
log10(100)
[1] 2
exp(1)
[1] 2.718282
```

```
pi # verdien pi
[1] 3.141593
```

Sette inn en variabel for x hvor 6.8 blir en skalar

```
x<-6.8
x
[1] 6.8
```

Man kan også skrive flere kommandoer på samme linje atskilt av semikolon (;):

```
x<-6.8;x
[1] 6.8
```

En variabel kan inneholde mange verdier og blir da en vektor (datavektor). Tallene kan skrives atskilt av komma (,) og deretter bindes disse sammen til en vektor med concatenere (lenke sammen) med kommandoen **c**. Her tilegnes noen tall til variabelen kalt x:

```
x<-c(5,7,9,11,13)
x
[1]  5  7  9 11 13
```

Alternativt kan man skrive inn verdiene forløpende etter å ha startet kommandoen **scan()**. Du avslutter tallrekken med to return:

```
y<-scan()
1: 2
2: 4
3: 6
4: 8
5: 10
6:
Read 5 items
y
[1]  2  4  6  8 10
```

Alle objekter har en mode (**mode()**) og en lengde (**length()**)

```
 tekst<-"Benytt dagen!"
 tekst
[1] "Benytt dagen!"
 mode(tekst)
[1] "character"
 mode(3<5)
[1] "logical"
 mode(cos)
[1] "function"
 kt<-3+2i;kt
[1] 3+2i
 mode(kt)
[1] "complex"
```

NaN er ikke et tall og Inf er uendelig, for eksempel logartimen til 0 er minus uendelig. Husk at R bruker log for naturlig logaritme (ln) med *e* som grunntall, og log10 for Briggske logaritmer.

```
 a<-log(0);a
[1] -Inf
 b<-0/0;b
[1] NaN
```

%/% er heltallsdivisjon

```
 7%/%2
[1] 3
```

Skal man finne resten etter en divisjon brukes **modulo**(%%) *e.g.* når 355 divideres på 9

```
 355%%9
[1] 4
```

1e-6 er det samme som 10^{-6}:

```
1e-6 * 1e-3
[1] 1e-09
```

Vi kan definere to datavektorer og etterpå kombinere dem:

```
a=c(65, 87, 93, 52, 48)
b=c(150,260, 320, 360, 450)
c(a,b)
 [1]  65  87  93  52  48 150 260 320 360 450
```

Den andre verdien i datavektor b:

```
b[2]
[1] 260
```

Hvis du ønsker å lage en tallrekkefølge fra 1 til 10:

```
z<-1:10
z
 [1]  1  2  3  4  5  6  7  8  9 10
```

Kolon (:) blir oppfattet som en serie heltall (integer). Tallene 1-10:

```
1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

Hvis du ønsker at datarekken skal gå motsatt vei:

```
rev(1:10)
 [1] 10  9  8  7  6  5  4  3  2  1
```

Hvis man ønsker å lage en tallrekkefølge uten heltall brukes kommandoen **seq**. For en serie som minsker brukes minus (-)

```
a<-seq(1,4,0.4)
a
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4 3.8

b<-seq(1,0,-0.2)
b
```

```
[1] 1.0 0.8 0.6 0.4 0.2 0.0
```

En rekke med oddetall:

```
oddetall=seq(1,15,by=2)
oddetall
[1]  1  3  5  7  9 11 13 15
```

Man kan også bruke navn i datasettet:

```
 kornslag=c("bygg","rug","ris","hvete","havre","mais")
 kornslag
[1] "bygg"  "rug"   "ris"   "hvete" "havre" "mais"
```

Kommandoen **rep(verdi,lengde)** kopierer/replikerer et objekt angitt i parentes et oppgitt antall ganger. En bokstavvektor består av tekst omgitt av gåseøyne.

```
 rep("b",5)
[1] "b" "b" "b" "b" "b"
 rep(1:4,2)
[1] 1 2 3 4 1 2 3 4
 rep(1:4,rep(3,4))
 [1] 1 1 1 2 2 2 3 3 3 4 4 4
```

Eller hvis man ønsker å repetere atskilt og vektorene må være like lange:

```
 rep(c(2,5,3,8),c(2,3,2,3))
 [1] 2 2 5 5 5 3 3 8 8 8
```

Hvis vi definerer et objekt x og replikerer denne så legg merke til forskjellen:

```
 x<-c(2,4,6)
 rep(x,4)
 [1] 2 4 6 2 4 6 2 4 6 2 4 6
 rep(x,2:4)
[1] 2 2 4 4 4 6 6 6 6
```

Man kan lage en aritmetisk rekke:

```
 x=1;y=2;n=6
 rekke=x+y*(0:n)
 rekke
[1]  1  3  5  7  9 11 13
```

Generere faktornivåer med kommandoen **gl**:

```
 gl(5,3)
 [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
Levels: 1 2 3 4 5
```

Ønsker man å gjenta dette som et mønster:

```
 gl(5,3,10)
 [1] 1 1 1 2 2 2 3 3 3 4
Levels: 1 2 3 4 5
```

gl betrakter automatisk vektoren som en faktor:

```
 is.factor(gl(5,3,10))
```

```
[1] TRUE

 x<-c(2,4,3,6,7,8,3,8,4,9,5,4)
 z<-12:1
```
Middeltallet til x:
```
 mean(x)
[1] 5.25
```
Variansen til x:
```
 var(x)
[1] 5.295455
```
Spredning til x:
```
 range(x)
[1] 2 9
```
Median eller midtverdi:
```
 median(x)
[1] 4.5
```
Median absolutt avvik:
```
 mad(x)
[1] 2.2239
```

Hvis x og z er like lange kan vektorene multipliseres med hverandre:

```
 x*z
 [1] 24 44 30 54 56 56 18 40 16 27 10  4
```

Hvis vektorene ikke er like lange blir den korteste gjentatt opp til nødvendig lengde

```
 x
 [1] 2 4 3 6 7 8 3 8 4 9 5 4
 z
 [1] 12 11 10  9  8  7  6  5  4  3  2  1
 x*2
 [1]  4  8  6 12 14 16  6 16  8 18 10  8
```

To vektorer kan kobles sammen med kommandoen c
```
 c(x,z)
 [1]  2  4  3  6  7  8  3  8  4  9  5  4 12 11 10  9  8  7  6  5  4  3  2
1
```

For å få del av en vektor angis dette i [] og nummeret i sekvensen:
```
 x[4]
[1] 6
 x[4:8]
[1] 6 7 8 3 8
```

Skal man ta ut for eksempel element 1,2,4,6:
```
 x[c(1,2,4,6)]
[1] 2 4 6 8
```

Hvis man ønsker å utelate et element brukes -, for eksempel fjerne det første elementet i x

```
x[-1]
 [1] 4 3 6 7 8 3 8 4 9 5 4
```

En måte å fjerne siste elementet i en vektor/array:

```
x[-length(x)]
 [1] 2 4 3 6 7 8 3 8 4 9 5
```

Logiske operatorer:

```
x[x>5]
[1] 6 7 8 8 9
```

Verdier for z hvor x>5:

```
z[x>5]
[1] 9 8 7 5 3
```

Logiske operatorer
Vi definerer først et objekt x og hvor **==** er lik og **!=** er ikke lik

```
x<-c(1,2,4,6,10,15);x
[1]  1  2  4  6 10 15
x==6
[1] FALSE FALSE FALSE  TRUE FALSE FALSE
x!=6
[1]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

< er mindre enn, **>** er større enn, **>=** lik eller større enn, **<=** lik eller mindre enn:

```
x>6
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE
x>=6
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

! er logisk nektelse, betyr forskjellig fra (logisk negasjon), **&** er elementvis og, **|** er elementvis eller.

Vi ønsker å ta ut alle elementene av x som ikke er multipler av. Hvis et tall er multippel av 3 så vil y modulo 3 være lik 0 y%%3=0!=

```
x
 [1] 2 4 3 6 7 8 3 8 4 9 5 4
x[x%%3!=0]
[1] 2 4 7 8 8 4 5 4
```

Lage en vektor

Generelt kan følgende kommandoer brukes til å lage vektorer:
seq(fra,til,by=x) hvor default-verdi er *by=1*:

```
x<-seq(1,20,by=2)
x
```

```
 [1]  1  3  5  7  9 11 13 15 17 19
```
Følgende gir samme resultat:
```
 x=seq(1,20,2)
```

Kombinere en rekke tall eller vektorer med konkatenering: **c(a,b,c,…):**
```
 z<-c(2,4,6,8,10)
 z
[1]  2  4  6  8 10
```
Eller koble flere vektorer:
```
 a<-seq(1,8,1);b<-seq(1,8,2);c<-seq(10,18,2);a;b;c;
[1] 1 2 3 4 5 6 7 8
[1] 1 3 5 7
[1] 10 12 14 16 18
 vektor<-c(a,b,c);vektor
 [1]  1  2  3  4  5  6  7  8  1  3  5  7 10 12 14 16 18
<
```
Bruke kommandoen **rep(a,b):**
```
 rep(1:5,1:5)
 [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

Med kommandoen **as.vector(x)** kan man omdannet et objekt til vektor.

Bibliotek (library)

En del av datapakkene som for eksempel **stats** blir lastet ned når du installerer R, men over hundre andre datapakker kan lastes ned fra CRAN via menylinjen Packages og Install package(s).. .
For å vise de forskjellige programpakker (bibliotek) som du har installert:
```
 library()
```
For å laste inn et spesielt bibliotek
```
 library(biblioteksnavn)
```

Vektorer og aritmetikk

Alle variable i R blir betraktet som objekter.
Regneoperasjonene regnes ut i rekkefølgen: potenser, kvadratrøtter, multiplikasjon, divisjon, addisjon og subtraksjon. Ønsker man en annen rekkefølge må det angis i parenteser () hva som skal regnes ut først.

Funksjon	Betydning
log(x)	Logaritmen til x med grunntall e

exp(x)	Eksponensialfunksjon. Antilog x (e^x)
log10(x)	Logaritmen til x med grunntall 10
sqrt(x)	Kvadratroten til x
abs(x)	Absoluttveriden til x
factorial (x)	x-fakultet (x!) (1·2·3·4·5…x
cos(x)	Cosinus til x (i radianer)
sin(x)	Sinus til x (i radianer)
tan(x)	Tangens til x (i radianer)
acos(x),asin(x),atan(x)	Inverse trigonometriske funksjoner
acosh(x),asinh(x)	Inverse hyperbolske trigenometriske f.
floor(x)	Største integer <x
ceiling(x)	Minste integer >x
choose(n,x)	Binomialkoffisienter n!/x!(n-x)!)
gamma(x)	(x-1)! for integer x (Γx)
lgamme(x)	Naturlig logaritme til gamma x
round(x,digits=0)	Avrunder verdien til x til en integer
trunc(x)	Nærmeste integer til x mellom x og 0
runif(n)	Lager n tilfeldige tall mellom 0 og 1
signif(x,digits=6)	Gir x med 6 signifikante tall
sum(variabel)	Summerer tallene i en variabel
length(variabel)	Finner lengden av en datavektor
max(variabel)	Finner maksverdien i en datavektor
range(variabel)	Finner både maks- og minimumsverdi

For eksempel en variabel avling som viser antall kilo bygg produsert per dekar på forskjellige forsøksfelt:

```
 avling=c(255,522,344,150,469)
 avling
[1] 255 522 344 150 469
 max(avling)
[1] 522
 range(avling)
[1] 150 522
 mean(avling)
[1] 348
```

Hvis vi har *n* objekter kan disse ordnes på *n!* (n fakultet) forskjellige måter. 10 objekter kan ordnes på 3.6288 millioner forskjellige måter.

```
x<-seq(1,10,1)
y<-factorial(x)
cbind(x,y)
      x       y
 [1,]  1       1
 [2,]  2       2
 [3,]  3       6
 [4,]  4      24
 [5,]  5     120
 [6,]  6     720
 [7,]  7    5040
 [8,]  8   40320
 [9,]  9  362880
[10,] 10 3628800
```

Operasjon	Betydning
max(x)	Maksimumsverdien i vektoren x
min(x)	Minimumsverdien i vektoren x
sum(x)	Summerer alle tallene i datavektoren x
mean(x)	Beregner aritmetisk middeltall i x
median(x)	Medianverdien i x
range(x)	Vektoren min(x) og max(x)
var(x)	Variansen til x
sort(x)	Sorterer x
rank(x)	Rangerer verdiene i x
order(x)	Heltallsvektor inneholder permutasjon av sortert x i stigende orden
quantile(x)	Minimum, 1.kv,median, 3.kv., maksimum

colMeans(x)	Kolonnemiddeltall av data i matrisen x
colSums	Kolonnesummen av data i matrisen x
colVars	Kolonnevariansen av data i matrisen x
rowSums	Radsummene av data i matrisen x
rowMeans	Radmiddeltallene av data i matrisen x
rowVars	Radvariansen av data i matrisen x
cor(x,y)	Korrelasjon mellom vektor x og y
length(x)	Lengden av datavektoren x
diff(x)	Beregner differansen av x
cumsum(x)	Beregner kumulativ sum i x
sd(x)	Standardavvik i vektoren x

Korrelasjon sier noe om styrken og retningen på sammenheng mellom to variable x og y, beskrevet med en **korrelasjonskoeffisient** r mellom -1 og +1. I R brukes kommandoen **cor**. Hvis r er positiv er det en positiv sammenheng mellom variablene.

$$r = \frac{1}{n-1} \cdot \sum_{i=1}^{n} \frac{(x_i - \bar{x})}{s_x} \cdot \frac{(y_i - \bar{y})}{s_y}$$

hvor standardavvikene her hhv. s_x og s_y. Vi ser at dette tilsvarer summen av z-skår for x og y. Akkurat som z-skår blir korrelasjonskoeffisienten r uten enheter.

```
 x<-c(46,54,66,50,73)
 y<-c(73,84,93,76,98)
 cor(x,y)
[1] 0.9871165
```

Vi kan også studere rangerte data. Rangeringen av tallene gjøres med kommandoen **rank**.

```
 rank(x)
[1] 1 3 4 2 5
 rank(y)
[1] 1 3 4 2 5
```

Spearman rang korrelasjon (Pearsons korrelasjonskoeffisient på rangerte data) finnes ved å bruke method="pearson", eller ved å kombindre cor og rank:

```
 cor(x,y,method="spearman")
[1] 1
 cor(rank(x),rank(y))
[1] 1
```

Kommandoen **rev** reverserer elementene i et objekt:

```
 a<-seq(1,10,1);a
 [1]  1  2  3  4  5  6  7  8  9 10
 b<-rev(a);b
 [1] 10  9  8  7  6  5  4  3  2  1

 n <- 10
```

```
xx <- c(0:n, n:0);xx
 [1]  0  1  2  3  4  5  6  7  8  9 10 10  9  8  7  6  5  4  3  2  1  0
```

Kommandoen cumsum lager en kumulativ sum e.g. summen av tallene 1 til 10:

```
a<-cumsum(1:10);a
 [1]  1  3  6 10 15 21 28 36 45 55
```

Komplekse tall

For å gjøre beregninger med komplekse tall er det flere funksjoner i R. Komplekse tall kan brukes i de fleste funksjoner. **Re(z)** finner den relle delen av det imaginære tallet og **Im(z)** den imaginære delen:

```
z<-2+3*1i
Re(z)
[1] 2
Im(z)
[1] 3
```

Polarkoordinatene modulus (**Mod(z)**) og argument (**Arg(z)**)
Hvis vi har det komplekse tallet z:

$$z = a + bi$$

$$Mod(z) = r = \sqrt{a^2 + b^2}$$

$$Arg(z) = \varphi\,; \quad a = r \cdot \cos\varphi \;\; ; b = r \cdot \sin\varphi$$

Den greske bokstaven phi (φ).

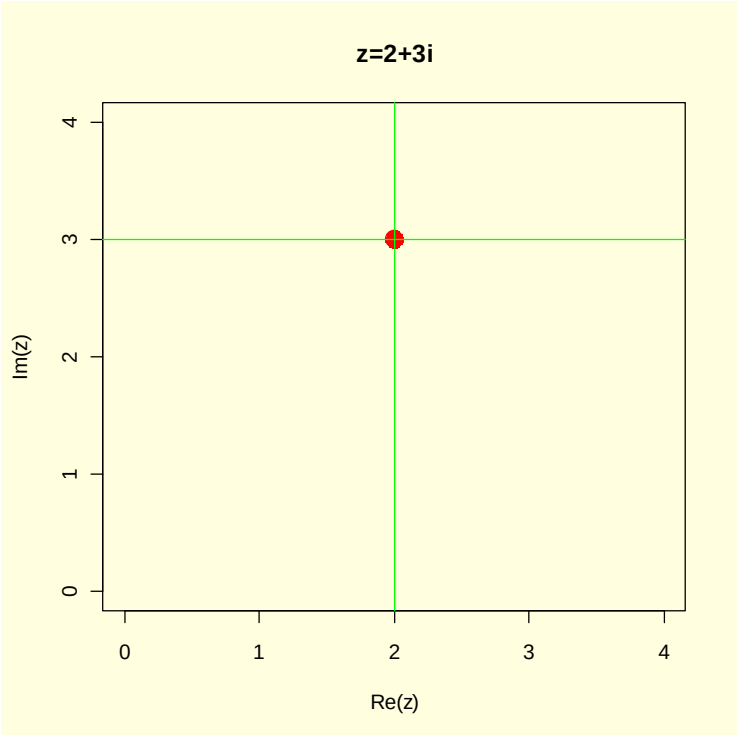
```
Mod(z)
[1] 3.605551
Arg(z)
[1] 0.9827937
```

Funksjonen **Conj(z)** gir det komplekse konjugatet:

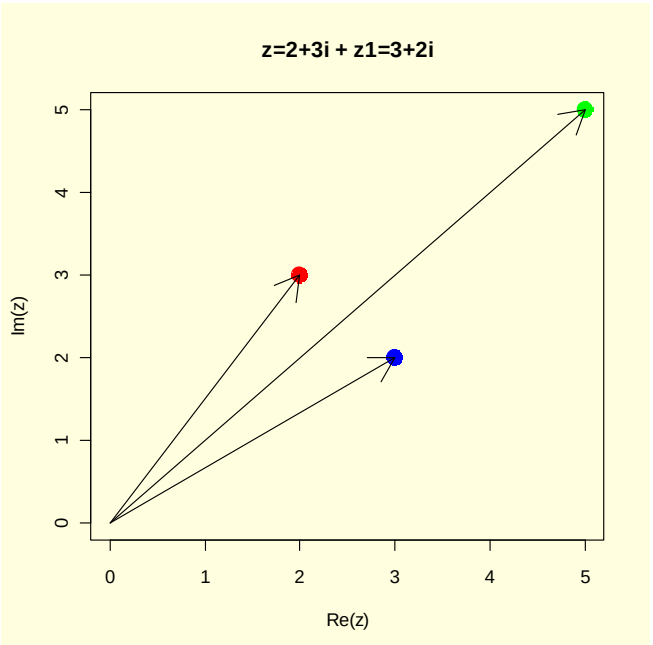
```
Conj(z)
[1] 2-3i
```

Vi kan plotte det imaginære tallet med en reell akse og imaginær akse:

```
par(bg="lightyellow")
plot(z,pch=19,cex=2,col="red",ylim=c(0,4),xlim=c(0,4),main="z=2+3i")
abline(v=2,h=3,col="green")
arrows(0,0,2,3)
```




```
z<-2+3*1i
z1<-3+2*1i
par(bg="lightyellow")
plot(z,pch=19,cex=2,col="red",ylim=c(0,5),xlim=c(0,5),main="z=2+3i + z1=3+2i")
arrows(0,0,2,3)
points(z1,pch=19,cex=2,col="blue")
arrows(0,0,3,2)
z3<-z+z1
points(z3,pch=19,cex=2,col="green")
arrows(0,0,5,5)
```



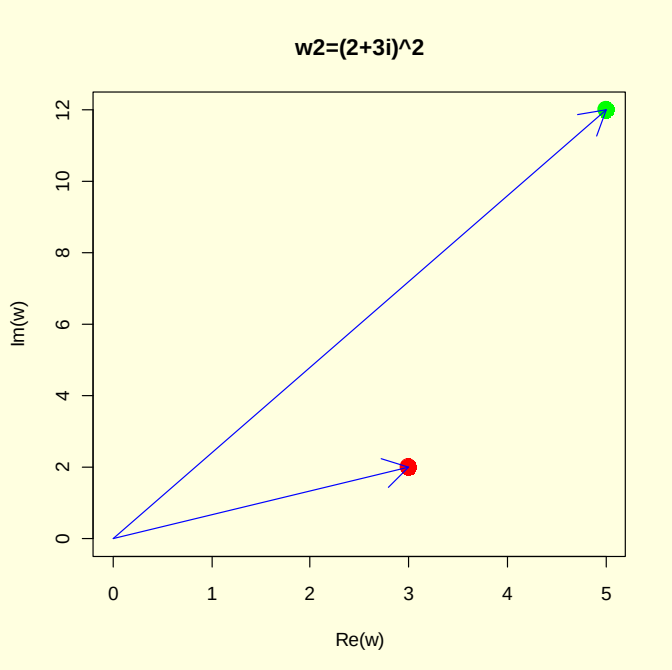
Figur. Summering av to komplekse tall. Re er reell akse og Im er imaginær akse. Origo er (0 + 0i). Summen av de komplekse tallene er diagonalen i et parallellogram med kanter dannet av de to komplekse tallene.

```
z1<-3+2*1i
z2<-z+z1;z2
[1] 5+5i
```

Vi kan multiplisere to komplekse tall:

$$(a+bi)^2 = (a+bi)\cdot(a+bi) = (a^2-b^2)+(2ab)i$$

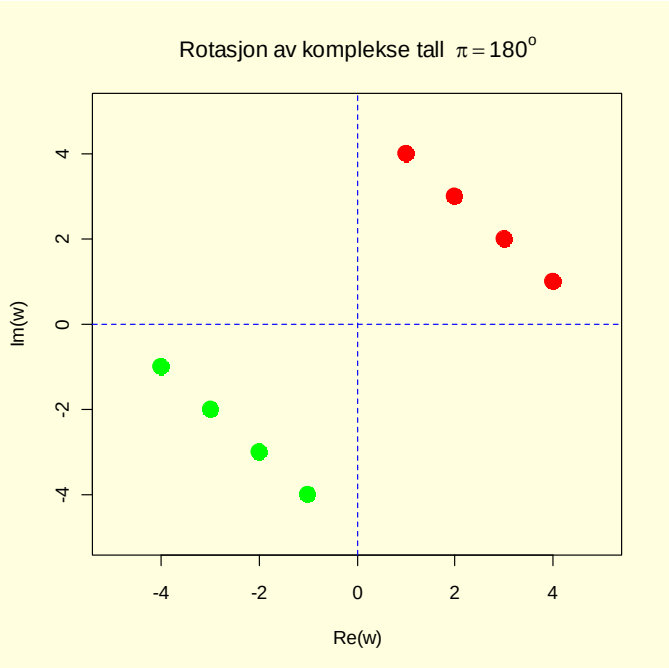
```
w<-3+2*1i
w1<-w^2;w1
[1] 5+12i
par(bg="lightyellow")
plot(w,pch=19,cex=2,col="red",ylim=c(0,12),xlim=c(0,5),main="w2=(2+3i)^2")
arrows(0,0,3,2,col="blue")
points(w1,pch=19,cex=2,col="green")
arrows(0,0,5,12,col="blue")
```



Figur. Multiplikasjon av et komplekst tall med seg selv dobler vinkelen og kvadrerer avstanden fra origo.

Rotasjon av komplekse tall (fra R-manualen):

```
w <-((1:4)+(4:1)*1i);w
[1] 1+4i 2+3i 3+2i 4+1i
Mod(w)
[1] 4.123106 3.605551 3.605551 4.123106
z<- complex(modulus = Mod(w),argument= Arg(w)+ pi);z
[1] -1-4i -2-3i -3-2i -4-1i
par(bg="lightyellow")
plot(w, col="red", pch=19,cex=2,xlim=c(-5,5),ylim=c(-5,5),
main = expression(paste("Rotasjon av komplekse tall "," ", pi
== 180^o)))
abline(h=0,v=0,col="blue",lty=2)
points(z, col="green",pch=19,cex=2)
```



Figur. Rotasjon av 4 komplekse tall 180°.

Arrays

3D-arrays med tallene 1-30 med 5 rader og 3 kolonner fordelt på to tabeller. Tallene kommer kolonnevis og elementene i array fylles opp i arrayet fra venstre mot høyre:

```
A<-array(1:30,c(5,3,2))
A
```

```
, , 1

     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15

, , 2

     [,1] [,2] [,3]
[1,]   16   21   26
[2,]   17   22   27
[3,]   18   23   28
[4,]   19   24   29
[5,]   20   25   30
```

En annen måte å lage matrise er å binde sammen vektorer til matriser. Man kan binde sammen kolonner med **cbind()** og rader med **rbind()**. Man kan bruke disse til å kombinere datavektorer laget med kommandoen **c()** som kombinerer en rekke tall i form av en vektor

Eksempel antall felte elg ifølge SSB:

```
 årstall<-c(1978,1983,1988,1993,1998,2003)
 felt<-c(14808, 24181,24972,38980,37957,38564)
 elg<-cbind(årstall,felt)
 elg
     årstall  felt
[1,]    1978 14808
[2,]    1983 24181
[3,]    1988 24972
[4,]    1993 38980
[5,]    1998 37957
[6,]    2003 38564
```

Funksjonen **sapply** returnerer en liste med samme lengde som X hvor hvert element er et resultat av behandling eller en funksjon på hvert element.

Utvalg av 2, og 3. kolonne i A:

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

     [,1] [,2]
[1,]    6   11
[2,]    7   12
[3,]    8   13
[4,]    9   14
[5,]   10   15

, , 2

     [,1] [,2]
[1,]   21   26
[2,]   22   27
[3,]   23   28
[4,]   24   29
[5,]   25   30
```

Utvalg av rad 2-4:

```
 A[2:4,2:3,]
, , 1

     [,1] [,2]
[1,]    7   12
[2,]    8   13
[3,]    9   14

, , 2

     [,1] [,2]
[1,]   22   27
[2,]   23   28
[3,]   24   29

 A[2:4,2:3,2]
     [,1] [,2]
[1,]   22   27
[2,]   23   28
```

```
[3,]   24   29
```

Vektorer subskriberes som [3], mens liste subskriberes som [[3]]. Se forskjellen i eksemplet nedenfor:

```
 blomster<-
list(c("rose","tulipan","nellik"),c(2003,2004,2005),c("hvit","rosa","blå","
rød"))
 blomster
[[1]]
[1] "rose"    "tulipan" "nellik"

[[2]]
[1] 2003 2004 2005

[[3]]
[1] "hvit" "rosa" "blå"  "rød"

 blomster[3]
[[1]]
[1] "hvit" "rosa" "blå"  "rød"
 blomster[[3]]
[1] "hvit" "rosa" "blå"  "rød"
 blomster[[3]][2]
[1] "rosa"
 blomster[3][2]
[[1]]
NULL
```

En **enhetsmatrise** (*I*) kan lages med kommandoen **diag**:

```
diag(1,nrow=3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Vi kan lage en matrise over gangetabellen:

```
x <- 1:10; names(x) <- x
tabell<-x %o% x;tabell
    1  2  3  4  5  6  7  8  9  10
1   1  2  3  4  5  6  7  8  9  10
2   2  4  6  8 10 12 14 16 18  20
3   3  6  9 12 15 18 21 24 27  30
4   4  8 12 16 20 24 28 32 36  40
5   5 10 15 20 25 30 35 40 45  50
6   6 12 18 24 30 36 42 48 54  60
7   7 14 21 28 35 42 49 56 63  70
8   8 16 24 32 40 48 56 64 72  80
9   9 18 27 36 45 54 63 72 81  90
10 10 20 30 40 50 60 70 80 90 100
```

Med kommandoen **outer()** kan man lage en matrise med tall som multipliseres med hverandre e.g. gangetabellen som foran eller potenser:

```
outer(1:10,1:10)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]    2    4    6    8   10   12   14   16   18    20
 [3,]    3    6    9   12   15   18   21   24   27    30
 [4,]    4    8   12   16   20   24   28   32   36    40
 [5,]    5   10   15   20   25   30   35   40   45    50
 [6,]    6   12   18   24   30   36   42   48   54    60
 [7,]    7   14   21   28   35   42   49   56   63    70
 [8,]    8   16   24   32   40   48   56   64   72    80
 [9,]    9   18   27   36   45   54   63   72   81    90
[10,]   10   20   30   40   50   60   70   80   90   100
```

Eller potenstabell:

```
y <- 2:8; names(y) <- paste(y,":",sep="")
outer(y, x, "^")
```

```
   1  2   3    4     5      6       7        8         9         10
2: 2  4   8   16    32     64     128      256       512       1024
3: 3  9  27   81   243    729    2187     6561     19683      59049
4: 4 16  64  256  1024   4096   16384    65536    262144    1048576
5: 5 25 125  625  3125  15625   78125   390625   1953125    9765625
6: 6 36 216 1296  7776  46656  279936  1679616  10077696   60466176
7: 7 49 343 2401 16807 117649  823543  5764801  40353607  282475249
8: 8 64 512 4096 32768 262144 2097152 16777216 134217728 1073741824
```

Funksjoner

Hvis man ønsker å lage en funksjon som skriver ut middeltall av forskjellig type og median
function står inne i {}. Linjer i koden atskilles med return. For å få en ny linje for neste utskrift må det brukes ”\n”
Funksjonen **cat** er det samme som **print**, men gir kontroll over utskriften

```
 sentral<-function (x) {
+ gm<-exp(mean(log(x)))
+ hm<-1/mean(1/x)
+ cat("Median",median(x),"\n")
+ cat("Aritmetisk middel",mean(x),"\n")
+ cat("Geometrisk middel",gm,"\n")
+ cat("Harmonisk middel",hm,"\n")}
 sentral(x)
```

```
Median 4.5
Aritmetisk middel 5.25
Geomatrisk middel 4.771715
Harmonisk middel 4.305239
```

Utregnet for

```
 x<-c(2,4,3,6,7,8,3,8,4,9,5,4)
```

Det er forskjell på sortere (**sort**) og ordne (**order**)

```
sort(x)
[1] 2 3 3 4 4 4 5 6 7 8 8 9

x
[1] 2 4 3 6 7 8 3 8 4 9 5 4

rev(sort(x))
[1] 9 8 8 7 6 5 4 4 4 3 3 2
```

Det er ingen måte å reversere sorteringen, derfor er det bedre å beholde den usortert og bruke order i stedet

```
order(x)
[1]  1  3  7  2  9 12 11  4  5  6  8 10
```

Order lager subskrift for x, altså lager ikke verdier for x
Den laveste har plass 1, neste laveste plass 3 osv.

Noen ganger ønsker man å vite hvor mange ganger et tall forekommer i en vektor og man bruker da funksjonen **table.**
For eksempel 1000 tilfeldige tall fra en negativ binomial fordeling med middeltall 1.2 og aggregasjonsparameter k=0.63
Prob= k/mu+k

Randomisering gir litt forskjellige tall hver gang:

```
vals<-rnbinom(10000,size=0.63,prob=0.63/1.83)
table(vals)
vals
   0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17
5066 2102 1165  662  370  258  149   94   37   29   34   11    7    7    3    3    2    1
```

Omdanne kontinuerlige variable til kategorisk variable med kommandoen **cut.**

```
sml<-cut(vals,3)
table(sml)
sml
(-0.018,5.99]     (5.99,12]       (12,18]
         9618           363            19
```

For oppsummerende statistikk er det vanlig å bruke funksjonen **tapply** som gir middel, varianse, prøvestørrelse etc.

Grafikk

En grafisk framstilling visualiserer data, synliggjør informasjon på en enkel måte slik at det er lett å se trender og mønstre.
En oversikt over grafikkmuligheter med:

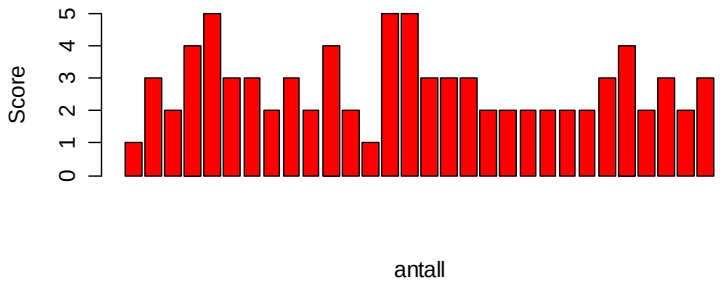
```
demo(graphics)
help(graphics)
```

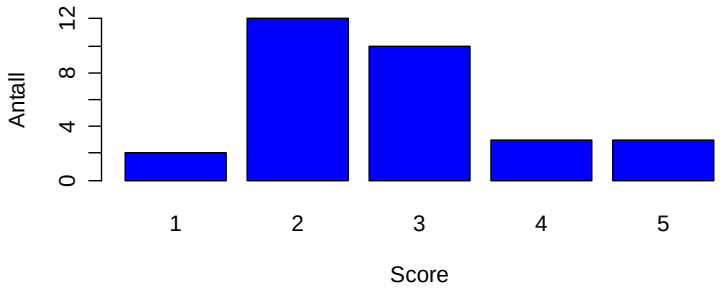
Leser inn data for fornøyd (1) til misfornøyd(5)

```
 antall=scan()
1: 1
2: 3
osv
antall

  [1] 1 3 2 4 5 3 3 2 3 2 4 2 1 5 5 3 3 3 2 2 2 2 2 2 3 4 2 3 2 3
 table(antall)
antall
 1  2  3  4  5
 2 12 10  3  3
```

I et **vertikalt stolpediagram** deles data i kategorier. Stolpene har samme bredde, det er avstand mellom stolpene og det er lett å se forskjeller. Et horisontalt stolpediagram gir bedre plass til lange kategorinavn.
Se de to forskjellige måtene å lage et stolpediagram av dataene:

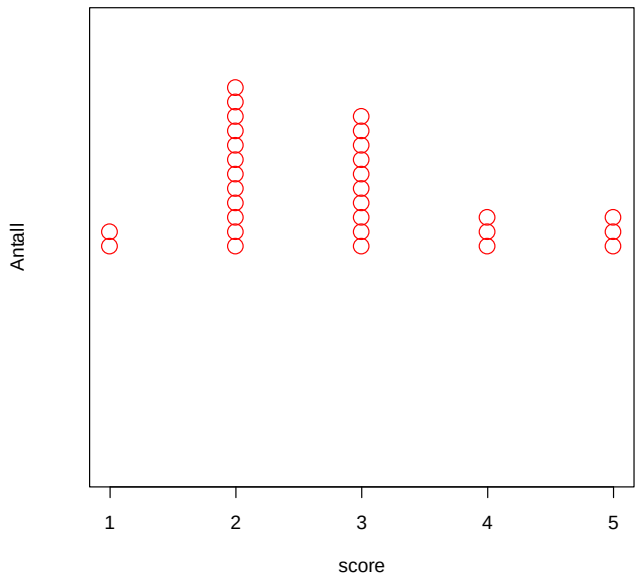
```
par(mfrow=c(2,1))
barplot(antall,col="red",xlab="antall",ylab="Score")
barplot(table(antall),col="blue",xlab="Score",ylab="Antall")
```



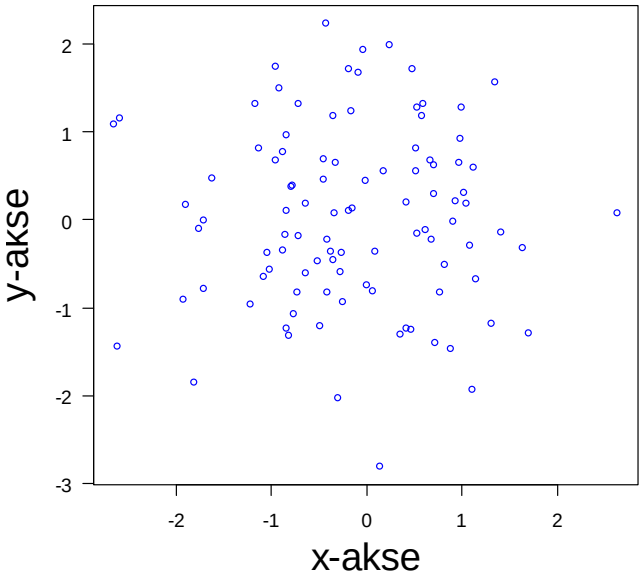


Strekkartplot lages med kommandoen **stripchart()** hvor det går an å tilføye **method="stack"**
Vi kan bruke antall som eksempel:

```
stripchart(antall,
col="red",method="stack",xlab="score",ylab="Antall",pch=1,cex=2)
```



Skal man ha et kvadratisk plot med verdiene på y-aksen skrevet horisontalt bruk kommandoen **par(las=1)** Størrelsen på akseteksten kan brukes ved kommandoen **cex.lab=1.5** eller et annet tall

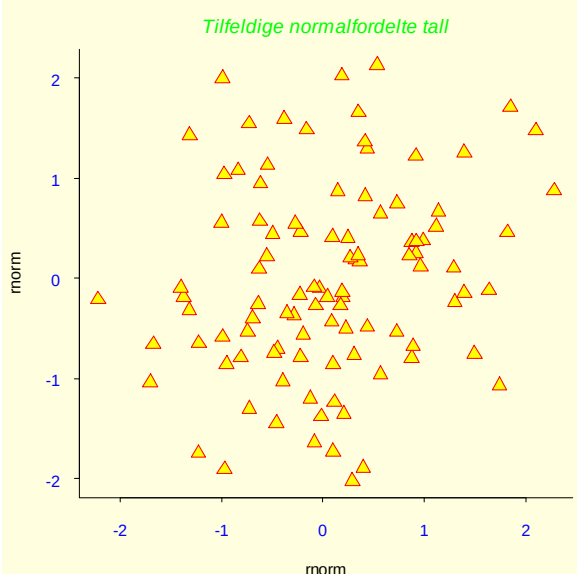



```
plot(rnorm(100),rnorm(100),ylab="y-akse",xlab="x-akse",col="blue",las=1,cex.lab=2)
```

Du kan sjekke hva som ligger bak plottet:

```
?par
```

```
plot(rnorm(100), rnorm(100), xlab="rnorm", ylab="rnorm",
pch=24, col="red", bg="yellow",bty="l", tcl=-.25, las=1,
cex=1.5)
title("Tilfeldige normalfordelte tall",
col.main="green",font.main=3)
```



xaxt="n" undertrykker akseteksten på x-aksen

Skal man ha inn greske tegn bruk kommandoene **expression()** eller **substitute ()**.

Vi kan tilegne en forklaringsvariabel x en tallrekke fra 0-12

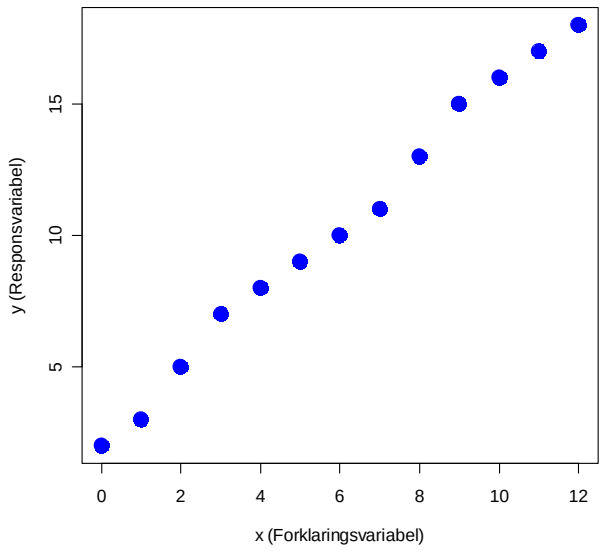
```
x<-0:12
x
[1]  0  1  2  3  4  5  6  7  8  9 10 11 12
```

Hvis vi har en responsvariabel y som skal tilegnes en vektor kan vi bruke konkatenere c.

```
y<-c(2,3,5,7,8,9,10,11,13,15,16,17,18)
y
[1]  2  3  5  7  8  9 10 11 13 15 16 17 18
```

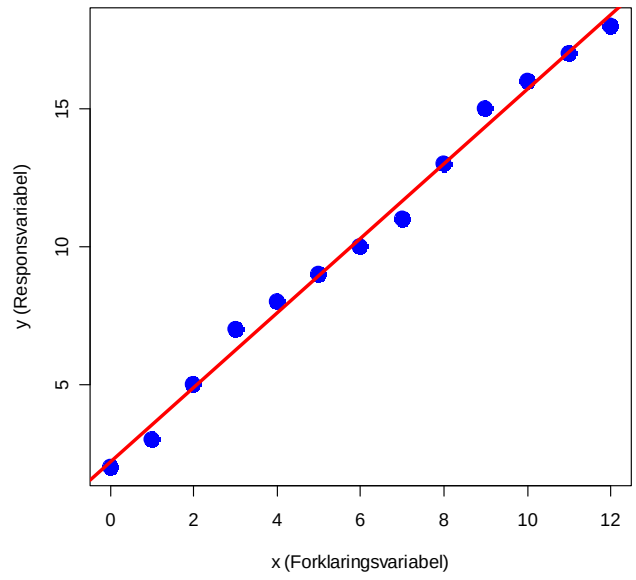
Hvis vi ønsker å lage et plot for de to variablene x og y med tilhørende aksetekst, bruker vi **xlab** for x-aksen og **ylab** for y-aksen og hva som skal stå på aksen settes i gåseøyne (")

```
plot(x,y, pch=19,cex=2,xlab="x (Forklaringsvariabel)",
ylab="y (Responsvariabel)",col="blue")
```



Hvis vi ønsker å legge en linje i plottet hvor vi antar at det er følgende linære modell y~x bruker vi kommandoen **abline.** Denne finnes i forskjellige utgaver: **abline(a,b)** hvor *a* er skjæringspunkt og *b* er stigningskoeffisient. Vertikal linje: abline(v=), horisontal linje:abline(h=)

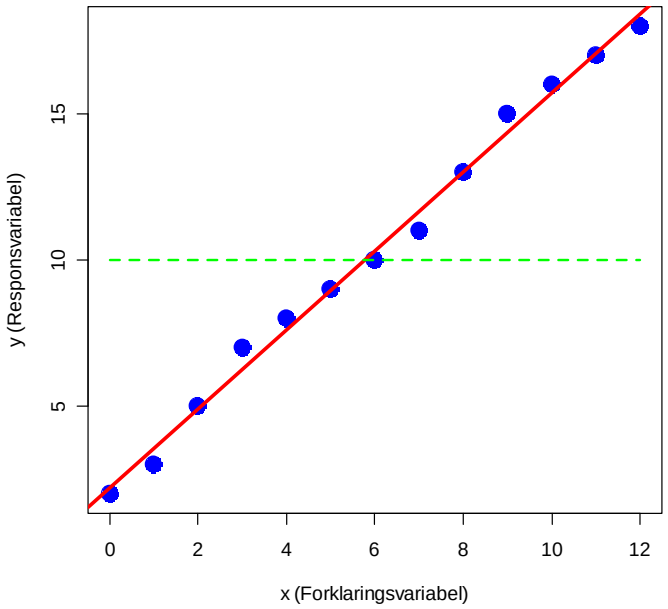
```
abline(lm(y~x),lwd=3,col="red")
```



I linjeplot vises trender når datapunktene er forbundet med en linje. I tidsserieplot er tiden på den horisontale x-aksen og dataverdi eller frekvens på den vertikale y-aksen

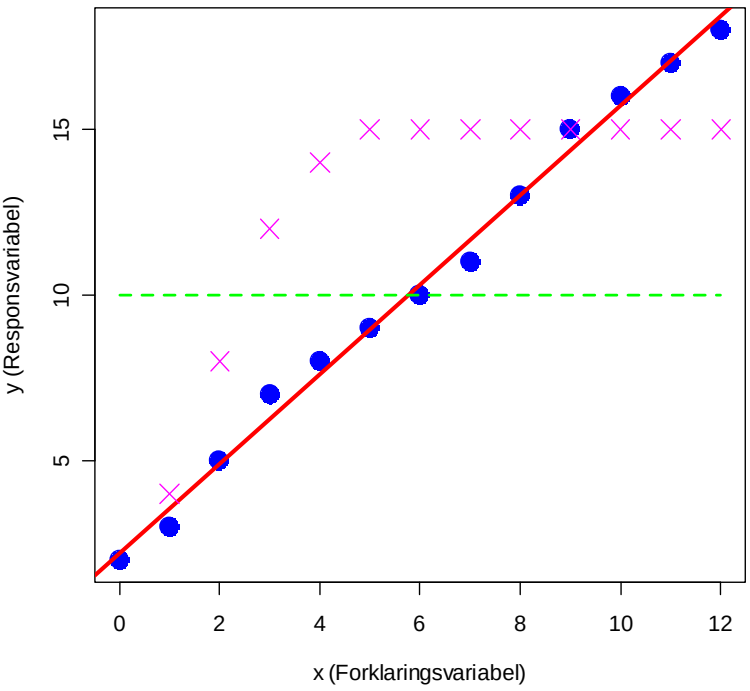
Hvis vi ønsker å trekke en linje fra koordinatene (0,10) til (12,10), så kobler vi sammen x-verdiene c(0,12) og tilsvarende for y-verdiene c(10,10) og bruker kommandoen **lines** og **lty=2** for å laget en prikket linje:

```
lines(c(0,12),c(10,10),col="green",lty=2)
```



Hvis vi nå ønsker å tegne en graf til på samme akseteksten brukes ikke kommandoen plot, da startes et nytt plot, men man bruker kommandoen **points**, **pch=4** gir kryss og **col** angir fargen.

```
a<-1:12
b<-c(4,8,12,14,15,15,15,15,15,15,15,15)
 points(a,b,pch=4,cex=2,col="magenta")
```



Kommandoen **plot** gir et punktskyplot hvis x er en kontinuerlig variabel og et boksplot hvis x er en kategorisk variabel (faktor). Hvis man angir **type="n"** plottes bare x- og y-aksen på plottet og man kan putte inn plot etterpå, mens **type="l"** lager linjer. Man kan begrense lengden av aksene ved **xlim** og **ylim** ,for eksempel begrense x-aksen fra 0-15: **xlim=c(0,15).**

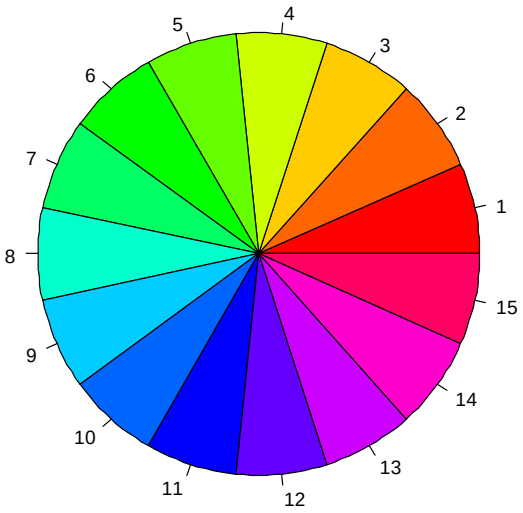
Plottesymboler med **pch= n** hvor n= 0-kvadrat, 2-triangel, 3-kryss, 4-x, 5-diamant, 6-omvendt triangel

Ønsker vi å få oversikt over et datasett er det mulig å lage et **stamme-blad-plot.** For hvert datapunkt registreres først tallene fra 0-9, 10-19 osv som stamme, og resten av tallet blir blad
Leser inn tall med **y=scan()**

```
y
[1]  6  7  4 12 15 19 25 24 29 31 33 59 64 57 39 37 36  0  9  8
stem(y)

 The decimal point is 1 digit(s) to the right of the |

 0 | 046789259
 2 | 45913679
 4 | 79
 6 | 4
```

Kommandoen **pie** brukes for kakediagram som deler data i grupper, og er best egnet når det er store forskjeller i andeler (frekvens). Husk at vår hjerne er lite flink til å tolke forskjeller i et kakedigram. Stolpediagram er egentlig bedre.

```
pie(rep(1,15),col=rainbow(15),radius=1)
```

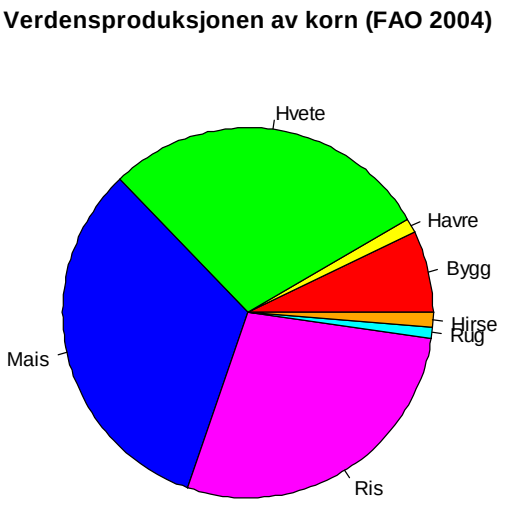


F.eks. andelen av verdensproduksjonen av korn oppgitt i 1000 tonn (FAO 2004)i % angitt som variabelen z. Radius angir størrelsen.

```
z<-c(155115,26961,624093,705293,608496,19545,27676)
 names(z)<-c("Bygg","Havre","Hvete","Mais","Ris","Rug","Hirse")
pie(z,col=rainbow(4),radius=0.9,main="Verdensproduksjonen av korn (FAO 2004)")
```

Eller hvis du vil bestemme fargene selv:

```
pie(z,col=c("red","yellow","green","blue","magenta","cyan","orange"),main="
Verdensproduksjonen av korn (FAO 2004)")
```



Hvis du vil se hvilke andre farger du kan velge mellom, og det er mange, skriv:

```
colours()
```

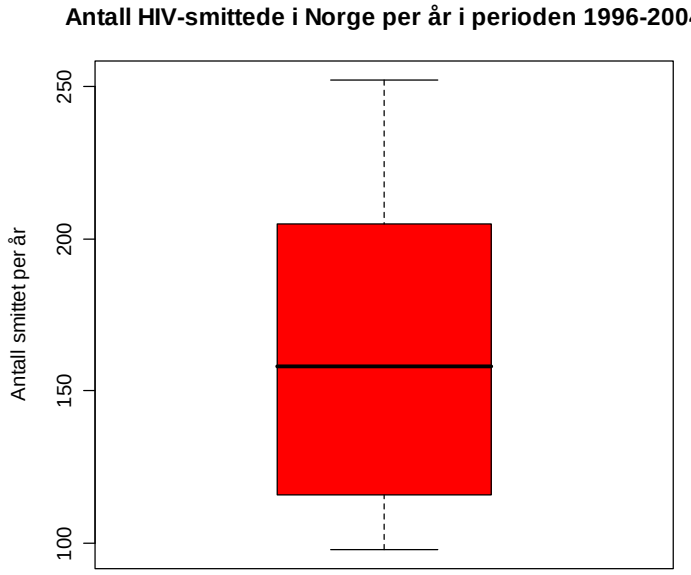
Boksplot (konstruert av John Tukey) er velgnet for å vise median-verdi som en strek i en boks (senter i datasettet), og boksen dekker interkvartilområdet (IQR)fra første til tredje kvartil. Første kvartil er den verdien som har 25% av dataene under seg. Tredje kvartil har 75% av dataene under seg. Spredningen på datasettet er vist ved tynne streker ut fra boksen, fra minimums- til maksimumsverdi, men siden man har bestemt at disse strekene ikke skal være mer enn 1.5 ganger lengden på boksen (interkvartilområde), så vil verdier som ligger utenfor dette bli angitt som **utligger-verdier** med separate datapunkter. Funksjonen **notch=T** gjør en signifikanstest om det er forskjeller mellom median-verdier i boksplottet.
Det er mange ferdige betingelseverdier som er innsatt på forhånd. Se for eksempel:

```
args(boxplot.default)
```

Boksplot av antall HIV-smittede i Norge ifølge Nasjonalt Folkehelseinstitutt (Statistisk årbok 2006, SSB)

```
hiv=scan()
1: 116 114 98 147 175 158 205 238 252
10:
Read 9 items
```

```
boxplot(hiv,col="red",ylab="Antall smittet per
år",main="Antall HIV-smittede i Norge per år i perioden 1996-
2004")
```



Kommandoen **pairs** lager en matrise av plot for alle kombinasjoner av variable i datasettet her kalt "filnavn":

```
pairs(filnavn,panel=panel.smooth)
```

Hvis man ønsker å ha flere plot på samme side bruker man kommandoene nedenfor før man begynner å plotte:

To plot ved siden av hverandre, og generelt gir **mfrow=c(*m*,*n*)** figurene som en (*m*,*n*) matrise:

```
par(mfrow=c(1,2))
```

Fire på på en side i en 2x2 matrise:

```
par(mfrow=c(2,2)
```

Vil man ha logaritmer på en eller begge av aksene bruk **log="x"**, **log="y"**, eller **log="xy"** i plot-kommandoen for å angi logaritmer på henholdsvis x-akse, y-akse eller på begge aksene.

Hvis man ønsker å sette inn tekst på en graf brukes kommandoen **text** etterfulgt av x,y-koordinatene der man vil teksten skal starte og teksten angitt i gåseøyne.

```
text(2,16,"Funksjon (x,y)")
```

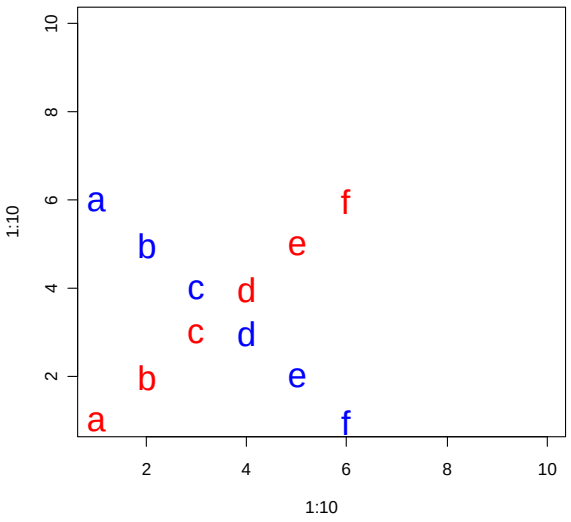
Man kan også plotte variable som bokstaver eller bokstavnavn. Skal man lage en liste med bokstavene i alfabetet tilegnet en variabel:

```
bokstav<-letters[1:6]
```

```
 bokstav
[1] "a" "b" "c" "d" "e" "f"
```

Hvis man ønsker å plotte disse bokstavene fra koordinatene (1,1) til (6,6) og lage de dobbelt så store **cex=2**, og deretter plotte de ved å starte ved (1,6) og slutte ved (6,1):

```
 plot(1:10,1:10,type="n")
 text(1:6,1:6,bokstav,cex=2,col="red")
 text(1:6,6:1,bokstav,cex=2,col="blue")
```

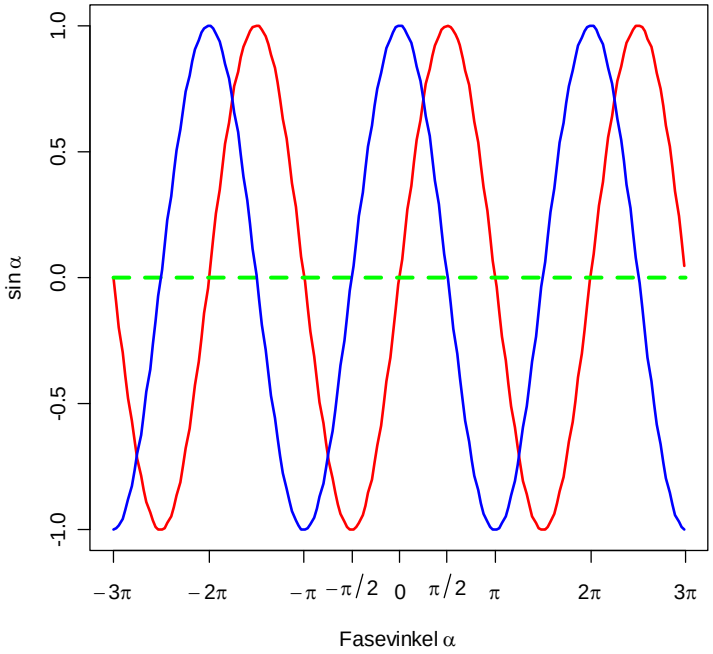


Skal man lage plot med greske symboler brukes kommandoen **expression:**

```
 x<-seq(-3*pi,3*pi,0.1)

plot(x,sin(x),col="red",lwd=2,type="l",xaxt="n",xlab=expressio
n(paste("Fasevinkel ",alpha)),ylab=expression("sin "*alpha))
```

Kommandoen lines trekker en rett linje fra x-koordinatene (x,x) til y-koordinatene (y,y):

```
 lines(c(-3*pi,3*pi),c(0,0),col="green",lwd=3,lty=2)

 axis(1,at=c(-3*pi,-2*pi,-pi,-
pi/2,0,pi/2,pi,2*pi,3*pi),labels=expression (-3*pi,-2*pi,-pi,-
pi/2,0,pi/2,pi,2*pi,3*pi))

 points(x,cos(x),col="blue",lwd=2,type="l")
```



Den røde linjen er sin(x) og den blå er cos(x), men hvor vinkelen er kalt α. Sinusfunksjonen begynner på likevektslinjen (grønn).

Lag et tomt plot med akser og aksetekst:

```
plot(-4:4,-4:4,type="n",xlab="x",ylab="y")
```

Trekk koordinatakser:

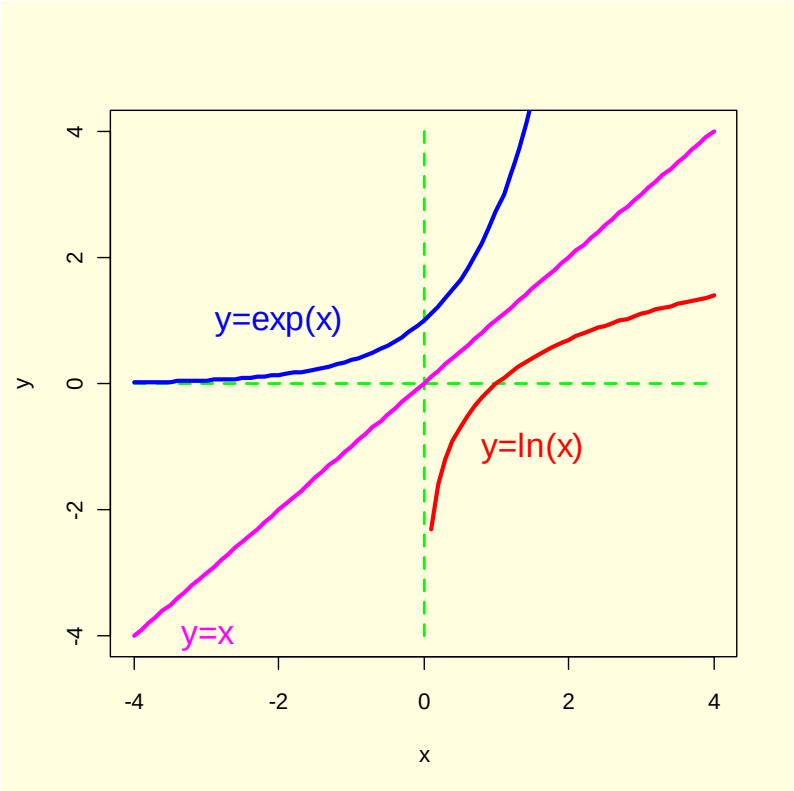
```
lines(c(-4,4),c(0,0),col="green",lty=2,lwd=2)
lines(c(0,0),c(-4,4),col="green",lty=2,lwd=2)
```

La x gå fra -4 til 4 og tegn eksponensialfunksjonen e^x (blå)og logaritmefunksjonen ln(x) (rød), som er omvendte funksjoner og y=x (magenta). Vi bruker nå kommandoen **points** i stedet for **plot** siden vi skal ha flere plot på samme figur. Legg merke til at ln(1)=0 og at ln(0)=-∞

```
x<-seq(-4,4,0.1)
 points(x,exp(x),col="blue",type="l",lwd=3)
 points(x,log(x),col="red",type="l",lwd=3)
 points(x,x,col="magenta",type="l",lwd=3)
```

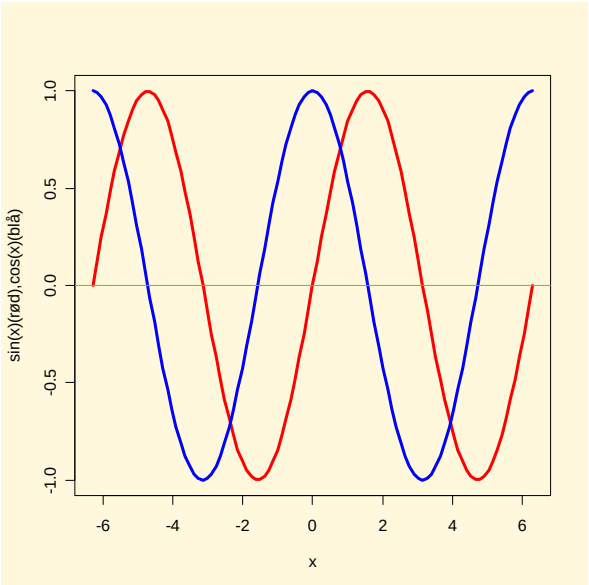
Sett på tekst på figurene:

```
text(-3,-4,"y=x",col="magenta",cex=1.5)
text(1.5,-1,"y=ln(x)",col="red",cex=1.5)
text(-2,1,"y=exp(x)",col="blue",cex=1.5)
```



Lager en grafisk framstilling av sinusfunksjonen fra -2π til 2π med kommandoen **curve**, og legger deretter på cosinusfunksjonen (**add=T**)og trekker en horisontal linje:

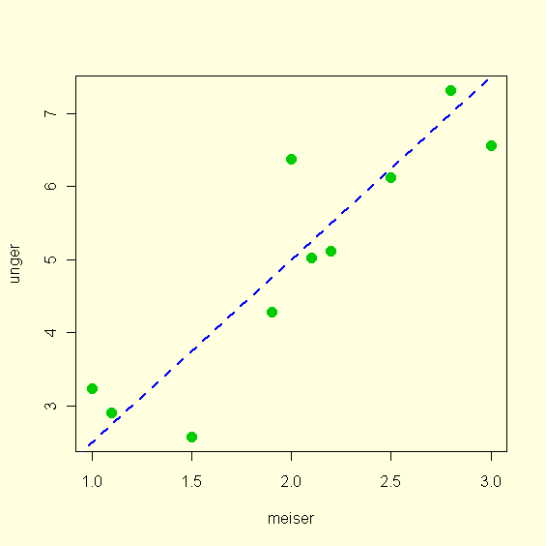
```
par(bg="cornsilk")
 curve(sin, -2*pi,
2*pi,col="red",lwd=3,ylab="sin(x)(rød),cos(x)(blå)")
 curve(cos, -2*pi, 2*pi,add=T,col="blue",lwd=3)
 abline(h=0,col="green")
```



Vi kan simulere et forsøk hvor vi har gitt antall kjøttmeispar per arealenhet. Antall unger regnes som en tilfeldig normalfordeling med gjennomsnitt 2.5 ganger antall kjøttmeispar, og standardavvik 0.5. Vi plotter resultatet og

siden vi har 2.5 ganger så mange unger kan vi bruke abline(a=0,b=2.5). Vi kan bruke en annen strategi med å lage en lineær modell. Programmet gir forskjellig resultat hver gang.

```
meiser<-c(1,1.1,1.5,1.9, 2,2.1,2.2,2.5,2.8,3);meiser
length(meiser)
summary(meiser)
unger<-rnorm(n=10,mean=2.5*meiser,sd=0.5);unger
par(bg="lightyellow")
plot(meiser, unger,pch=16,cex=1.5,col=3)
abline(a=0,b=2.5,lty=2,col=4,lwd=2)
#alternativ til abline over
modell<-lm(unger~meiser) #lineær modell
abline(modell)
```



John Stirling (1692-1770) var venn av Newton,skrev *Methodus differentialis* (1730) og har fått flere formler oppkalt etter seg bl.a. **Stirlings formel** for n fakultet (n!)hvor *n* er et positivt heltall

$$\sqrt{2\pi}\cdot n^{n+\frac{1}{2}}\cdot e^{-n} < n! < \sqrt{2\pi}\cdot n^{n+\frac{1}{2}}\cdot e^{-n}\left(1+\frac{1}{4n}\right)$$

så vil brøken nedenfor nærme seg √2π når n→∞

$$\frac{n!}{n^{n+\frac{1}{2}}\cdot e^{-n}}$$

eller hvis vi ser på logaritmen:

$$\log(n!)-\left(log\left(n^{n+\frac{1}{2}}\right)-n\right)\approx log\left(\sqrt{2\pi}\right)$$

Som blir:

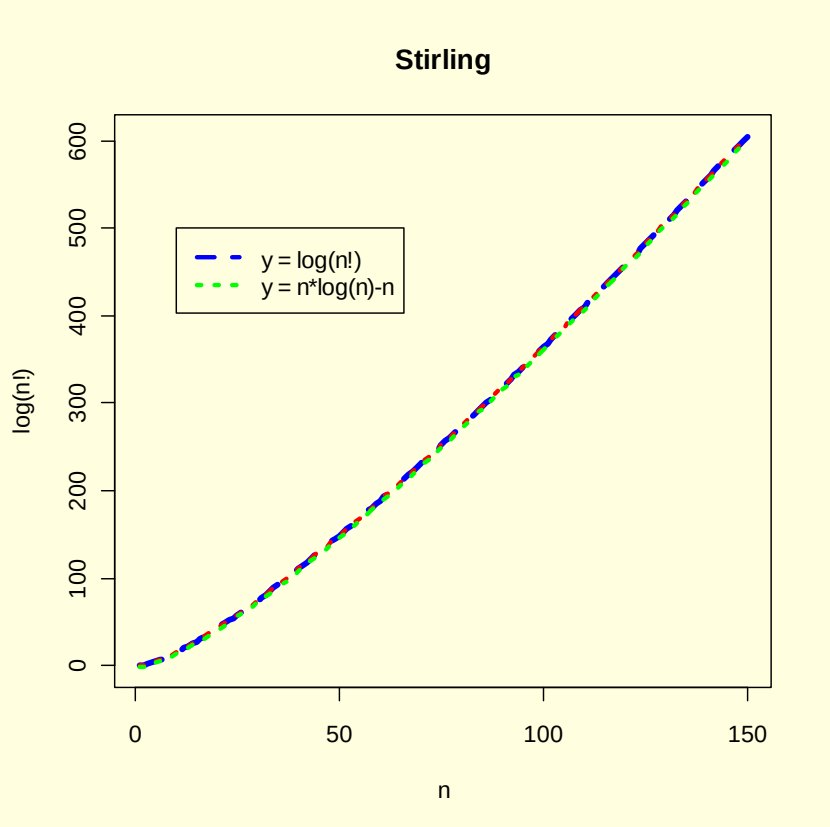
$$\log(n!)=n\cdot\log(n)-n+\frac{1}{2}\cdot\log(n)+log\left(\sqrt{2\pi}\right)$$

Eller enda enklere:

$$\log(n!)\approx n\cdot\log(n)-n$$

På figuren ser vi at det er god overensstemmelse:

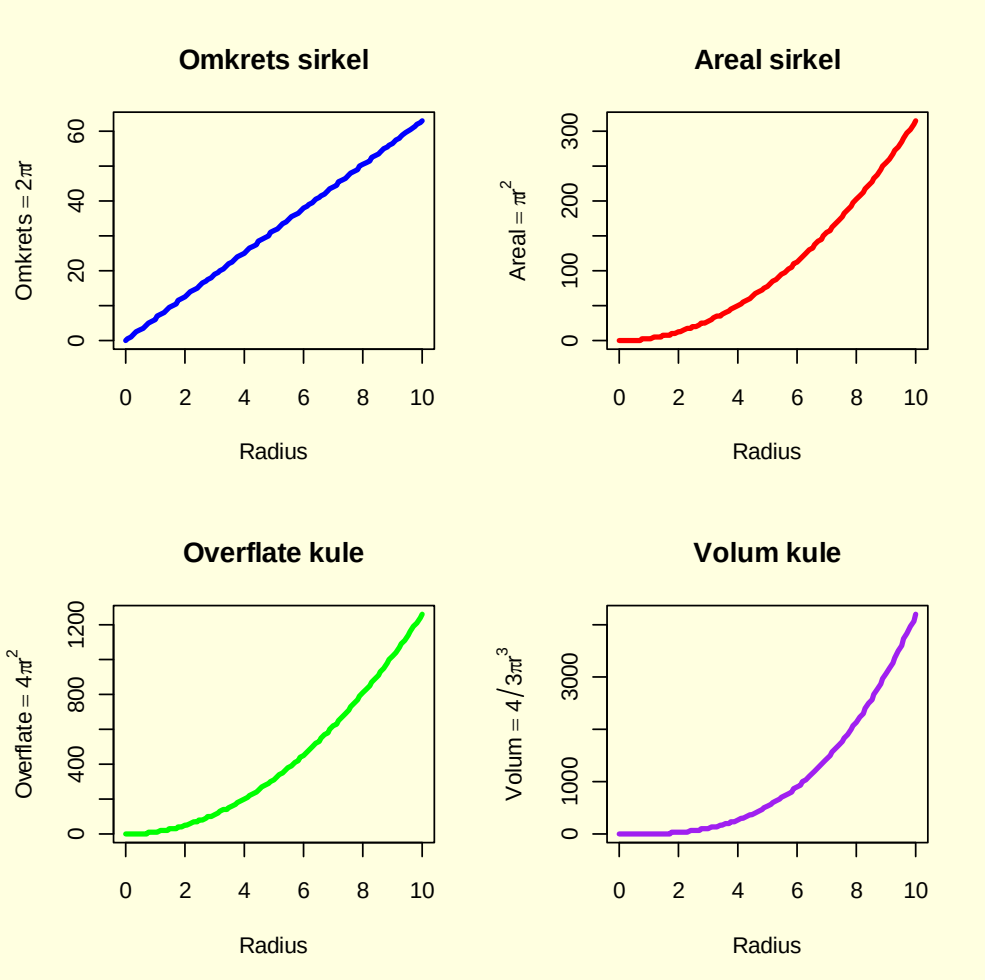
```
n<-seq(1,150,1)
y<-log(factorial(n))
par(bg="lightyellow")
plot(n,y,type="l",lty=2,col="blue",lwd=4,ylab="log(n!)",main="
Stirling")
y2<-n*log(n)-n+1/2*log(n)+log(sqrt(2*pi))
y3<-n*log(n)-n
lines(n,y2,col="red",lty=3,lwd=3)
lines(n,y3,col="green",lty=3,lwd=3)
legend(10,500,c("y = log(n!)","y = n*log(n)-n"),lty=c(2,3),
lwd=c(3,3),col=c("blue","green"))
```



Overflate og omkrets av en sirkel, samt overflate og volum av en kule med radius r. Se størrelsesorden på måleenhetene på y-aksen. Kommandoen **expression()** kan brukes til å skrive greske bokstaver og formler.

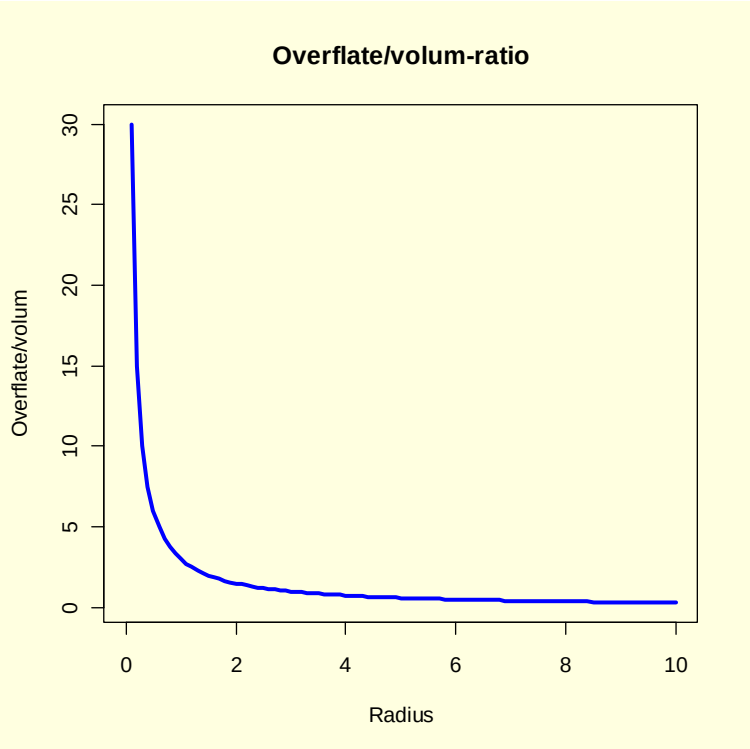
```
r<-seq(0,10,0.1)
sirkelareal<-pi*r^2
sirkelomkr<-2*pi*r
par(mfrow=c(2,2),bg="lightyellow")
plot(r,sirkelomkr,type="l",lwd=3,col="blue",
xlab="Radius",ylab=expression(Omkrets==2*pi*r),main="Omkrets
sirkel")
plot(r,sirkelareal,type="l",col="red",lwd=3,xlab="Radius",ylab
=expression(Areal==pi*r^2),main="Areal sirkel"))
kuleoverfl<-4*pi*r^2
kulevol<-4/3*pi*r^3
```

```
plot(r,kuleoverfl,type="l",lwd=3,col="green",
xlab="Radius",ylab=expression(Overflate==4*pi*r^2),main="Overf
late kule"))
plot(r,kulevol,type="l",lwd=3,col="purple",
xlab="Radius",ylab=expression(Volum==4/3*pi*r^3),main="Volum
kule"))
```



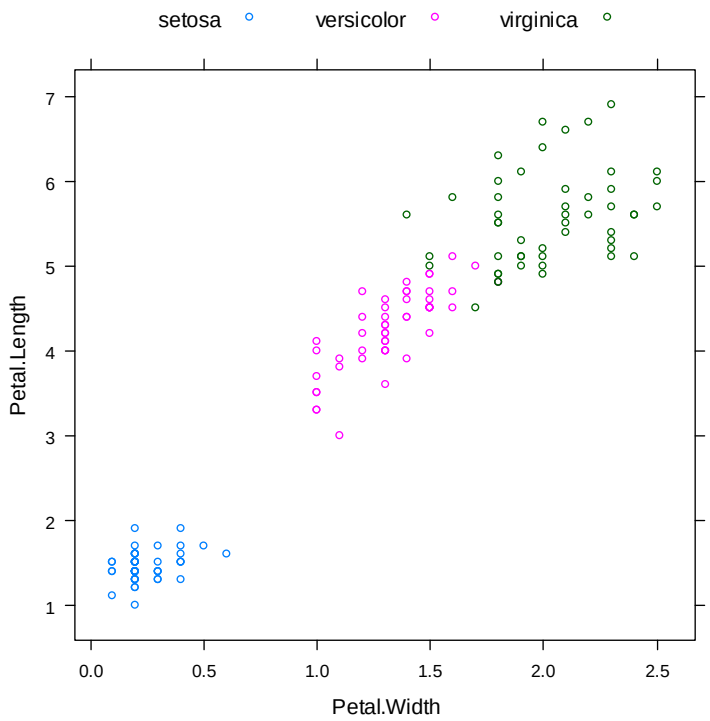
Grunnen til at isbjørn har så stor kropp sammenlignet med bjørner som lever mer sydover skyldes at i kalde strøk er det gunstig med liten overflate i forhold til volum. For organismer som lever i vann og tar opp stoffer gjennom overflaten er det fordel å ha stor overflate i forhold til volumet, og disse blir så små som mulig.

```
par(bg="lightyellow")
plot(r,kuleoverfl/kulevol,type="l",lwd=3,col="blue",
xlab="Radius",ylab="Overflate/volum",main="Overflate/volum-
ratio"))
```

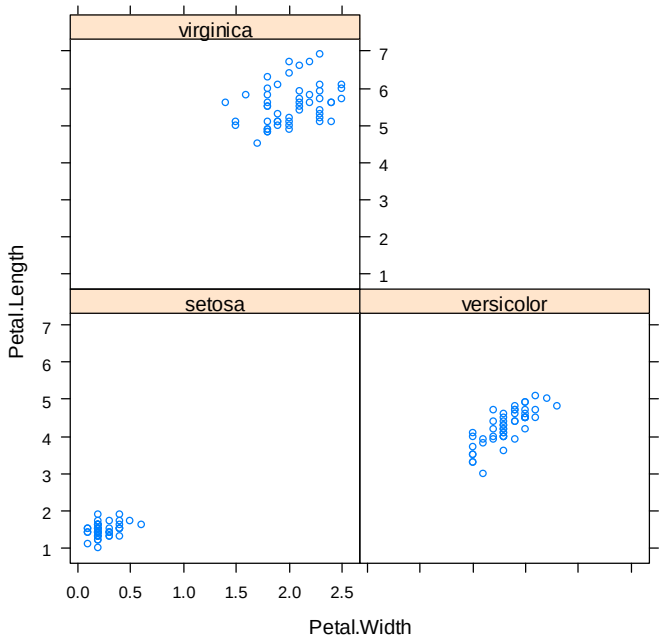


I **library(lattice)** er det mulig å lage trellis-plot. Med **library(grid)** kan man lage et rutenettverk.

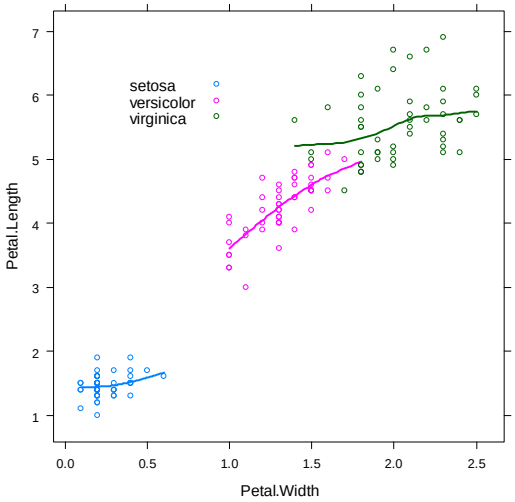
```
library(help=lattice)
library(help=grid)
library(datasets);data(iris);attach(iris);names(iris)
xyplot(Petal.Length~Petal.Width,data=iris,groups=Species,auto.
key=list(columns=3))
```



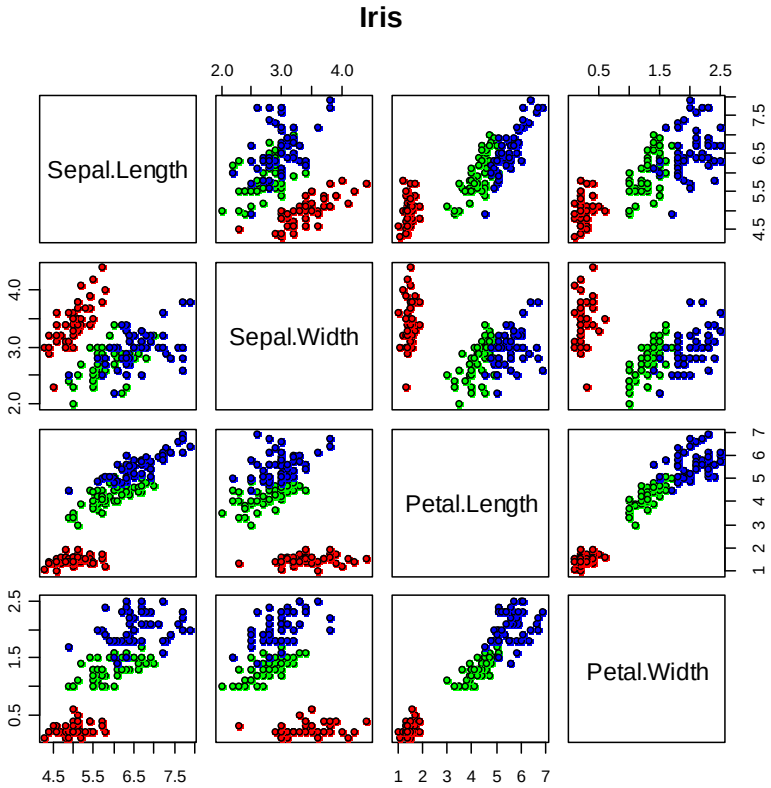

```
xyplot(Petal.Length~Petal.Width|Species)
```




```
xyplot(Petal.Length ~ Petal.Width, groups=Species,panel =
panel.superpose,type = c("p", smooth"),lwd=2,span=.75,auto.key
= list(x = 0.15, y = 0.85))
```




```
pairs(iris[1:4],main = "Iris",pch = 21,bg =
c("red","green","blue")[unclass(iris$Species)])
```



I lattice er det en rekke typer plot som dotplot, stripplot, barchart, densityplot, bwplot,qqmath, splom, parallel, cloud og wireframe.

Innlesing av data

Den enkleste måten å få data inn i R er ved kommandoen **read.table**. Filen må være i ascii-format i form av *.txt fil,

og kan editeres i NotePad med Edit og Replace, og man må bruke punktum (.) i stedet for komma (,) i desimaltall. Har du laget en datafil i Excel kan du lagre den som Text(Tab delimited (*.txt)). Har man en datafil hvor det er mellomrom i variabelnavnene på kategoriske variable (faktor) så bruk innlesingen **read.csv** i stedet. Det er viktig å være klar over i hvilken mappe filen ligger som skal leses inn i R. ved å bruke. Filen kan deretter åpnes ved funksjonen **read.table**. Kommandoen **header=T** sier at første linje i datasettet inneholder variabelnavn. Kommandoen **attach** brukes for å gjøre variabelnavnene tilgjengelig for R og **names** lager en liste med variabelnavnene.

Lese data fra fil

Data kan skrives inn direkte via kommandoen **scan**, eller via kommandoen **read.table** for å lese en fil. Hele stikatalogen til datamappen hvor du har lagret txt-filen må være omgitt av gåseøyne (") og start alltid med en dobbelslæsj (//).
Det må ikke være mellomrom i variabelnavn, bruk . i stedet for å binde sammen to navn du ønsker som variabelnavn
Bruk **NA** ("not available") for å indikere utelatte verdier ("missing values"). Vi kan telle og erstatte NA med en annen verdi for eksempel 0

```
x<-c(2,4,6,NA)
is.na(x)
[1] FALSE FALSE FALSE  TRUE
sum(is.na(x))
[1] 1
x[is.na(x)]=0;x
[1] 2 4 6 0
```

For å kunne bruke variablene i dataarket gi kommandoen **attach** og deretter **names**

```
datavariabel<-read.table("C://mappe/filnavn.txt",header=T)
attach(datavariabel)
```

For å få med navnene på variblene:

```
 names(datavariabel)
```

For å fjerne datasettet datavariabel og variabelnavn

```
detach(datavariabel)
```

Når du starter R er det flere programpakker som ikke blir lastet opp for å spare hukommelse i datamaskinen. En liste over disse pakkene får du med kommandoen **library()**

```
 library()
```

Du kan laste opp pakkene med kommandoen **library()** eller **require()** hvor du angir hvilken pakke (pkg) i parentesen **library(pkg)**. MASS er en pakke laget av Venables and Ripley,2002, som inneholder mange dataset og funksjoner.

Ferdige datasett i R

R inneholder også en rekke ferdige datasett. Du ser den på listen etter å ha skrevet kommandoen library() hvor den er kalt datasets. Du kan få oversikt over alle datasettene i en pakke med kommandoen **data()**.

```
library(datasets)
data()
```

Vil du se hva et spesifikt datasett inneholder bare skriv navnet på datasettet etter prompten i R, for eksempel datasettet CO2 som inneholder verdier for fotosyntese ved forskjellige konsentrasjoner av CO2 for planter som har stått kaldt (chilled) sammenlignet med kontrollplanter.

```
CO2
```

Med kommandoen **data(ds)** laster man opp datasettet ds, for eksempel CO2

Skal man se på de første eller siste radene i en fil brukes **head(filnavn,n=5)**, hvis 5 linjer, eller **tail(filnavn)**

Skal du lagre kommandoer og data i Word bruk skrifttypen ***Courier New*** for å beholde kolonner intakt.

Det er også mulig å laste filen fra klippebordet. Marker filområdet du ønsker å få inn i R med kopier (Ctrl+C) og deretter. For eksempel data fra oppgave 1.

```
navn<-read.table("clipboard",header=T)
```

Har du laget en graf kan du ved å klikke på den med høyre musetast og velg "copy as a metafile" kan grafen limes inn i word.

Man kan kopiere innholdet i en variabel for eksempel variabelen y inn i en tekstfil som blir lagret i det direktoriet du befinner deg med kommandoen **dump("y","filnavn.txt")**. I hvilken mappe dette er finner du med kommandoen **getwd()**

Kommandoen **source()** leser inn en fil med R-kommandoer.

Med kommandoene **read.table(), scan(), source(), read.csv()** brukes file= for å spesifisere filnavnet, men filnamet kan også bestemmes interaktivt med kommandoen **file.choose**()

Du kan også hente datafiler fra internett med kommandoen **url()**

```
site="http://www.osv/data/datanavn.txt"
read.table(file=url(site),header=TRUE)
```

Hjelp får det ved å skrive **help** etterfulgt av navnet på kommandoen du ønsker å vite mer om:

```
help(kommando)
```

eller

```
?kommando
```

Kommandoer atskilles med semikolon (;) eller ny linje
Kommentarer kan settes inn ved å starte med #
Hvis en kommando ikke er fullført ved slutten av linjen gir R prompt +
Vertikale piler på tastbordet kan brukes til å bevege seg gjennom tidligere kommandoer.
Gamle kommendoer kan redigeres med del-tasten og ved å skrive inn nye bokstaver.
Arbeidsdirektoriet er "work"
Oversikt over objektene som R har brukt får du med:

```
objects()
```

Eller du kan liste opp objektene med:

```
 ls()
```

Du kan også vise en liste presentert i en nettleser (”browser”):

```
browseEnv()
```

Samlingen av objekter som er lagret kalles "workspace"
Objekter kan fjernes med kommandoen **rm**:
rm(navnpåobjektet, eventuelt, atskilt, med, komma, hvis flere objekter)
Objekter kan lagres permanent, og du blir spurt om dette når du avslutter R.
Kommando som fjerner alt:

```
 rm(list=ls())
```

Hvis du lagrer blir objektene skrevet inn i en fil .RData i direktoriet du befinner deg og kommandoene du har brukt blir lagret som .Rhistory
Kommandoen **history** gir en oversikt over de siste 25 kommandoene du har gitt:

```
 history()
```

Kommandoen **search** gir en oversikt over objekter og pakker som er lastet opp. .GlobalEnv er arbeidsrommet "workspace"

```
 search()
 [1] ".GlobalEnv"        "oppg1"             "package:methods"
 [4] "package:stats"     "package:graphics"  "package:grDevices"
 [7] "package:utils"     "package:datasets"  "Autoloads"
[10] "package:base"
```

R opererer med navngitte datastrukturer, hvorav den enkleste er en vektor som består av en ordnet rekke med tall.

Har man skrevet inn en vektor eller datamatriske kan man bruke et enkelt editeringsvindu med kommandoen **data.entry()** og i parentesen angis navnet på vektoren eller datamatrisen. For eksempel for datavektoren x:

```
 data.entry(x)
```

Alle variable i R blir betraktet som objekter. Regneoperasjonene regnes ut i rekkefølgen: potenser, kvadratrøtter, multiplikasjon, divisjon, addisjon og subtraksjon. Ønsker man en annen rekkefølge må det angis i parenteser () hva som skal regnes ut først.

Funksjon	Betydning
log(x)	Logaritmen til x med grunntall e
exp(x)	Eksponensialfunksjon. Antilog x (e^x)
log10(x)	Logaritmen til x med grunntall 10
sqrt(x)	Kvadratroten til x
abs(x)	Absoluttveriden til x
factorial (x)	x-fakultet (x!) (1·2·3·4·5…x
cos(x)	Cosinus til x (i radianer)
sin(x)	Sinus til x (i radianer)
tan(x)	Tangens til x (i radianer)
acos(x),asin(x),atan(x)	Inverse trigonometriske funksjoner
acosh(x),asinh(x)	Inverse hyperbolske trigenometriske f.
floor(x)	Største integer <x
ceiling(x)	Minste integer >x
choose(n,x)	Binomialkoffisienter n!/x!(n-x)!)
gamma(x)	(x-1)! for integer x (Γx)
lgamme(x)	Naturlig logaritme til gamma x
round(x,digits=0)	Avrunder verdien til x til en integer
trunc(x)	Nærmeste integer til x mellom x og 0
runif(n)	Lager n tilfeldige tall mellom 0 og 1
signif(x,digits=6)	Gir x med 6 signifikante tall

Operasjon	Betydning
max(x)	Maksimumsverdien i x
min(x)	Minimumsverdien i x
sum(x)	Summerer alle tallene i x
mean(x)	Beregner aritmetisk middeltall i x
median(x)	Medianverdien i x
range(x)	Vektoren min(x) og max(x)
var(x)	Variansen til x
sort(x)	Sorterer x
rank(x)	Rangerer verdiene i x
order(x)	Heltallsvektor inneholder permutasjon av sortert x i stigende orden
quantile(x)	Minimum, 1.kv,median, 3.kv., maksimum
colMeans(x)	Kolonnemiddeltall av data i matrisen x
colSums	Kolonnesummen av data i matrisen x
colVars	Kolonnevariansen av data i matrisen x
rowSums	Radsummene av data i matrisen x
rowMeans	Radmiddeltallene av data i matrisen x
rowVars	Radvariansen av data i matrisen x
cor(x,y)	Korrelasjon mellom vektor x og y

Det er forkjellige måter å beregne sentrum av en prøve med måleverdier. Hovedtendensen er gjenomsnittet også kalt **middelverdien** (aritmetisk middeltall eller aritmetisk gjennomsnitt) som er summen av alle målingene dividert på antall observasjoner eller måleverdier(n). Σ er summetegnet Sigma:

$$\mu = \overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

En annen måte å beregne hovedtendensen er **median** (midtverdien), den verdi hvor halvparten av målingene ligger under og den andre halvparten over, samt **mode** (modalverdi) som er det tallet i prøven som forekommer med størst frekvens. Median splitter dataene i to halvdeler, halvparten er mindre enn midtverdien og den andre halvparten er større enn midtverdien.

Kvadratsummen SS ("sum of square") er kvadrerte avvik fra middeltallet:

$$SS = \sum_{i=1}^{n} (x_i - \overline{x})^2$$

Variasjonen beskrives som **varianse** (s^2) som er kvadratsummen dividert på antall frihetsgrader ($n-1$).

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$

Standardavvik (s eller SD) er et mål på spredningen av dataene eller observasjonene rundt det sanne middeltallet (μ) eller et estimat av middeltallet (x med strek over ¯)

$$s = \sqrt{s^2}$$

Standardfeil (SE) er et mål på spredning av middeltall rundt det sanne middeltallet μ:

$$SE = \frac{s}{\sqrt{n}}$$

Sentralgrenseteoremet sier at hvis observasjonene eller dataene fra et forsøk er normalfordelte så vil også middelverdiene fra en rekke tilsvarende like forsøk også bli normalfordelte.

Middelverdien av en tallrekke kan beregnes slik:

```
 x=scan()
1: 26 8 6 5 4 3 2 2
9:
Read 8 items
```

Eller *x* kan også bestemmes slik:

```
 x<-c(26,8,6,5,4,3,2,2);x
[1] 26  8  6  5  4  3  2  2
```

Det aritmetiske gjenomsnitt:

```
 sum(x)/length(x)
[1] 7
```

Eller enklere kan gjennomsnittsverdien bestemmes:

```
 mean(x)
[1] 7
```

Median-verdi:

```
 median(x)
[1] 4.5
```

Mode kan finnes på følgende måte:

```
 which(table(x)==max(table(x)))
2
1
```

Eller:

```
 which.max(table(x))
2
1
```

Som betyr at 2 er mode og det er det første tallet i rekken hvis tallrekken blir sortert.

Altså, for tallrekken *x* foran er middeltall =7, medianverdi=4.5 og mode=2.

Hvis vi har en tallrekke z så kan vi finne spredningen av verdiene med kommandoen **range()** og forskjellen mellom største og minste tall med kommandoen **diff(range()).**

```
 z=scan()
1: 4 6 9 12 15 5 31 56 66 5 3 7 26 11
15:
Read 14 items
```

Tallvektoren kan også leses inn på denne måten:

```
 z<-c(4,6,9,12,15,5,31,56,66,5,3,7,26,11);z
 range(z)
[1]  3 66
 diff(range(z))
[1] 63
```

I den samme tallrekken z kan vi finne variansen og standardavviket (sd):

```
 var(z)
[1] 398.3736
 sd(z)
[1] 19.9593
```

Kvantiler gir et mål på spredningen av et datasett, hvor **p-te kvantil** i det etter størrelse sorterte datasettet er:

$1 + p(n-1)$

hvor *p* har verdier fra 0-1. p-te kvantil deler datasettet slik at 100p% av datasettet er mindre enn denne og 100(1-p)% er større enn denne. Kvantilen 0.5 er det samme som midtverdien (median). **Persentiler** er egentlig det samme som kvantiler, men hvor verdiene oppgis i %, p-persentilverdien til en fordeling er verdien slik at p% av observasjonene faller under denne. Det vil si at median er lik 50. persentil. **Kvartilene** Q_1, Q_2 og Q_3 deler de sorterte verdiene i 4 like store deler og tilsvarer persentilene 0, 25, 50, 75 og 100 persentilen. Vi har tidligere sett at median angir midtverdien. Første kvartil (Q_1) har ¼ av observasjonene under seg og ¾ av observasjonene over seg. Andre kvartil (Q_2) er median og skiller de nedre 50% fra de øvre 50%. Tredje kvartil (Q_3) skiller de nedre 7%5 av verdiene fra de øvre 25%. **Kvintiler** tilsvarer persentilene 0, 20, 40, 60, 80 og 100.

Vi lager et datasett med tallene fra 0-50, og vi finner kvantil-verdien med kommandoen **quantile().**

```
 x=0:50
 median(x)
[1] 25
```

```
 quantile(x)
  0%  25%  50%  75% 100%
 0.0 12.5 25.0 37.5 50.0
 quantile(x,0.25)
 25%
12.5
```

Men kan få denne fem-inndelingen også med kommandoen **fivenum()**

```
 fivenum(x)
[1]  0.0 12.5 25.0 37.5 50.0
```

Interkvartilområdet er forskjellen mellom første og tredje kvartil og inkluderer median og 25% av verdiene over og under median. Bestemmes i R med kommandoen **IQR()**

```
 IQR(x)
[1] 25
```

Med kommandoen **summary()** får vi minimums-og maksimumsverdi, kvartiler, middelverdi og midtverdi:

```
 summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    12.5    25.0    25.0    37.5    50.0
```

Matriser og matriseoperatorer

En matrise er en firkantet tabell med tall ordnet i rader og kolonner. En *m x n* matrise har *m* rader og *n* kolonner. Matriser med bare en kolonne eller en rad kalles en vektor En vektor er en *1 x n* matrise og en kolonnevektor er en *n x 1* matrise.
En 1x1 matrise er bare et tall, en **skalar**.
Vi har en *mxn* matrise kalt A med *m* rader og *n* kolonner:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix}$$

Hvis det er like mange rader som kolonner (m=n) så har vi en nxn **kvadratmatrise.**

En vektor (x_1, x_2,..,x_n) er en *1xn* matrise (radmatrise)
En kolonnevektor:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

er en *nx1* matrise (kolonnematrise).

Vi kan **transposere** matrisen A vi startet med ved å bytte rader og kolonner og får den tranposerte matrisen $^{t}\mathbf{A}$:

$$\begin{pmatrix} a_{11} & a_{21} & a_{31} & \cdots & a_{m1} \\ a_{12} & a_{22} & a_{32} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & a_{3n} & \cdots & a_{mn} \end{pmatrix}$$

Transposering av en radvektor gir en kolonnevektor, og transposering av en kolonnevektor gir en radvektor.

Hvis vi multipliserer en radvektor med den transposerte kolonnevektoren får vi et skalart kvadratprodukt:

$$^tB \cdot B = (b_1 \quad b_2 \quad \ldots \quad b_n) \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = b_1^2 + b_2^2 + \cdots + b_n^2$$

En matrise kalles **symmetrisk** hvis den er lik den transposerte matrisen: $^tA=A$.

For eksempel er følgende matrise symmetrisk:

$$\begin{pmatrix} 1 & -1 & 2 \\ -1 & 0 & 3 \\ 2 & 3 & 7 \end{pmatrix}$$

```
A<-matrix(c(1,-1,2,-1,0,3,2,3,7),ncol=3);A
     [,1] [,2] [,3]
[1,]    1   -1    2
[2,]   -1    0    3
[3,]    2    3    7
t(A)
     [,1] [,2] [,3]
[1,]    1   -1    2
[2,]   -1    0    3
[3,]    2    3    7
```

En kvadratmatrise kalles en **diagonalmatrise** hvis alle komponentene er lik 0 bortsett fra diagonalen:

$$\begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{pmatrix}$$

Hvis alle tallene i diagonalen er lik 1 og resten av komponentene er lik 0 har vi en **identitetsmatrise** (I) eller I_n:

$$I_n = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

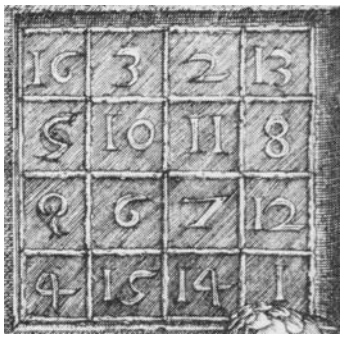
Hvis *A* er en *nxn* kvadratmatrise så kalles *A* **inverterbar** eller **ikke-singulær** hvis det finnes en **invers matrise** A^{-1} slik at A ganger A^{-1} er lik identitetsmatrisen:

$$A \cdot A^{-1} = I_n$$

Hvis $A=(a_{ij})$ er en *nxn* kvadratmatrise så er **trace A** (tr(A)) lik summen av diagonalelementene:

$$tr(A) = a_{11} + a_{22} + \cdots + a_{nn}$$

Lag en "magisk" matrise jfr. Albrecht Düreres *Melencolia I* med et magisk kvadrat, med innslag av alkymistenes mystikk.



Det finnes ikke noe magisk i kvadratet, men har den interessante egenskapen at tallet 34 går igjen i flere av summeringene.

Fra Donald Duck nr. 10/1995

- Alle hjørnene summeres til 34
- De fire tallene i midten summeres til 34
- 3 og 2 i første rad som vender mot 15 og 14 i fjerde rad summeres til 34
- 5 og 9 i første kolonne som vender mot 8 og 12 i fjerde kolonne summeres til 34
- De fire kvadratene i hvert hjørne adderes til 34
- Summeres kolonnene blir dette 34
- 15+9+2+8 = 34
- Summen av diagonalene blir 34

```
16     3    2   13
 5    10   11    8
 9     6    7   12
 4    15   14    1
```

```
v<-c(16,5,9,4,3,10,6,15,2,11,7,14,13,8,12,1)
magisk<-matrix(v,nrow=4)
magisk
     [,1] [,2] [,3] [,4]
[1,]   16    3    2   13
[2,]    5   10   11    8
[3,]    9    6    7   12
[4,]    4   15   14    1
```

Summer kolonner og rader i matrisen

```
colSums(magisk)
[1] 34 34 34 34
rowSums(magisk)
[1] 34 34 34 34
```

Diagonalen til matrisen:

```
diag(magisk)
[1] 16 10  7  1
```

Summer diagonalen:

```
sum(diag(magisk))
[1] 34
```

Transposering av en matrise forandrer rader til kolonner og kolonner til rader:
Transposer matrisen:

```
t(magisk)
     [,1] [,2] [,3] [,4]
[1,]   16    5    9    4
[2,]    3   10    6   15
[3,]    2   11    7   14
[4,]   13    8   12    1
```

En alternativ måte å lage den magiske matrisen er ved kommandoen **rbind()** som kombinerer datavektorer i rader:

```
magisk2<-rbind(c(16,3,2,13),c(5,10,11,8),c(9,6,7,12),c(4,15,14,1))
magisk2
     [,1] [,2] [,3] [,4]
[1,]   16    3    2   13
[2,]    5   10   11    8
[3,]    9    6    7   12
[4,]    4   15   14    1
```

Man kan adressere til en matrise akkurat som en vektor, men man må oppgi hvilke rad(er) eller kolonne(r)det er snakk om:

```
rad1<-magisk[1,];rad1
[1] 16  3  2 13
kolonne2<-magisk[,2];kolonne2
[1]  3 10  6 15
```

To matriser kan multipliseres med hverandre **%*%**

```
magisk%*%magisk
     [,1] [,2] [,3] [,4]
[1,]  341  285  261  269
[2,]  261  301  309  285
[3,]  285  309  301  261
[4,]  269  261  285  341
```

Vi lager to enkle matriser A og B:

```
A<-rbind(c(2,3,4),c(1,4,6));A
     [,1] [,2] [,3]
[1,]    2    3    4
[2,]    1    4    6
B<-rbind(1,2,1);B
     [,1]
[1,]    1
[2,]    2
[3,]    1
A%*%B
     [,1]
[1,]   12
[2,]   15
```

Det som skjer er følgende:

$$A * B = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 3 \cdot 2 & 4 \cdot 1 \\ 1 \cdot 1 & 4 \cdot 2 & 6 \cdot 1 \end{bmatrix} = \begin{bmatrix} 12 \\ 15 \end{bmatrix}$$

Generelt kan to matriser A og B av samme størrelse (NB!) adderes (A+B), subtraheres (A-B), multipliseres (A*B), divideres (A/B) og opphøyes (A^B) ledd for ledd.

```
A<-rbind(c(1,1,1),c(2,4,9),c(16,25,36))
A
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    4    9
```

```
[3,]   16   25   36
```

```
B<-rbind(c(1,1,1),c(2,2,2),c(3,3,3))
B
```

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
[3,]    3    3    3
```

```
A*B
```

```
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    4    8   18
[3,]   48   75  108
```

```
A+B
```

```
     [,1] [,2] [,3]
[1,]    2    2    2
[2,]    4    6   11
[3,]   19   28   39
```

Vi kan multiplisere matrisene A og B med hverandre:

$$A = \begin{pmatrix} 2 & 1 & 5 \\ 1 & 3 & 2 \end{pmatrix} * \; B = \begin{pmatrix} 3 & 4 \\ -1 & 2 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 15 & 15 \\ 4 & 12 \end{pmatrix}$$

```
A<-rbind(c(2,1,5),c(1,3,2));A
```

```
     [,1] [,2] [,3]
[1,]    2    1    5
[2,]    1    3    2
```

```
B<-rbind(c(3,4),c(-1,2),c(2,1));B
```

```
     [,1] [,2]
[1,]    3    4
[2,]   -1    2
[3,]    2    1
```

```
A%*%B
```

```
     [,1] [,2]
[1,]   15   15
[2,]    4   12
```

Vi kan summere to matriser:

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 2 & 3 & 4 \end{pmatrix} + B = \begin{pmatrix} 5 & 1 & -1 \\ 2 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 6 & 0 & -1 \\ 4 & 4 & 3 \end{pmatrix}$$

```
A<-matrix(c(1,2,-1,3,0,4),ncol=3);A
```

```
     [,1] [,2] [,3]
[1,]    1   -1    0
[2,]    2    3    4
```

```
B<-matrix(c(5,2,1,1,-1,-1),ncol=3);B
```

```
     [,1] [,2] [,3]
[1,]    5    1   -1
[2,]    2    1   -1
```

```
A+B
```

```
     [,1] [,2] [,3]
[1,]    6    0   -1
[2,]    4    4    3
```

Man kan lage en matrise av en tallvektor. Kommandoen **dim()** endrer dimensjonen på vektoren:

```
y<-1:16;dim(y)<-c(4,4);y
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Kommandoen **table()**brukes til å lage tabeller (kontingenstabeller) ved å kombinere andre datasett.

Lage en matrise med tallene 1-16 ordnet rekkevis, og sette små og store bokstaver som kolonne- og radnavn:

```
matrise<-matrix(1:16,nrow=4,byrow=T)
rownames(matrise)<-LETTERS[1:4]
colnames(matrise)<-letters[1:4]
matrise
   a  b  c  d
A  1  2  3  4
B  5  6  7  8
C  9 10 11 12
D 13 14 15 16
```

Lage en 10x10 matrise med bare 1-tall:

```
matrix(1,10,10)
```

Kommandoen **diag(x,n)** lager en *n x n*-matrise med vektoren *x* på diagonalen. For eksempel en **identitetsmatrise** A med 1 på diagonalen, for øvrig 0:

```
A<-diag(1,8);A
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    0    0    0    0    0    0    0
[2,]    0    1    0    0    0    0    0    0
[3,]    0    0    1    0    0    0    0    0
[4,]    0    0    0    1    0    0    0    0
[5,]    0    0    0    0    1    0    0    0
[6,]    0    0    0    0    0    1    0    0
[7,]    0    0    0    0    0    0    1    0
[8,]    0    0    0    0    0    0    0    1
```

Hvis du skal editere matrisen kan man bruke data.entry

```
data.entry(A)
```

Hvis vi har 3 variable hver med 3 observasjoner ordnet i kolonner i en matrise A:

```
A<-matrix(c(-3,-5,4,1,3,2,2,1,4),ncol=3);A
     [,1] [,2] [,3]
[1,]   -3    1    2
[2,]   -5    3    1
[3,]    4    2    4
```

Vi kan finne **kovariansematrisen** til A som vi ser er en matrise med variansene plassert i diagonalen. Matrisen som blir

symmetrisk omkring diagonalen kalles også **varianse-kovariansematrise**. Kovariansene mellom hvert par med variable står utenfor diagonalen.

	x_1	x_2	x_3	x_4
x_1	var(x_1)	cov(x_1,x_2)	cov(x_1,x_3)	cov(x_1,x_4)
x_2	cov(x_2,x_1)	var(x_2)	cov(x_2,x_3)	cov(x_2,x_4)
x_3	cov(x_3,x_1)	cov(x_3,x_2)	var(x_3)	cov(x_3,x_4)
x_4	cov(x_4,x_1)	cov(x_4,x_2)	cov(x_4,x_3)	var(x_4)

En kovariansematrise kan betegnes som cov(x_i,x_j)=$\sum$, og den inverse kovariansematrisen $\sum$-1. Produktet av kovariansematrise og invers kovariansematrise blir lik identitetsmatrisen: $\sum \cdot \sum^{-1} = I$.

Hvis X og Y er tilfeldige variable så vil variansen til summen være lik:

$$var(X+Y) = var(X) + var(Y) + 2 \cdot cov(X,Y)$$

Kovariansen er et forventet (E) produkt:

$$cov(x,y) = E[(x-\bar{x}) \cdot (y-\bar{y})]$$
$$cov(x,y) = E(xy) - E(x) \cdot E(y)$$

Når x og y er ikke-korrelert er $E(xy)=E(x)E(y)$ blir kovariansen lik 0.

Kovarianse til matrise A ovenfor:

```
kovA> cov(A);kovA
          [,1] [,2]      [,3]
[1,] 22.333333 -1.0  7.166667
[2,] -1.000000  1.0 -0.500000
[3,]  7.166667 -0.5  2.333333
```

Variansene står i diagonalen:

```
 var<-diag(cov(A));var
[1] 22.333333  1.000000  2.333333
```

Vi kan finne variansen for den første kolonnen i matrisen A:

```
 var1<-var(A[,1]);var1
[1] 22.33333
```

Eller vi kan alternativt finne alle variansene samtidig:

```
 varb<-apply(A,2,var);varb
[1] 22.333333  1.000000  2.333333
```

En radmatrise med alle middelverdiene:

```
 m<-apply(A,2,mean);m
[1] -1.333333  2.000000  2.333333
```

Korrelasjonskoeffisienten r er lik:

$$r = \frac{cov(x,y)}{\sqrt{s_x^2 \cdot s_y^2}}$$

I en **korrelasjonsmatrise** har man korrelasjonskoeffisientene (ρ) i stedet for kovarianse, og tallene i diagonalen blir lik 1.
Hvis alle variablene i en kovariansematrise er i form av enhetsvarianse så blir kovariansematrisen lik korrelasjonsmatrisen.
Egenvektorene til en kovariansematrise er lik prinsipalkomponentene til fordelingen.
Vi kan finne egenverdiene og egenvektoren til kovarianse matrisen:

```
 eigen(kovA)
$values
[1] 2.468777e+01 9.788921e-01 3.424937e-16

$vectors
            [,1]        [,2]       [,3]
[1,]  0.95092219  0.09767338 -0.2936101
[2,] -0.04660094  0.98325677  0.1761661
[3,]  0.30590086 -0.15383771  0.9395524
```

Vi kan sammenligne egenvektorene til kovariansematrisen med prinsipalkomponentene (PC) til matrise A og ser at de blir like:

```
 prcomp(A)
Standard deviations:
[1] 4.968679e+00 9.893897e-01 6.957723e-17

Rotation:
             PC1         PC2        PC3
[1,]  0.95092219 -0.09767338  0.2936101
[2,] -0.04660094 -0.98325677 -0.1761661
[3,]  0.30590086  0.15383771 -0.9395524
```

Første prinsipalkomponent viser retningen (vektoren) gjennom datasettet som forklarer det meste av variasjonen.

Hvis man har en **kvadratmatrise** A så har vi følgende sammenheng:

$$A \cdot v = \lambda \cdot v$$

hvor v er **egenvektoren**, og λ er en skalar kalt **egenverdi**
matrise ·egenvektor = skalar · egenvektor
v er en egenvektor med egenverdi λ.
Kommandoen **eigen** gir sorterte egenverdier hvor den med størst absoluttverdi kommer først. Absluttverdien kan også beregnes for komplekse tall a+bi.

$$|a + bi| = \sqrt{a^2 + b^2}$$

Hvis vi har en 2x2 matrise (kvadratmatrise) med tallene a,b,c og d:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Så vil **determinanten** for matrisen (det(A)) være lik:

$$det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

En kvadratmatrise A har en **invers matrise** (A^{-1}) slik at

$$AA^{-1} = I$$

Hvor I er en **identitetsmatrise.**

Vi har videre determinanten med egenverdien λ og enhetsmatrisen *I*:

$|A - \lambda I| = 0$

Egenverdier og egenvektorer for en matrise:

```
 A<-matrix(c(2,1,3,4),nrow=2);A
     [,1] [,2]
[1,]    2    3
[2,]    1    4
 eigen(A)
$values
[1] 5 1

$vectors
           [,1]       [,2]
[1,] -0.7071068 -0.9486833
[2,] -0.7071068  0.3162278
```

Vi kan finne determinanten til A (|A|):

```
prod(eigen(A)$values)
[1] 5
```

Vi kan lage den inverse matrisen (A^{-1}) med kommandoen **ginv** i pakken MASS:

```
library(MASS)
?ginv
ginv(A)
     [,1] [,2]
[1,]  0.8 -0.6
[2,] -0.2  0.4
```

Vi har at den inverse av inversmatrisen gir den opprinnelige matrisen $(A^{-1})^{-1}$=A

```
ginv(ginv(A))
     [,1] [,2]
[1,]    2    3
[2,]    1    4
```

Vi kan også finne determinanten |A| som stemmer med det vi fant tidligere:

```
1/det(ginv(A))
[1] 5
```

Pakken *boot* (**library(boot)**) har kommendoen **iden(n)** for å lage en *nxn* identitetsmatrise og **zero(n,m)** for å lage en *nxm* matrise med bare 0.

Generelt kan matriser benyttes til å løse *m* lineære ligninger med *n* ukjente:

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_n$$

Som betyr at:

$$Ax = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = b$$

I matriseform blir løsningen $x=A^{-1}b$

I en matrise (**matrix()**) er elementer av samme mode ordnet i rader og kolonner, men **data.frame ()** er en matrise, tabell med data i kolonnevektoer, hvor kolonnene kan ha forskjellig mode.

Løse ligninger med to ukjente

6x + 12y = 20
-8x + 4y = 24
Lager en kvadratmatrise A med koeffisientene 6,-8,12 og 4:

```
A<-matrix(c(6,-8,12,4),nrow=2)
A
```

```
     [,1] [,2]
[1,]    6   12
[2,]   -8    4
```

Lager en kolonnevektor B:

```
B<-matrix(c(20,24),nrow=2)
B
```

```
     [,1]
[1,]   20
[2,]   24
```

Løser ligningen:

```
x<-solve(A,B)
x
```

```
          [,1]
[1,] -1.733333
[2,]  2.533333
```

x=-1.733, y= 2.533
Det vil si det samme som A*x=B(%*% multiplisering av matriser):

```
A%*%x
```

```
     [,1]
[1,]   20
[2,]   24
```

Se at dette stemmer på en figur:
y= 5/3 - 1/2x
y= 6 - 1/2x

Lager et tomt plot:

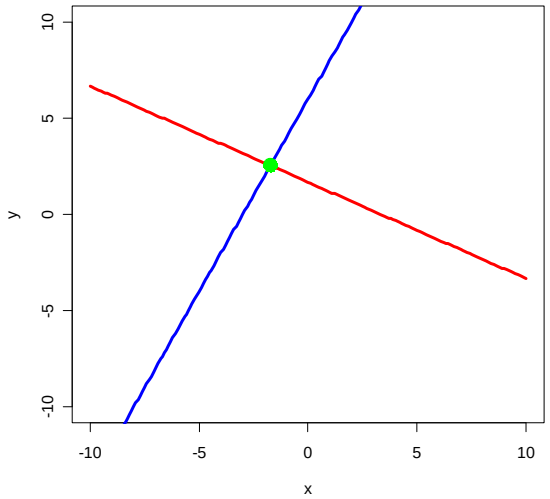
```
plot(-10:10,-10:10,type="n",xlab="x",ylab="y")
```

Lar x gå fra -10 til 10 i trinn på 0.1:

```
x<-seq(-10,10,0.1)
```

Trekker linjene (rød og blå) og setter på koordinatene for skjæringspunktet mellom linjene (fra løsningen av linjene) angitt med et grønt punkt:

```
points(x,(5/3-x/2),type="l",col="red",lwd=3)
points(x,(6+2*x),type="l",col="blue",lwd=3)
points(-1.7333,2.5333,pch=19,cex=2,col="green")
```



Vi kan i stedet bruke **rootSolve** med tilhørende **Newton-Raphson-metode**. Library(rootSolve) må først være lastet ned fra CRAN. Siden røttene finnes ved iterasjoner må man starte med en gjetning hvor røttene befinner seg. Vi kan starte med en gjetning (0,0). Vi ser at vi finner samme resultat:

```
library(rootSolve)
ligning<- function(x){
f1=6*x[1]+12*x[2]-20
f2=-8*x[1]+4*x[2]-24
c(f1=f1,f2=f2)
}
multirot<-multiroot(f=ligning,start=c(0,0));multirot
$root
[1] -1.733333  2.533333
$f.root
          f1           f2
4.294513e-07 1.458593e-07
$iter
[1] 2
$estim.precis
[1] 2.876553e-07
```

Regneoperasjoner på matriser:

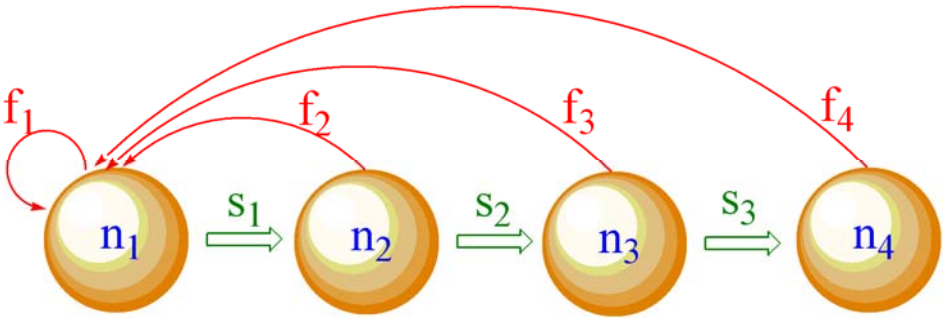
Operasjon	Betydning
solve(A)	Invers matrise til A
solve(A,B)	Løser lineære ligninger. A er koeff.matrise og B er kolonnematrise
backsolve(A,B)	Løser lineære ligninger
eigen(A)	Egenverdier og egenvektorer til kvadratmatrisen A
diag(A)	Diagonalen til en matrise A
sum(diag(A))	Summerer diagonalen i matrise A
prod(eigen(A)$values)	Determinanten til matrise A
svd(A)	Liste med singulære verdier dekompositering av A
qr(A)	En ortogonal (enhets-)matrise og triangulær matrise hvis produkt er lik A
cho(A)	En øvre triangulær matrise som er Choleski nedbrytning av en symmetrisk positiv
kronecker(A,B)	En matrise med dimensjon produktene av dimensjonen av matrisene A og B. Hvert element i A erstattes med det elementet ganger hele matrisen B.
t(A)	Transposerer en matrise A

Til studier av tetthetsuavhengig vekst av populasjoner kan man benytte **Leslie matriser** (etter P.H. Leslie). Populasjonen deles inn i livsstadier eller aldersklasser. Matrisene er en kolonnematrise (populasjonsvektor) som viser antall individer n i hver aldersklasse x, n_x, ved et tidspunkt t, samt en kvadratisk Lesliematrise (L) som viser andelen av individer som overlever fra en aldersklasse til den neste (s_x, aldersavhengig overlevelsesrate) og fekunditet (f_x) som er antall hunnlig avkom per capita født av mødre i aldersklasse x.
For eksempel for fire aldersklasser:

$$\begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}_{t+1} = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \\ s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{bmatrix}_t$$

Som kan skrives :

$$N_{t+1} = L \cdot N_t$$



Aldersklasser n_1-n_4, aldersavhengig overlevelsesrate s_1-s_3 og fekunditet (f_1-f_4)

Det er også mulig å sette inn i Leslie-matrisen antall hanner og deres overlevelsesrate.

```
 L<-rbind(c(0,2,1,1),c(0.6,0,0,0),c(0,0.5,0,0),c(0,0,0.4,0));L
[,1] [,2] [,3] [,4]
[1,]  0.0  2.0  1.0    1
[2,]  0.6  0.0  0.0    0
[3,]  0.0  0.5  0.0    0
[4,]  0.0  0.0  0.4    0
```

hvor hver hunn i alderklasse 2 produserer 2 hunnlige avkom som overlever til neste aldersklasse, alderklasse 3 og 4 får 1 avkom som overlever til neste aldersklasse. 60% av de i første aldersklasse går over til neste aldersklasse osv.
En kolonnevektor inneholder antall individer i hver årsklasse:

```
 n<-rbind(8,12,11,4);n
     [,1]
[1,]    8
[2,]   12
[3,]   11
[4,]    4
```

Når man multipliserer matriser brukes %*%
Hvis man har en Leslie-matrise L som skal multipliseres med en kolonnematrise med aldersstrukturen med en populasjonsstørrelse lik n.
Antall individer neste år fordelt på de forskjellige aldersklassene:

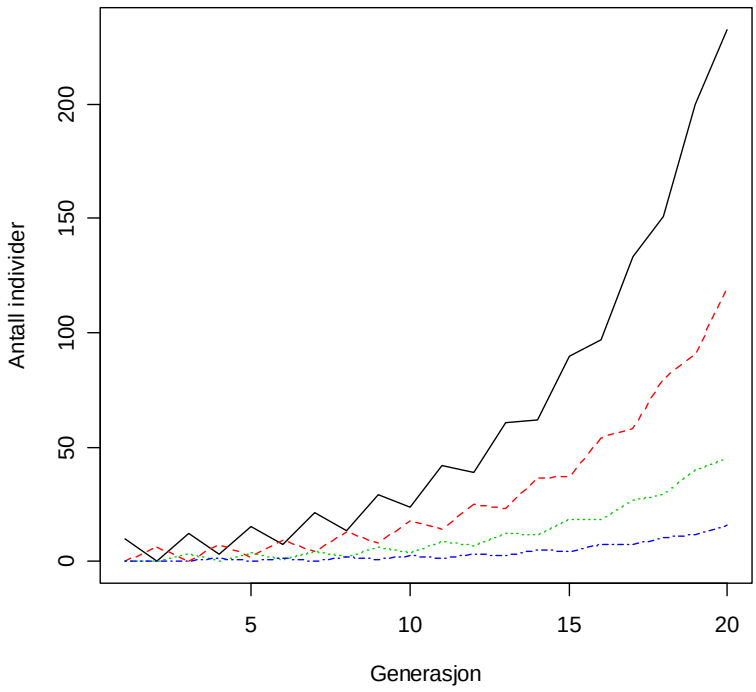
```
 n1<-L%*%n;n1
     [    [,1]
[1,] 39.0
[2,]  4.8
[3,]  6.0
[4,]  4.4
```

Antall individer i første og andre år:

```
 na<-sum(n);na
[1] 35
 nb<-sum(n1);nb
[1] 54.2
 r<-nb/na;r
[1] 1.548571
```

Vi kan nå se på hvordan denne populasjonen vil utvikle seg de neste 20 årene. Vi lager en tom matrise A med plass til 4x20= 80 tall, og starter med 10 hunner i aldersklasse 1, Leslie-matrise som foran i følgende script-fil. Under File på menylinjen finner man New script som kan brukes til å lage **script-filer** som kan eksekveres i kommandovindu.

```
n<-c(10,0,0,0)
A<-matrix(c(0,0,0,0),20,4)
A[1:4]<-n;A
L<-rbind(c(0,2,1,1),c(0.6,0,0,0),c(0,0.5,0,0),c(0,0,0.4,0))
vekst<-function(x)L%*%x
for (i in 2:20){
n<-vekst(n);
A[i,]<-n
cat(i,n,"\n");
}
matplot(1:20,A,type="l",xlab="Generasjon",ylab="Antall
individer")
```



Figuren viser antall individer i de fire forskjellige aldersklassene etter forskjellige generasjoner.
Veksthastigheten mellom 19. og 20. generasjon:

```
r40<-sum(A[20,])/sum(A[19,]);r40
[1] 1.209971
```

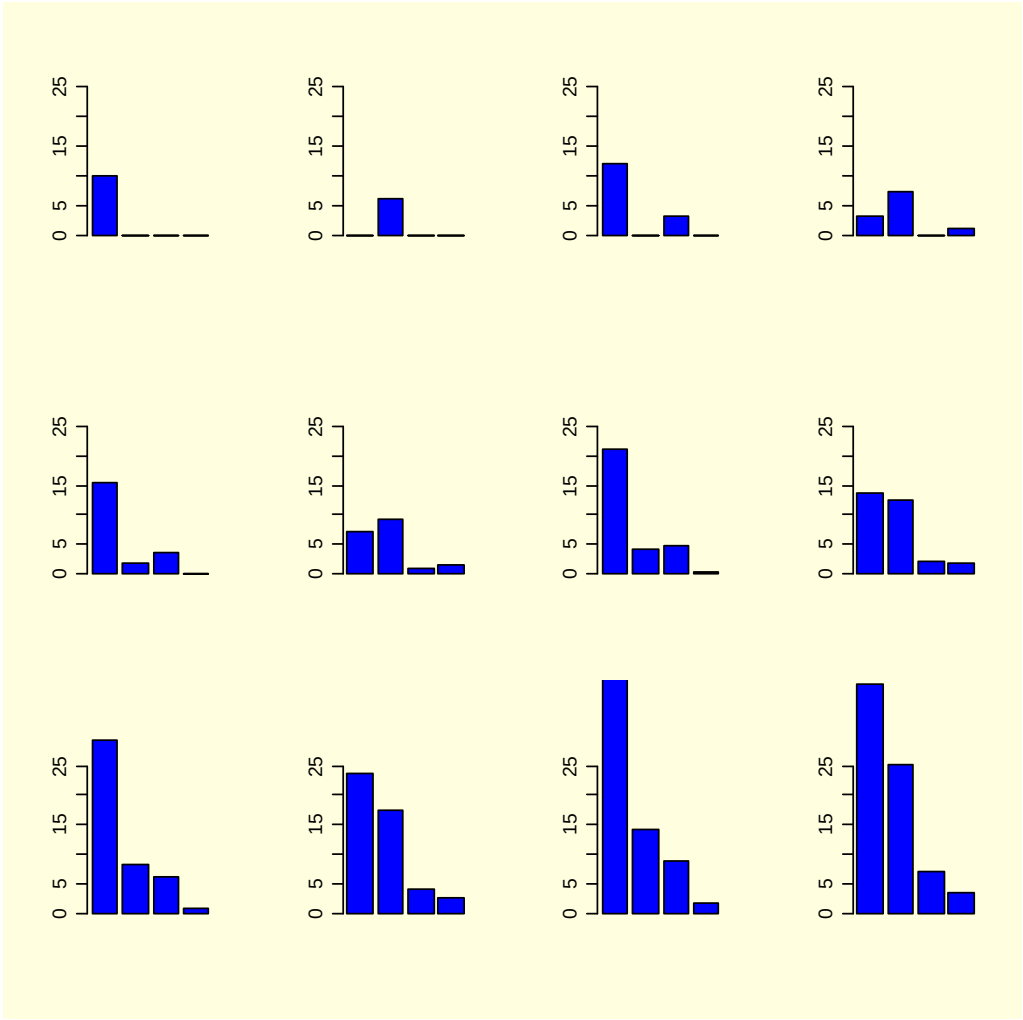
Prosentfordelingen av hunner i 20. generasjon:

```
A[20,]/sum(A[20,])
[1] 0.56221856 0.28977152 0.10935259 0.03865734
```

Et barplot for de 10 første generasjonene:

```
par(mfrow=c(3,4),bg="lightyellow")
barplot(A[1,],i[1],col="blue",ylim=c(0,25))
```

Osv.



Aldersfordeling ved start av 10 hunndyr og Lesliematrise som foran.

Kvadratfunksjon

En **kvadratisk funksjon** er på formen:

$$f(x) = ax^2 + bx + c$$

hvor a, b og c er relle tall og $a \neq 0$, og som har løsning:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Slike kvadratiske funksjoner beskriver en rekke fenomener i naturen. En grafisk framstilling gir en parabel som åpner seg oppover hvis $a>0$ og nedover hvis $a<0$.

Vi kan se litt nærmere på den kvadratiske funksjonen:

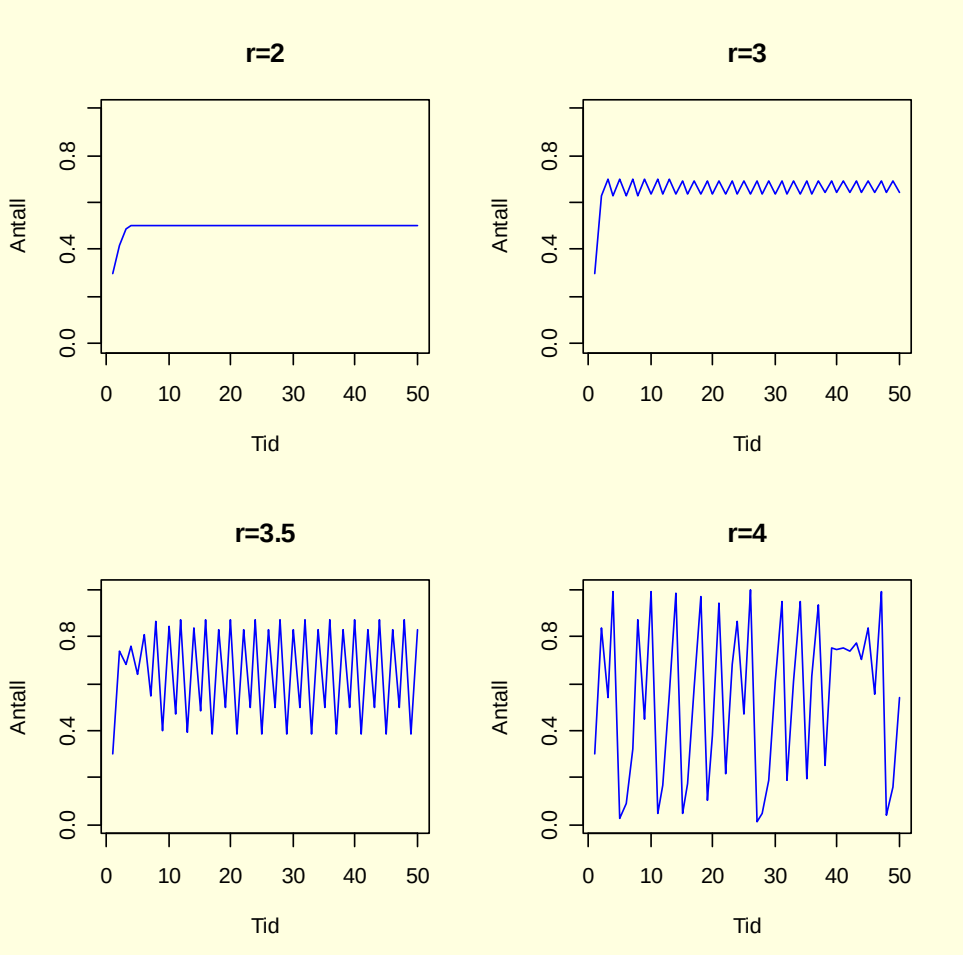
$$f(x) = r \cdot x(1 - x)$$

Hvis vi bruker den til å studere veksten i en populasjon hvor veksthastigheten per capita $r>0$ og x_n er populasjonen i år n og x_{n+1} er populasjonen året etter:

$$x_{n+1} = r \cdot x_n(1 - x_n)$$

Vi skal iterere denne og resultatet påvirkes både av utgangsverdi for x og r. Biologen Robert May viste at populasjonsveksten beskrevet av en logistisk ligning blir kaotisk ved høye verdier av r. Edvard Lorenz som var meterolog ved MIT hadde allerede vist i 1961 i studiet av differensialligninger som beskrev været at inititalbetingelsene var avgjørende for kaos. Systemet oscillerer først mellom to stadier, bifurkasjoner, deretter fire stadier, og i et bifurkasjonsdiagram framkommer det periodiske vinduer i kaos.

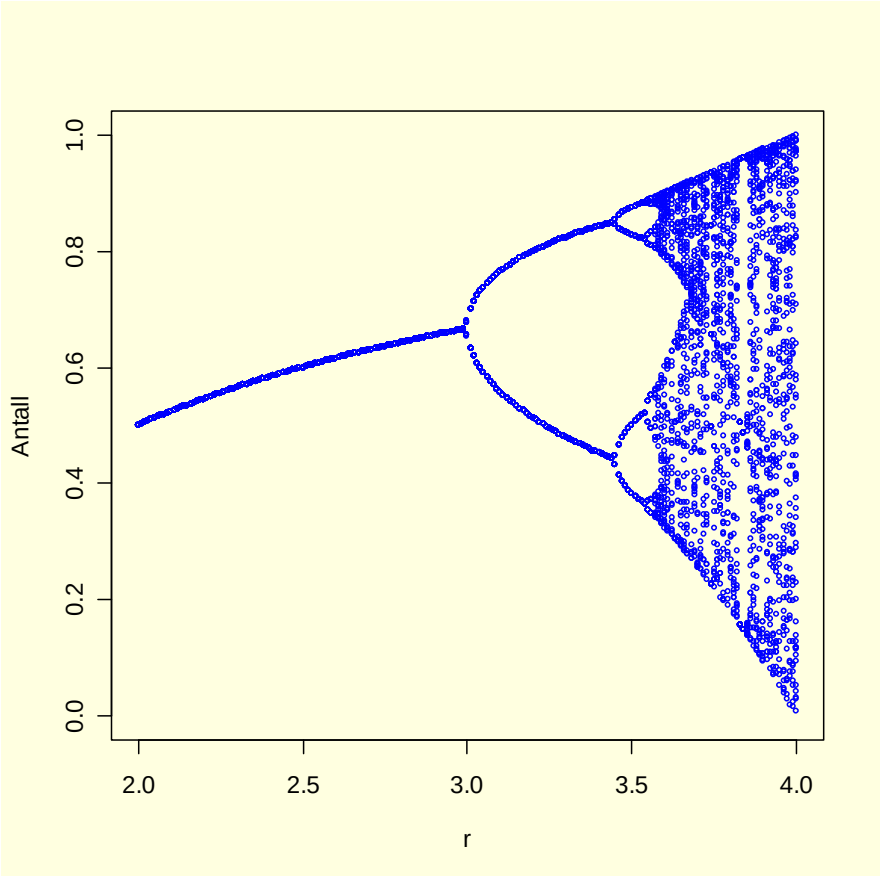
```
par(mfrow=c(2,2),bg="lightyellow")
 x<-rep(0,100)
 r<-2
 x[1]=0.3
for (t in 2:100) x[t]<-r*x[t-1]*(1-x[t-1])

plot(1:100,x,type="l",col="blue",ylim=c(0,1),xlab="Tid",ylab="
Antall",main="r=2")
```



Ved r=2 stabiliserer populasjonen seg, ved r=3 stabiliserer populasjonen seg men svinger mellom to ytterligheter (stabil bane med periode 2), ved r=3.5 er det svining mellom fire ytterpunkter (stabil bane periode 4), og ved r=4 er det en kaotisk svining uten tilsynelatende system.

Vi kan variere *r*=2-4 og plukke ut de tilsvarende antall individer i populasjonen i følgende script-program:

```
populasjon<-function (r){
x<-rep(0,400)
x[1]=0.3
for(t in 2:400) x[t]<-r*x[t-1]*(1-x[t-1])
x[351:400]}
par(mfrow=c(1,1),bg="lightyellow")
plot(c(2,4),c(0,1),type="n",xlab="r",ylab="Antall")
for (r in seq(2,4,0.01))
points(rep(r,50),sapply(r,populasjon),col="blue",cex=0.5)
```

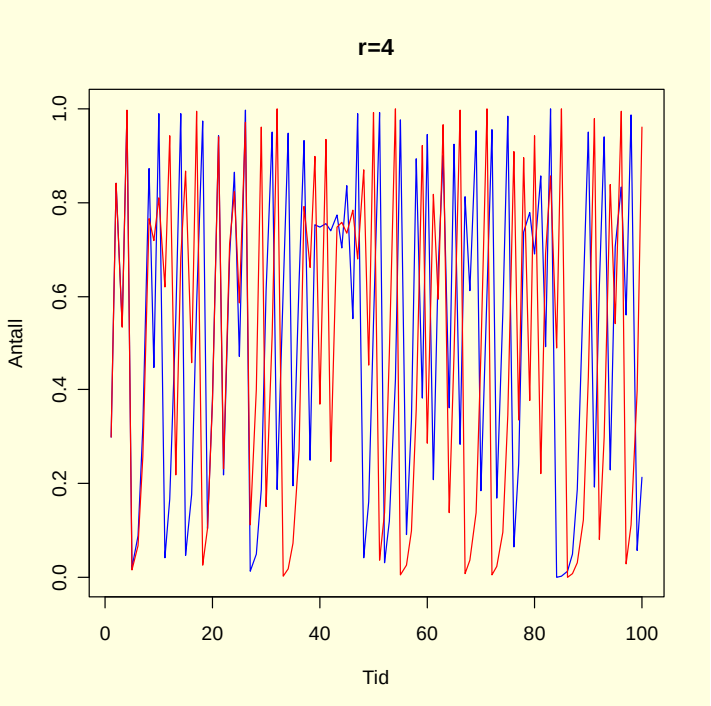


Figuren viser det samme som figurene foran hvor r opptil 3 gir en stabil populasjonsmengde, for r>3 blir det en **bifurkasjon** med to stabile verdier, når r=3.5 har vi 4 stabile konfigurasjoner og for høyere verdier oppfører systemet seg **kaotisk.**

For å se hvor forutsigbart systemet er kan vi har to utgangsverdier for r, r=0.3 og r=0.301 som ikke synes særlig

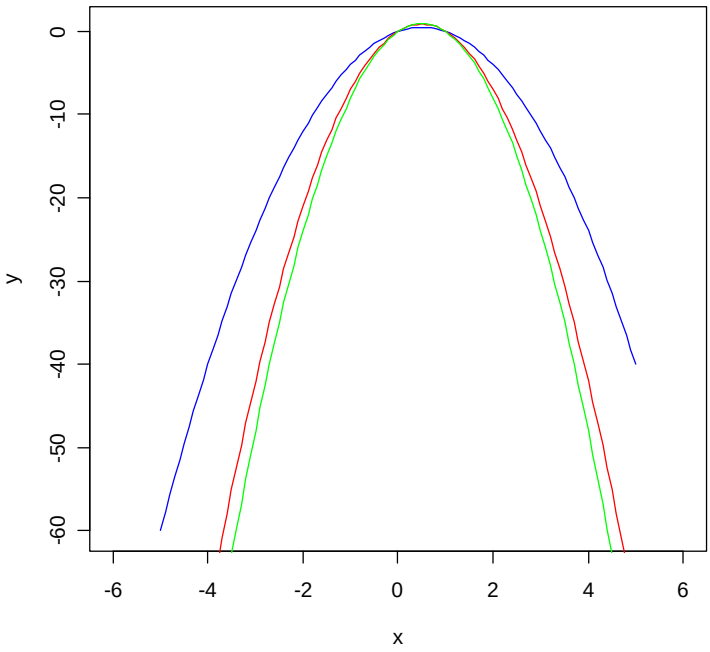
forskjellig, men resultatet blir svært forskjellige når systemet oppfører seg kaotisk:

```
x<-rep(0,100)
r<-4
x[1]=0.3
for (t in 2:100) x[t]<-r*x[t-1]*(1-x[t-1])
par(mfrow=c(1,1),bg="lightyellow")
plot(1:100,x,type="l",col="blue",ylim=c(0,1),xlab="Tid",ylab="
Antall",main="r=4")
x[1]=0.301
for (t in 2:100) x[t]<-r*x[t-1]*(1-x[t-1])
points(1:100,x,type="l",col="red")
```



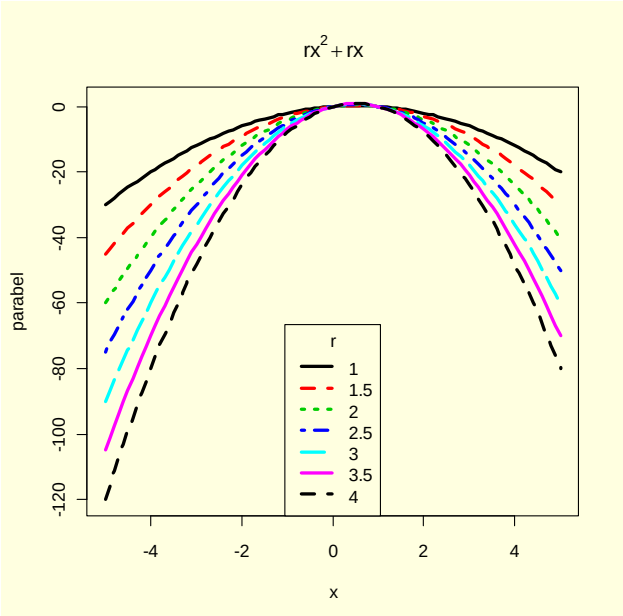
Vi har nå sett hvordan iterasjoner av en kvadratisk ligning oppfører seg.
Ser vi bare på plot av den tilsvarende ligningen $y= -rx^2+rx$ for r=2, 3.5 og 4 blir de parabler:

```
x<-seq(-5,5,0.1)
r<-2
plot(x,-r*x^2+r*x,xlim=c(-
6,6),col="blue",type="l",xlab="x",ylab="y")
r<-3.5
points(x,-r*x^2+r*x,xlim=c(-6,6),col="red",type="l")
r<-4
points(x,-r*x^2+r*x,xlim=c(-6,6),col="green",type="l")
```



Man kan i stedet plotte alle samtidig ved forskjellige verdier av r ved å bruke kommandoene **outer** og **matplot:**

```
x<-seq(-5,5,0.1)
r<-seq(1,4,0.5)
parabel<-outer(x,r, function(x,r)-r*x^2+r*x)
par(bg="lightyellow")
matplot(x,parabel,type="l",lwd=3,main=expression(r*x^2+r*x))
legend("bottom",as.character(r),title="r",lty=1:5,col=1:6,lwd=3)
```



Sannsynlighetsregning og fordelinger

Oversikt over all statistikk:
help(stats)
I pakken stats finnes en rekke statistiske fordelinger: normal-fordeling (**norm**), F-(**f**), binomial- (**binom**),Poisson-(**pois**), students t- (**t**), Beta- (**beta**), kji-kvadrat- (**chisq**), negativ binomial- (**nbinom**), Gamma- (**gamma**),logistisk- (**logis**), Uniform- (**unif**), Wilcoxon rangeringssum- (**wilcox**), eksponensial- (**exp**), Cauchy- (**cauchy**), geometrisk (**geom**),hypergeometrisk (**hyper**),lognormal- (**lnorm**), multinomial- (**multinom**), Weibull- (**weibull**). Andre sannsynlighetsfordelinger finnes i pakken **MCMC** som Dirichlet (**dirichlet**), inverse Gamma (**invgamma**), Wishart (**wish**) og invers Wishart (**iwish**). I pakken **evir** finnes generalisert ekstremverdifordeling (**gev**) og generalisert Pareto (**gpd**).
I tilknytning til til fordelingene er det et prefiks hvor
p er sannsynlighetsfunksjonen
q er kvantiler
d er tetthetsfunksjonen (sannsynligheten for diskrete tilfeldige variable)
r er tilfeldige simulerte verdier

hvor *x* kan være enten bokstaven **p**, **d**, **q** eller **r**: normal-(**xnorm**), poisson-(**xpois**), binomial-(**xbinom**), negativ binomial-(**xnbinom**),kjikvadrat-(**xchisq**), eksponential-(**xexp**), F-(**xf**), logistisk –(**xlogis**), Students t-(**xt**), beta-(**xbeta**), Cauchy-(**xcauchy**), gamma-(**xgamma**), Wilcoxon rangeringssum-(**xwilcox**), Wilcoxon signrangering- (**xsignrank**), Weibull-(**xweibull**),lognormal – (**xlnorm**), uniform – (**xunif**), geometrisk –(**xgeom**), og hypergeometrisk- (**xhyper**).

I tilknytning til disse er sannsynlighetsfordeling med forbokstav ***p***, *e.g.* **pnorm**, tetthetsfunksjonen ***d*** *e.g.* **dnorm**, kvantiler ***q*** *e.g.* **qnorm**, og randomiserte tall ***r*** *e.g.* **rnorm**. Analogt for de andre fordelingene nevnt ovenfor. Vil du vite mer om sannsynlighetsfordelingen e.g. om normalfordelingen dnorm skriv
?dnorm
Informasjon om en uniform fordeling fra intervallet min til max angis med kommandoene **runif()**, **dunif()**, **punif()**, og **qunif()**.
En sannsynlighetsfunksjon (diskret eller kontinuerlig) er en ikke-negativ funksjon som har areal lik 1 integrert over hele domene.
Eksempler på **diskrete sannsynlighetsfordelinger** er Bernoulli-, Binomial-, Negativ binomial-, Multinomial-, Geometrisk-, Hypergeometrisk-, og Poissonfordeling.

Kontinuerlige sannsynlighetsfordelinger er: Normal-, Uniform-, Eksponential-, Gamma-, Log-normal-, Beta-, og Kjikvadrat-fordeling.

Sannsynlighettetthetskurver *f(x)=P(x)* beskriver sannsynligheten til fordelingen av en kontinuerlig stokastisk variabel. Sannsynligheten for at en variabel eller hendelse skal komme mellom *a* og *b* er lik integralet under kurven:

$$P(a < x < b) = \int_a^b f(x)\,dx$$

Det totale arealet under en sannsynlighetstetthetskurve fra minimum til maksimum er lik 1:

$$\int_{-\infty}^{+\infty} f(x)\,dx = 1$$

Forventet verdi (gjennomsnitt, middelverdi, senteret) for en **kontinuerlig stokastisk variabel** er lik integrering av hvert utkomme x ganger sannsynlighetstetthetfunksjonen *f(x)*:

$$E(X) = \mu = \int_{-\infty}^{+\infty} x \cdot f(x)dx$$

For en diskret tilfeldig variabel blir forventet verdi:

$$E(X) = \sum_{i=1}^{n} x_i \cdot f(x_i)$$

Variansen til en til tilfeldig variabel X med sannsynlighetsfordeling f(x) er lik:

$$Var(X) = E[(X - E(X))^2] = E[(X - \mu)^2] = E(X^2) - [E(X)]^2$$

Variansen (spredningen) til en kontinuerlig stokastisk (tilfeldig) variabel er lik integralet av kvadrert avstand fra middeltallet ganger *f(x)*:

$$Var(X) = E(X - \mu)^2 = E(X)^2 - \mu^2 = \int_{-\infty}^{+\infty} (x - \mu)^2 \cdot f(x)dx = \left(\int_{-\infty}^{+\infty} x^2 f(x)dx \right) - \mu^2$$

Gjennomsnitts- eller middelverdien er et estimat av forventningen hvor n er lik prøvestørrelsen:

$$E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variansen blir:

$$Var(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - E(x_i)^2$$

Standardavviket *SD(X)* er lik kvadratroten til variansen:

$$SD(X) = \sqrt{Var(X)}$$

Den sanne verdien for standardavviket som vi lager et estimat av kalles σ (sigma), den sanne verdien for gjennomsnittet kalles μ (mu).

Noen regneregler for forventet verdi(gjennomsnitt, middelverdi) og varianse, hvor *X* og *Y* er to variable og *c* er en konstant. Formlene gjelder for diskrete (diskontinuerlige) sannsynlighetsfordelinger, mens for kontinuerlige gjelder tilsvarende integraler:

$E(X+Y) = E(X) + E(Y)$	Forventning av sum
$E(X \cdot Y) = E(X) \cdot E(Y)$	Forventning av produkt
$Var(X+Y) = Var(X) + Var(Y)$	X og Y er uavhengig variable
$Var(X+Y) = Var(X) + Var(Y) + 2Cov(XY)$	X og Y er kovariable
$Var(c \cdot X) = c^2 \cdot Var(X)$	c er en konstant
$E(c \cdot X) = c \cdot E(X)$	c er en konstant
$Var(c) = 0$	c er en konstant
$Cov(X \cdot Y) = E(X \cdot Y) - E(X) \cdot E(Y)$	Kovarianse mellom X og Y
$E(c) = c$	c er en konstant

Statistisk inferens

Statistisk inferens er en implikasjon, en logisk konsekvens eller konklusjon. Sannsynlighet skyldes usikkerhet pga. variasjon. Blaise Pascal (1623-1662) og Pierre Fermat (1601-1665) regnes som grunnleggerne av sannsynlighetsregningen. Ved myntkast er det to **mulige utfall**, kron eller mynt. Vi kan kalle kron for **gunstig utfall**. Sannsynligheten *P* for et bestemt utfall (hendelse eller event) er forholdet mellom gunstige utfall dividert på antall mulige utfall.

$$P = \frac{gunstig\ utfall}{mulige\ utfall}$$

Antall gunstige utfall kan aldri bli flere enn antall mulige utfall som betyr at 0 ≤ P ≤ 1. Sannsynligheten har grenseverdier 0 og 1, og 0 betyr at hendelsen aldri skjer, og 1 betyr at hendelsen alltid skjer.
Summen av alle sannsynlighetene for et utkomme i et prøverom =1.0
Ved kast av en mynt er sannsynlighet for kron 1/2=0.5 det vil si 50%, og mynt=1/2=0.5, det vil si 50%. Vi kan kalle samlingen av alle mulige utfall for **prøverommet** *S*:

$$S = \{(Kron), (Mynt)\}$$

Har man fått mynt i et kast øker ikke dette sannsynligheten for å få kron i neste kast.

Summen av alle sannsynlighetene for et utkomme i et prøverom =1. Hvis vi har to utfall så kalles sannsynligheten for det ene utfall *p* og sannsynligheten for det andre utfall *q*, og *p*+*q* = 1

Fenomener som terningkast og myntkast er **stokastiske** (tilfeldige) i motsetning til **deterministiske** fenomener hvor vi kjenner til resultatet på forhånd e.g. flo-fjære, soloppgang-solnedgang.

Hvis vi kaster en terning er det seks mulige utfall og **prøverommet** *S* blir:

$$S = \{(1), (2), (3), (4), (5), (6)\}$$

Sannsynligheten for å få en sekser ved terningkast er:1/6

I en kortstokk er det 52 mulige utfall og prøverommet blir:

$$S = \{(13\ kløver, 13\ ruter, 13\ hjerter, 13\ spar)\}$$

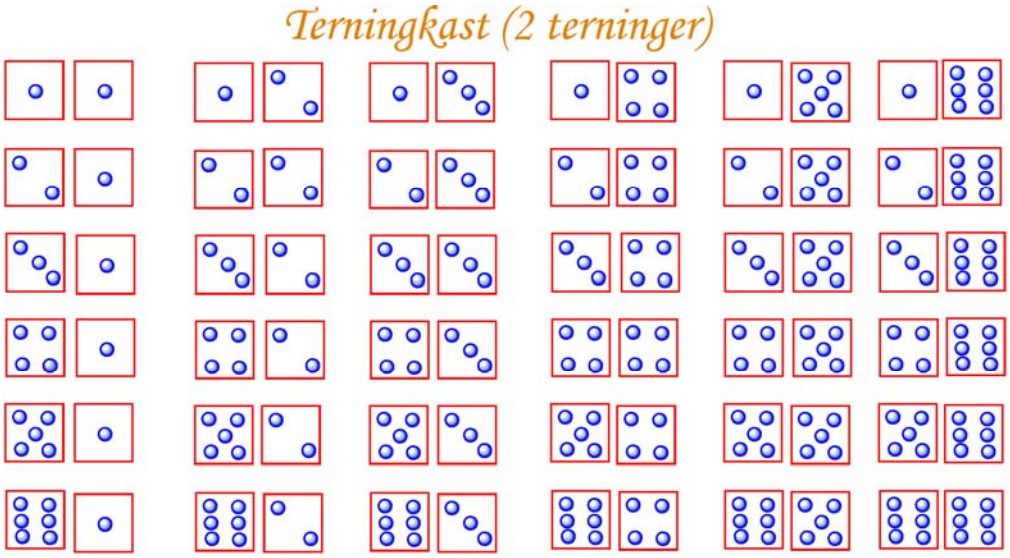
Hva er sannsynligheten for å trekke 4 ess ut av en kortstokk ? Det vil si at det er 4 gunstige utfall og 52 mulige utfall og sannsynligheten blir derfor 1/52+1/52+1/52+1/52= 4/52= 1/13 som tilsvarer ca. 7.7%

Sannsynligheten for komplekse hendelser er summen av sannsynlighetene for enkelthendelser

Vi skal nå se på mer komplekse hendelser som består av en sum av enkelthendelser i et prøverom. Delte hendelser vil si at flere enkelthendelser skjer samtidig, som hvis vi går fra en mynt nevnt foran til to mynter. Kaster du to mynter så blir sannsynligheten for 2 kron lik 1/4 =0.25 det vil si 25%, sannsynligheten for 2 mynt blir 1/4 er lik 25% , mens sannsynligheten for en kron og en mynt blir 2/4=1/2 det samme som 50%. Grunnen er at det er 4 mulige utfall: kron-kron, mynt-kron, kron-mynt og mynt-mynt. Vi forutsetter at utfallet fra den ene mynten er **uavhengig** av utfallet av den andre mynten.

Analogt gjelder for fordeling av alleler *A* og *a* i Hardy-Weinbergs lov hvor $P(AA)=p^2$, $P(Aa)=2pq$ og $P(aa)=q^2$ hvor $p+q=1$

Det samme skjer hvis vi har to terninger og skal beregne sannsynligheten for å få seks.



To terninger betyr 36 mulige utfall, men det er bare 5 gunstige utfall (2+4, 4+2, 1+5, 5+1, 3+3). Sannsynligheten for en sekser ved å kaste to terninger blir derfor 5/36 som tilsvarer ca. 14%.

På tippekupongen er det 12 kamper, hver har tre mulige utfall: hjemmeseier (H), uavgjort (U) og borteseier (B). Det betyr at det er 3^{12} = 531441 mulige tipperekker. Bare en rekke er gunstig utfall (12 rette) og sannsynlighten for å få dette blir $1/3^{12} = 1.88 \cdot 10^{-6}$ som tilsvarer 0.00000188%. Imidlertid er sannsynligheten for å få 12 rette i tipping høyere enn dette for dem som har kunnskaper om fotballagene og som vet hvor dyktige lagene er til å spille fotball. Dessuten er det lettere å vinne på hjemmebane enn bortebane, slik at dette blir en Bayesiansk analyse. Sannsynligheten for 11 rette har 24 mulige utfall $24/3^{12}$ og sannsynligheten for 10 rette har 264 mulige utfall $264/3^{12}$, forutsatt at man ikke har noen forhåndskunnskaper om fotballagene.

Den tyske matematikeren G. Cantor (1845-1918) etablerte teorien for **mengdelære**. En **mengde** er en samling av bestemte atskilte **objekter** (elementer) som kan betraktes som et hele. Hvis objektet x er medlem av mengden A skrives dette:

$$x \epsilon A$$

Mengden omgis av klammeparenteser { }. En tom mengde (**nullmengden**) har symbolet:

$$\emptyset$$

Hvis vi har to mengder A og B så vil **A union B** ($A \cup B$) være mengden av alle x som er medlem av mengden A eller medlem av mengden B:

$$A \cup B = \{x | x \epsilon A \text{ eller } x \epsilon B\}$$
$$\{1,2,3\} \cup \{1,3,4\} = \{1,2,3,4\}$$

A snitt B ($A \cap B$) vil si alle x som er medlem av både mengden A og mengden B

$$A \cap B = \{x | x \epsilon A \text{ og } x \epsilon B\}$$
$$\{1,2,3\} \cap \{1,3,4\} = \{1,3\}$$

Denne sammenhengen kan vises i et **Venn-diagram** (John Venn 1834-1923).

```
A<-c(1,2,3)
B<-c(1,3,4)
union(A,B)
[1] 1 2 3 4
intersect(A,B)
[1] 1 3
```

Hvis sannsynligheten for to hendelser *P(A)* og *P(B)*er **uavhengig** av hverandre er sannsynligheten for at begge hendelser inntreffer *P(A∩B)* (A snitt B) lik produktet av de individuelle hendelsene. Snittet betyr at begge hendelser skjer samtidig.

$$P(A \cap B = P(A) \cdot P(B)$$

Sannsynligheten for hendelser som skjedde samtidig var summen av hendelsene, union, men hvis det er overlapp mellom hendelsene må vi trekke fra felles hendelser slik at de ikke blir telt to ganger:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Hvis det ikke er overlappende hendelser :

$$P(A \cup B) = P(A) + P(B)\ hvor\ A \cap B = \{\emptyset\}$$

I mange tilfeller har vi erfaringsmessig (empirisk) forhåndskunnskap (*a priori*) om den ene av sannsynlighetene. Sannsynligheten for A (P(A)) gitt sannsynligheten *P(B)* skrives som *P(A|B)*. Sannsynligheten for A gitt B blir:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Som vi kan regne om til:

$$P(A \cap B) = P(A|B) \cdot P(B)$$

Statistikk hvor man allerede på forhånd før eksperimentet utføres har forhåndskunnskap om sannsynligheter danner grunnlaget for **Bayesiansk statistikk** oppkalt etter Thomas Bayes (1702-1761), publisert etter hans død: *Essay towards solving a problem in the doctrine of chances* (Phil.Trans 53(1763)370-418)

En generell form av **Bayes teorem** er:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Sannsynligheten for at *A* skal skje gitt *B* (*P(A|B)*) er lik sannsynligheten for at både *A* og *B* skal skje (*P(A∩B)*) dividert på sannsynligheten for at *B* skal skje (*P(B)*).

Bayes teorem blir hvis vi erstatter A med Hypotese og B med data:

$$P(Hypotese|data) = \frac{P(Hypotese) \cdot P(data|Hypotese)}{P(data)}$$

P(Hypotese) kalles **prior sannsynlighetsfordeling** til hypotesen, sannsynlighet før eksperimentet utføres. *P(Hypotese|data)* kalles **posterior sannsynlighetsfordeling** for å få et parameterestimat som hypotesen tilsier, gitt data vi har fått.

P(data|Hypotese) kalles "**likelihood**" til data, sannsynligheten for observerte data gitt hypotesen
P(data) er en normaliseringskonstant (skaleringskonstant) som viser sannsynligheten for å få data gitt alle mulige hypoteser.

$$Posterior = Prior \cdot likelihood$$

Ved bruk av tilgjengelig informasjon skaffer man seg en prior. Deretter innsamles nye data og man bestemmer posterior sannsynlighetsfordeling. Deretter oppdateres prior, man skaffer nye data osv.
Ved Bayesiansk analyse ønsker vi å finne sannsynlighetsfordelingen begrenset av 0 og 1 for å få en kron i gjentatte myntkast, altså ikke den eksakte sannsynligheten som vi har funnet tidligere. Denne forhåndssannsynlighetsfordelingen (prior) kan være en uniform fordeling mellom 0 og 1. Hvis vi nå gjør noen forhåndsforsøk e.g. kaster mynten 10 ganger og får 4 kron og 6 mynt så vil vi ha sannsynlighetsfordelingen (likelihood)
L= P^4*(1-p)^6
Vi kan bruke betafordelingen til å modellere en fordeling mellom 0 og 1

Hvis vi har 3 uavhengige hendelser A, B og C så har vi

$$P(A,B,C) = P(A) * P(B) * P(C)$$

Diskrete sannsynlighetsfordelinger beskriver systemer hvor hvor det er tilfeldige variable som har diskrete utfall. Diskrete betyr at utfallene er heltall fra 0 til *n*. Den enkleste diskrete sannsynlighetsfordelingen er Bernoulli prøven med eksperimenter med to dikotome utfall, oppkalt etter matematikeren Jacob Bernoulli (1654-1705). Hans arbeider ble utgitt etter hans død i *Ars conjectandi*. Han hadde også en matematikkbror, Johann Bernoulli. To utfall av Bernoulli prøven, binære responsvariable, kan *e.g.* være riktig-galt, død-levende, svart-hvit, kron-mynt, gutt-jente eller på-av. Responsen har bare to verdier: det er 1 med sannsynlighet *p*, ofte kalt suksess, og den andre utfallet er 0 med sannsynlighet *1-p=q*, ofte kalt feil. Vi skal nå se på flere diskrete sannsynlighetsfordelinger: binomial fordeling, multinomial fordeling, hypergeometrisk fordeling, geometrisk fordeling, negativ binomial fordeling og Poissonfordeling.

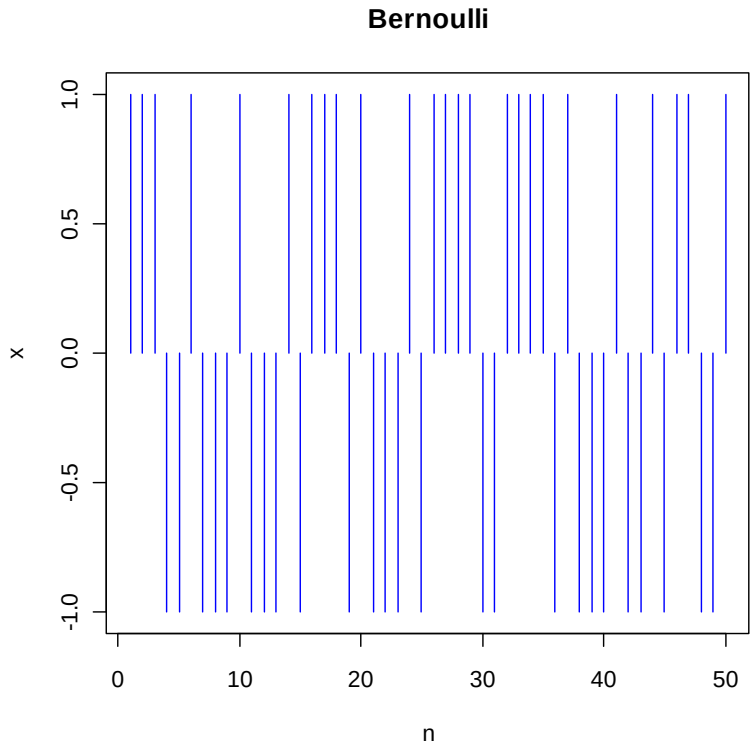
Bernoullifordeling

Tetthetsfunksjonen for Bernoullifordelingen er:

$$p(x) = p^x \cdot (1-p)^{1-x}$$

Variansen for en Bernoullifordeling er lik *pq*. **Bernoulli-fordelingen** har fått navn etter matematikeren og astronomen Jacob Bernoulli (1654-1705) som skrev *Ars conjectandi* (1713), samt en bok om sannsynlighetsregning. Han hadde også en kjent bror Johann som også var matematiker. Bernoullifordelingen har to utkomme: kron-mynt, galt-riktig, tilstede-fraværende, reproduksjon-ikke reproduksjon osv. De to sannsynlighetene blir *p* for *x(1)* og *1-p* for *x(0)*. Vi kan simulere Bernoulli ved bruke kommandoen **sample**, og trekke n=50 ut tilfeldige prøver *x* av 1 og -1, med tilbakeplassering. Figuren endrer seg fra gang til gang man kjører scriptet:

```
n <- 50
x <- sample(c(-1,1), n, replace=T)
plot(x, type="h",col="blue", xlab="n", main="Bernoulli")
```



Vi kan også generere 20 tilfeldige 0 og 1 med sannsynlighet p=0.5

```
bernoulli<-rbinom(20,size=1,p=0.5);bernoulli
 [1] 0 0 0 0 1 0 1 1 1 0 0 1 1 1 0 1 1 0 1 0
```

Mange Bernoulli-forsøk gir en **binomial fordeling.**

Binomial fordeling

Binomial fordeling består av en serie med uavhengige forsøk som har to mulige utkomme (Bernoulli-eksperimenter). Sannsynligheten *P(x)* for å få *x* av en sort (suksess) og *n-x* av en annen sort (ikke-suksess) er:

$$P(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

som er den generelle formen av den binomiale fordelingen, hvor p^x er sannsynligheten for å få x uavhengige suksess, hver med sannsynlighet p; og $(1-p)^{n-x}$ er sannsynligheten for $n-x$ ikke-suksess, hver med sannsynlighet $1-p$. Summen av suksess og ikke-suksess blir lik n som er lik antall Bernoulliforsøk.

$$\binom{n}{x}$$

kalles **binomialkoeffisienter**.
Forventet verdi for binomialfordelingen:

$$E(X) = n \cdot p$$

Variansen for binomialfordelingen:

$$Var(X) = n \cdot p(1-p)$$

Hvis vi en undersøkelse med to mulige utfall ikke kjenner p så kan vi lage et estimat av p ved p ≈ E(X)/n

Variansen blir alltid mindre enn middelverdien.

Antall måter å kombinere x objekter valgt ut fra n forskjellige objekter (binomialkoffisientene n over x inne i en stor parentes) blir ifølge kombinasjonsregelen:

$$\binom{n}{x} = \frac{n!}{x! \cdot (n-x)!}$$

n! kalles *n fakultet* og beregnes som n·(n-1)·(n-2)··3·2·1.
Binomialkoeffisientene kan plasseres i en **Pascals trekant** hvor tallene i neste horisontale linje er summen av de ovenfor og havner midt mellom dem:

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

$$\binom{0}{0}$$
$$\binom{1}{0}\binom{1}{1}$$
$$\binom{2}{0}\binom{2}{1}\binom{2}{2}$$
$$\binom{3}{0}\binom{3}{1}\binom{3}{2}\binom{3}{3}$$
$$\binom{4}{0}\binom{4}{1}\binom{4}{2}\binom{4}{3}\binom{4}{4}$$

og slik kan man fortsette nedover. Vi ser at trekanten er symmetrisk om midtlinjen og

$$\binom{n}{x} = \binom{n}{n-x}$$

Summen av to koeffisienter som står ved siden av hverandre i en horisontal rekke havner midt mellom dem i raden under:

$$\binom{n}{x}+\binom{n}{x+1}=\binom{n+1}{x+1}$$

Vi ser også at summerer vi tallene som står i de horisontale linjene får vi tallene 1, 2, 4, 16, 64 osv.
Det vil si:

$$\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\binom{n}{3}+\cdots+\binom{n}{n}=2^n$$

Vi har en boks med 6 kuler nummerert fra 1-6. På hvor mange måter kan man trekke ut 3 kuler fra de 6 ? Svar: 20

```
 choose(6,3)
[1] 20
```

$$\binom{n}{x}=\binom{6}{3}=\frac{n!}{x!\cdot(n-x)!}=\frac{6!}{3!\cdot(6-3)!}=\frac{6\cdot5\cdot4\cdot(3\cdot2\cdot1)}{3\cdot2\cdot1\cdot(3\cdot2\cdot1)}=\frac{6\cdot5\cdot4}{3\cdot2\cdot1}=20$$

Myntkast er eksempel på binomial fordeling med to mulige utfall, kron-mynt, hver med sannsynlighet 0.5. Sannsynligheten for å få 5 kron ved å kaste en mynt 5 ganger er:

Alle utfall har samme sannsynlighet 0.5 (kron), 0.5 (mynt) 5 ganger:
Alle mulige kombinasjoner av de 5 myntkastene:

```
 choose(5,0:5)
[1]  1  5 10 10  5  1
```

De tilsvarende sannsynlighetene:

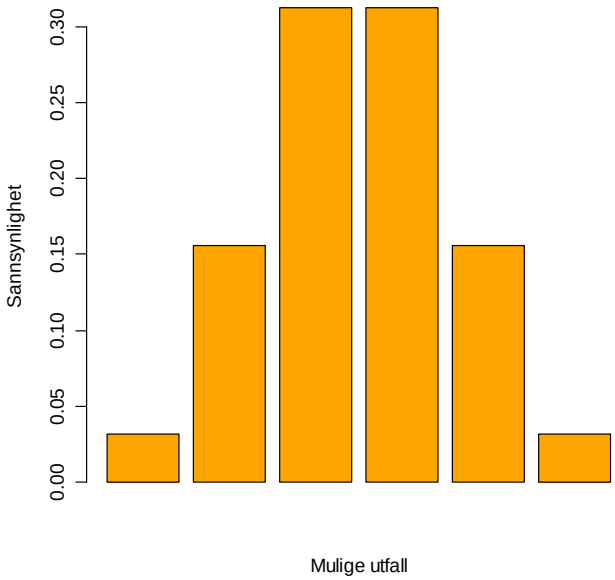
```
 choose(5,0:5)*0.5^5
[1] 0.03125 0.15625 0.31250 0.31250 0.15625 0.03125
```

Sannsynligheten for 5 mynt eller 5 kron er ca. 3%.
Sannsynligheten for 3 kron og 2 mynt er ca. 31% o.s.v.

Summen av alle sannsynlighetene er lik 1:

```
 sum(choose(5,0:5)*0.5^5)
[1] 1
```

```
 barplot(choose(5,0:5)*0.5^5, xlab="Mulige utfall",
ylab="Sannsynlighet",col="orange")
```



Hvis det er 12 personer i et selskap og alle skal hilse på alle blir det 66 håndtrykk:

```
 choose(12,2)
[1] 66
```

For eksempel Det er 52 forskjellige måter å dele ut kortene i en kortstokk: 52! (52 faktultet). Kortene deles ut på 4 spillere med 13 kort hver. Hver hånd kan deles ut på 13! forskjellige måter. Det er 4 hender slik at antall mulige kombinasjoner blir $(13!)^4$. Totalt antall mulige bridgehender blir:

```
 factorial<-function(x)max(cumprod(1:x))
 factorial(52)/(factorial(13)^4)
[1] 5.364474e+28
```

I Lotto skal man velge ut 7 tall fra 1-34. Hvor mange måter er det å kombinere de 34 tallene ?

```
 choose(34,7)
[1] 5379616
```

Sannsynligheten for å vinne i Lotto:

```
 1/choose(34,7)
[1] 1.858869e-07
```

Hva er sannsynligheten for å få 4 kron og 5 mynt i 9 myntkast ?

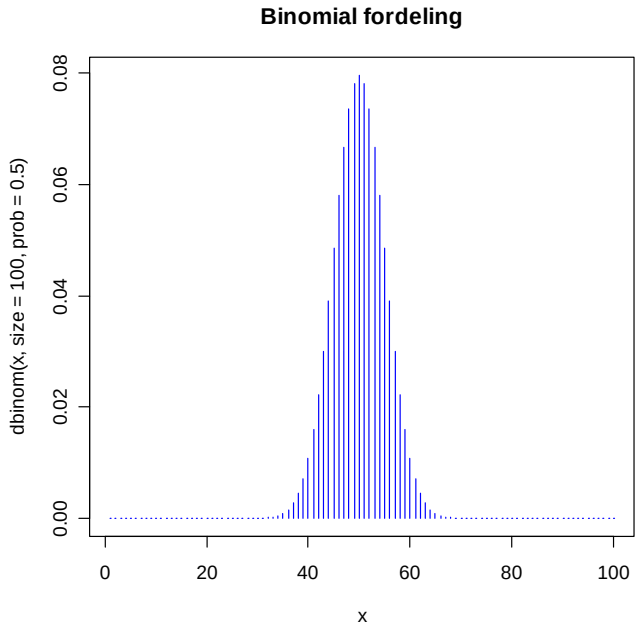
$$Pr = \binom{n}{x} \cdot p^x \cdot (1-p)^{n-x} = \frac{n!}{x! \cdot (n-x)!} \cdot p^x \cdot (1-p)^{n-x} = \frac{9!}{4! \cdot 5!} \cdot 0.5^4 \cdot 0.5^5$$
$$= 0.2460938$$

```
factorial(9)/(factorial(4)*factorial(5))*0.5^4*0.5^5
```

```
[1] 0.2460938
```
Kombinasjoner av 9 myntkast:
```
 choose(9,0:9)
 [1]   1   9  36  84 126 126  84  36   9   1
```
Og de tilsvarende sannsynligheter:
```
 choose(9,0:9)*0.5^9
 [1] 0.001953125 0.017578125 0.070312500 0.164062500 0.246093750 0.246093750
 [7] 0.164062500 0.070312500 0.017578125 0.001953125
```
Sannsynlighet for 9 kron (0.19%), 9 kron + 1 mynt (1.76%)osv.

Kommandoen **prod** beregner produktet av en vektor med tall.
En tilfeldig prøve med prøvestørrelse *n* fra en diskret fordeling kan lages med kommandoen **sample()**. Hvis prøven lages ved å putte tilbake tallet som blir tatt ut angir vi dette med **replace=TRUE.**
For eksempel kaste en terning 20 ganger:
```
 sample(1:6,size=20,replace=TRUE)
 [1] 2 2 6 2 1 3 1 2 3 6 2 2 1 2 1 3 5 5 5 1
```
Eller kaste en mynt 20 ganger med utfall mynt/kron:
```
 sample(0:1,size=20,replace=TRUE)
 [1] 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1
```
Eller:
```
 sample(c("K","M"),20,replace=T)
 [1] "M" "K" "M" "M" "M" "K" "M" "K" "M" "K" "M" "K" "K" "M" "K" "K" "K" "K" "M"
[20] "M"
```

En binomial fordeling har diskrete verdier og en graf av binomial fordeling med n=100 og p=0.5 blir
```
 x<-seq(1,100,1)
plot(x,dbinom(x,size=100,prob=0.5),type="h",col="blue",main="B
inomial fordeling")
```



R kan generere tilfeldige tall fra en angitt type fordeling.
For eksempel binomial fordeling

```
rbinom(20,size=1,p=0.5)
 [1] 1 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0
```

De tilfeldige tallene som blir generert blir forskjellige hver gang, avhengig av et starttall som kalles "frø". Hvis man ønsker å bruke samme starttall som "frø" for flere påfølgende generering av slumptall kan man bruke kommandoen **set.seed()**. Her fra en Poisson-fordeling:

```
set.seed(7568)
rpois(30,5)
 [1] 5 7 9 7 7 4 2 6 4 5 7 1 6 7 2 7 3 6 9 7 6 2 3 5 3 3 3 4 6 2
```

En annen måte er å trekke ut tall fra en angitt populasjon av tall og bruke **replace=FALSE** for å forhindre at samme tall blir trukket ut mer enn en gang. For eksempel trekke ut 10 tall fra 1-3000

```
sample(1:3000,10,replace=FALSE)
 [1] 2173  430 1169 2241  504  715 1261 1585 2637  437
```

En binomial fordeling med størrelse n og sannsynlighet p for suksess i hver prøve har sannsynlighetstetthet:

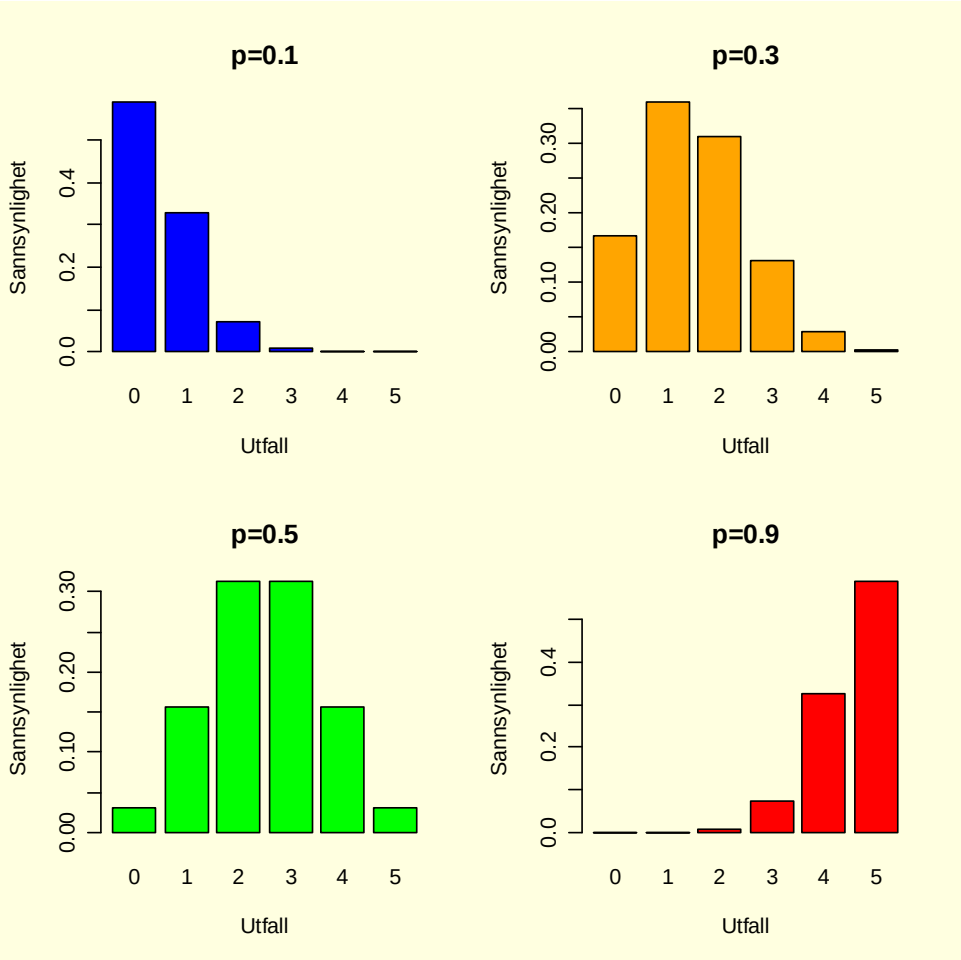
$$p(x) = choose(n,x)p\text{^}x(1-p)\text{^}(n-x)$$

Se R-manualen:

```
?dbinom
```

Sannsynligheten for et spesielt utfall med forskjellige sannsynligheter p for suksess og 5 forsøk (n=5):

```
n<-5
p<-0.1
x<-0:n
px<-choose(n,x)*p^x*(1-p)^(n-x)
barplot(px,col="blue",names=as.character(0:5),xlab="Utfall",yl
ab="Sannsynlighet",main="p=0.1")
```

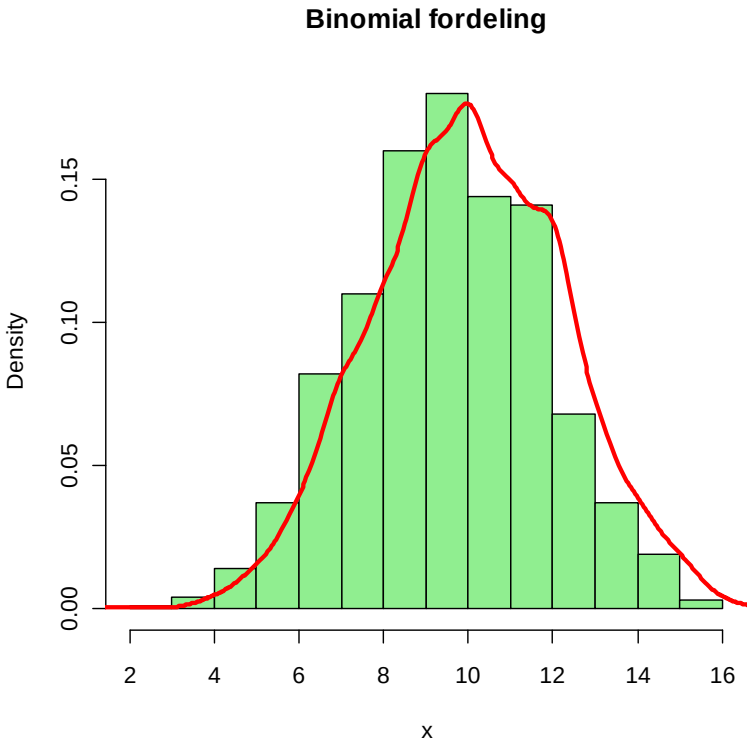


Binomial fordeling. Hver stolpe angir sannsynligheten for et bestemt antall utfall. Når p=0.5 blir den binomiale fordelingen symmetrisk. Når *p* nærmer seg 0 eller 1 får fordelingen skew.

Utplukking av 1000 tall fra 0-20 med sannsynlighet p=0.5

```
x <- rbinom(1000,size=20,p=0.5)
hist(x,probability = TRUE,col = "lightgreen",main = "Binomial
fordeling")
```

```
lines(density(x), col="red",lwd = 3)
```



Hvis det i en klasse med 30 elever som følger binomaial fordeling er 12 jenter og 18 gutter kan vi si at det statistisk er flere jenter enn gutter i klassen ? Vi tester nullhypotesen om p=0.5. Vi kan ikke si at det er flere jenter enn gutter i klassen

```
 binom.test(12,30,0.5)

        Exact binomial test

data:  12 and 30
number of successes = 12, number of trials = 30, p-value = 0.3616
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.2265576 0.5939651
sample estimates:
probability of success
                   0.4
```

Ifølge tabll 61 Folkemengd, etter kjønn og alder (http://www.wwb.no/aarbok/tab/tab-061.html) var i 2008 forholdet mellom gutter og jenter i årsklassen 0-9 år henholdsvis 303016 gutter og 289748 jenter. Ifølge Fisher skulle vi forvente et kjønnsratio 50%. Vi forsøker først en binomial test:

```
 303016+289748
[1] 592764
 binom.test(289748,592764,0.5)
```

```
        Exact binomial test
data:  289748 and 592764
number of successes = 289748, number of trials = 592764, p-value <
2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.4875350 0.4900818
sample estimates:
probability of success
             0.4888084
```

Vi forkaster nullhypotesen om sex-forhold 50:50
Vi kan også bruke GLM til å teste dette og lager en nullmodell:

```
barn<-c(289748, 303016)
model<-glm(barn~1,poisson)
summary(model)
Call:
glm(formula = barn ~ 1, family = poisson)
Deviance Residuals:
     1       2
-12.23   12.14
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 12.599404   0.001299    9700   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)
    Null deviance: 297.01  on 1  degrees of freedom
Residual deviance: 297.01  on 1  degrees of freedom
AIC: 327.88
Number of Fisher Scoring iterations: 3
```

Vi sammenligner residual avviket med kjikvadrattest med 1df:

```
 1-pchisq(297.01,1)
[1] 0
```

Vi forkaster hypotesen om 50:50 forhold gutter og jenter i aldersklassen 0-9 år. Estimatet er exp(12.599404)= 296381.9

Mendel fant i 1865 følgende resultat i F2 generasjon: 6022 grønne erter og 2001 gule erter. Er dette forskjellig fra ¾ grønne erter og ¼ gule erter ? Nei. Nullhypotesen er at grønne erter (suksess) har p=3/4 og nullhypotesen beholdes.

```
 binom.test(c(6022, 2001), p = 3/4)
        Exact binomial test

data:  c(6022, 2001)
number of successes = 6022, number of trials = 8023, p-value = 0.9076
alternative hypothesis: true probability of success is not equal to 0.75
95 percent confidence interval:
 0.7409722 0.7600295
sample estimates:
probability of success
              0.750592
```

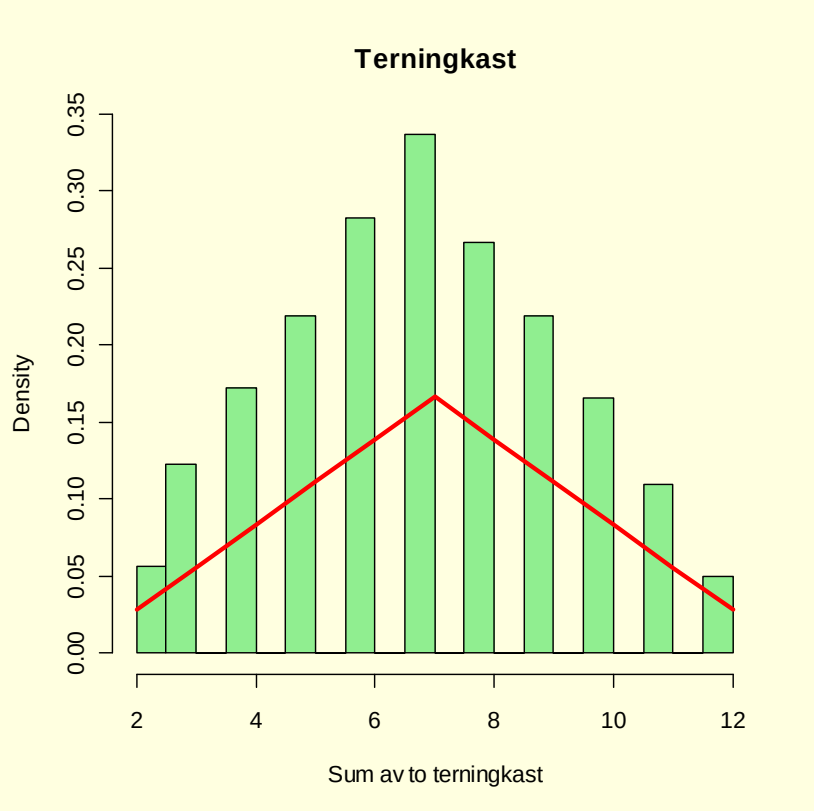
Hvis vi har en familie med 2 barn hva er sannsynligheten for at en av dem er en jente (J)? Vi har følgende mulige utfall: JJ, JG, GJ, GG:

```
jente<-dbinom(0:2,size=2,p=0.5);jente
[1] 0.25 0.50 0.25
```
Det er 75% sannsynlig. 25% av familiene har to jenter, 50% har en gutt og ei jente, og 25% av familiene har to gutter.

Vi kaster to terninger (t1 og t2) og summerer tallene

```
n<-10000
t1<-sample(c(1,2,3,4,5,6),n,replace=T)
t2<-sample(c(1,2,3,4,5,6),n,replace=T)
sum<-t1+t2
par(bg="lightyellow")
hist(sum,probability=T,col="lightgreen",xlab="Sum av to
terningkast",main="Terningkast")
lines(x,PX,col="red",lwd=3)
```

PX kan først tegnes inn etter at den er regnet (se nedenfor)



Det mest sannsynlige utfall er at summen blir lik 7 (1+6,2+5,3+4,4+3,5+2,6+1). Den rød linjen viser de teoretiske sannsynlighetene med en topp ved sum=7, og figuren blir symmetrisk.

Sannsynligheten for et utfall for en terning er 1/6 og for to terninger blir utfallet 1/6*1/6, for eksempel 1+1. For 3 er det to muligheter 1+2 og 2+1 dvs. 1/6*1/6+1/6*1/6osv.
Vi kan finne sannsynlighetene PX for x=2-12,xp er x ganger sannsynligheten.

```
p2<-1/6*1/6
PX<-numeric(11)
```

```
PX[1]<-p2
n<-11
for(i in 2:n){
if(i<7)(PX[i]<-i*p2)
else(PX[i]<-PX[6]-(i-6)*p2)
}
x<-seq(2,12,1)
xp<-x*PX
ex<-sum(xp)
var<-(x-ex)^2*PX
P<-cbind(x,PX,xp,var);P
```

```
       x         PX         xp       var
 [1,]  2 0.02777778 0.05555556 0.6944444
 [2,]  3 0.05555556 0.16666667 0.8888889
 [3,]  4 0.08333333 0.33333333 0.7500000
 [4,]  5 0.11111111 0.55555556 0.4444444
 [5,]  6 0.13888889 0.83333333 0.1388889
 [6,]  7 0.16666667 1.16666667 0.0000000
 [7,]  8 0.13888889 1.11111111 0.1388889
 [8,]  9 0.11111111 1.00000000 0.4444444
 [9,] 10 0.08333333 0.83333333 0.7500000
[10,] 11 0.05555556 0.61111111 0.8888889
[11,] 12 0.02777778 0.33333333 0.6944444
```

Variansen blir:

```
varx<-sum(var);varx
```

```
[1] 5.833333
```

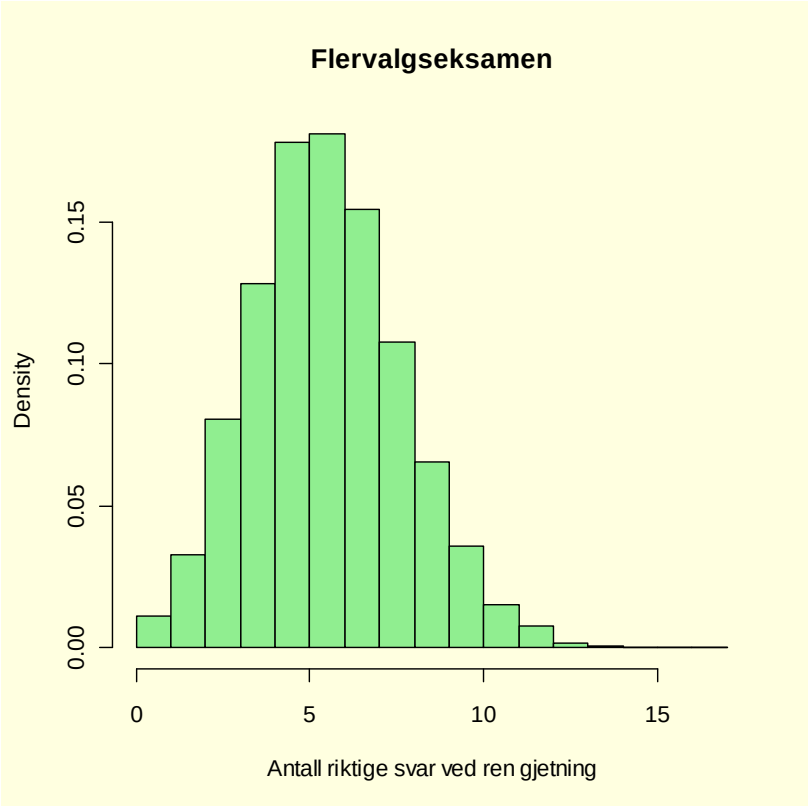
Forventet verdi:

```
 ex
```

```
[1] 7
```

I et kurs med flervalgsoppgaver er det 30 spørsmål, hver med 5 valgmuligheter. Svaret på en oppgave blir en Bernoulli-prøve med sannsynlighet for suksess (riktig svar): 1/5=0.2, og sannsynlighet for ikke-suksess: 4/5=0.8. Vi kan simulere en eksamen ved å ta ut prøver fra en uniform fordeling. Når man ganger med 1 (x<- 1*(svar<.2)) så blir resultatet TRUE (1)eller FALSE (0), og vi kan telle opp antall 1. Vi går deretter opp til eksamen 10000 ganger.

```
fasit<-numeric(10000)
for(i in 1:10000){
svar<-runif(30)
x<- 1*(svar<.2)
riktig<-length(x[x=="1"])
fasit[i]<-riktig
}
par(bg="lightyellow")
hist(fasit,probability=T,col="lightgreen",xlab="Antall riktige
svar ved ren gjetning",main="Flervalgseksamen")
```



Det vil si at det er mulig å få 5-6 riktige svar ved ren gjetning, så dette bør være grensen for stryk.

Maksimum likelihood estimering i binomial fordeling

Maksimum likelihood estimering (MLE) vil si å tilpasse en matematisk modell til data. Vi begynner med å lage en **likelihoodfunksjon L(X|p)** med ukjente modellparametere (p) og vi ønsker å finne parameterverdier som passer best overens med beskrivelsen av data (X) vi allerede har. Legg merke til forskjellen til probabilitet hvor vi kjenner parameterverdiene for et utfall på forhånd, men hvor vi ønsker å forutsi utfallet av fremtidige hendelser: **sannsynlig utfall P(p|X).**
Det er mer praktisk å regne med den naturlige logaritmen til likelihood, og log likelihood blir alltid negativ. Hvis vi har verdier $x=y\cdot z$ blir dette skrevet som logaritmer $\log(x)=\log(y)+\log(z)$.
Eller hvis vi har et ratio $x=y/z$ blir dette $\log(x)=\log(y)-\log(z)$. Av praktiske grunner opererer vi med $-2\cdot$loglikelihood , som følger kji-statistikk og vi kan bruke kjikvadrattest for å teste sannsynlighet. Variansen er lik den inverse av den andrederiverte til $-2\cdot$loglikelihood

Hvis vi har et normalfordelt $N(\mu,\sigma^2)$datasett X $(x_1, x_2,\ldots,x_n)$hvor man kjenner variansen s^2, men ikke μ så ønsker man med MLE å finne hvilken μ som gjør datasettet mest sannsynlig.
Metoden for maksimum likelihood ble innført først av R.A. Fisher. Modellen er $f(x_i|p)$ hvor vi skal se på et eksempel hvor p er sannsynlighet for å få kron i et myntkast.
Likelihoodfunksjonen L er gitt ved:

$$L(x_1, x_2, \ldots x_n|p) = \prod_{i=1}^{n} f(x_i|p)$$

Hvor $\prod$ er produkttegn.
Vi kan simulere 10 myntkast som følger Bernoulli-fordeling:

```
X<-rbinom(10,size=1,p=0.5);X
 [1] 1 1 1 0 1 0 1 1 1 0
```

X={1 1 1 0 1 0 1 1 1 0} hvor 1 står for kron, og verdiene x_1=1 (kron), x_2=1(kron),…x_{10}=0(mynt).
Bernoullifordelingen har formen:

$$f(x_i|p) = p^{x_i} \cdot (1-p)^{(1-x_i)}$$

For myntkast blir likelihoodfunksjonen:

$$L(x_1, x_2, \ldots x_n|p) = \prod_{i=1}^{n} f(x_i|p) = p^{x_1} \cdot (1-p)^{1-x_1} \cdot p^{x_2} \cdot (1-p)^{1-x_2} \cdot\cdot p^{x_n} \cdot (1-p)^{1-x_n}$$

$$= p^{\sum_{i=1}^{n} x_i} \cdot (1-p)^{1-\sum_{i=1}^{n} x_i}$$

Vi tar logaritmen på begge sider av likhetstegnet:

$$\log\left(L(p|x_1, x_2, \ldots x_n)\right) = \log p \cdot \sum_{i=1}^{n} x_i + log(1-p) \cdot \left(n - \sum_{i=1}^{n} x_i\right)$$

Vi får maksimal likelihood når p= antall kron. Vi kan også derivere likelihoodfunksjonen med hensyn på p og vi har maksimal likelihhood når den deriverte er lik 0. Vi ser på formen på loglikelihoodfunksjonen rundt topp-punktet.
Vi beregner likelihood ratio:

$$\Delta = \frac{max[L(x|H_0: p = 0.5)]}{max[L(x|H_A: p \neq 0.5)]}$$

Når vi tar logΔ (husk naturlige logaritmer!) får vi ratio som en differanse i loglikelihood.
Ved kast av en mynt uten feil vil sannsynligheten for kron være p=0.5 og sannsynligheten for mynt er 1-p=0.5. Ved MLE har vi allerede et datasett *e.g.* kast av en mynt 100 ganger som gir 59 kron og 41 mynt og vi ønsker å finne en parameterverdi *p* som best beskriver dette datasettet, L(X|p). Vi kan bruke myntkast som trivielt eksemplet og prinsipp for likelihoodestimering.
For en binomial fordeling blir likelihoodfunksjonen

$$\binom{n}{x} \cdot p^x \cdot (1-p)^{1-x} = \frac{n!}{x! \cdot (n-x)!} \cdot p^x \cdot (1-p)^{1-x}$$

Vi kan la *x* være antall kron, og i vårt eksempel med p=0.5 (teoretisk verdi) får vi likelihood for nullhypotesen H_0: p=0.5

$$\frac{100!}{59! \cdot 41!} \cdot 0.5^{59} \cdot (1 - 0.5)^{41} = 0.01586907$$

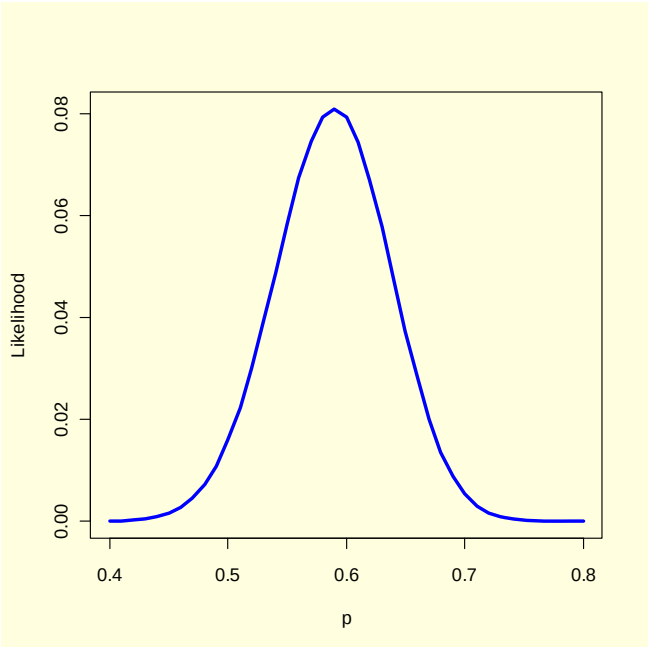
Loglikelihood (LL) for H_0 p=0.5 blir: -4.143383

Vi kan plotte de forskjellige likelihood ved å variere sannsynligheten *p*:

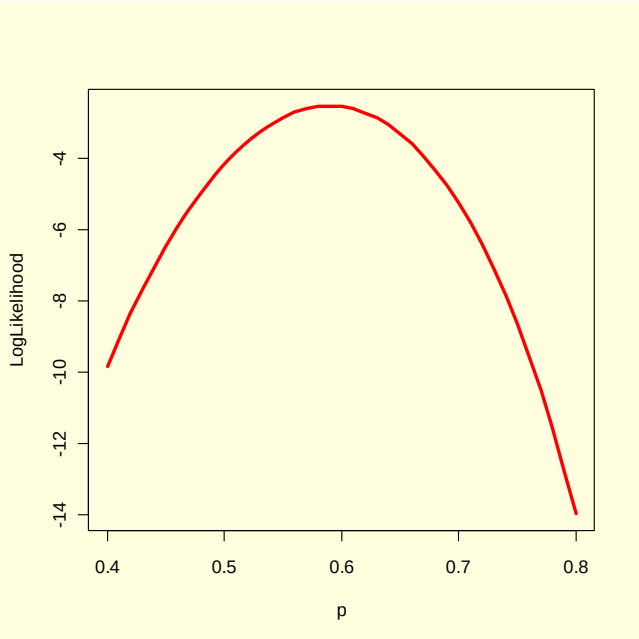
```
n<-100
x<-59
p<-seq(0.4,0.8,0.01)
L<-choose(n,x)*p^x*(1-p)^(n-x)
LL<- log(L);LL
plot(p,L,type="l",col="blue", lwd=3,xlab="p",
ylab="Likelihood")
plot(p,LL,type="l",col="red", lwd=3,xlab="p",
ylab="LogLikelihood")
```

For å beregne L kan også følgende kommando brukes:

```
L<-factorial(n)/(factorial(x)*factorial(n-x))*p^x*(1-p)^(n-
x);L
```



Vi kan også plotte loglikelihood:



Vi kan finne maksimum likelihood:

```
a<-cbind(p,L);a
max(a[,2])
```

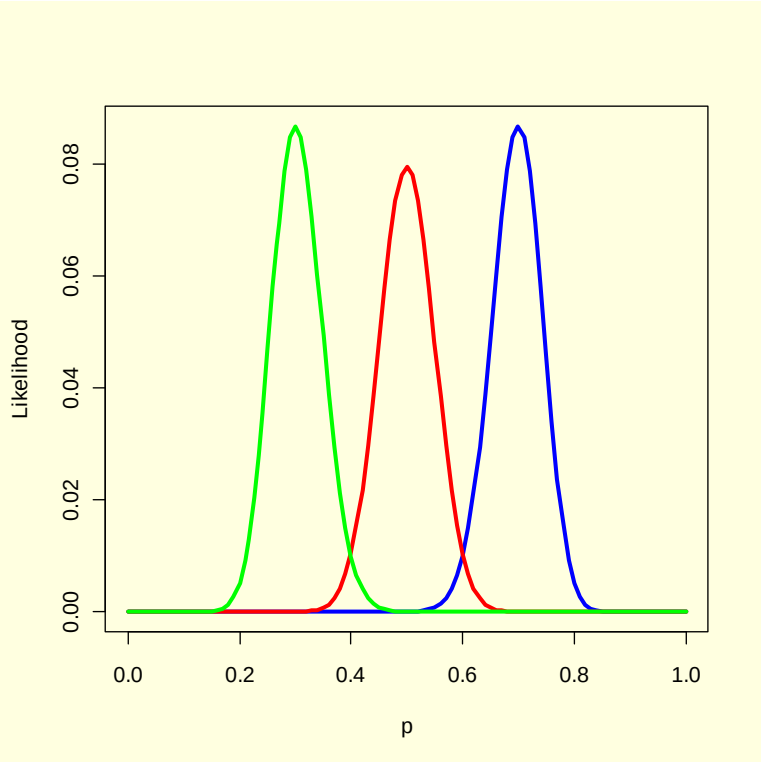
og finner at p=0.59 beskriver datasettet best.
Vår alternative hypotese H_A:p ≠ 0.5
For H_A med p=0.59 får vi likelihood: 0.08090176 og loglikelihood (LL): -2.51452
Vi har da 2*differansen i loglikelihood (2*LL_A-LL_0), kalles også loglikelihoodratio:

```
 2*-2.51452 -(-4.143383)= -0.885657
```

Sien vi har bare en p-verdi for hver hypotese blir df=1 og tabellverdi for kjikvadratet blir:

```
 qchisq(0.95,1)
[1] 3.841459
```

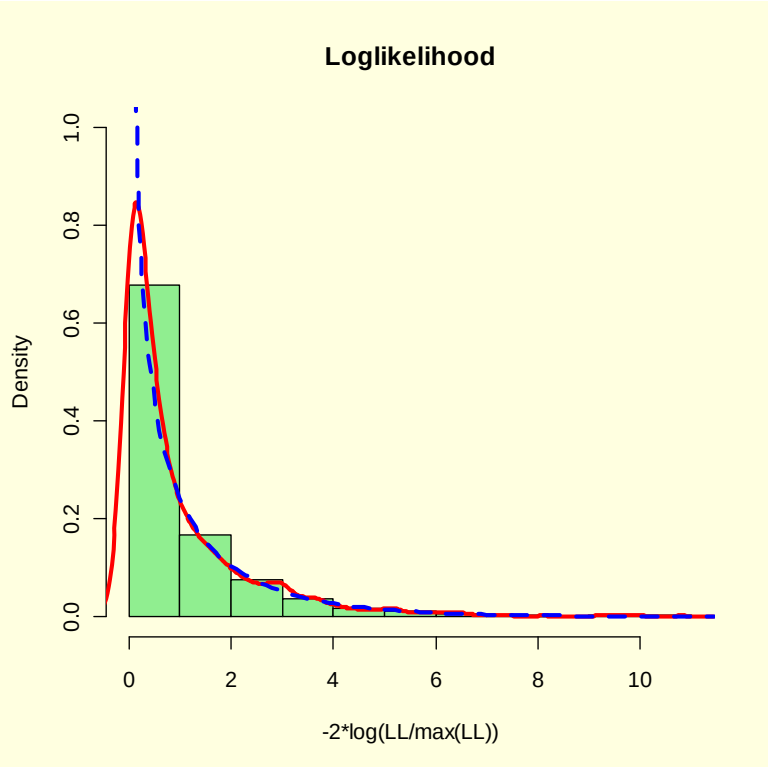
Siden vår test-statistikk (-0.88) er mindre enn tabellverdien beholdes nullhypotesen H_0.



Figur. Likelihood for 100 myntkast, men hvor antall kron er 30 (grønn), 50 (blå) og 70 (blå). Vi ser at likelihood blir lik et estimat av p for antall kron.

Vi simulerer 1000 myntkast, teller opp antall kron (k) og bestemmer likelihood for å få et slikt antall kron.
LL:likelihood, LLR:2*log (likelihoodratio)

```
LL<-numeric(1000)
for(i in 1:1000){
n<-1000
x<-rbinom(n,size=1,p<-0.5)
kron<-x[x=="1"]
k<-length(kron)
p<-0.5
L<-choose(n,k)*p^k*(1-p)^(n-k)
LL[i]<-L
LLR<--2*log(LL/max(LL))
}
par(bg="lightyellow")
hist(LLR,probability=T,col="lightgreen",ylim=c(0,1),xlab="-
2*log(LL/max(LL))",main="Loglikelihood")
lines(density(LLR),col="red",lwd=3)
y<-seq(0,12,0.1)
lines(y,dchisq(y,1),lty=2,col="blue",lwd=3)
```



Figur. Viser sannsynlighettetthet (rød) for -2*log(likelihoodratio) 1000 myntkast, frekvens, samt vi ser at denne følger kjikvadratfordeling med df=1 (stiplet blå linje). Den kritiske verdien for kji-kvadratfordelingen er

```
qchisq(0.95,1)
[1] 3.841459
```

Vi kan se på figuren at ved -2*log likelihood ca. 4 har vi fått med ca. 95% av fordelingen.

Multinomial fordeling

I noen tilfeller kan det være nyttig å ha mulighet for flere enn to utfall. I tilfelle x uavhengige prøver:

$$p(n_i) = \frac{n!}{n_1!\,n_2!\,n_3!\,\ldots n_x!} \cdot {p_1}^{n_1} {p_2}^{n_2} {p_3}^{n_3} \ldots . {p_x}^{n_x}$$

Multinomial fordeling gir mulighet for dette *e.g.* hvis man ønsker å se på alleler etter krysningsforsøk hvor allelkombinasjonene i avkommet er enten *AA*, *Aa* eller *aa*, og vi kan la sannsynlighetene for de 3 utkommene være p_1=0.25, p_2=0.5, og p_3=0.25.

Forventningen i en multinomial fordeling:

$$E(X_i) = n \cdot p_i$$

$$Var(X_i) = n \cdot p_i \cdot (1 - p_i)$$

Geometrisk fordeling

Både den geometriske fordeling og den negative binomiale fordelingen angir hvor lang tid det tar før man får et spesielt utkomme. Denne bygger på en rekke Bernoulliprøver. Sannsynlighetsfordelingen er:

$$p(x) = p \cdot (1-p)^{x-1}$$

Når suksess kommer i *x*te forsøk krever det ikke-suksess i de forgående x-1 tilfellene. $x \geq 1$
Forventet verdi *E(X)*(gjennomsnitt, middelverdi) er:

$$E(x) = \frac{1}{p}$$

Variansen (*var(X)*) til en geometrisk tilfeldig variabel er:

$$Var(X) = \frac{1-p}{p^2}$$

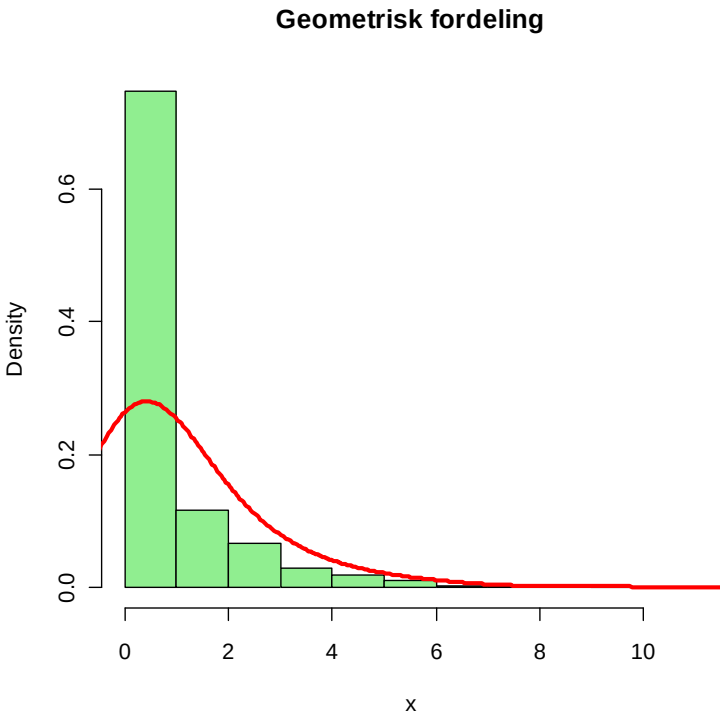
Et eksempel på geometrisk fordeling er antall mislykkede parringer før en vellykket, forutsatt at antall parringer er konstant over tid og det ikke skjer læring fra mislykkede parringer. Et annet er hvor mange år det går før en utrydningstruet art når ekstinksjon.

Geometrisk fordeling angir hvor lang tid det tar før en hendelse skjer i et Bernoulli-forsøk for eksempel hvor lang tid det tar før man får en kron (1)

```
bernoulli<-rbinom(20,size=1,p=0.5);bernoulli
 [1] 0 0 1 1 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 1
```

Det vil si to mynt før gunstig utfall:

```
x <- rgeom(1000,p=0.5)
hist(x, probability=T,col="lightgreen",main="Geometrisk
fordeling")
lines(density(x,bw=1), col='red', lwd=3)
```



Negativ binomial fordeling

Negativ binomial fordeling brukes på telledata hvor variansen er mye større enn middeltallet.
Både geometrisk og negativ binomival fordeling beskriver ventetiden før en spesiell hendelse skjer.
Sannsynligheten *P(x)=f(x)* for en negativ binomial fordeling er en serie Bernoulli-forsøk hvor r suksess på x-te forsøk:

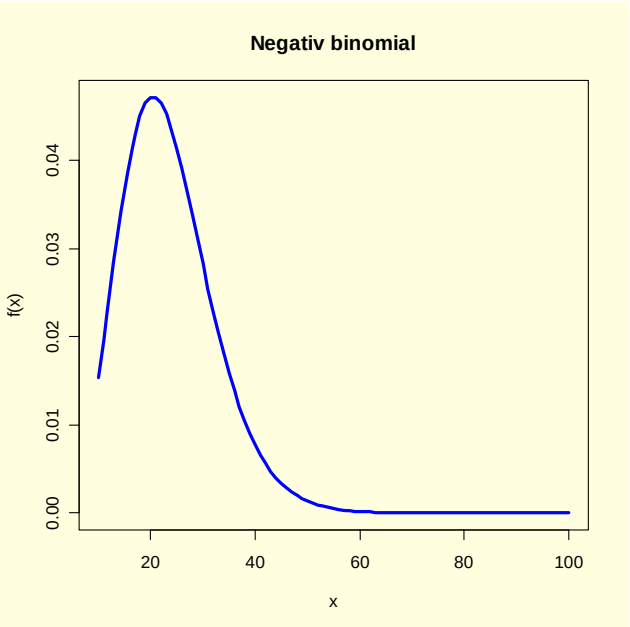
$$f(x) = \binom{x-1}{r-1} \cdot p^r \cdot (1-p)^{x-r}$$

$$E(X) = \frac{r}{p}$$

$$Var(X) = \frac{r \cdot (1-p)}{p^2}$$

Kommandoen dnbinom(x, størrelse,sannsynlighet) angir antall ganger *x* hvor det ikke skjer det som er forventet ved en gitt sannsynlighet.
Anta at det går 10 forsøk før suksess og vi har en prøvestørrelse på 100 med p=0.3:

```
negbinom<-dnbinom(10:100,10,0.3)
par(bg="lightyellow")
plot(10:100,negbinom,type="l",xlab="x",ylab="f(x)",col="blue",lwd=3,main="Negativ binomial")
```



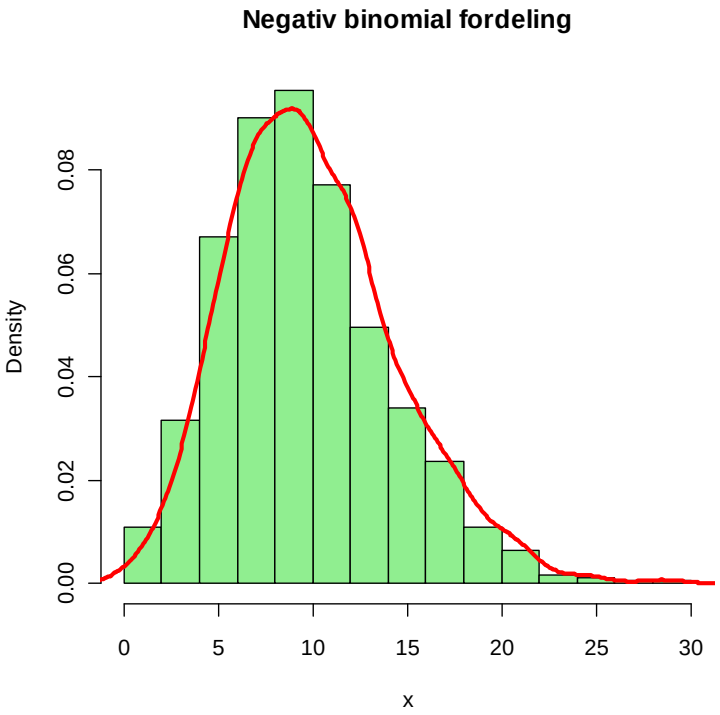
Figuren viser at man må vente til hendelse 20 for ny suksess.
Man kan generere data til en negativ binomial fordeling:

```
 binomtelling<-rnbinom(100,1,0.4)
 table(binomtelling)
binomtelling
 0  1  2  3  4  5  6  8 10
43 24 14  8  4  2  2  2  1
```

Vi ser at variansen er større enn middeltallet:

```
 mean(binomtelling)
[1] 1.4
 var(binomtelling)
[1] 3.737374
```

```
x <- rnbinom(1000,10,p=0.5)
hist(x, probability=T,col="lightgreen",main="Negativ binomial
fordeling")
lines(density(x,bw=1), col="red", lwd=3)
```



Hypergeometrisk fordeling

I den binomiale fordelingen hvor man trekker ut *n* individer fra en populasjon som har *p* individer av en spesiell type, så forutsetter det at man legger tilbake den man har trukket ut før man foretar en ny trekning. Hvis man ikke legger tilbake har man en hypergeomtrisk fordeling og denne kan brukes til å karakterisere det å trekke kuler ut av en urne eller eske, eller beregne sannsynligheten for å vinne i et lotteri (vinnersjansen er mye mindre enn det loddselgeren vil få oss til å tro). I statslotteri beholdes ofte 50% av spillinnsatsen og 50% deles ut igjen som gevinst. Den hypergeometriske fordelingen er prøvetaking uten tilbakelegging. Anta at vi har *N* fargete kuler *e.g.* klinkekuler, bestående av *b* blå kuler og *r* røde kuler i urnen, og man trekker tilfeldig *n* kuler fra urnen uten å legge dem tilbake. *N= b + r*. Funksjonen f(x) = P(x) som angir sannsynligheten for at *x* av de *n* kuler er blå blir:

$$f(x) = \frac{\binom{b}{x}\binom{r}{n-x}}{\binom{b+r}{n}}$$

Forventet verdi for hypergeometrisk fordeling:

$$E(X) = n \cdot p$$

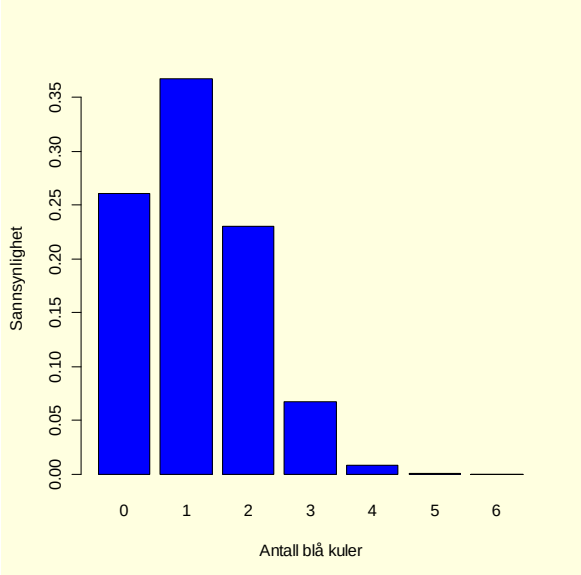
$$Var(X) = n \cdot p(1-p) \cdot \frac{N-n}{N-1}$$

hvor *N=a+b* i eksemplet ovenfor.

Hypergeometrisk-, multinomial- og binomial fordeling beskriver antall som faller innen hver kategori.

Fra R-manualen **?dhyper:**
hyper(x,b,r,n)
phyper(q,b,r,n)
qhyper(p,b,r,n)
rhyper(nn,b,r,n)
Hvor *p* er sannsynlighet 0-1,*nn* er antall observasjoner. Når *b* og *r* (og *N*) blir meget store vil fordelingen nærme seg en binomial fordeling, siden det ikke å legge tilbake får da mindre betydning, *b/N → p* og *r/N → 1-p*. Anta at vi har 10 blå kuler og 20 røde kuler, og vi trekker ut n=6 kuler. Hva er sannsynligheten for at disse 6 er blå ?

```
x<-numeric(7)
b<-10;r<-20;n<-6
for(i in 0:6) x[i]<-dhyper(i,b,r,n)
par(bg="lightyellow")
barplot(x,names=as.character(0:6),
col="blue",ylab="Sannsynlighet",xlab="Antall blå kuler")
```



Det er liten sannsynlighet for at flere enn to kuler er blå.
Sannsynligheten kan også regnes ut fra:

```
px<-choose(b,x)*choose(r,n-x)/choose(b+r,n)
```

nn=30 ganger Monte Carlo simulering av det samme eksperimentet:

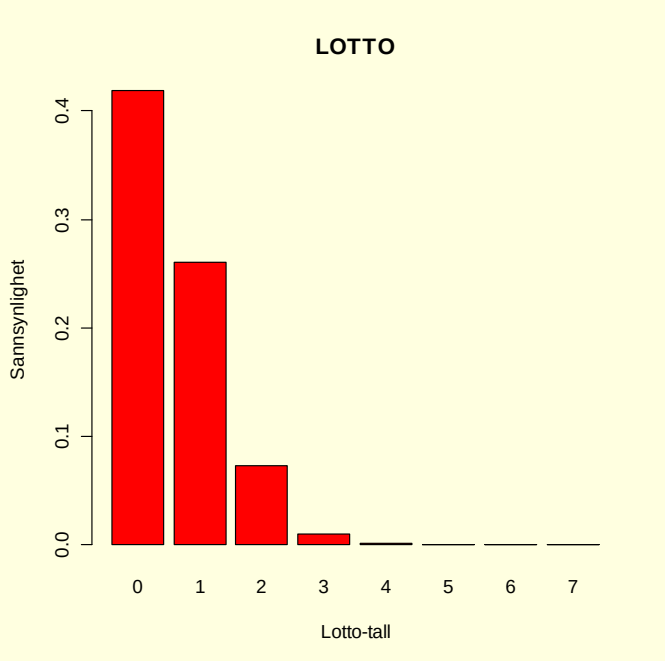
```
rhyper(30,10,20,6)
[1] 2 2 2 0 3 2 2 2 1 1 2 3 4 0 3 3 3 3 2 3 3 3 0 2 3 3 1 1 2 2
```

I Lotto skal man plukke ut 7 tall fra tallrekken 1-34.

```
x<-numeric(8)
```

```
for(i in 0:7) x[i]<-dhyper(i,7,34,7)
par(bg="lightyellow")
barplot(x,names=as.character(0:7),
col="red",ylab="Sannsynlighet",xlab="Lotto-tall",main="LOTTO")
```

Sannsynligheten for å plukke ut flere enn 3 riktige tall blir forsvinnende liten.



Poisson-fordeling

Oppkalt etter den franske matematikeren Siméon-Denis Poisson (1781-1840): *Recheches sur la probabilité des jugements en matière criminelle et en matière civile*. Poisson-fordeling er en utgave av Binomial fordeling med et meget stort antall (n) Bernoulli-forsøk, men hvor sannsynligheten for suksess (p)er meget liten og sannsynligheten for ikke-suksess er meget stor. Poissonfordelingen har en tetthetsfunksjon som angir sannsynligheten for å et tall x når gjennomsnittstelling per enhet er lambda (λ). Det er λ som bestemmer Poisson-fordelingen

$$p(x) = \frac{\lambda^x \cdot e^{-\lambda}}{x!}$$

$$E(X) = \mu = Var(X)$$

Poisson-fordeling beskriver antall hendelser som skjer innen en bestemt tidsperiode eller på et bestemt areal, dvs. tellinger i tidsperioder eller på areal. Når lambda (λ) beskriver antall hendelser per tidsenhet så vil forventet hendelse være $\mu = \lambda \cdot t$.
Hvis λ beskriver tetthet av hendelser per arealenhet så vil forventet antall (μ) på totalarealet A være lik: $\mu = \lambda \cdot A$

$$\lambda = n \cdot p$$

$$p = \frac{\lambda}{n}$$

hvor n er meget stor og p er meget liten. Man behøver ikke å vite verdien av n eller p, bare λ. Vi har også:

$$p(x) = \binom{n}{x}\left(\frac{\lambda}{n}\right)^x\left(1-\frac{\lambda}{n}\right)^{n-x}$$

Poissonfordelingen gjelder for tellinger av individer eller hendelser som forekommer uavhengig av hverandre i tid og rom *e.g.* adioaktiv desintegrasjon, telling av røde blodceller i et tellekammer, telling av trafikkulykker, telling av antall ganger en blomst blir besøkt av et insekt, antall ganger et dyr gjør en spesiell atferd, antall ganger det skjer en ny mutasjon i et genom.
osv.
Sannsynligheten for å få en telling lik 0:

$$p(0) = e^{-\lambda}$$

Sannsynligheten for å få en telling lik 1:

$$p(1) = \lambda \cdot e^{-\lambda}$$

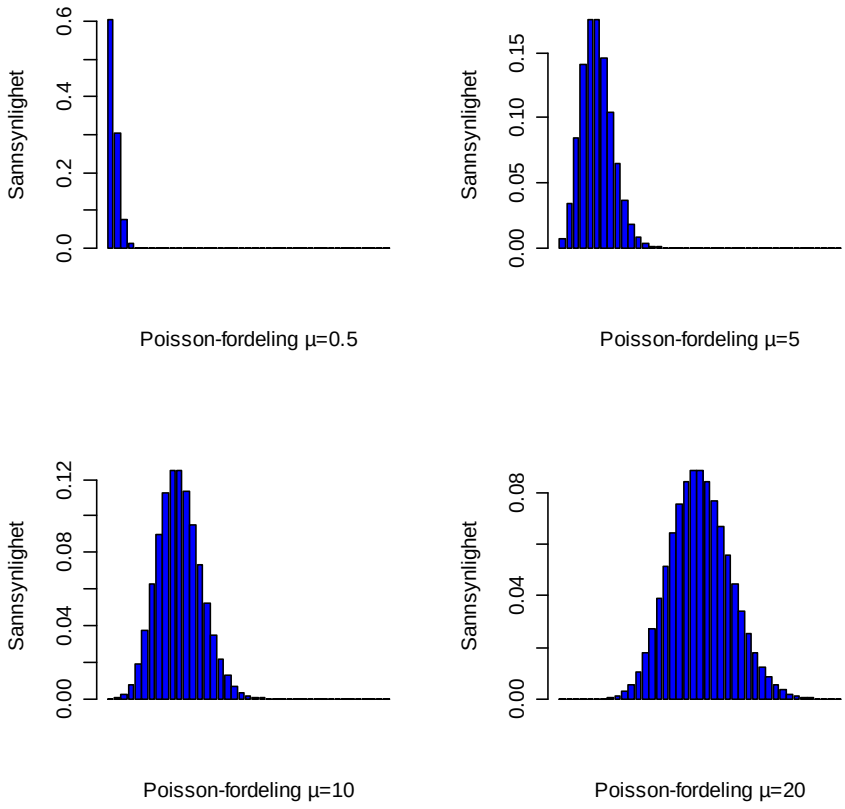
Sannsynligheten for å få en telling lik 2:

$$p(2) = \frac{\lambda^2 \cdot e^{-\lambda}}{2!}$$

Poissonfordelingen bestemmes bare av middeltallet og middeltallet=variansen. Poisson-fordelingen brukes for å finne romlig fordeling og tetthetsmønstere (klumpet, tilfeldig). Poisson-fordelingen brukes også i studiet av tellinger hvor hendelsene er sjeldne og uavhengig av hverandre. For telledata analysert med **glm** er error-fordelingen Poisson-fordelt.
Kommandoer for Poisson-fordeling hvor lambda er middelverdien:
Sannsynlighetsfordeling: **depois(x,lambda)**
Generering av tilfeldige tall: **rpois(n,lambda)**
Kumulativ fordeling: **ppois(q,lambda)**
Kvantil fordeling: **qpois(p,lambda)**

Eksempel: For x fra 0-40 endrer vi middeltallet fra 0.5 til 60 og ser hvordan sannsynlighetsfordelingen endrer seg og nærmer seg normalfordeling

```
 par(mfrow=c(2,2))
 barplot(dpois(0:40,0.5),xlab="Poisson-fordeling
μ=0.5",ylab="Sannsynlighet",col="blue")
 barplot(dpois(0:40,5),xlab="Poisson-fordeling
μ=5",ylab="Sannsynlighet",col="blue")
 barplot(dpois(0:40,10),xlab="Poisson-fordeling
μ=10",ylab="Sannsynlighet",col="blue")
 barplot(dpois(0:40,20),xlab="Poisson-fordeling
μ=20",ylab="Sannsynlighet",col="blue")
```



La oss simulere et eksempel hvor vi finner i gjennomsnitt 0.75 planter per 10 meter veistrekning, og vi måler over 1 km=1000 meter :

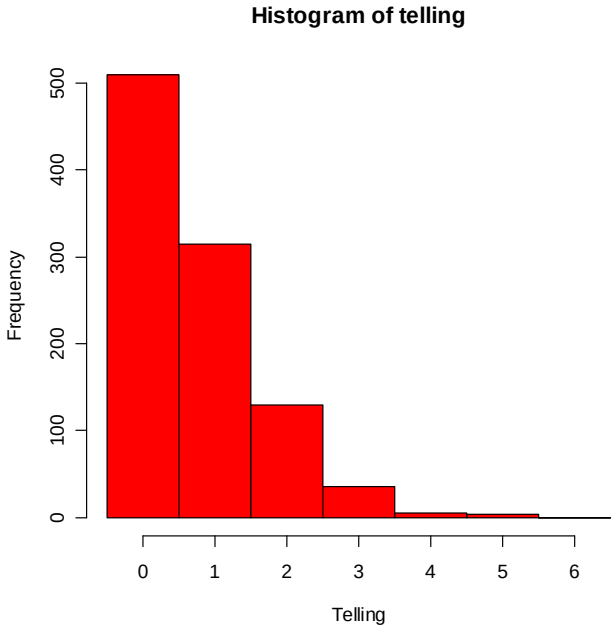
```
telling<-rpois(1000,0.75)
```

Vi lager en tabell over resultatet:

```
table(telling)
```

```
telling
  0   1   2   3   4   5
509 315 130  36   6   4
```

```
hist(telling,breaks=-0.5:7,xlab="Telling",col="red")
```



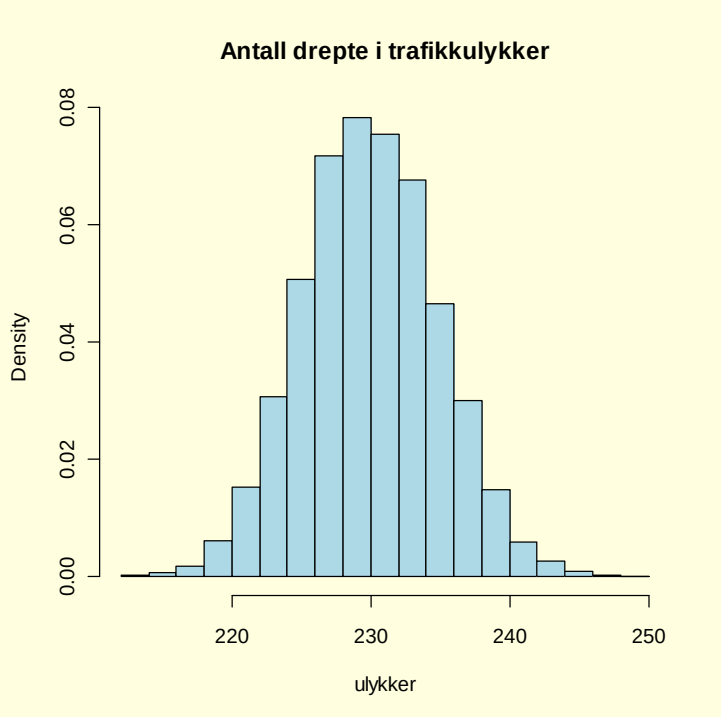

```
 exp(-0.75)
[1] 0.4723666
 exp(-0.75)*0.75
[1] 0.3542749
 (0.75^2*exp(-0.75)/2)
[1] 0.1328531
```

Hvis gjennomsnittelig 230 personer per år dør i trafikk-ulykker vil fordeling av ulykker over 10 år bli:

```
 x<-rpois(10,230);x
 [1] 249 259 241 232 219 215 217 242 251 229
```

Vi kan simulere dette 10.000 ganger:

```
ulykker<-numeric(10000)
for (i in 1:10000){
x<-rpois(10,230)
m<-mean(x)
ulykker[i]<-m
}
par(bg="lightyellow")
hist(ulykker,probability=TRUE,col="lightblue",main="Antall
drepte i trafikkulykker")
```



Minus log-likelihoodfunksjonen LL ($-\ln(L(x_1, x_2, ..x_n|\lambda_1, \lambda_2, \ldots\lambda_n))$ til en Poissonfordeling blir (husk naturlige logaritmer ln i likelihoodfunksjonen, ln er log() i R):

$$LL = -log\big(L(x_1, x_2, \ldots)|(\lambda_1, \lambda_2, .., \lambda_n)\big) = \prod_{i=1}^{n} P(x_i|\lambda_i) = -\sum_{i=1}^{n} log\left(\frac{\lambda_i^{x_i} \cdot e^{-\lambda_i}}{x_i!}\right) =$$

$$= -\sum_{i=1}^{n} x_i \cdot log(\lambda_i) - \lambda_i - log(x!) = -\sum_{i=1}^{n} x_i \cdot log(\lambda_i) - \lambda_i - x_i \cdot log(x_i) + x_i$$

Normalfordelingen

Tidligere har vi sett på diskrete sannsynlighetsfunksjoner, og nå skal vi se på kontinuerlige sannsynlighetsfunksjoner og starter med normalfordelingen (Gaussfordeling, Karl Friedrich Gauss (1777-1885)). Fordelingen ble først beskrevet av Abraham de Moivre (1667-1754) i *The doctrine of chances* (1738, i 1738 av Pierre Simon Laplace (1749-1829. Poincare ga den navnet normal-.
Når funksjonen *f(x)* nedenfor plottes mot *x* fås den klokkeformete normalfordelingskurven ("The bell curve"). Normalfordelingskurven er bare bestemt av **gjennomsnittsverdien** (μ) og **standardavviket** (σ). **Variansen** er lik σ^2. Totalarealet under normalfordelingskurven $N(\mu, \sigma^2)$ er lik 1. En **tetthetskurve** er en matematisk modell av en fordeling hvor totalt areal

under kurven er lik 1=100%. Toppen av kurven er ved x=μ og høyden av toppen er lik $1/(\sigma\sqrt{2\pi})$.

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Funksjonen *f(x)* kalles sannsynlighetstetthetsfunksjonen og beskriver sannsynlighetsfordelingen for en kontinuerlig tilfeldig variabel. Legg merke til at funksjonen inneholder de berømte matematiske uttrykk pi(*π*=3.14159..), det naturlige tallet *e*= 2.71828..som danner grunntallet i naturlige logaritmer, samt kvadratroten av 2 (*√2*).
Sannsynligheten *P* for at en variabel *x* skal falle mellom to punkter *a* og *b* er lik arealet under normalfordelingskurven:

$$P(a \leq x \leq b) = \int_a^b f(x) \cdot dx$$

Hvis et middeltall *m* ligger mellom -1.96·SD og +1.96·SD så tilsvarer dette 95% av arealet under normalfordelingskurven, og p=0.05. Hvis μ=0 og σ=1 får vi **standard normalfordeling** (N(0,1)). For hver måleverdi *x* beregnes en ny verdi *z* kalt **z-skår**. Z-skår er et mål på antall standardavvik (σ) som en dataverdi er vekk fra middelverdien μ.

$$z = \frac{x - \mu}{\sigma}$$

Standard normalsannsynlighetstabell angir arealet under normalfordelingskurven til venstre for z. De standardiserte z-verdiene har ingen måleenhet.

$$x = \mu + z \cdot \sigma$$

Hvis *x* er 2 standardavvik større enn middeltallet har vi *x=μ+2σ*. Setter vi denne x inn i formelen får vi *z=(μ+2σ-μ)/σ=2*.

I et **normal-kvantilplot** rangeres datapunktene og percentilrang omdannes til z-skår. Hvis z-skår plottes på x-aksen og datapunktene på y-aksen så vil dette gi en rett linje hvis dataene er normalfordelte.

Gjennomsnittsverdien eller forventet verdi i en normalfordeling er:

$$E(X) = \mu = \int_{-\infty}^{+\infty} x \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx$$

Variasjonen til en normalfordeling er:

$$var(X) = \sigma^2 = \int_{-\infty}^{+\infty} (x - \mu)^2 \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx$$

Generelt vil for en forventet verdi være:

$$E(X) = \int x \cdot f(x)dx = \mu$$

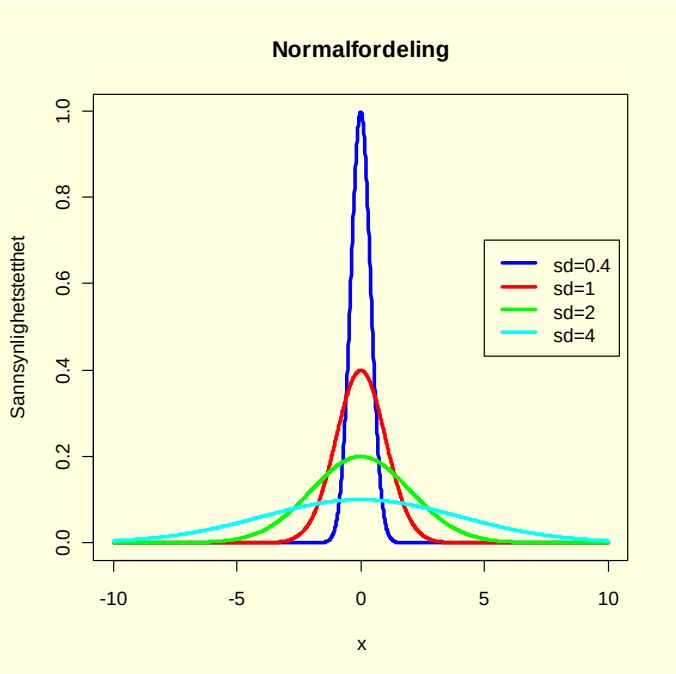
og variansen vil være:

$$var(X) = \int x^2 \cdot f(x)dx - E^2(X) = \sigma^2$$

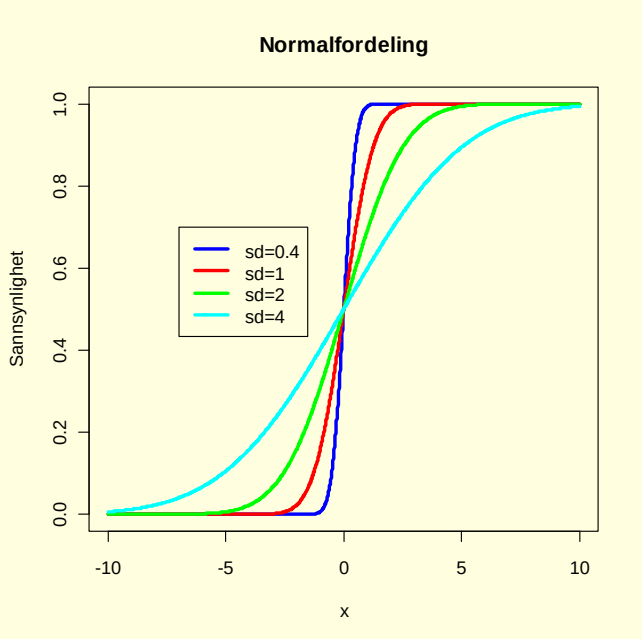
Kommandoen **rnorm** lager tilfeldige utvalg fra en normalfordeling. R bruker **r** (**rnorm**)for å lage en tilfeldig prøve fra en fordeling, **p** brukes for å lage en kumulativ tetthetsfordeling (**pnorm**), **d** brukes for å lage en sannsynlighetstetthetsfordeling (**dnorm**) og **q** brukes for å lage kvantiler (**qnorm**). Enhver ikke-negativ funksjon hvor arealet under grafen til funksjonen fra -∞ til +∞ er lik 1 kalles en sannsynlighetstetthetsfunksjon. Begrepet tetthet skriver seg fra analogi med stoff i rommet som har en tetthet. Normalfordelingskurven er symmetrisk rundt μ, og sannsynlighetstettheten minsker når man beveger seg vekk fra μ. Teoretisk fortsetter de to halene uendelig i begge retninger, men dette skjer ikke i relle datasett.

Forventet verdi (E-ekspektert) for den sentrale tendensen i den variable normalfordelingen er E[x]=μ, men denne sier ingenting om spredningen eller variansen. I figuren under er det samme sentrale tendens, men forskjellig varianse. Variansen er et mål på spredningen av verdiene til variablene rundt middeltallet, var(x)= $E[(x-\mu)^2]=\sigma^2$.

```
x<-seq(-10,10,0.01)
norm<-dnorm(x,mean=0,sd=0.4)
par(bg="lightyellow")
plot(x,norm,type="l",
col="blue",lwd=3,ylab="Sannsynlighetstetthet",main="Normalford
eling")
lines(x,norm,type="l",col="cyan",lwd=3)
legend(5,0.7,c("sd=0.4","sd=1","sd=2","sd=4"),lty=
c(1,1,1,1),lwd=c(3,3,3,3),col=c("blue","red","green","magenta"
,"cyan"))
```



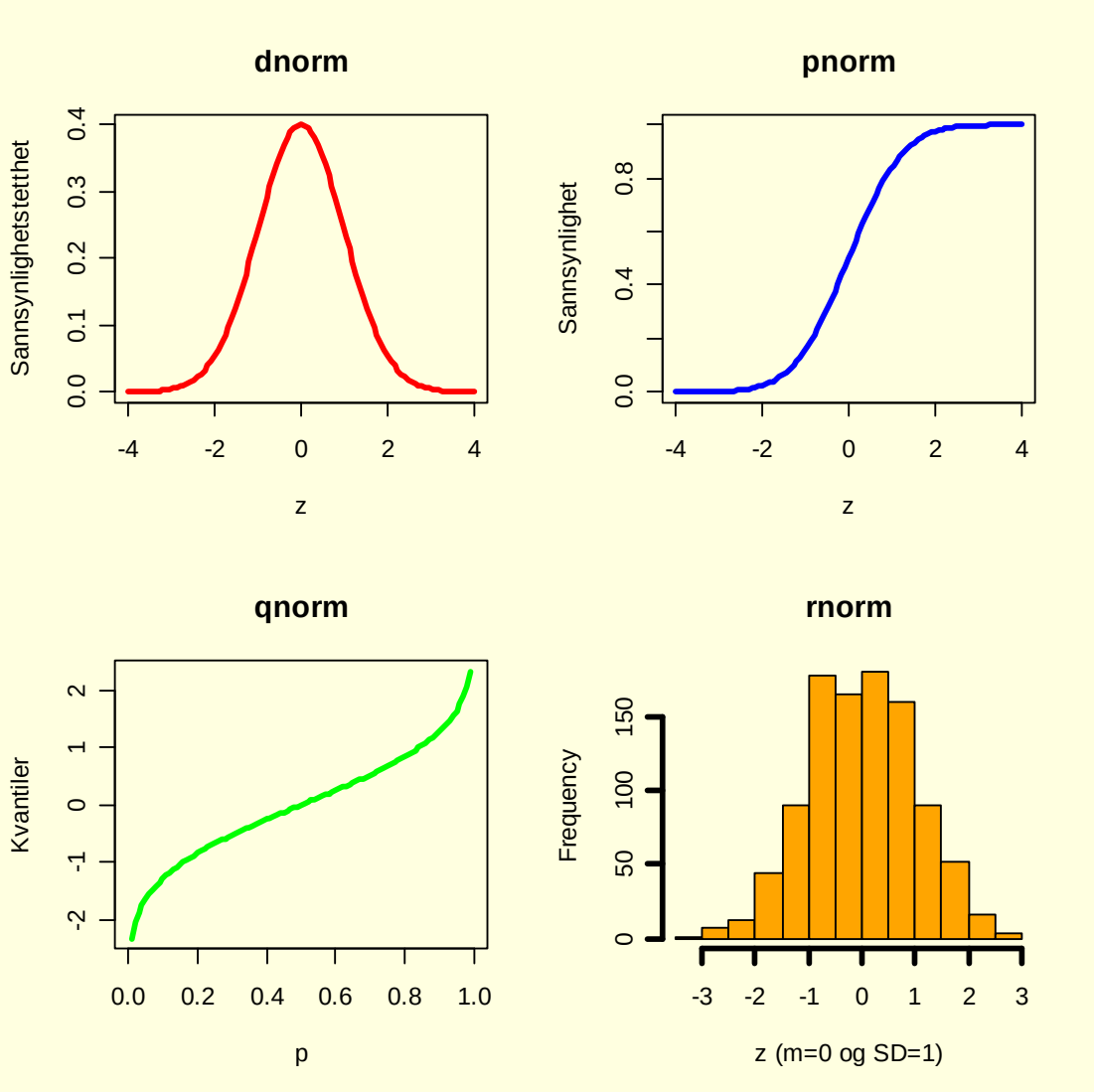
Figur. Standard normalfordeling med forskjellig standardavvik (forskjellig varianse), men samme middeltall (mean=0. Sannsynlighetsfordelingen har klokkeform.



Figur. Kumulativ sannsynlighetsfordeling (pnorm)
Samme middeltall (mean=0),men forskjellig varianse.

```
par(mfrow=c(2,2),bg="lightyellow")
curve(dnorm,-
4,4,xlab="z",ylab="Sannsynlighetstetthet",col="red",lwd=3,main
="dnorm")
curve(pnorm,-
4,4,xlab="z",ylab="Sannsynlighet",col="blue",lwd=3,main="pnorm
")
```

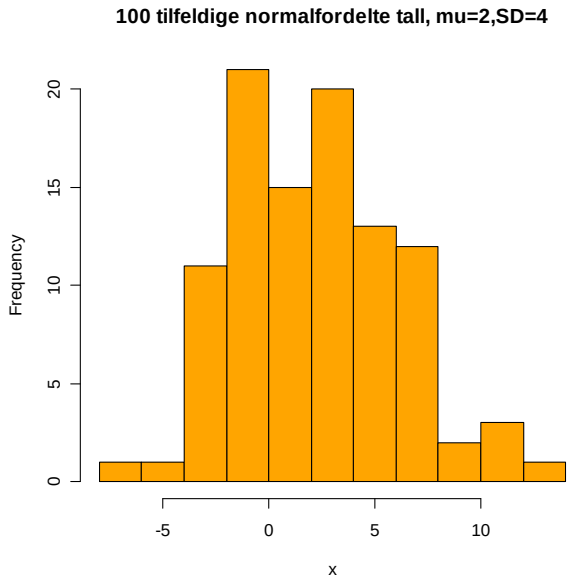
```
curve(qnorm,0,1,xlab="p",ylab="Kvantiler",col="green",lwd=3,ma
in="qnorm")
x<-rnorm(1000,0,1)
hist(x,col="orange",lwd=3,xlab="z (m=0 og SD=1)",main="rnorm")
```




```
x<-rnorm(40)
```

Hvis vi skal ha 100 tilfeldige tall hentet fra en normalfordeling med *e.g.* middeltall=2 og standardavvik=4:

```
x<-rnorm(100,2,4)
hist(x,col="orange",main="100 tilfeldige normalfordelte tall,
mu=2,SD=4")
```



Hvis man ønsker å lage en matrise med tilfeldige tall kan dette gjøres med **matrix()**.

Kommandoen **pnorm** angir hvor stor del av en standard normalfordelingskurve (middeltall=0, standardavvik=1) vi har hvis vi oppgir verdien av z eller en kvantil. Dvs. vi får oppgitt sannsynligheten. Det er pnorm vi benytter til hypotesetesting

```
pnorm(1.96)
[1] 0.9750021
```
Dvs. 97.5%

Det resterende arealet av normalfordelingskurven hvor vi ser på en av halene.

```
 1-pnorm(1.96)
[1] 0.02499790
```
Dvs. 2.5 %

Hvis vi skal finne arealet av normalfordelingskurven som ligger mellom +- ett standardavvik:_

```
 pnorm(1)-pnorm(-1)
[1] 0.6826895
```
Som tilsvarer ca. 68.3 % av arealet under normalfordelingskurven.

Eller arealet som ligger mellom +-1.96 SD som vi ser blir lik 95% altså p=0.05

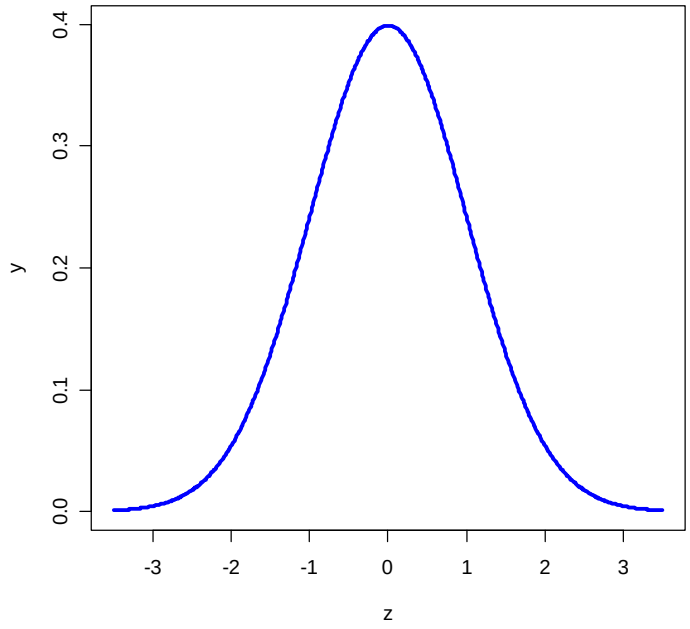
```
 pnorm(1.96)-pnorm(-1.96)
[1] 0.9500042
```

Kommandoen **dnorm** angir hvor stor sannsynlighet det er å finne en gitt z i en standard normalfordeling gitt middeltall og standardavvik.
Sannsynlighet for å få z=0

```
dnorm(0)
[1] 0.3989423
```

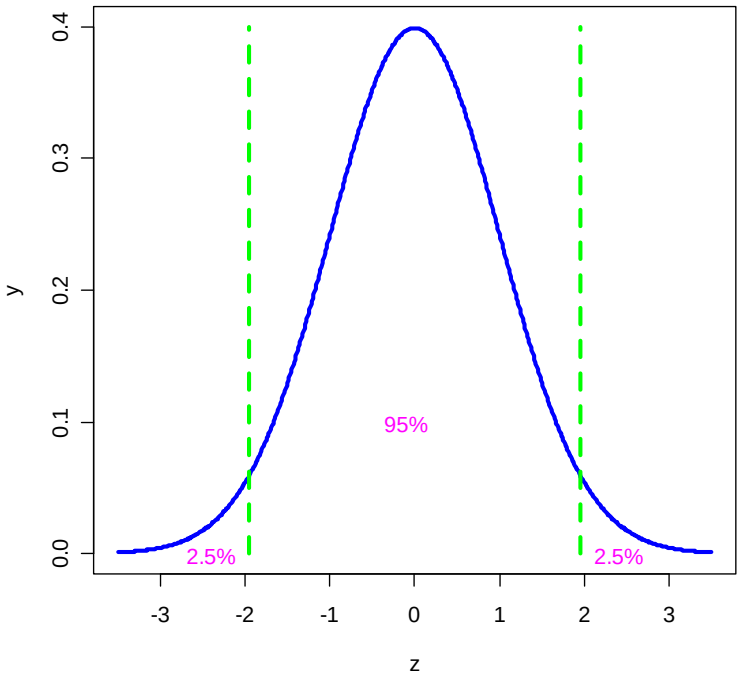
Vi kan lage en figur av fordelinger av z fra -3.5 til +3.5 som gir en standard normalfordelingskurve:

```
z<-seq(-3.5,3.5,0.01)
y<-dnorm(z)
plot(z,y,type="l",col="blue",lwd=3)
```



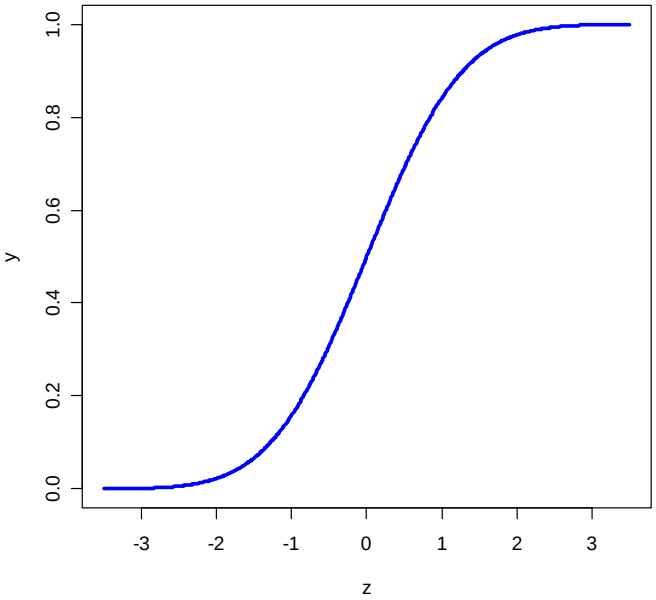
Vi trekker vertikale prikkete linjer fra z=1.96 og z=-1.96

```
lines(c(-1.96,-1.96),c(0,0.4),lty=2,col="green",lwd=3)
lines(c(1.96,1.96),c(0,0.4),lty=2,col="green",lwd=3)
text(-0.1,0.1,"95%",col="magenta")
text(2.4,0,"2.5%",col="magenta")
text(-2.4,0,"2.5%",col="magenta")
```



Kommandoen **pnorm** angir den kumulative sannsynligheten:

```
y<-pnorm(z)
plot(z,y,type="l",col="blue",lwd=3)
```



Vi ser at sannsynligheten summeres til 1.

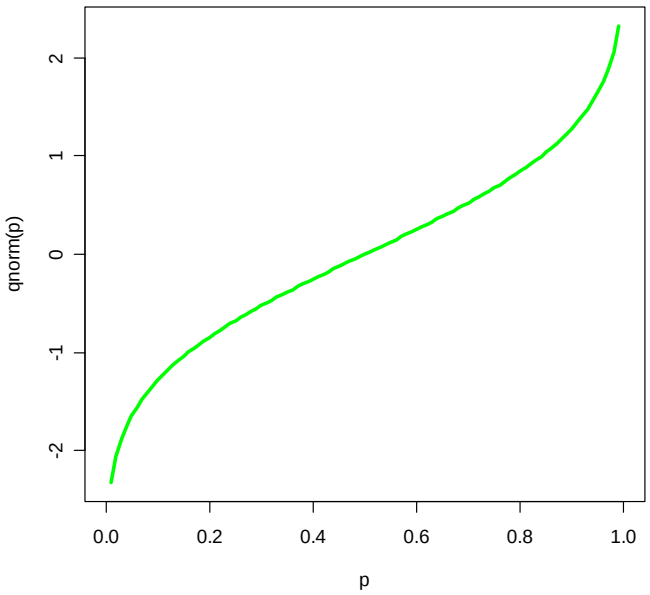
Kvantiler til normalfordelingen er det inverse av sannsynlighet og kjenner vi sannsynligheten vil kommandoen

qnorm oppgi verdien av z som tilsvarer sannsynligheten. Det er på denne måten man finner de kritiske verdiene for z. Intervallet fra 0.05-0.95:

```
 qnorm(c(0.05,0.95))
[1] -1.644854  1.644854

 qnorm(0.5)
[1] 0

 qnorm(1)
[1] Inf
```

Plot av kvantiler versus sannsynlighet:

```
 p<-seq(0,1,0.01)
 plot(p,qnorm(p),type="l",col="green",lwd=3)
```



Konfidensintervall er alltid tohalet. Statistisk inferens vil si å trekke konklusjoner om hele populasjonen ut fra vår prøve. I konfidensintervallet brukes standardfeilen (=standardavviket til middeltallene omkring den sanne verdi μ).

$$z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

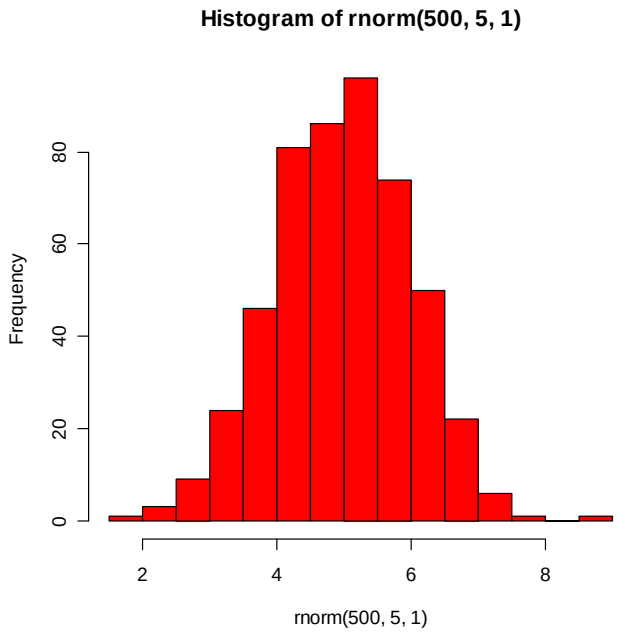
Konfidensintervallet:

$$\bar{x} \pm z \cdot \frac{\sigma}{\sqrt{n}}$$

Kommandoen **rnorm** generer tilfeldige tall. Velegnet for å lage kunstige datasett. Man angir hvor mange tilfeldige tall man vil ha, og i tillegg oppgis middeltallet og standardavvik:

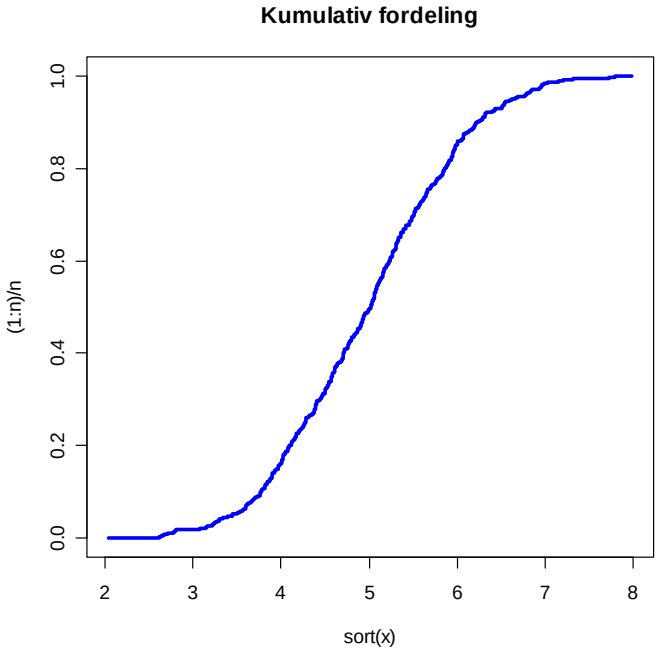
Et **histogram** viser grafisk formen på en fordeling. I et histogram er det ingen avstand mellom kategoriene (stolpene) og arealet er proporsjonal med mengde. Vi kan selv bestemme bredde på stolpene.
Vi lager et histogram over 500 normalfordelte tall med middeltall=5 og standardavvik=1:

```
hist(rnorm(500,5,1),col="red")
```



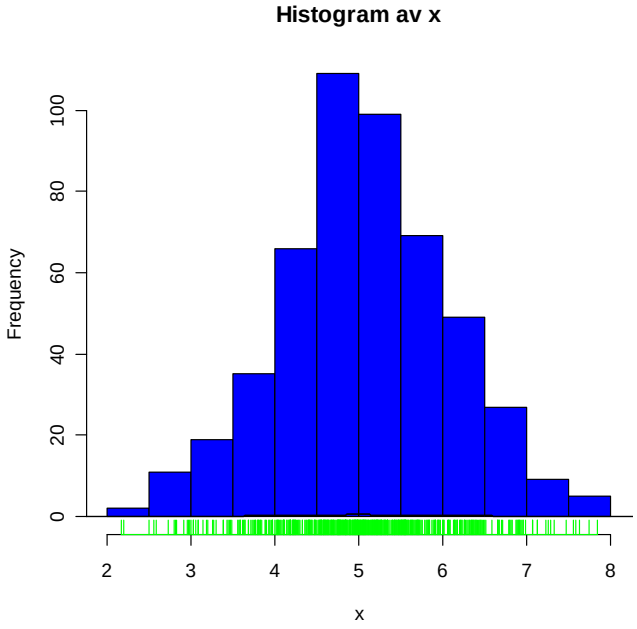
Kumulativ fordeling av 500 tilfeldige tall med betingelser som over:

```
x=rnorm(500,5,1)
n<-length(x)
plot(sort(x),(1:n)/n,type="s",ylim=c(0,1),col="blue",lwd=3,main="Kumulativ fordeling")
```



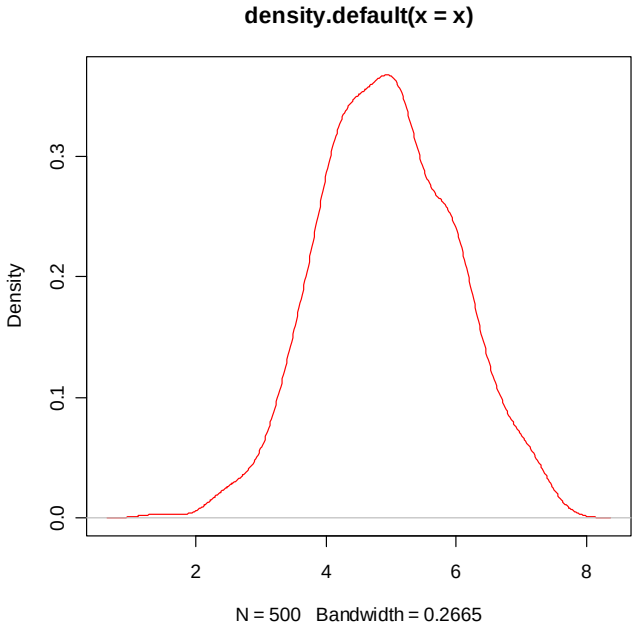

```
x=rnorm(500,5,1)
hist(x,col="blue",main="Histogram av x")
rug(x,col="green")
```

Kommandoen **rug()** viser datapunktene som streker på x-aksen.



Med kommandoen og funksjonen **plot(density())** kan man se tettheten i datasettet

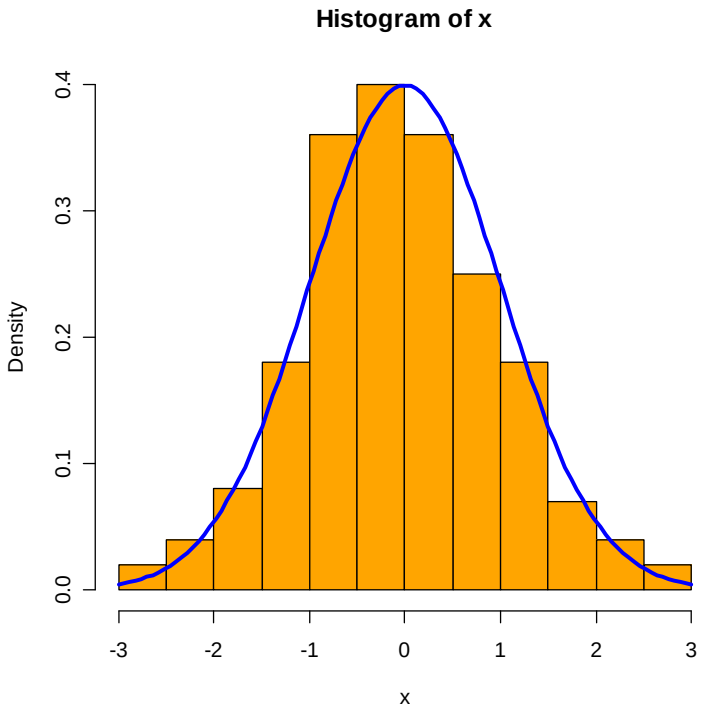
```
plot(density(x),col="red")
```



Kommandoen **Cut()** deler x-verdier opp i intervaller.

Velger ut 200 normalfordelte tall, lager et histogram over tallene og med kommandoen curve lages et plot over normalfordelte x.

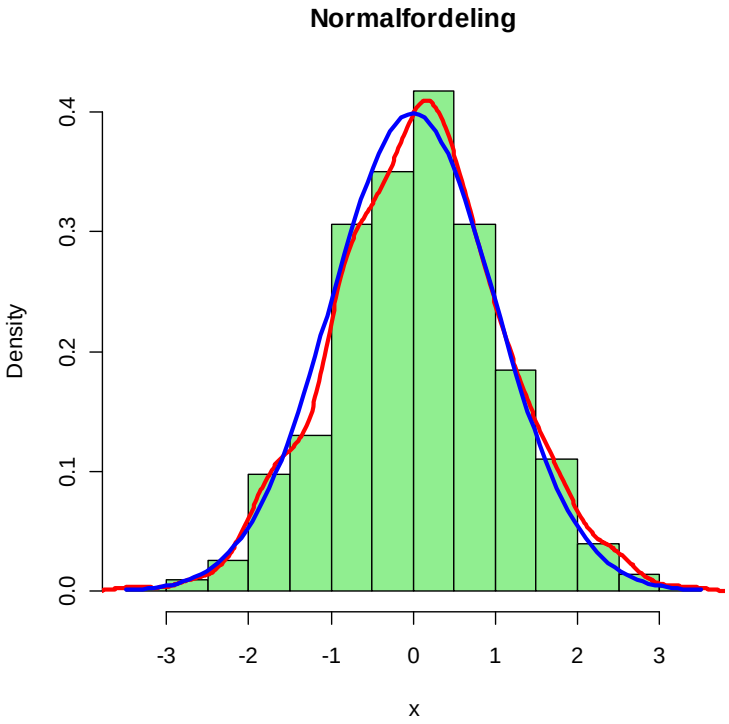
```
x<-rnorm(200)
hist(x,freq=F,col="orange")
curve(dnorm(x),add=T,col="blue",lwd=3)
```



Fordeling av normalfordelte tall, med teoretisk tetthetsfordeling og tetthetsfordelingen fra prøven:

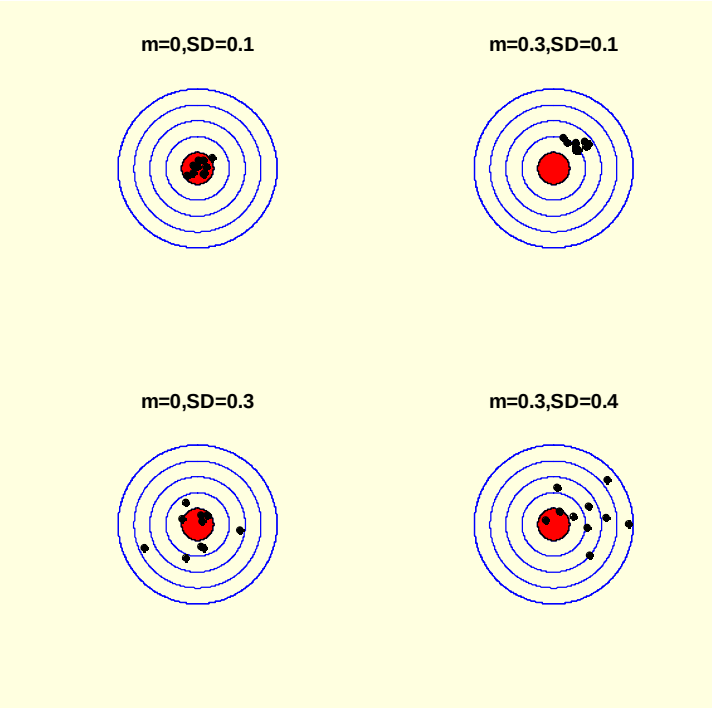
```
x<- rnorm(1000)
```

```
hist(x,probability=T,col="lightgreen",main="Normalfordeling")
lines(density(x),col="red",lwd=3)
curve(dnorm(x),add=T,col="blue",lwd=3)
```



Vi kan bruke eksemplet med en blink hvordan målepunktene har forskjellig samling og spredning, hvor middeltall m=0 tilsvarer blinken. Skuddene tilsvarer en matrise med 20 tilfeldige tall ordnet i to kolonner.

```
a<-seq(0,2*pi,0.01)
x<-sin(a)
y<-cos(a)
par(mfrow=c(2,2),bg="lightyellow")
plot(x,y,type="l",xlab="",ylab="",axes=F,
asp=1,col="blue",main="m=0,SD=0.1")
for (i in seq(0.2,1,0.2)) lines(i*x,i*y,col="blue")
polygon(x*0.2,y*0.2,col="red")
treff<- matrix(ncol=2,rnorm(20,0,0.1))
points(treff,pch=19)
#Endre SD og m og kjør på nytt
```



Students t-fordeling

Students t-fordeling publisert av W.S. Gosset under pseudonymet "Student". Gosset arbeidet ved Guinness-bryggeriet i Dublin.

$$t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{n}}} = \frac{\bar{x} - \mu}{SE}$$

Students t-fordeling er en symmetrisk og tar hensyn til små prøveverdier (<30).

Lager en sekvens av z:

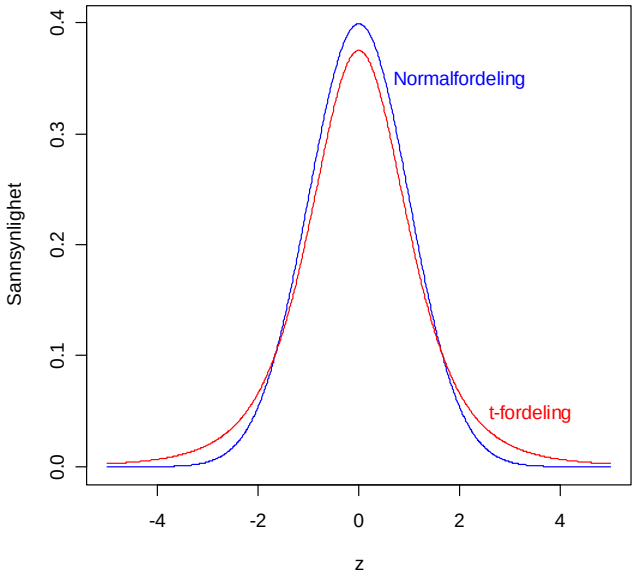
```
z<-seq(-5,5,0.01)
```

Lager en normalfordeling av z:

```
plot(z,dnorm(z),type="l",lty=2,xlab="z",ylab="Sannsynlighet",col="blue")
```

Lager en t-fordeling av z med df=4:

```
lines(z,dt(z,df=4),col="red")
text(2,0.35,"Normalfordeling",col="blue")
text(3.4,0.05,"t-fordeling",col="red")
```



Antall standardavvik μ ± S.D.	Areal under normalfordelings-kurven	Kumulativ sannsynlighet	t-fordeling med 4 d.f.
1.0	68.3%	84.1%	0.5145
1.96	95%	97.5%	2.7764
2.58	99%	99.5%	4.6041
3.29	99.9%	99.95%	8.6103

Tabell som viser hvilke areal middeltallet +/- 1SD utgjør av normalfordelingskurven = 68.3%. Middeltallet +/-1.96·SD utgjør 95% av arealet under normalfordelingskurven. De tilsvarende t-verdiene er vist for 4 frihetsgrader (d.f.)

```
qt(0.975,4)
[1] 2.776445
```

Tetthetsfunksjonen til t-fordelingen med *n* er lik antall frihetsgrader (df):

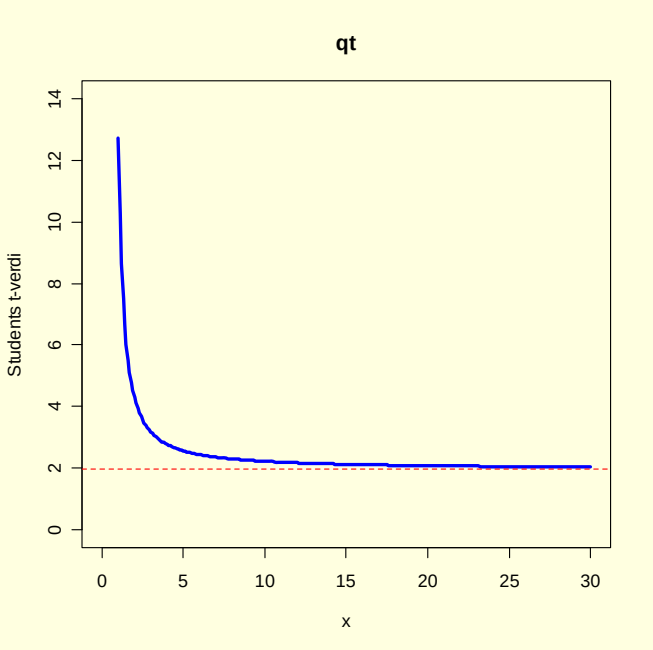
$$f(x) = y = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{n \cdot \pi} \cdot \Gamma\frac{n}{2}} \cdot \left(1 + x^{\frac{2}{n}}\right)^{-\left(\frac{n+1}{2}\right)}$$

Se R-manualen:

```
?dt
```

Denne funksjonen ser komplisert ut, men har form som en normalfordelingskurve med svært brede haler.
Vi kan plotte t-verdiene mot prøvestørrelsen, p=0.025 dvs. 0.975, og da ser vi at t-verdiene nærmer seg verdiene fra normalfordelingen når n>30. Det er satt inn en horisontal strek for 1.96 som tilsvarer p=0.05 i normalfordelingen.
Students t-fordeling tar hensyn til eksperimenter hvor man har få prøveverdier og ofte er *n* mye mindre enn 30. Ettersom antall frihetsgrader øker så nærmer t-fordelingen seg normalfordelingen

```
x<-seq(0,30,0.1)
par(bg="lightyellow")
plot(x,qt(0.975,x),ylim=c(0,14),type="l",col="blue",lwd=3,ylab
="Students t-verdi",main="qt")
abline(h=1.96,col="red",lty=2)
```

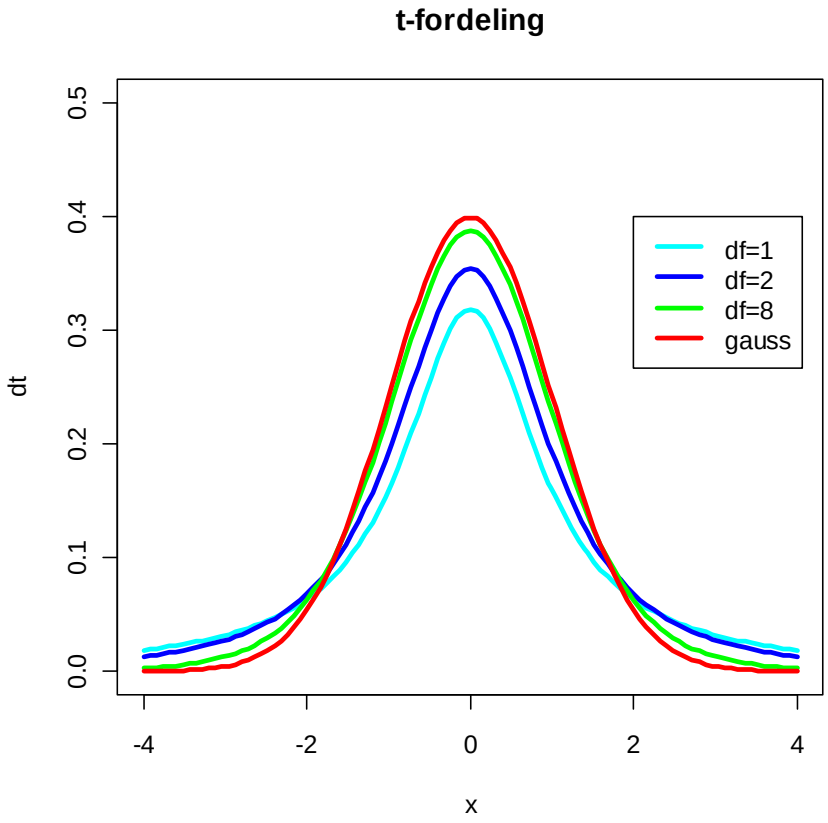


Vi kan se hvordan t-fordelingen nærmer seg normalfordelingen når antall frihetsgrader (df) øker.

```
x<-seq(-5,5,0.01)
plot(x,dnorm(x),type="l",lty=2,xlab="x",ylab="Sannsynlighet",c
ol="blue")
lines(x,dt(x,df=30),col="red")
lines(x,dt(x,df=10),col="cyan")
lines(x,dt(x,df=5),col="tomato")
lines(x,dt(x,df=2),col="green")
lines(x,dt(x,df=1),col="orange")
title(main="Student's t-fordeling")
```

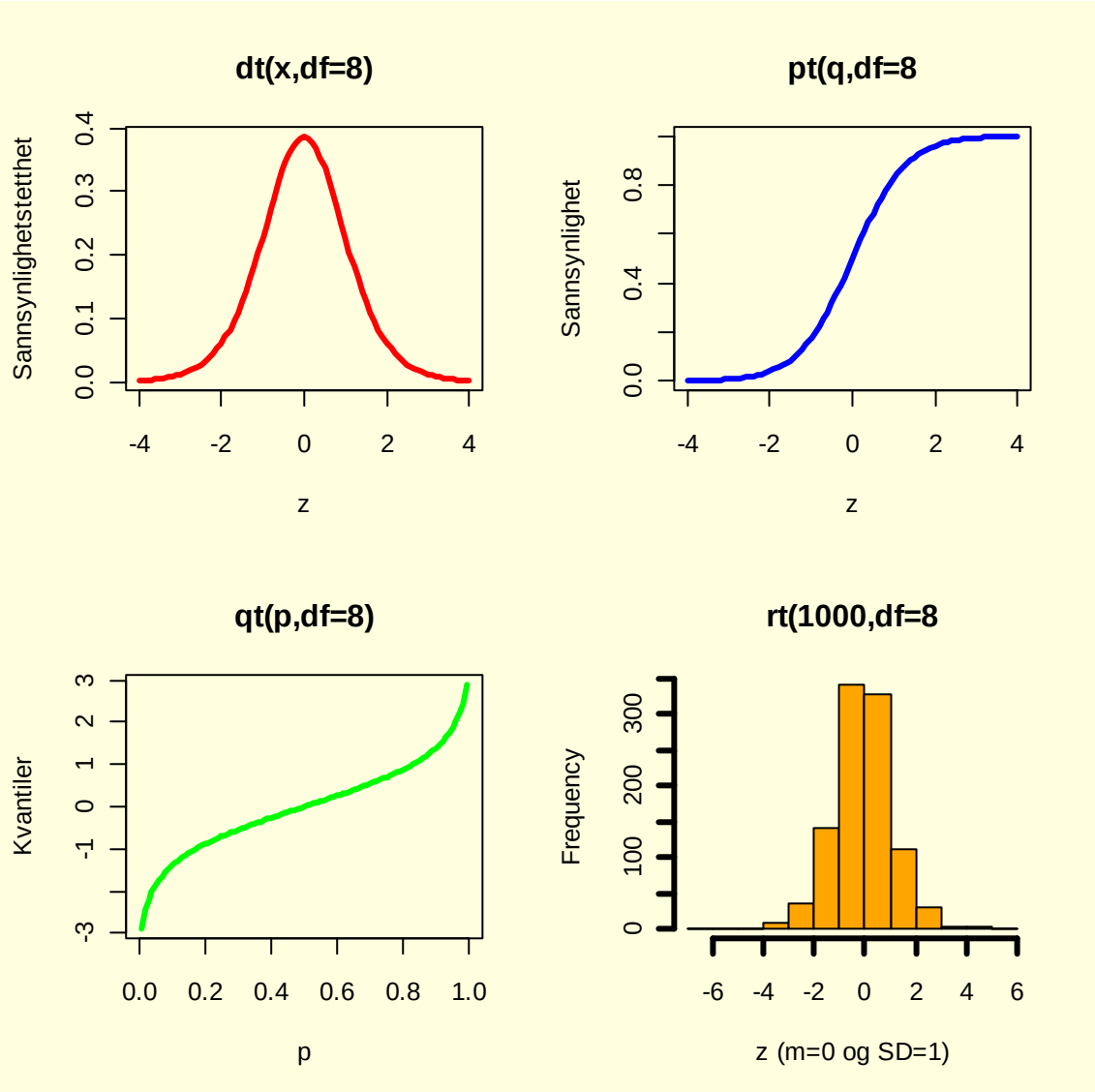
Eller via følgende figur:

```
x<-seq(-4,4,0.1)
curve(dt(x,df=1),col="cyan",xlim=c(-
4,4),ylim=c(0,0.5),lwd=3,ylab="dt",
main="t-fordeling")
curve(dt(x,df=2),add=T,col="blue",lwd=3)
curve(dt(x,df=8),add=T,col="green",lwd=3)
curve(dnorm(x),add=T,col="red",lwd=3)
legend(2,0.4,c("df=1","df=2","df=8","gauss"),lty=c(1,1,1,1),
lwd=c(3,3,3,3),col=c("cyan","blue","green","red"))
```



Sannsynlighetstetthet: **dt(x,df)**
Sannsynlighetsfordeling: **pt(q,df)**
Kvantil: **qt(p,df)**
Randomiserte: rt**(n,df)**
Det er valgfritt om man ønsker å ta med en ikke-sentralparameter ncp: **dt(x,df,ncp)**

```
par(mfrow=c(2,2),bg="lightyellow")
x<-seq(-4,4,0.1)
plot(x,dt(x,8),type="l",xlab="z",ylab="Sannsynlighetstetthet",
col="red",lwd=3,main="dt(x,df=8)")
plot(x,pt(x,8),type="l",xlab="z",ylab="Sannsynlighet",col="blu
e",lwd=3,main="pt(q,df=8")
p<-seq(0,1,0.01)
plot(p,qt(p,8),type="l",xlab="p",ylab="Kvantiler",col="green",
lwd=3,main="qt(p,df=8)")
x<-rt(1000,8)
hist(x,col="orange",lwd=3,xlab="z (m=0 og
SD=1)",main="rt(1000,df=8")
```



Students t-test gjøres med kommandoen **t.test**(x,y).
Hvis variansene er like: **t.test(x,y,var.equal=TRUE)**.
Er det parvis t-test: **t.test(x,y,paired=TRUE,alt="two.sided")**.

Wilcoxon rangeringstest gjøres med kommandoen **wilcox.test**(x,y) for eksempel **wilcox.test(x,mu=…,alt="two.sided")** eller **wilcox.test(x,y,alt="two.sided")**.

Kommandoen **prop.test** tester om to proporsjoner er forskjellige og p-verdien blir gitt i svaret.
prop.test(c(a,b),c(a1,b2)))

Teststyrke: Type I og type II feil

Vi ønsker å gjøre statistisk inferens om vi skal beholde eller forkaste den **statistiske nullhypotesen** (H_0) som velger et konservativt utgangspunkt om at det ikke er noen forskjell mellom to prøver.Imidlertid er det alltid statistisk

usikkerhet. **Type I feil** (α) vil si at vi forkaster en nullhypotese som er sann. **Type II feil** (β) vil si å beholde en nullhypotese som er feil. Når man beregner p-verdi lager man implisitt et estimat av signifikansen α. Ved konvensjon velges $p \leq 0.05$ som grense for å forkaste nullhypotesen. Hvis $p > 0.05$ beholdes nullhypotesen.

	Behold H_0	Forkast H_0
H_0 sann	Korrekt avgjørelse	Type I feil (α)
H_0 falsk	Type II feil (β)	Korrekt avgjørelse

Teststyrke (1-β) er lik sannsynligheten for å forkaste nullhypotesen når den er falsk. Teststyrken på en t-test kan finnes med kommandoen **power.t.test()**.
Vanligvis velges σ=0.05 og β=0.2, og da blir teststyrken 1-β=0.8dvs. 80%. Dette kan brukes til å beregne prøvestørrelse. Man må kjenne variansen s^2 fra et pilotstudium eller forkunnskap, og man ønsker å finne en forskjell δ. Antall replikater (*n*) som trengs for å forkaste nullhypotesen med teststyrke 0.8 blir:

$$n \approx \frac{8 \cdot s^2}{\delta^2}$$

I R er det funksjoner som kan brukes til å undersøke teststyrke **power.t.test()** for en- og to-prøve t-test, **power.prop.test()** for to-prøvetest for proporsjoner, samt **power.anova.test()** for balansert en-veis ANOVA.
Hvis middeltallet for en prøve er 8 med standardavvik 1.2, og vi ønsker å finne en forskjell på δ=0.8 mellom middeltallene til to prøver, dvs. ±10% for teststyrke 80%, så må prøven være n=36.

```
power.t.test(delta=0.8,sd=1.2,power=0.8)
     Two-sample t test power calculation
              n = 36.3058
          delta = 0.8
             sd = 1.2
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
 NOTE: n is number in *each* group
```

Teststyrke 0.8 vil si 80% sannsynlig å finne et signifikant resultatet.

Hvis vi ønsker en forskjell på 1 mellom to middeltall δ=0.8 hvordan endrer teststyrken seg med n ?
Hvis man har flere uavhengige variable må man ha tilsvarende økning i prøvestørrelse.

Hvis vi ønsker forskjell delta=1 mellom to middeltall, så kan vi beregne teststyrken (1-β) når *n* varierer fra 2-40:

```
n<-c(2,10,20,30,40)
styrke<-power.t.test(n,delta=1);styrke
    Two-sample t test power calculation
            n = 2, 10, 20, 30, 40
```

```
          delta = 1
             sd = 1
      sig.level = 0.05
          power = 0.09131778, 0.56198462, 0.86895280, 0.96770825, 0.99298477
    alternative = two.sided
NOTE: n is number in *each* group
```

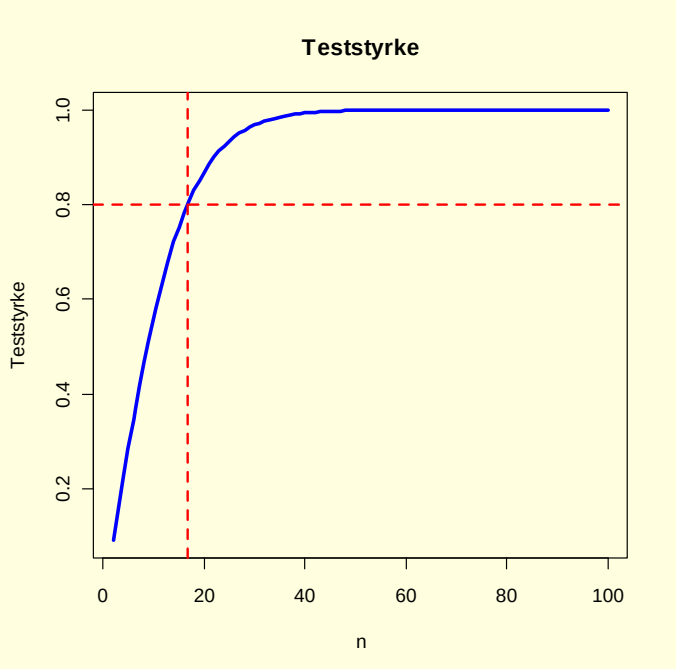
Vi kan la n=1-100 og plotte teststyrke for en tosidig to-prøve t-test som funksjon av n:

```
n<-seq(2,100,1)
styrke<-power.t.test(n,delta=1,type ="two.sample",alternative
= "two.sided")
par(bg="lightyellow")
plot(n,styrke$power,type="l",col="blue",lwd=3,ylab="Teststyrke
",main="Teststyrke")
abline(h=0.8,lty=2,col="red",lwd=2)
m<-power.t.test(delta=1,power=0.8,type
="two.sample",alternative = "two.sided");m
```

```
     Two-sample t test power calculation
              n = 16.71477
          delta = 1
             sd = 1
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

```
abline(v=m$n,lty=2,col="red",lwd=2)
```

Prøvestørrelsen bør være n=17 for å få en teststyrke 0.8



Teststyrke for en en-veis ANOVA:

```
 styrke2<-power.anova.test(groups=3,
between.var=0.95,within.var=1.25,power=.80)
 styrke2
```

```
     Balanced one-way analysis of variance power calculation
         groups = 3
              n = 7.4342
    between.var = 0.95
     within.var = 1.25
```

```
      sig.level = 0.05
          power = 0.8
NOTE: n is number in each group
```

Students t-test

Vi ønsker å foreta statistisk inferens om to grupper er fra samme eller forskjellig populasjon. Vi har to uavhengige og normalfordelte prøver x_1 og x_2 med henholdsvis prøvestørrelse n_1 og n_2. Nullhypotesen er at det ikke er noen forskjell mellom de to prøvene (H_0: $\mu_1=\mu_2$). Vi benytter de greske bokstavene my (μ) for middeltall og sigma (σ) for standardavvik når vi snakker om de sanne verdiene for populasjonen, som vi aldri finner men som vi lager et estimat for ut fra vår prøve. Vi kan summere variansene til de to prøvene og beregne to-prøve t-statistik:

$$t_{n1+n2-2} = \frac{differanse}{SE\ til\ differensen} = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} = \frac{\overline{x_1} - \overline{x_2}}{SE(\overline{x_1} - \overline{x_2})}$$

Det tilsvarende konfidensintervallet (CI) for μ_1-μ_2 blir:

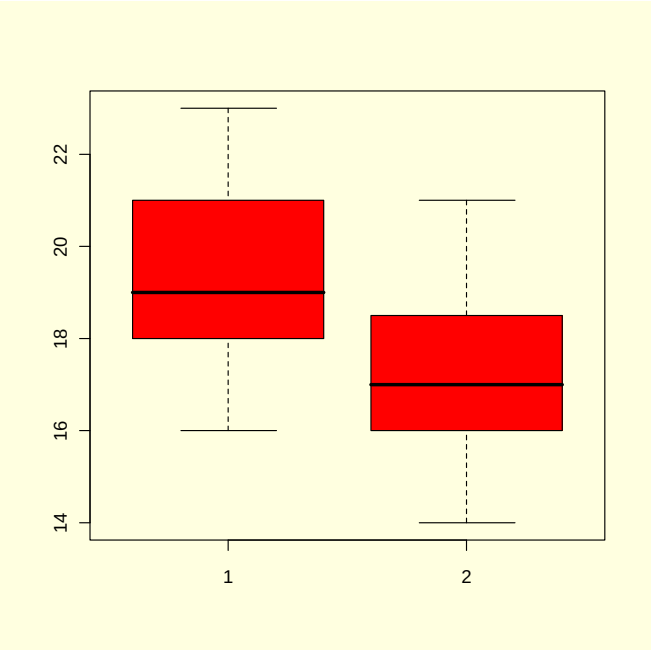
$$CI = (\overline{x_1} - \overline{x_2}) \mp t \cdot \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

Hvis standardavviket (σ)er kjent på forhånd bruker vi z-verdien:

$$z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

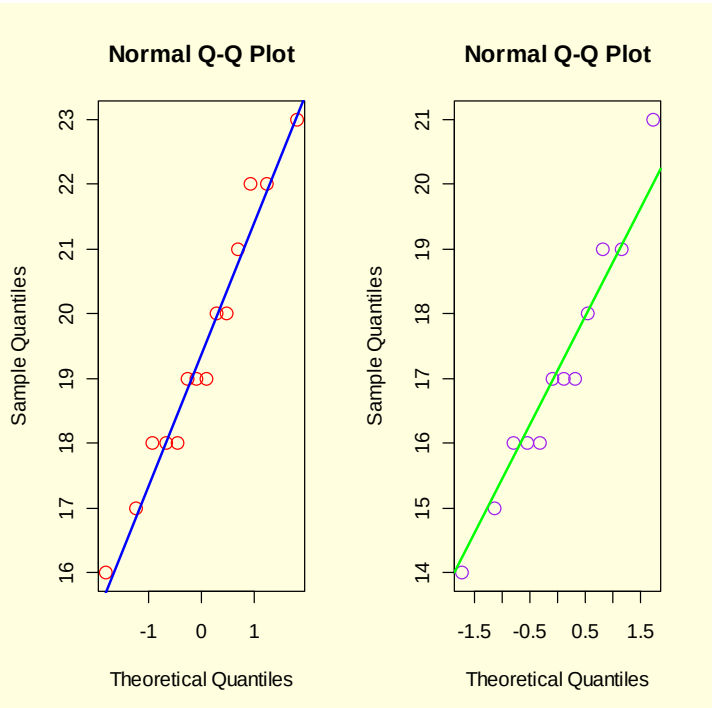
Vi har to prøver A og B som vi ønsker å undersøke om kommer fra samme populasjon. Vi lager først en grafisk framstilling som indikerer at det kan være en signifikant forskjell:

```
A<-c(19, 22, 21, 22, 23, 20, 18, 17, 19, 16, 18, 19, 20, 18)
B<-c(17, 16, 15, 17, 14, 21, 16, 19, 16, 19, 17,
18)par(bg="lightyellow")
boxplot(A, B, col="red")
```



Vi ser om dataene er normalfordelte:

```
par(mfrow=c(1,2),bg="lightyellow")
qqnorm(A,col="red",cex=1.5)
qqline(A,col="blue",lwd=2)
qqnorm(B,col="purple",cex=1.5)
qqline(B,col="green",lwd=2)
```



Generelt er oppsummerende statistikk nyttig:

```
summary(A)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  16.00   18.00   19.00   19.43   20.75   23.00
```

Vi må undersøke om variansen fra de to prøvene er like og gjør en F-test. Konstant varianse (**homoskedastisitet**)er en viktig forutsetning for mange tester, inkludert ANOVA og regresjon.

```
var.test(A,B)
```

```
        F test to compare two variances
data:  A and B
F = 1.1049, num df = 13, denom df = 11, p-value = 0.8791
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3257629 3.5329109
sample estimates:
ratio of variances
          1.104899
```

Som viser at de to variansene er like (p=0.8791)
Vi kan også teste for normalfordeling med Shapiro-Wilk normalitetstest:

```
shapiro.test(A)
        Shapiro-Wilk normality test
data:  A
W = 0.9648, p-value = 0.8003
```

Vi gjør en t-test:

```
t.test(A,B)
        Welch Two Sample t-test
data:  A and B
t = 3.0188, df = 23.708, p-value = 0.005985
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.7407978 3.9496784
sample estimates:
mean of x mean of y
 19.42857  17.08333
```

Noe som indikerer signifikant forskjell mellom prøve A og B (p=0.005985). Vi kan også regne ut *t* etter formelen over:

```
t<-(mean(A)-mean(B))/sqrt((var(A)/length(A))+(var(B)/length(B)));t
[1] 3.018801
```

Fortegnet på *t* har ingen betydningen når vi tester signifikans. Tabellverdien for kritisk verdi av t:

```
qt(0.975,24)
[1] 2.063899
```

Sannsynligheten for å få en t-verdi vi har fått hvis nullhypotesen H_0:A=B er riktig, det vil si vi forkaster nullhypotesen og beholder den alternative hypotesen om at A er forskjellig fra B:

```
pt<-pt(3.018801,24);pt
[1] 0.9970328
```

Noe som tilsvarer 99.7% av arealet under t-fordelingskurven. Sannsynligheten for t-verdien vi har fått (tohalet, *2):

```
2*(1-pt)
[1] 0.005934367
```

Har vi gitt en avstand fra middeltallet e.g. ±1.96·SE kan vi finne ut hvor stor del av arealet under normalfordelingskurven dette tilsvarer, dvs. 95%

```
pnorm(1.96)-pnorm(-1.96)
[1] 0.9500042
```

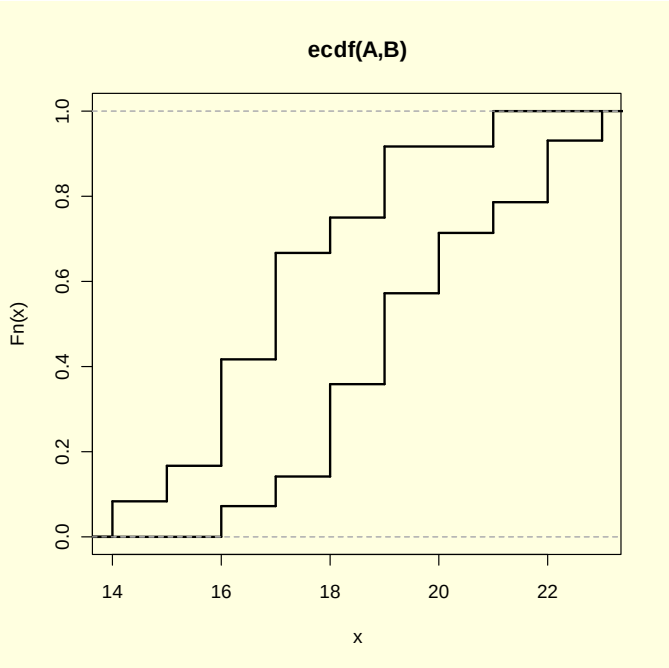
Har vi gitt et areal under normalfordelingskurven kan vi beregne hvilke grense dette tilsvarer e.g. p=0.975 (vi ser på bare den ene siden av normalfordelingskurven):

```
 qnorm(0.975)
[1] 1.959964
```
Vi kan gjøre tilsvarende for en t-fordeling med df=5 og ±2.57
```
 pt(2.57,5)-pt(-2.57,5)
[1] 0.9499647
 qt(0.975,5)
[1] 2.570582
```

Vi kan gjøre en to-prøve t-test, og ser at prøve A og B er signifikant forskjellige p=0.006107:
```
 t.test(A, B, var.equal=TRUE)
        Two Sample t-test
data:  A and B
t = 3.0068, df = 24, p-value = 0.006107
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.7354276 3.9550486
sample estimates:
mean of x mean of y
 19.42857  17.08333
```

t-test forutsetter normalfordeling. Ikke-parametriske tester som Wilcoxon rangeringsumtest forutsetter ikke normalfordeling, men at prøven kommer fra en kontinuerlig fordeling.
```
 wilcox.test(A, B)
        Wilcoxon rank sum test with continuity correction
data:  A and B
W = 135, p-value = 0.00876
alternative hypothesis: true location shift is not equal to 0
Warning message:
In wilcox.test.default(A, B) : cannot compute exact p-value with ties
```
Vi ser at vi får en advarsel som indikerer at de to prøvene kommer fra to diskrete fordelinger.
I Kolmogorov-Smirnov test ks.test(x,y)undersøker man nullhypotesen om at x og y kommer fra samme kontinuerlige fordeling. For A og B blir såvidt p>0.05, og vi beholder nullhypotesen
```
 ks.test(A,B)
        Two-sample Kolmogorov-Smirnov test
data:  A and B
D = 0.5238, p-value = 0.05769
alternative hypothesis: two-sided
Warning message:
In ks.test(A, B) : cannot compute correct p-values with ties
```

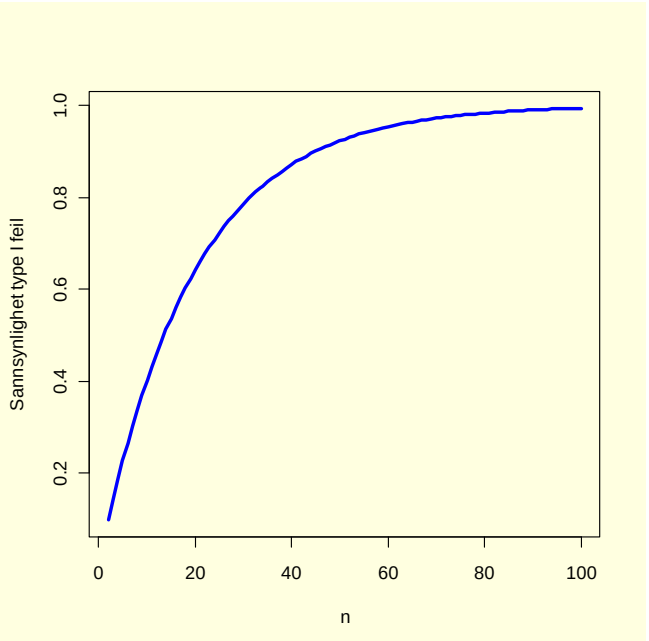
Vi kan også plotte den empiriske kumulative fordelingsfunksjonen **ecdf()** for de to prøvene A og B, se **?ecdf.**
```
par(bg="lightyellow")
plot(ecdf(A), do.points=FALSE,
lwd=2,verticals=TRUE,xlim=range(A, B),main="ecdf(A,B)")
plot(ecdf(B), do.points=FALSE, lwd=2,verticals=TRUE,add=TRUE)
```



Muligheten for å gjøre en type I feil øker med antall prøver:

$$1-(1-\alpha)^n$$

```
n<-seq(2,100,1)
feil<-1-(1-0.05)^n
par(bg="lightyellow")
plot(n,feil,type="l",col="blue",lwd=3, ylab="Sannsynlighet
type I feil")
```



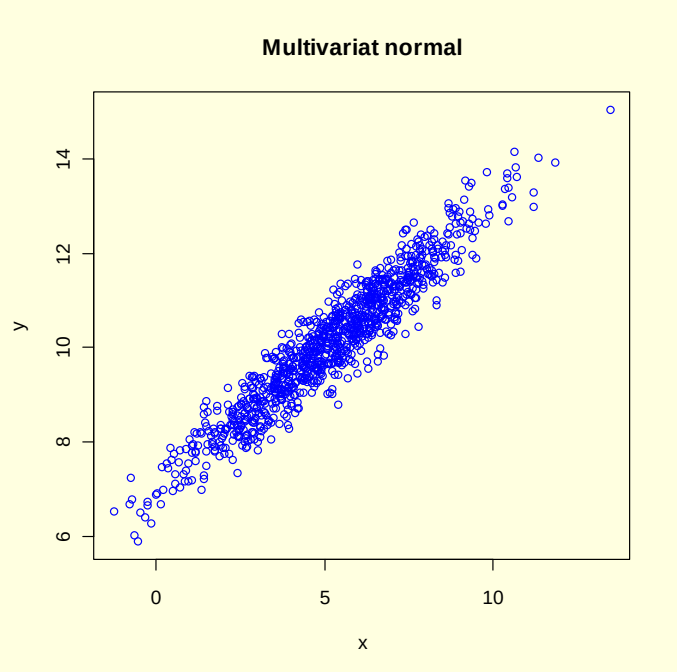
Derfor bruker man istedet ANOVA fremfor en rekke t-tester.

Multivariat normalfordeling

Hvis man ønsker å lage vektorer eller normalfordelte tall som er korrelert med hverandre kan man bruke **mvrnorm (n,mu,sigma)**

fra library(MASS). Den skiller seg fra rnorm ved at man må angi kovariansematrise (sigma) og middelverdi (mu) for hver variabel.

```
library(MASS)
sigma <- matrix(c(5,3,3,2),2,2);sigma
xy<-mvrnorm(1000,mu=c(5,10),sigma)
     [,1] [,2]
[1,]    5    3
[2,]    3    2
x<-xy[,1]
y<-xy[,2]
par(bg="lightyellow")
plot(x,y,col="blue",xlab="x",ylab="y")
```



Ved å se på variansen kan vi sammenligne denne med kovariansematrisen og ser at det er god overensstemmelse, kovarianse=3 og varianse henholdsvis 5 og 2.

```
var(xy)
         [,1]     [,2]
[1,] 4.969884 2.978175
[2,] 2.978175 1.976354
```

Kovariansen til x og y (cov(x,y))henger sammen med **korrelasjonskoeffisienten ρ** og de to variansene for x og y som følgende:

$$cov(x,y) = \rho \cdot \sqrt{s_x^2 s_y^2}$$

```
cor(x,y)*sqrt(var(x)*var(y))
[1] 2.978175
```

Som stemmer godt med kovariansen=3
Hvis x og y hadde vært uavhengig av hverandre hadde variansen til summen av de to variablene vært lik summen til de to

variansene, noe de ikke er siden x og y er korrelert med hverandre:

```
 var(x+y)
[1] 12.90259
 var(x)+var(y)
[1] 6.946238
```

Vi ser at summen at de to variansene er ca. 7 (5+2) som vi satte i kovariansematrisen.

```
 var(x)
[1] 4.969884
 var(y)
[1] 1.976354
```

Hvis x og y er to uavhengige variable forventer vi

$$var(x+y) = var(x) + var(y) + 2 \cdot cov(x,y)$$

I vårt tilfelle for x og y er kovariable får vi:

```
 var(x)+var(y)*2*cov(x,y)
[1] 16.8218
```

Vår kovarianse-matrise er:

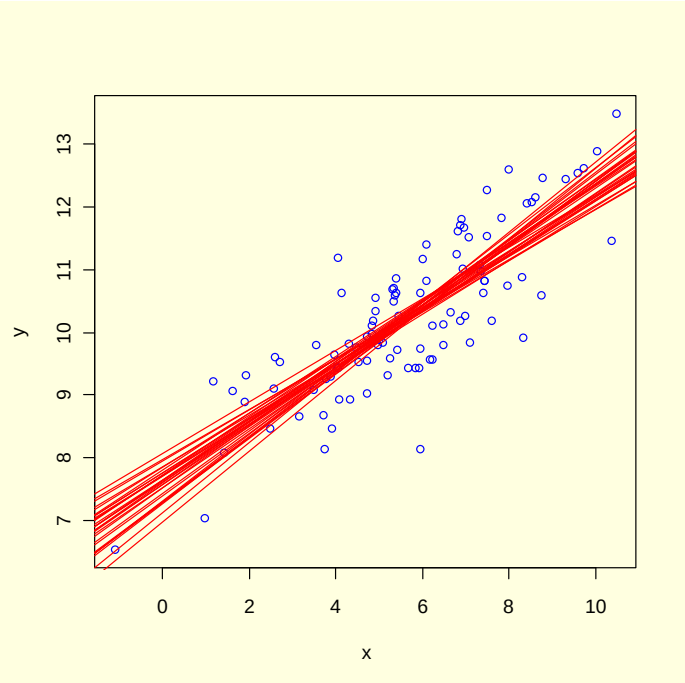
$$\begin{pmatrix} var(x) & cov(x,y) \\ cov(x,y) & var(y) \end{pmatrix}$$

Generelt blir variansene liggende på diagonalen i kovarianse-matrisen, og kovariansene utenfor diagonalen:

$$\begin{pmatrix} var(x_1) & cov(x_1,x_2) & \dots & cov(x_1,x_n) \\ cov(x_1,x_2) & var(x_2) & \dots & cov(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(x_1,x_n) & cov(x_2,x_n) & \dots & var(x_n) \end{pmatrix}$$

Vi kan bruke boot-strap for å lage forskjellige regresjonslinjer gjennom den multivariate fordelingen:

```
sigma <- matrix(c(6,3,3,2),2,2);sigma
xy<-mvrnorm(100,mu=c(5,10),sigma)
x<-xy[,1]
y<-xy[,2]
par(bg="lightyellow")
plot(x,y,col="blue",xlab="x",ylab="y")
for (i in 1:30) {
n<-sample(1:length(x),replace=TRUE)
xn<-x[n]
yn<-y[n]
abline(lm(yn~xn),col="red")
}
```



Vi har normalfordelingsfunksjonen for flere variable:

$$f(x_1, x_2, \dots, x_n; \mu, \sigma) = \frac{1}{\left(\sigma\sqrt{2 \cdot \pi}\right)^n} \cdot \prod_{i=1}^{n} exp\left(-\frac{1}{2} \cdot \left(\frac{x_i - \mu}{\sigma}\right)^2\right)$$

Kjikvadrat-fordeling

En variabel *X* har kjikvadratfordeling med *n* frihetsgrader hvis den kan uttrykkes som:

$$X = Z_1^2 + Z_2^2 + \cdots + Z_n^2$$

hvor $Z_1, Z_2, .. Z_n$ er uavhengige tilfeldige variable med normalfordeling.
Forventet middelverdi for kjikvadratfordelingen $E(X) = n$ og forventet varianse $var(X) = 2n$.
Varianse som følger normalfordeling er en skalert utgave av kjikvadratfordelingen og beskrives av en sum av kvadrerte ledd. Kjikvadratfordelingen brukes til å finne konfidensintervallet for variansen.
Variansen s^2 til en prøve er lik:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Vi har et estimat av populasjonsvariansen $E(s^2) = \sigma^2$.
Forholdet mellom varianse og estimatet av populasjonsvariansen ganger *n-1* frihetsgrader følger ca. en kjikvadratfordeling.

$$\chi^2_{n-1} = \frac{(n-1)\cdot s^2}{\sigma^2}$$

Tetthetsfunksjonen for kjikvadratfordelingen er en avart av gammafordelingen med formparameter *a=n/2* og skalarparameter *λ=b=1/2*

$$y = f(x) = \frac{\left(\frac{1}{2}\right)^{\frac{1}{2}}}{\Gamma\left(\frac{n}{2}\right)} \cdot x^{\left(\frac{n}{2}-1\right)} \cdot e^{-\frac{x}{2}}$$

hvor *Γ* er gammafunksjonen, *n* er middeltallet lik antall frihetsgrader og variansen er *2n*. Kjikvadratfordelingen er bare bestemt av en parameter, *n*. F-fordelingen som brukes i ANOVA er uttrykt som forholdet mellom to kjikvadratfordelinger.

Ikke-sentral chikvadratfordeling har en ikke-sentral parameter ncp=λ, som er lik summen av kvadratet av 3 normalfordelte middeltall.

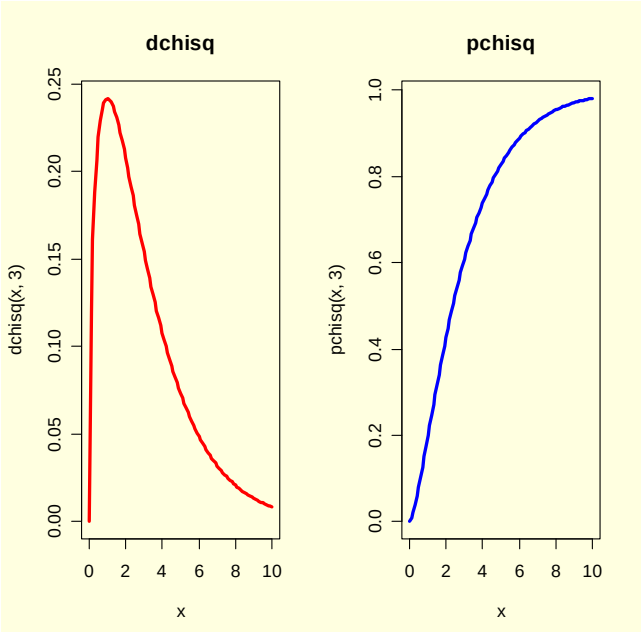
Hvis for eksempel variansen s^2 er 12.3 med df=10 blir konfidensintervallet for σ2:

```
 12.3*9/qchisq(0.975,9)
[1] 5.819342
 12.3*9/qchisq(0.025,9)
[1] 40.99409
```

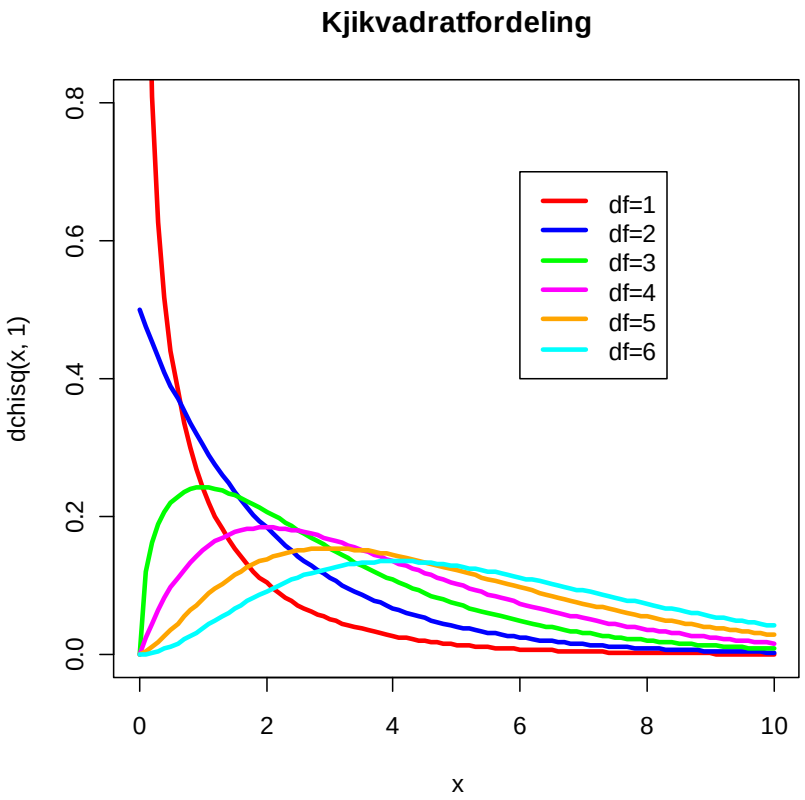
$5.81 \leq \sigma^2 \leq 40.99$

Figur som viser tetthetsfunksjon og kumulativ sannsynlighet for kjikvadratfordeling med df=3 og uten ncp:

```
x<-seq(0,10,0.1)
par(mfrow=c(1,2),bg="lightyellow")
plot(x,dchisq(x,3),type="l",col="red",lwd=3,main="dchisq")
plot(x,pchisq(x,3),type="l",col="blue",lwd=3,main="pchisq")
```



```
x<-seq(0,10,0.1)
curve(dchisq(x,1),col="red",
lwd=3,ylim=c(0,0.8),main="Kjikvadratfordeling")
curve(dchisq(x,2),add=T,col="blue",lwd=3)
curve(dchisq(x,3),add=T,col="green",lwd=3)
curve(dchisq(x,4),add=T,col="magenta",lwd=3)
curve(dchisq(x,5),add=T,col="orange",lwd=3)
curve(dchisq(x,6),add=T,col="cyan",lwd=3)
legend(6,0.7,c("df=1","df=2","df=3","df=4","df=5","df=6"),lty=
c(1,1,1,1,1,1),
lwd=c(3,3,3,3,3,3),col=c("red","blue","green","magenta","orang
e","cyan"))
```



Det er to hovedtyper kji-kvadrattester, for forskjeller og for assosiasjoner. I en kji-kvadrat ($\chi 2$) test for forskjeller undersøkes sannsynligheten for en observert frekvens (O) fra en forventet frekvens (E)

$$\chi^2 = \sum \frac{(Observert - forventet)^2}{forventet} = \sum \frac{(O-E)^2}{E}$$

Denne fordelingen kalles **Pearson kjikvadratstatistikk** (Karl Pearson 1857-1936) og følger kjikvadratfordeling. Kjikvadrattest brukes for å teste hypotesen om uavhengige variable i *2x2* kontingenstabeller, et mål på hvor mye observert frekvens er forskjellig fra forventet frekvens. Dataene er presentert i en *RxC* tabell med *R* rader og *C* kolonner. En **fisher.test** benyttes på 2x2 tabeller (sjekk

forutsetninger). For større kontingenstabeller benyttes **chisq.test**. Antall frihetsgrader er (R-1)·(C-1).

Kji-kvadrattest gjøres med kommandoen **chisq.test()** på en matrise: **chisq.test(x,p=…)** hvor p er sannsynlighet, og tabellen er matrisen x. Hvis dataene er lagret som to variable y og z: **chisq.test(y,z)** eller **chisq.test(table(y,z))**.
Med tilleggsargumentet simulate.p.value=TRUE gjøres en Monte Carlo simulering.

Korrelasjon mellom to prøver med **cor.test**

Kommandoen **ks.test** gjør en Kolmogorow-Smirnov rangeringstest: **ks.test(x,y="name",…)**.

Brukes til å undersøke data for eksempel fra en dihybrid krysning om disse avviker fra forventet ratio 9:3:3:1

```
 dihybrid<-matrix(c(82,29,37,11,90,30,30,10),nrow=2,byrow=T)
 colnames(dihybrid)<-c("AABB","AAbb","aaBB","aabb")
 rownames(dihybrid)<-c("Observert","Forventet")
 dihybrid
          AABB AAbb aaBB aabb
Observert   82   29   37   11
Forventet   90   30   30   10

 chisq.test(dihybrid)

        Pearson's Chi-squared test

data:  dihybrid
X-squared = 1.1649, df = 3, p-value = 0.7614
```

Det er ingen signifikant forskjell mellom observert og forventet frekvens, siden chi-kvadrat er mindre enn den kritiske verdien med 3d.f.

I Mendels klassiske eksperiment med dihybrid krysning og to uavhengige 3:1 segregasjoner av alleler ga dette 556 erter med følgende fordeling: 315 gule runde erter, 108 grønne runde erter, 101 gule rynkete erter og 32 grønne rynkete erter. Følger dette forholdet 9:3:3:1 ?

```
 erter<-c(315,108,101,32);erter
[1] 315 108 101  32
 chisq.test(erter,p=c(9,3,3,1),rescale.p=TRUE)

       Chi-squared test for given probabilities

data:  erter
X-squared = 0.47, df = 3, p-value = 0.9254
```

Det vil si at forholdet 315:108:101:32 er ikke signifikant forskjellig fra 9:3:3:1
Forventede frekvenser:

```
 forventet<-556*c(9,3,3,1)/16;forventet
```

```
[1] 312.75 104.25 104.25  34.75
```

Vi kan også lage en 2x2 matrise av datasettet og bruke Fisher:

```
erter2<-matrix(c(315,101,108,32),nrow=2)
colnames(erter2)<-c("gule","grønne")
rownames(erter2)<-c("runde","rynket");erter2
       gule grønne
runde   315    108
rynket  101     32
 fisher.test(erter2)
        Fisher's Exact Test for Count Data
data:  erter2
p-value = 0.819
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.5667874 1.4806148
sample estimates:
odds ratio
 0.9242126
```

Eller vi kan gjøre en kji-kvadrattest med Yates korreksjon:

```
 chisq.test(erter2)
        Pearson's Chi-squared test with Yates' continuity correction
data:  erter2
X-squared = 0.0513, df = 1, p-value = 0.8208
```

Det blir en liten forskjell i p-verdi for de to testene, men hovedkonklusjonen er det samme, Mendels uavhengige sortering av farge og form, hver i forholdet 3:1 og totalt 9:3:3:1, og det er ingen interaksjon mellom farge og form.

Log-lineære modeller kan benyttes hvis man har enkle kontingenstabeller. Vi lager faktorer for farge og form. Lager deretter to modeller, med (mettet modell) og en modell uten interaksjonsledd). Vi ser at i den mettede modellen er residual deviansen lik 0. Deviansen, dataavik for den tilpassete modellen, er forskjellig for de forskjellige GLM. Devianse defineres generelt som -2*loglikelihood(nullmodell minus mettet modell med maksimal tilpasning). For en log-lineær modell er:

$$devianse = 2\sum O \cdot ln\left(\frac{O}{E}\right)$$

Vi lager en maksimalmodell (model1) og en minimalmodell (modell2):

```
farge<-factor(c("gule","grønne","gule","grønne"));farge
form<-factor(c("runde","runde","rynket","rynket"));form
model1<-glm(erter~farge*form,poisson)
summary(model1)
Call:
glm(formula = erter ~ farge * form, family = poisson)
Deviance Residuals:
[1]  0  0  0  0
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        4.68213    0.09623  48.658  < 2e-16 ***
fargegule          1.07044    0.11151   9.600  < 2e-16 ***
```

```
formrynket           -1.21640    0.20127  -6.044 1.51e-09 ***
fargegule:formrynket  0.07894    0.23148   0.341    0.733
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for poisson family taken to be 1)
    Null deviance: 3.0239e+02  on 3  degrees of freedom
Residual deviance: 1.5543e-14  on 0  degrees of freedom
AIC: 33.876
Number of Fisher Scoring iterations: 2
 model2<-glm(erter~farge+form,poisson)
 summary(model2)
Call:
glm(formula = erter ~ farge + form, family = poisson)
Deviance Residuals:
       1         2         3         4
-0.08378   0.14396   0.14892  -0.25928
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.66825    0.08780   53.17   <2e-16 ***
fargegule    1.08904    0.09771   11.15   <2e-16 ***
formrynket  -1.15702    0.09941  -11.64   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)
    Null deviance: 302.38754  on 3  degrees of freedom
Residual deviance:   0.11715  on 1  degrees of freedom
AIC: 31.993
Number of Fisher Scoring iterations: 3
 anova(model1,model2,test="Chi")
Analysis of Deviance Table
Model 1: erter ~ farge * form
Model 2: erter ~ farge + form
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         0  1.554e-14
2         1    0.11715 -1 -0.11715   0.73215
```

Det er ingen interaksjon mellom form og farge (p=0.73215), noe som vi forventer fra uavhengig sortering av alleler.
Ifølge koeffisienttabellen for log-link (Poisson) har vi:
log(erter)~ 4.66825+1.08904*farge-1.15702*form.
Vi bruker eksponentialfunksjonen på begge sider:
Erter ~$e^{4.66825}$ +$e^{1.08904}$*farge $e^{-1.15702}$*form.
Erter~106.5112 + 2.97142*farge + 3.180441*form.
Det er et lite avvik fra 3:1, nemlig forhold gule-grønne (farge) 2.97:1 og runde-rynket (form) 3.18:1

Chi-kvadrat test om det er forskjell i hårfarge hos menn og kvinner ut fra følgende datasett, hvor vi a priori ikke har kjennskap til forventet frekvens:

```
 haar<-matrix(c(19,31,24,25),nrow=2,byrow=T)
 colnames(haar)<-c("Blond","Brun")
rownames(haar)<-c("Menn","Kvinner")
 haar
        Blond Brun
Menn       19   31
Kvinner    24   25
```

```
chisq.test(haar)
```

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  haar
X-squared = 0.8085, df = 1, p-value = 0.3686
```

Det er ingen relasjon eller interaksjon mellom hårfarge og kjønn.
Vi kan vise forventet frekvens, $expected, eller $observed:

```
chisq.test(haar)$expected
```

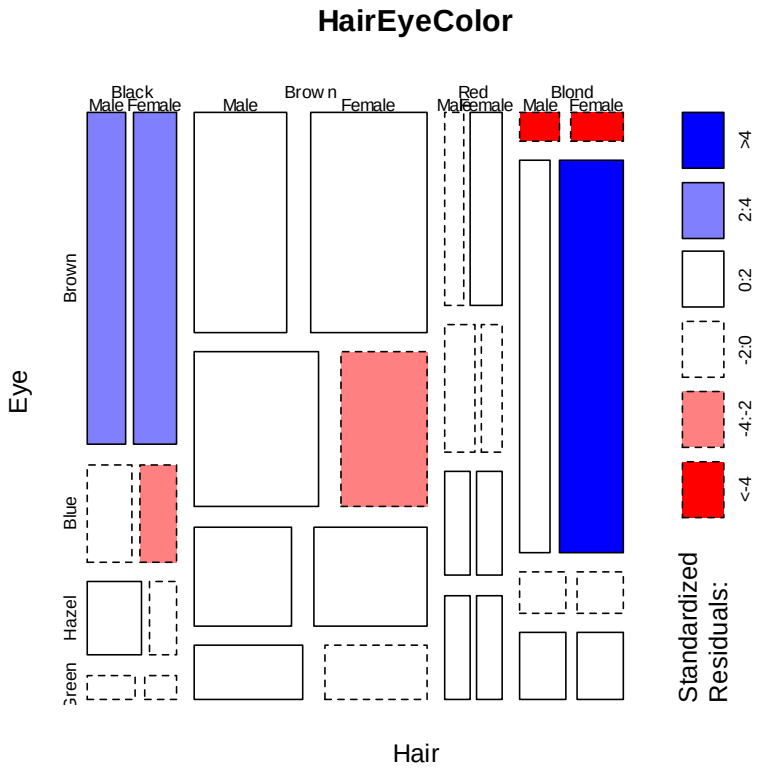
```
          Blond     Brun
Menn    21.71717 28.28283
Kvinner 21.28283 27.71717
```

Kommandoen **mosaicplot()**kan brukes for kontingenstabeller, for eksempel sammenhengen mellom øye- og hårfarge hentet fra library(datasets)

```
library(datasets)
data()
 HairEyeColor
```

```
, , Sex = Male
       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female
       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

```
mosaicplot(HairEyeColor,shade=TRUE,color=TRUE)
```



Plottet viser at det er signifikant flere blåøyde blonde kvinner og færre brunøyde blonde kvinner enn forventet hvis uavhengighet.
Vi kan slå isammen de to tabellene for menn og kvinner til en tabell med **margin.table():**

```
 farge<-margin.table(HairEyeColor,c(1,2));farge
```

```
       Eye
Hair    Brown Blue Hazel Green
  Black    68   20    15     5
  Brown   119   84    54    29
  Red      26   17    14    14
  Blond     7   94    10    16
```

```
 chisq.test(farge)
```

```
        Pearson's Chi-squared test

data:  farge
X-squared = 138.2898, df = 9, p-value < 2.2e-16
```

Hårfarge	Øyefarge				
	Brun	Blå	Brungrønn	Grønn	Radtotal
Sort	68	20	15	5	108
Brun	119	84	54	29	286
Rød	27	17	14	14	72
Blond	7	94	10	16	127

Kolonnetotal	221	215	93	64	593

Forventet frekvens

Hårfarge	Øyefarge				
	Brun	Blå	Brungrønn	Grønn	Rad
Sort	(108·221)/593	(108·215)/593	(108·93)/593	(108·64)/593	108
Brun	(286·221)/593	(286·215)/593	(286·93)/593	(286·64)/593	286
Rød	(72·221)/593	(72·215)/593	(72·93)/593	(72·64)/593	72
Blond	(127·221)/593	(127·215)/593	(127·93)/593	(127·64)/593	127
Kolonne	221	215	93	64	593

Forventet frekvens:

Hårfarge	Øyefarge				
	Brun	Blå	Brungrønn	Grønn	Rad
Sort	40.24958	39.15683	16.93761	11.65599	108
Brun	106.5868	103.6931	44.85329	30.86678	286
Rød	26.83305	26.10455	11.29174	7.770658	72
Blond	47.33052	46.04553	19.91737	13.70658	127
Kolonne	221	215	93	64	593

Forventet frekvens:

$$Forventet\ frekvens = \frac{Radtotal \cdot kolonnetotal}{Stortotal}$$

Tabell over $(O-E)^2/E$

Hårfarge	Øyefarge				
	Brun	Blå	Brungrønn	Grønn	Rad
Sort	19.13277	9.372162	0.2216566	3.80081	108
Brun	1.445653	3.740058	1.865243	0.1129003	286
Rød	0.001038730	3.175417	0.6495608	4.993747	72
Blond	34.36579	49.94255	4.938113	0.3837409	127
Kolonne	221	215	93	64	593

```
19.13277+1.445653+0.001038730+34.36579+9.372162+3.740058+3.175417+
49.94255+0.2216566+1.865243+0.6495608+4.938113+3.80081+0.1129003+
4.993747+0.3837409
[1] 138.1412
```

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Hvor O er observert og E er forventet frekvens.
I kontingenstabeller er antall frihetsgrader df:

$$df = (rader - 1) \cdot (kolonner - 1)$$

For en 4x4 kontingenstabell blir df=9

```
qchisq(0.95,9)
[1] 16.91898
```

Vår teststatistikk 138.1412 er større enn tabellverdien og vi forkaster nullhypotesen, det vil si det er interaksjon mellom hårfarge og øyefarge.

Vi lager kolonner av kontingenstabellen:

```
farge2<-as.data.frame.table(farge)
 farge2
   Hair   Eye Freq
```

```
1  Black Brown   68
2  Brown Brown  119
3    Red Brown   26
4  Blond Brown    7
5  Black  Blue   20
6  Brown  Blue   84
7    Red  Blue   17
8  Blond  Blue   94
9  Black Hazel   15
10 Brown Hazel   54
11   Red Hazel   14
12 Blond Hazel   10
13 Black Green    5
14 Brown Green   29
15   Red Green   14
16 Blond Green   16
```

```
attach(farge2)
model<-glm(Freq~Hair+Eye,family=poisson)
summary(model)
```

```
Call:
glm(formula = Freq ~ Hair + Eye, family = poisson)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.3263  -2.0646  -0.2120   1.2353   6.1724

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.69225    0.11007  33.544  < 2e-16 ***
HairBrown    0.97386    0.11294   8.623  < 2e-16 ***
HairRed     -0.41945    0.15279  -2.745  0.00604 **
HairBlond    0.16206    0.13089   1.238  0.21569
EyeBlue     -0.02299    0.09590  -0.240  0.81054
EyeHazel    -0.86103    0.12369  -6.961 3.37e-12 ***
EyeGreen    -1.23474    0.14202  -8.694  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 453.31  on 15  degrees of freedom
Residual deviance: 146.44  on  9  degrees of freedom
AIC: 241.04

Number of Fisher Scoring iterations: 5
```

Vi kan prediktere forventede verdier fra modellen:

```
yv<-predict(model,type="response");yv
```

```
       1         2         3         4         5         6         7
 40.135135 106.283784  26.385135  47.195946  39.222973 103.868243  25.785473
        8         9        10        11        12        13        14
 46.123311  16.966216  44.929054  11.153716  19.951014  11.675676  30.918919
       15        16
  7.675676  13.729730
```

Disse stemmer godt overens med tallene i tabellen over for forventet frekvens.

Imidlertid ser vi at Residual deviance 146.44 er mye større enn antall frihetsgrader, noe som indikerer overdispersering. Vi starter med det opprinnelige datasettet og lager kolonner av dette:

```
data(HairEyeColor)
```

```
farge3<-as.data.frame.table(HairEyeColor)
farge3
```

```
   Hair   Eye    Sex Freq
1  Black Brown   Male   32
2  Brown Brown   Male   53
3    Red Brown   Male   10
4  Blond Brown   Male    3
5  Black  Blue   Male   11
6  Brown  Blue   Male   50
7    Red  Blue   Male   10
8  Blond  Blue   Male   30
9  Black Hazel   Male   10
10 Brown Hazel   Male   25
11   Red Hazel   Male    7
12 Blond Hazel   Male    5
13 Black Green   Male    3
14 Brown Green   Male   15
15   Red Green   Male    7
16 Blond Green   Male    8
17 Black Brown Female   36
18 Brown Brown Female   66
19   Red Brown Female   16
20 Blond Brown Female    4
21 Black  Blue Female    9
22 Brown  Blue Female   34
23   Red  Blue Female    7
24 Blond  Blue Female   64
25 Black Hazel Female    5
26 Brown Hazel Female   29
27   Red Hazel Female    7
28 Blond Hazel Female    5
29 Black Green Female    2
30 Brown Green Female   14
31   Red Green Female    7
32 Blond Green Female    8
```

```
attach(farge3)
```

Vi lager først en maksimal modell, og forsøker deretter å fjerne ledd i modellen som er ikke-signifikante.

```
model3<-glm(Freq~Hair*Eye*Sex,family=poisson)
summary(model3)
```

```
Call:
glm(formula = Freq ~ Hair * Eye * Sex, family = poisson)

Deviance Residuals:
 [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
[26]  0  0  0  0  0  0  0

Coefficients:
                            Estimate Std. Error z value Pr(>|z|)
(Intercept)                  3.46574    0.17678  19.605  < 2e-16 ***
HairBrown                    0.50456    0.22387   2.254 0.024210 *
HairRed                     -1.16315    0.36228  -3.211 0.001325 **
HairBlond                   -2.36712    0.60381  -3.920 8.84e-05 ***
EyeBlue                     -1.06784    0.34951  -3.055 0.002249 **
EyeHazel                    -1.16315    0.36228  -3.211 0.001325 **
EyeGreen                    -2.36712    0.60381  -3.920 8.84e-05 ***
SexFemale                    0.11778    0.24296   0.485 0.627825
HairBrown:EyeBlue            1.00957    0.40128   2.516 0.011874 *
HairRed:EyeBlue              1.06784    0.56759   1.881 0.059923 .
HairBlond:EyeBlue            3.37043    0.69916   4.821 1.43e-06 ***
HairBrown:EyeHazel           0.41173    0.43603   0.944 0.345021
HairRed:EyeHazel             0.80648    0.61164   1.319 0.187323
HairBlond:EyeHazel           1.67398    0.81522   2.053 0.040033 *
HairBrown:EyeGreen           1.10488    0.67091   1.647 0.099590 .
HairRed:EyeGreen             2.01045    0.77938   2.580 0.009893 **
HairBlond:EyeGreen           3.34795    0.90715   3.691 0.000224 ***
HairBrown:SexFemale          0.10158    0.30504   0.333 0.739128
HairRed:SexFemale            0.35222    0.47067   0.748 0.454253
HairBlond:SexFemale          0.16990    0.80147   0.212 0.832120
EyeBlue:SexFemale           -0.31845    0.51093  -0.623 0.533098
EyeHazel:SexFemale          -0.81093    0.59919  -1.353 0.175935
EyeGreen:SexFemale          -0.52325    0.94465  -0.554 0.579642
HairBrown:EyeBlue:SexFemale -0.28657    0.58692  -0.488 0.625365
```

```
HairRed:EyeBlue:SexFemale    -0.50822    0.81634  -0.623 0.533569
HairBlond:EyeBlue:SexFemale   0.78846    0.94517   0.834 0.404169
HairBrown:EyeHazel:SexFemale  0.73999    0.68376   1.082 0.279150
HairRed:EyeHazel:SexFemale    0.34093    0.89847   0.379 0.704351
HairBlond:EyeHazel:SexFemale  0.52325    1.15860   0.452 0.651543
HairBrown:EyeGreen:SexFemale  0.23489    1.03173   0.228 0.819905
HairRed:EyeGreen:SexFemale    0.05324    1.15783   0.046 0.963321
HairBlond:EyeGreen:SexFemale  0.23557    1.31366   0.179 0.857686
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance:  4.7512e+02  on 31  degrees of freedom
Residual deviance: -1.5099e-14  on  0  degrees of freedom
AIC: 202.88

Number of Fisher Scoring iterations: 3
```

Dette ble en omfattende modell og vi ønsker å fjerne interaksjonsleddet: Hair:Eye:Sex og sammenligner om det er noen signifikant forskjell med modell 3

```
model4<-update(model3,~.-Hair:Eye:Sex)
anova(model3,model4,test="Chi")
Analysis of Deviance Table

Model 1: Freq ~ Hair * Eye * Sex
Model 2: Freq ~ Hair + Eye + Sex + Hair:Eye + Hair:Sex + Eye:Sex
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         0  -1.51e-14
2         9     6.7613 -9  -6.7613    0.6620
```

Som viser at det ikke er noen signifikant forskjell, og modell 4 er en enklere og bedre modell enn modell 3. Slik kan man arbeide videre med å forenkle modellen til den best mulig beskriver datasettet.

Vi ser også at modell 4 har lavere AIC-verdi:

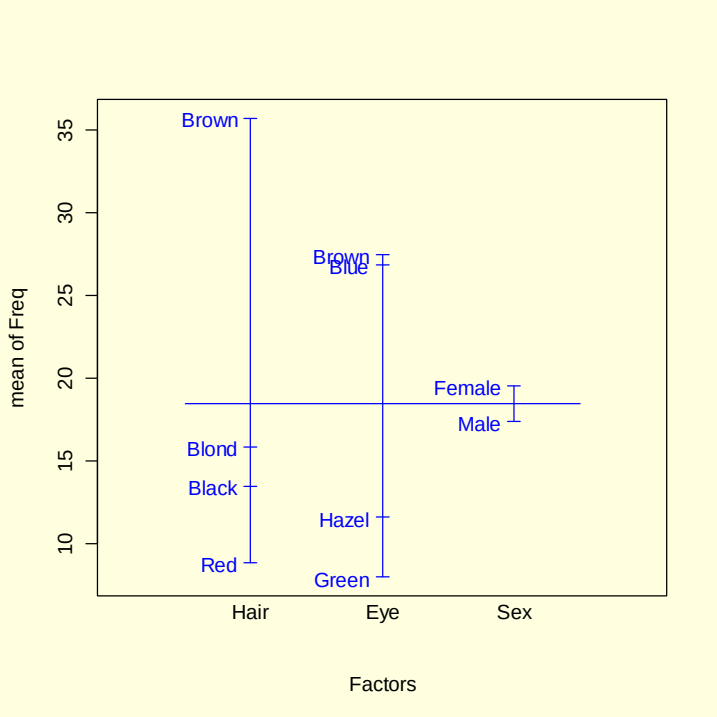
```
 AIC(model3,model4)
       df      AIC
model3 32 202.8787
model4 23 191.6400
 summary(model4)
Call:
glm(formula = Freq ~ Hair + Eye + Sex + Hair:Eye + Hair:Sex +
    Eye:Sex, family = poisson)

Deviance Residuals:
      Min         1Q     Median         3Q        Max
-0.783971  -0.359510   0.001240   0.376772   0.785480

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)         3.49020    0.16122  21.648  < 2e-16 ***
HairBrown           0.47102    0.19552   2.409 0.015995 *
HairRed            -1.11437    0.29033  -3.838 0.000124 ***
HairBlond          -2.83462    0.44346  -6.392 1.64e-10 ***
EyeBlue            -1.02682    0.27264  -3.766 0.000166 ***
EyeHazel           -1.35667    0.30777  -4.408 1.04e-05 ***
EyeGreen           -2.38540    0.47982  -4.971 6.65e-07 ***
SexFemale           0.07106    0.20519   0.346 0.729096
HairBrown:EyeBlue   0.89281    0.29283   3.049 0.002297 **
HairRed:EyeBlue     0.82807    0.40424   2.048 0.040513 *
HairBlond:EyeBlue   3.91224    0.47155   8.296  < 2e-16 ***
HairBrown:EyeHazel  0.73463    0.32984   2.227 0.025932 *
HairRed:EyeHazel    0.91478    0.43851   2.086 0.036967 *
HairBlond:EyeHazel  1.93743    0.57269   3.383 0.000717 ***
HairBrown:EyeGreen  1.21834    0.50854   2.396 0.016586 *
HairRed:EyeGreen    2.02488    0.57160   3.542 0.000396 ***
HairBlond:EyeGreen  3.54294    0.65296   5.426 5.76e-08 ***
HairBrown:SexFemale 0.16460    0.23087   0.713 0.475881
```

```
HairRed:SexFemale    0.27704    0.31302   0.885 0.376119
HairBlond:SexFemale  0.89744    0.29703   3.021 0.002517 **
EyeBlue:SexFemale   -0.42354    0.21763  -1.946 0.051640 .
EyeHazel:SexFemale  -0.32429    0.25192  -1.287 0.197987
EyeGreen:SexFemale  -0.49207    0.29661  -1.659 0.097125 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 475.1180  on 31  degrees of freedom
Residual deviance:   6.7613  on  9  degrees of freedom
AIC: 191.64

Number of Fisher Scoring iterations: 4
```

Vi ser at Redisual deviance er i samme størrelsesorden som antall frihetsgrader noe som indikerer at vi ikke har overdispersjon, og vi trenger heller ikke utforske family=quasibinomial.

Vi kan plotte design på datasettet:

```
par(bg="lightyellow")
plot.design(farge3,col="blue")
```



Vår teststatistikk 138.1412 er større enn tabellverdien og vi forkaster nullhypotesen, det vil si det er interaksjon mellom hårfarge og øyefarge.
Ønsker man å lage tabellen på egen hånd kan dette gjøres som følger:

```
farge3<-c(32,53,10,3,11,50,10,30,10,25,7,5,3,15,7,8,
36,66,16,4,9,34,7,64,5,29,7,5,2,14,7,8)
dim(farge3)<-c(4,4,2)
dimnames(farge3)[[3]]<-list("Male","Female")
dimnames(farge3)[[2]]<-list("Brown","Blue","Hazel","Green")
dimnames(farge3)[[1]]<-list("Black","Brown","Red","Blond")
```

```
farge3
ftable(farge3)
farge4<-as.data.frame.table(farge3)
names(farge4)<-c("Hair","Eye","Sex","n")
farge4
attach(farge4)
```
Deretter kan man gjøre GLM på datasett farge4.

Gammafordelingen

Gamma-fordelingen er avhengig av to parametere form (a) og skala (s) og beskriver ikke-normalfordelte data med positiv skew (lang hale til høyre).
Tetthetsfunksjonen av gammafordelingen:

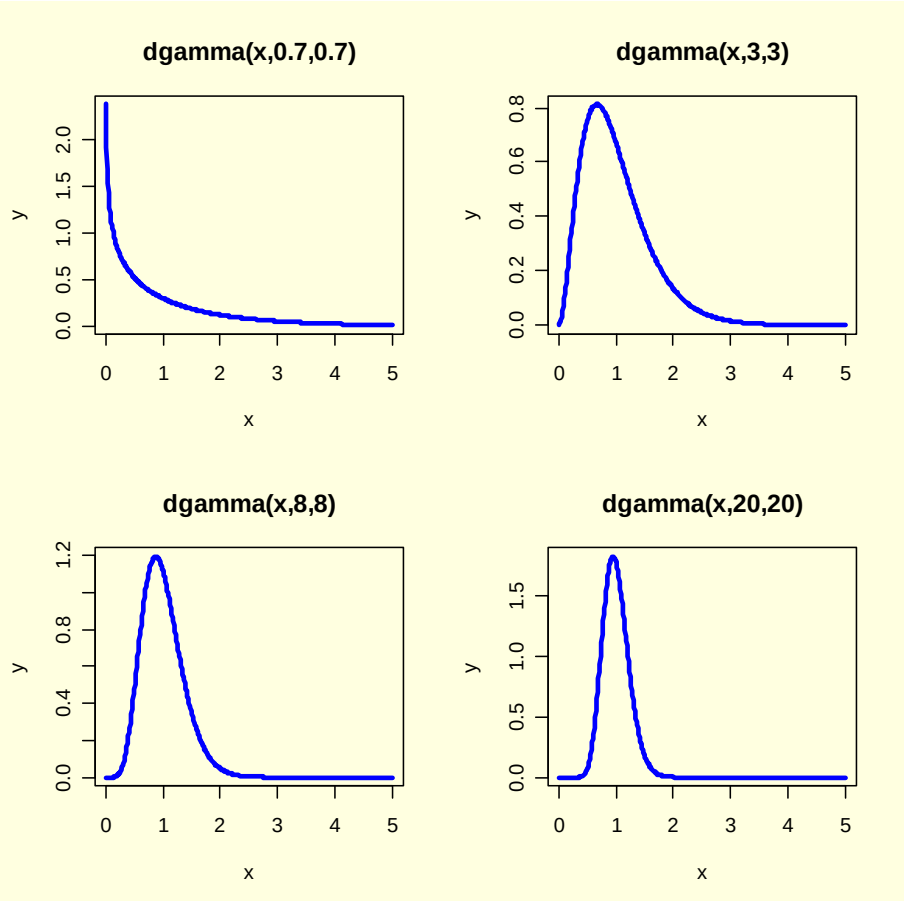
$$f(x) = y = \frac{1}{s^{a \cdot \Gamma a}} \cdot x^{a-1} . e^{-\frac{x}{s}}$$

For detaljer se R-manualen

```
?dgamma
```

Sannsynlighetsfordelingen er gitt ved kommandoen **dgamma**(x,form,rate,skala). Figur som viser hvordan gammafordelingen endrer seg med form og rate. Når verdien for *form* er liten blir kurven L-formet, og når form-verdien øker blir kurven mer klokkeformet:

```
x<-seq(0.01,5,0.01)
par(mfrow=c(2,2),bg="lightyellow")
y<-dgamma(x,0.7,0.7)
plot(x,y,type="l",col="blue",lwd=3,main="dgamma(x,0.7,0.7)")
y<-dgamma(x,3,3)
plot(x,y,type="l",col="blue",lwd=3,main="dgamma(x,3,3)")
y<-dgamma(x,8,8)
plot(x,y,type="l",col="blue",lwd=3,main="dgamma(x,8,8)")
y<-dgamma(x,20,20)
plot(x,y,type="l",col="blue",lwd=3,main="dgamma(x,20,20)")
```



En annen utgave av gammafordelingen er :

$$f(x) = \frac{\alpha^{\beta}}{\Gamma(\beta)} \cdot x^{\beta-1} \cdot e^{-\alpha \cdot x}$$

hvor x er tiden før en hendelse β skjer, og hendelsene skjer med en konstant tidsrate α per tidsenhet. Γ er gammafunksjonen. Når β er lik 1 så blir gammafunksjonen til 1 lik 1 og gammafordelingen blir lik eksponentialfordelingen.
Forventning (gjennomsnitt) i en gammafordeling:

$$E(X) = \frac{\beta}{\alpha}$$

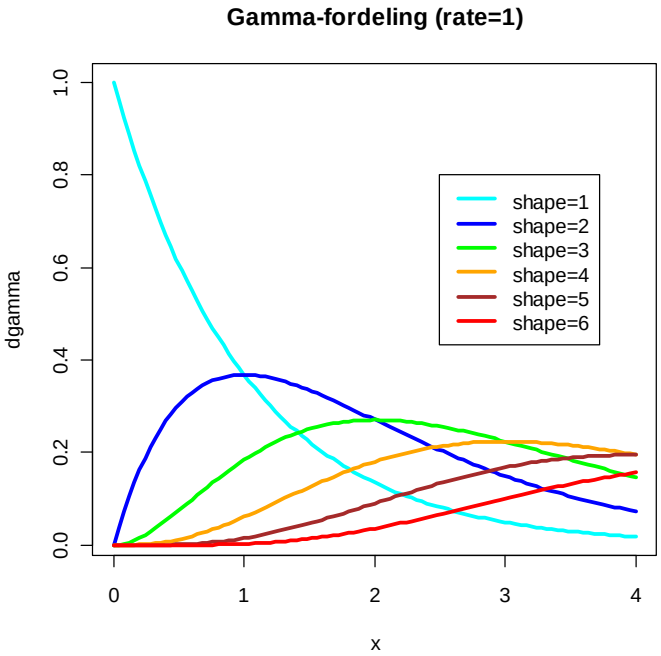
Variansen til en gammafordeling:

$$Var(X) = \frac{\beta}{\alpha^2}$$

Her tilsvarerβ=form og α=skala.
Chikvadrat-fordelingen (form=df/2, middel=df, varianse=2df) og eksponensialfordelingen(middel=s, varianse=s^2, a=1) er spesialtilfeller av gammafordelingen. Rate=1/skala . Skew er 2/√form.
Gammafordeling med forskjellig form og rate=1. Form =1 blir lik eksponensialfunksjonen:

```
x<-seq(0,4,0.1)
curve(dgamma(x,1,1),col="cyan",lwd=3,ylab="dgamma",xlim=c(0,4)
,
main="Gamma-fordeling (rate=1)")
curve(dgamma(x,2,1),add=T,col="blue",lwd=3)
```

```
curve(dgamma(x,3,1),add=T,col="green",lwd=3)
curve(dgamma(x,4,1),add=T,col="orange",lwd=3)
curve(dgamma(x,5,1),add=T,col="brown",lwd=3)
curve(dgamma(x,6,1),add=T,col="red",lwd=3)
legend(2.5,0.8,c("shape=1","shape=2","shape=3","shape=4","shap
e=5","shape=6"),
lty=c(1,1,1,1,1,1), lwd=c(3,3,3,3,3,3),
col=c("cyan","blue","green","orange","brown","red"))
```



Som man ser av figuren kan gammafunksjonen har varierende form og kan derved brukes hvis man skal studere fordelingen av en ukjent parameter. Hvis man skal studere en type atferd 50 ganger hos dyr, og atferden forekommer 5 ganger per time så vil studiet ta 10±2 timer

Fishers F-fordeling

F-fordelingen, oppkalt etter R.A. Fisher, hvor F er forholdet mellom to varianser (middelkvadrater, hvor den største variansen divideres med den minste variansen) inngår i variansanalyse (anova). Den kritiske verdien for F med antall frihetsgrader i teller og nevner inngår i hypotesetesting. Vi finner *e.g.* den kritiske verdien av F for p=0.05 dvs. 0.95, og henholdsvis 2 og 27 frihetsgrader:

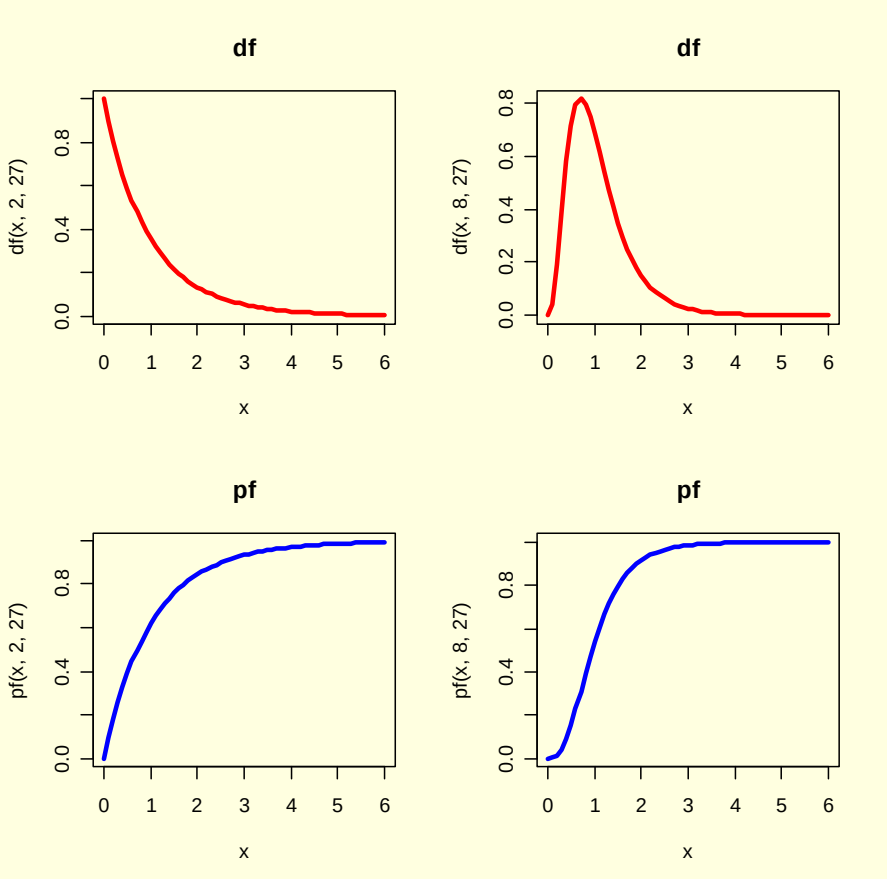
```
 qf(0.95,2,27)
[1] 3.354131
```

Tetthetsfunksjonen for F-fordelingen med to frihetsgrader df1 = n1 og df2=n2 hvor Γ er gammafunksjonen er: Se R-manualen **?df**

$$f(x) = y = \frac{\Gamma\frac{n1+n2}{2}}{\Gamma\frac{n1}{2}\Gamma\frac{n2}{2}} \cdot \left(\frac{n1^{\frac{n1}{2}}}{n2}\ x^{\frac{n1}{2}-1}\right) \cdot \left(1+\frac{n1}{n2}\right)$$

Når det bare er en verdi for df i teller blir F-verdien tilnærmet lik kvadratet av *t* i Students t-fordeling ($F=t^2$)
Tetthetsfordeling og sannsynlighetsfordeling for F *e.g.* med henholdsvis 2 og 27 frihetsgrader:

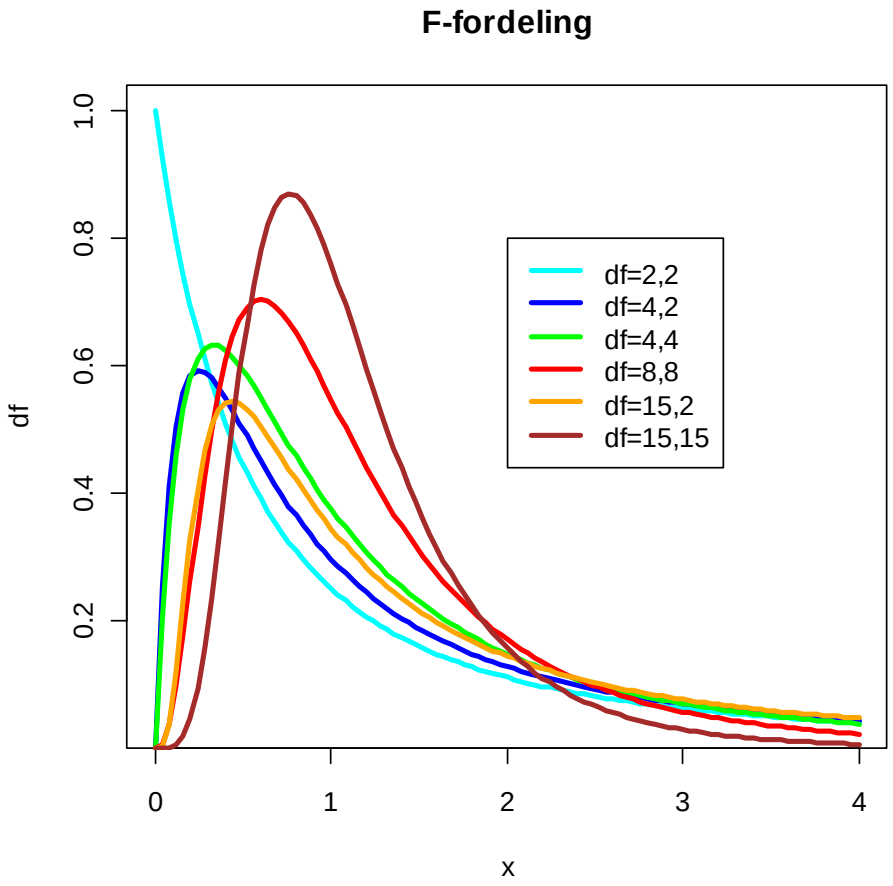
```
x<-seq(0,6,0.1)
par(mfrow=c(2,2),bg="lightyellow")
plot(x,df(x,2,27),type="l",col="red",lwd=3,main="df")
plot(x,df(x,8,27),type="l",col="red",lwd=3,main="df")
plot(x,pf(x,2,27),type="l",col="blue",lwd=3,main="pf")
plot(x,pf(x,8,27),type="l",col="blue",lwd=3,main="pf")
```



Fishers F-fordeling (største varianse/minste varianse)for forskjellige kombinasjoner av frihetsgrader (df):

```
x<-seq(0,4,0.1)
curve(df(x,2,2),col="cyan",lwd=3,ylab="df",xlim=c(0,4),
main="F-fordeling")
curve(df(x,4,2),add=T,col="blue",lwd=3)
curve(df(x,4,4),add=T,col="green",lwd=3)
curve(df(x,8,8),add=T,col="red",lwd=3)
curve(df(x,15,2),add=T,col="orange",lwd=3)
curve(df(x,15,15),add=T,col="brown",lwd=3)
legend(2,0.8,c("df=2,2","df=4,2","df=4,4","df=8,8","df=15,2","
df=15,15"),
```

```
lty=c(1,1,1,1,1,1), lwd=c(3,3,3,3,3,3),
col=c("cyan","blue","green","red","orange","brown"))
```



Eksponential-fordelingen

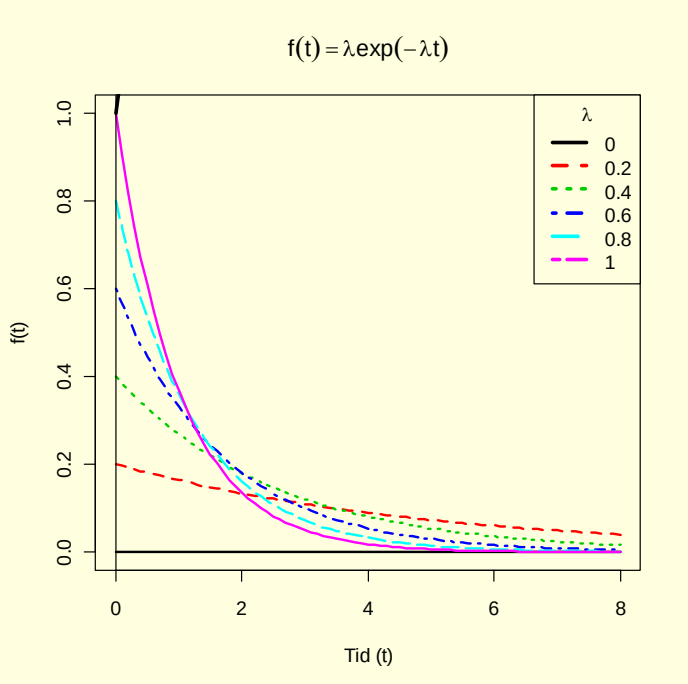
Eksponential-fordelingen er et spesialtilfelle av gamma-fordelingen og brukes i studie av overlevelse. Denne fordelingen får man når tiden til en hendelse måles på en kontinuerlig tidsskala.Gjelder bare når hendelser per tidsenhet er konstant.
Tetthetsfunksjonen for eksponentialfordelingen er:

$$f(x) = y = \lambda \cdot e^{-\lambda \cdot x}$$

Ventetiden er x før en hendelse skjer.
x kan erstattes med t for å indikere tid.

```
par(bg="lightyellow")
t<-seq(0,8,0.1)
lambda<-seq(0,1,0.2)
eks<-outer(t,lambda, function(t,lambda)lambda*exp(-lambda*t))
matplot(t,eks,type="l",ylab="f(t)",xlab="Tid (t)",lwd=2,
main=expression(f(t)==lambda*exp(-lambda*t)))
lines(t,exp(t),lwd=3)
```

```
abline(v=0)
legend("topright",as.character(lambda),title=expression(lambda),lty=1:6,col
=1:6,lwd=3)
```



Figur. Eksponentialfordeling ved forskjellige verdier av λ.

Forventning(gjennomsnitt) til eksponentialfordelingen:

$$E(X) = \frac{1}{\lambda}$$

Variansen til eksponentialfordelingen:

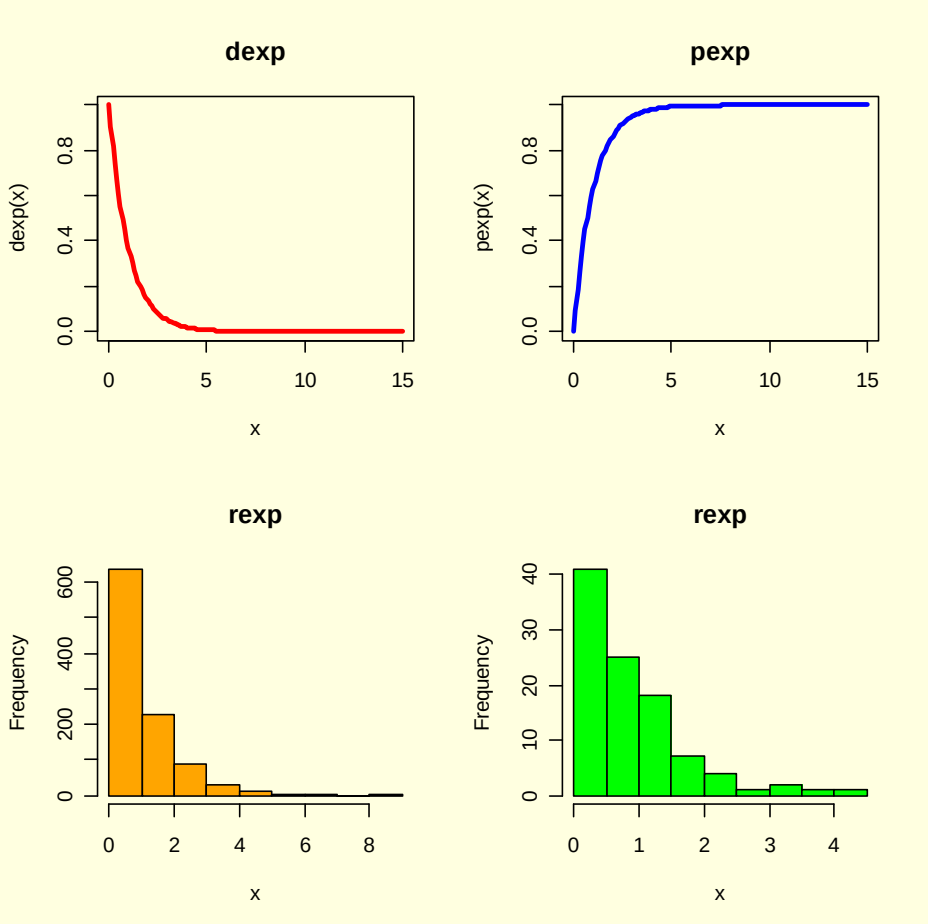
$$Var(X) = \frac{1}{\lambda^2}$$

Se R-manualen:

```
?dexp
```

Tetthets og sannsynlighetsfordeling av eksponentialfordelingen:

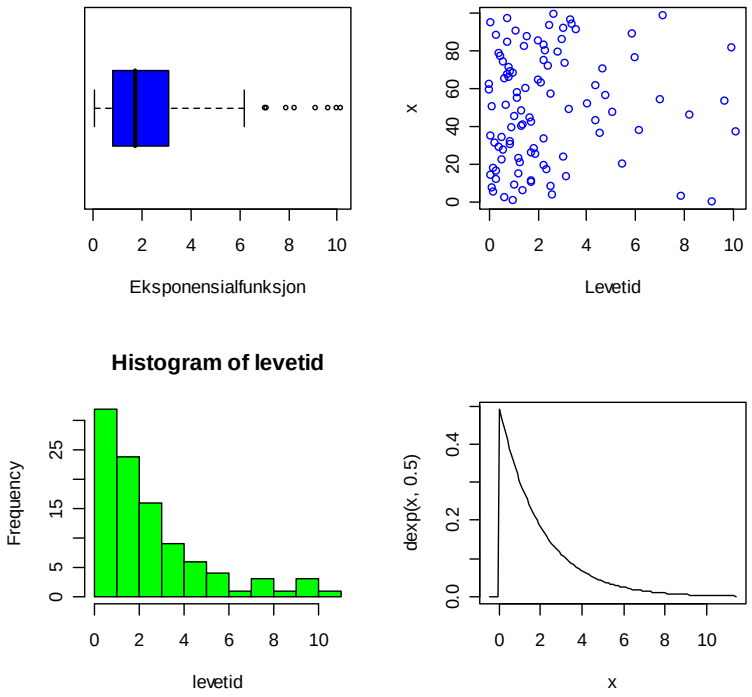
```
x<-seq(0,15,0.1)
par(mfrow=c(2,2),bg="lightyellow")
plot(x,dexp(x),col="red",type="l",lwd=3,main="dexp")
plot(x,pexp(x),col="blue",lwd=3,type="l",main="pexp")
x<-rexp(1000)
hist(x,col="orange",main="rexp")
x<-rexp(100)
hist(x,col="green",main="rexp")
```



Eksponentialfordelingen starter ved en høyde lambda (λ) og synker eksponentielt med rate λ. Middeltallet er lik 1/λ og variansen er lik $1/\lambda^2$.
Eksponentialfordelingen kan brukes i generering av tilfeldige tall med kommandoen **rexp** og kan brukes i Monte Carlo-simuleringer i forsikringsmatematikk *e.g.* tid før død når man kjenner dødsrisikoen (resiprok verdi av gjennomsnittsalder ved død). I eksponentialfordelingen måles det på en kontinuerlig tidslinje, mens på en diskret geometrisk fordeling måles tiden som diskrete aldersklasser. Vi kan omregne dødlighetsrate λ til dødlighetsrisiko per år:

$$p(\text{død per år}) = \int_0^1 \lambda \cdot e^{-\lambda \cdot x} dx = 1 - e^{-\lambda}$$

```
x<-seq(0,99,1)
levetid<-rexp(100,0.5)
par(mfrow=c(2,2))

boxplot(levetid,col="blue",horizontal=TRUE,bty="n",xlab="Ekspo
nensialfunksjon")
  plot(levetid,x,col="blue",xlab="Levetid")
hist(levetid,col="green")
curve(dexp(x,.5)
```



Hvis dødsraten er λ=0.02 per år, så er sannsynligheten for å dø i løpet av et år ca. 2%:

```
 1-exp(-0.02)
[1] 0.01980133
```

Gjennomsnittelig levetid er 1/λ og her lik 50 år.

```
 1/0.02
[1] 50
```

Betafordelingen

Betafordelingen inneholder to konstanter form1=a og form2=b Tetthetsfunksjonen for betafordelingen hvor Γ er gammafunksjonen:

$$f(x) = y = \frac{\Gamma(a+b)}{\Gamma a \cdot \Gamma b} \cdot x^{a-1} \cdot (1-x)^{b-1}$$

Forventet verdi (gjennomsnitt) for Betafordelingen:

$$E(x) = \frac{a}{a+b}$$

Variansen for Betafordelingen:

$$Var(X) = \frac{a \cdot b}{(a+b)^2 \cdot (1+a+b)}$$

Betafordelingen brukes ofte som forhånds sannsynlighetsfordeling (prior) eller betaposterior i Bayesiansk analyse. **Dirichlet-fordelingen** er en generell

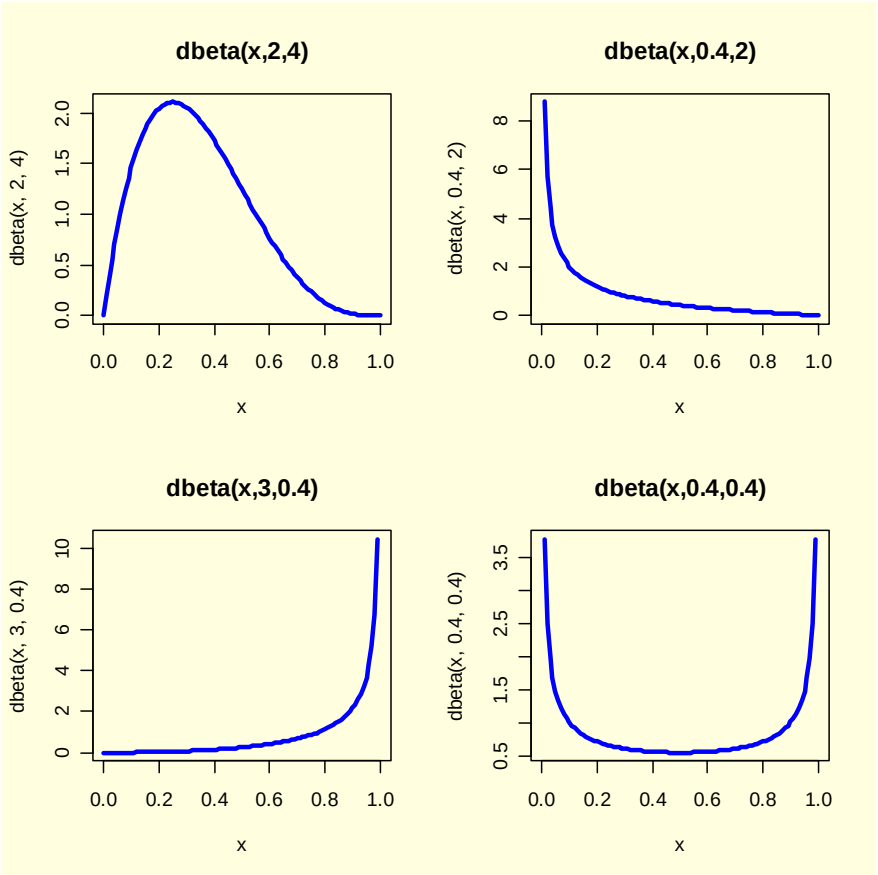
flerdimensjonal utgave av Betafordelingen som også benyttes i Bayesiansk analyse.
Se R-manualen:

```
?dbeta
```

Betafordelingen har middelverdi a/(a+b) og variansen er lik $(ab)/((a+b)^2(1+a+b))$
Kommandoen **dbeta(x,a,b)** gir forskjellig form på tetthetskurven avhengig av verdiene for *a* og *b*:

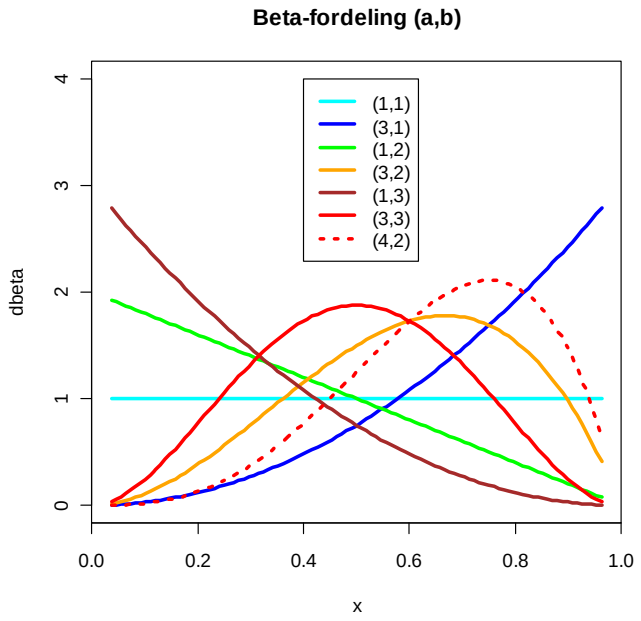
```
x<-seq(0,1,0.01)
par(mfrow=c(2,2),bg="lightyellow")
plot(x,dbeta(x,2,4),col="blue",type="l",lwd=3,main="dbeta(x,2,4)")
plot(x,dbeta(x,0.4,2),col="blue",type="l",lwd=3,main="dbeta(x,0.4,2)")
plot(x,dbeta(x,3,0.4),col="blue",type="l",lwd=3,main="dbeta(x,3,0.4)")
plot(x,dbeta(x,0.4,0.4),col="blue",type="l",lwd=3,main="dbeta(x,0.4,0.4)")
```

Kurven er U-formet når *a* og *b* har omtrent lik størrelse og er mindre enn 1, den er L-formet når *b* er mye større enn *a* og J-formet når *a* er mye større enn *b*, samt klokkeformet når *a* og *b* er tilnærmet like store.




```
x<-seq(0,1,0.01)
curve(dbeta(x,1,1),col="cyan",lwd=3,ylim=c(0,4),ylab="dbeta",
main="Beta-fordeling (a,b)")
```

```
curve(dbeta(x,3,1),add=T,col="blue",lwd=3)
curve(dbeta(x,1,2),add=T,col="green",lwd=3)
curve(dbeta(x,3,2),add=T,col="orange",lwd=3)
curve(dbeta(x,1,3),add=T,col="brown",lwd=3)
curve(dbeta(x,3,3),add=T,col="red",lwd=3)
curve(dbeta(x,4,2),add=T,col="red",lty=3,lwd=3)
legend(0.4,4,c("(1,1)","(3,1)","(1,2)","(3,2)","(1,3)","(3,3)"
,"(4,2)"),
lty=c(1,1,1,1,1,1,3),lwd=c(3,3,3,3,3,3,3),
col=c("cyan","blue","green","orange","brown","red","red"))
```



Cauchyfordelingen

Cauchyfordelingen med skala=*s* og lokalisering=*l* har tetthetsfordeling:

$$f(x) = y = \frac{1}{\pi \cdot s \cdot \left(1 + \left(\frac{(x-l)}{s}\right)^2\right)}$$

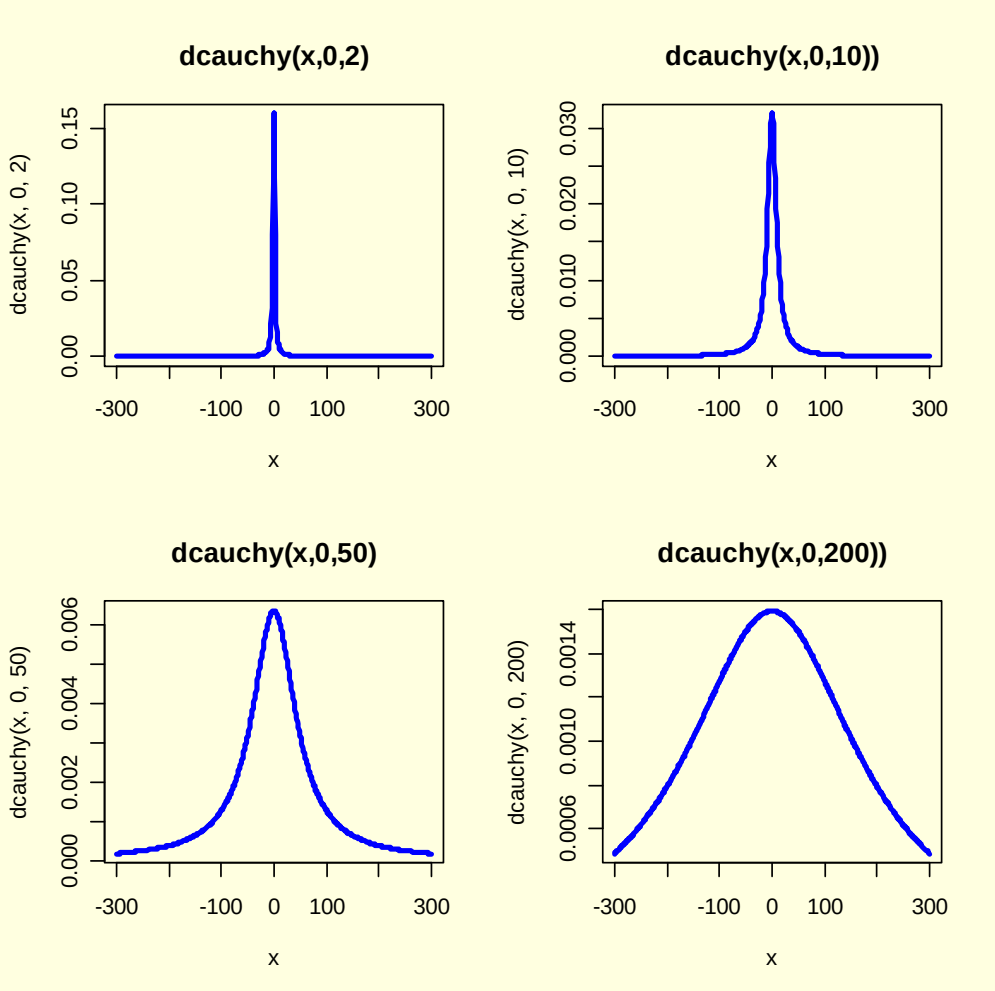
Se R-manualen:

```
?dcauchy
```

Tetthetsfunksjonen er symmetrisk omkring *l* som er lik medianverdien. Hvis l=0 og s=1 (standard Cauchyfordeling) så blir denne lik Students t-fordeling.
Cauchyfordelingen inngår ifm. med Brownske bevegelser.

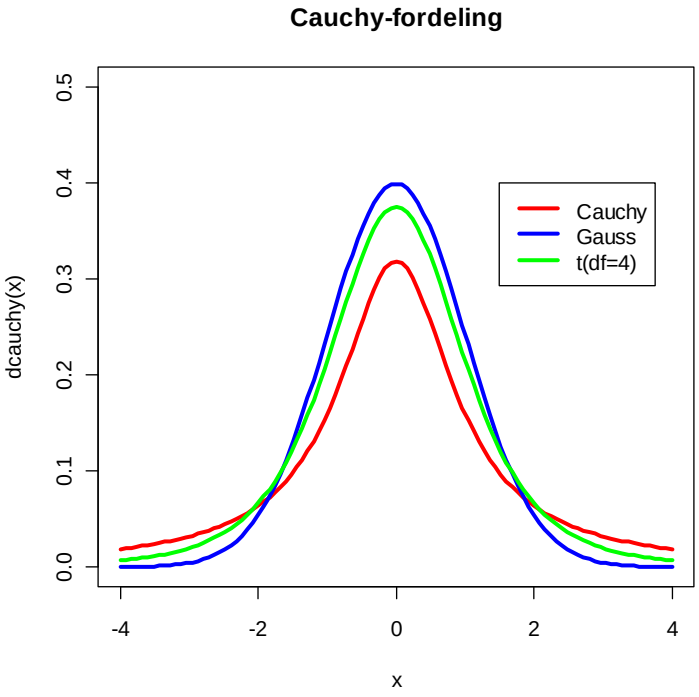
```
x<-seq(-300,300,1)
par(mfrow=c(2,2),bg="lightyellow")
plot(x,dcauchy(x,0,2),col="blue",type="l",lwd=3,main="dcauchy(
x,0,2)")
plot(x,dcauchy(x,0,10),col="blue",type="l",lwd=3,main="dcauchy
(x,0,10))")
```

```
plot(x,dcauchy(x,0,50),col="blue",type="l",lwd=3,main="dcauchy
(x,0,50)")
plot(x,dcauchy(x,0,200),col="blue",type="l",lwd=3,main="dcauch
y(x,0,200))")
```



Cauchy-fordelingen har lenger hale enn normalfordelingen:

```
x<-seq(-4,4,0.1)
curve(dcauchy(x),col="red",lwd=3,xlim=c(-4,4),ylim=c(0,0.5),
main="Cauchy-fordeling")
curve(dnorm(x),add=T,col="blue",lwd=3)
curve(dt(x,df=4),add=T,col="green",lwd=3)
legend(1.5,0.4,c("Cauchy","Gauss","t(df=4)"),
lty=c(1,1,1), lwd=c(3,3,3),
col=c("red","blue","green"))
```



Lognormalfordelingen

Lognormalfordelingen har tetthetsfunksjonen hvor mu (μ) er middelverdien og sigma (σ) er standardavviket til logaritmen.

$$f(x) = \frac{1}{\sigma X\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{\ln(X)-\mu}{\sigma}\right)^2}$$

Forventet middeltall for lognormalfordelingen for en tilfeldig variabel X:

$$E(X) = e^{\mu+\frac{1}{2}\sigma^2}$$

Variansen:

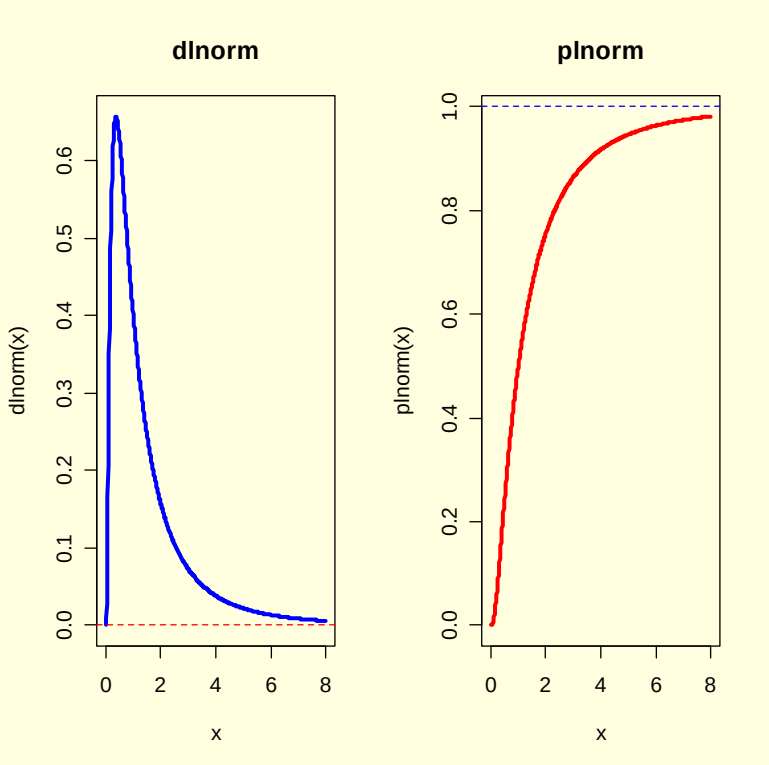
$$var(X) = e^{2\mu+\sigma^2} \cdot \left(e^{\sigma^2} - 1\right)$$

Se R-manualen:

```
?dlnorm
```

Lognormalfordelingen har positive verdier, har positiv skew med en lang hale til høyre.

```
par(mfrow=c(1,2),bg="lightyellow")
x<-seq(0,8,0.01)
plot(x,dlnorm(x),type="l",col="blue",lwd=3,main="dlnorm")
abline(h=0,lty=2,col="red")
plot(x,plnorm(x),type="l",col="red",lwd=3,main="plnorm")
abline(h=1,lty=2,col="blue")
```

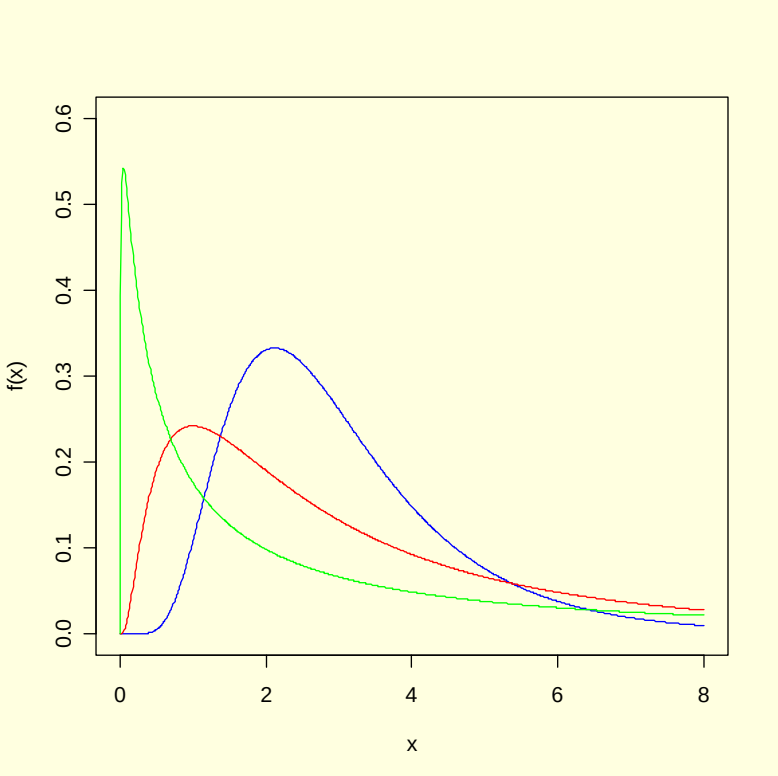


Kommandoer for lognormalfordeling er **lnorm**, middelverdi er **meanlog=** og standardavvik er **sdlog=**.
En annen utgave av lognormalfordelingen er:

$$f(x) = \frac{e^{-(\ln(x)-\mu)^2/2\sigma^2}}{x \cdot \sqrt{2\pi\sigma^2}}$$

Figur som viser lognormalfordeling med middeltall=1 og forskjellig standardavvik: SD=0.5, 1 og 2 som viser hvordan den endrer seg med variansen:

```
x<-seq(0,8,0.01)
par(bg="lightyellow")
plot(x,dlnorm(x,1,0.5),ylim=c(0,0.6),type="l",col="blue",ylab=
"f(x)")
points(x,dlnorm(x,1,1),type="l",col="red")
points(x,dlnorm(x,1,2),type="l",col="green")
```



Logistisk fordeling

Se R-manualen:

```
?dlogis
?plogis
```

Denne fordelingen er linkfunksjon i generaliserte lineære modeller. Den kumulative sannsynlighetsfordelingen p er gitt ved:

$$p = \frac{e^{a+bx}}{1+e^{a+bx}}$$

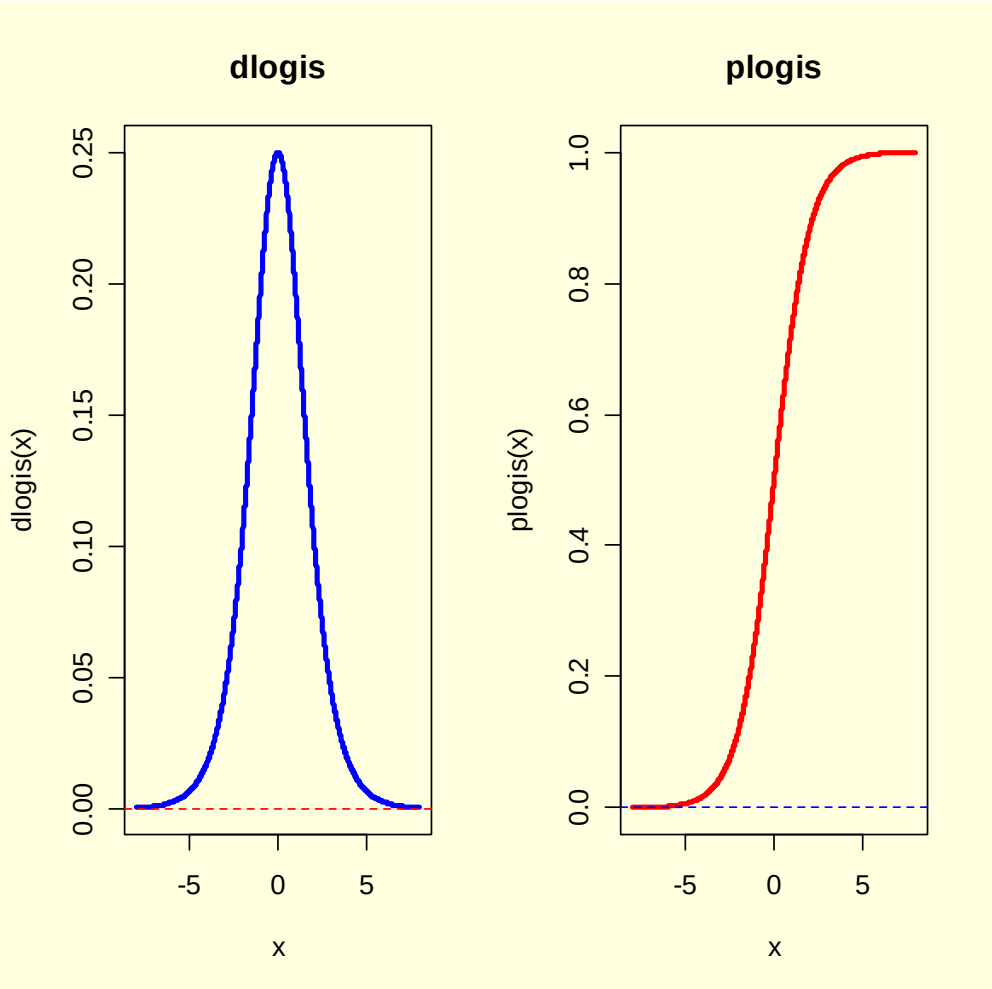
og sannsynligheten $q=1-p$. I veddmålssammenheng brukes $odds=p/q$
Vi kan vise at:

$$ln\left(\frac{p}{q}\right) = a + bx$$

som er en rett linje. Vi har dessuten:

$$logit\ p = ln\frac{p}{1-p} = ln\left(\frac{p}{q}\right)$$

```
par(mfrow=c(1,2),bg="lightyellow")
x<-seq(-8,8,0.01)
plot(x,dlogis(x),type="l",col="blue",lwd=3,main="dlogis")
abline(h=0,lty=2,col="red")
plot(x,plogis(x),type="l",col="red",lwd=3,main="plogis")
abline(h=0,lty=2,col="blue")
```



Vi kan se at tetthetsfunksjonen **dlogis** er mye bredere enn normalfordelingen **dnorm**.

Weibullfordelingen

Weibullfordelingen benyttes i demografiske studier og overlevelseanalyse hvor en formparameter α beskriver hasardfunksjonen.

$$f(x) = \alpha\lambda(\lambda x)^{\alpha-1} \cdot e^{-(\lambda x)^{\alpha}}$$

Hvorα er en formparameter og λ er en skalarparameter.
Forventet middelverdi:

$$E(X) = \frac{1}{\lambda}\Gamma\left(1+\frac{1}{\alpha}\right)$$

Forventet varianse:

$$var(X) = \frac{1}{\lambda^2}\left[\Gamma\cdot\left(1+\frac{2}{\alpha}\right) - \left(\Gamma\cdot\left(1+\frac{1}{\alpha}\right)\right)^2\right]$$

Hvis $\alpha = 1$ er hasard konstant (eksponensialfordeling), hvis $\alpha>1$ øker hasard med alder, og hvis $\alpha<1$ minsker hasard med

alder. Weibull og lognormal er eksempler på fordeling med positiv skew.
Weibull er en avart av eksponensialfunksjonen hvor vekstreduksjonen (hasard) ikke er konstant

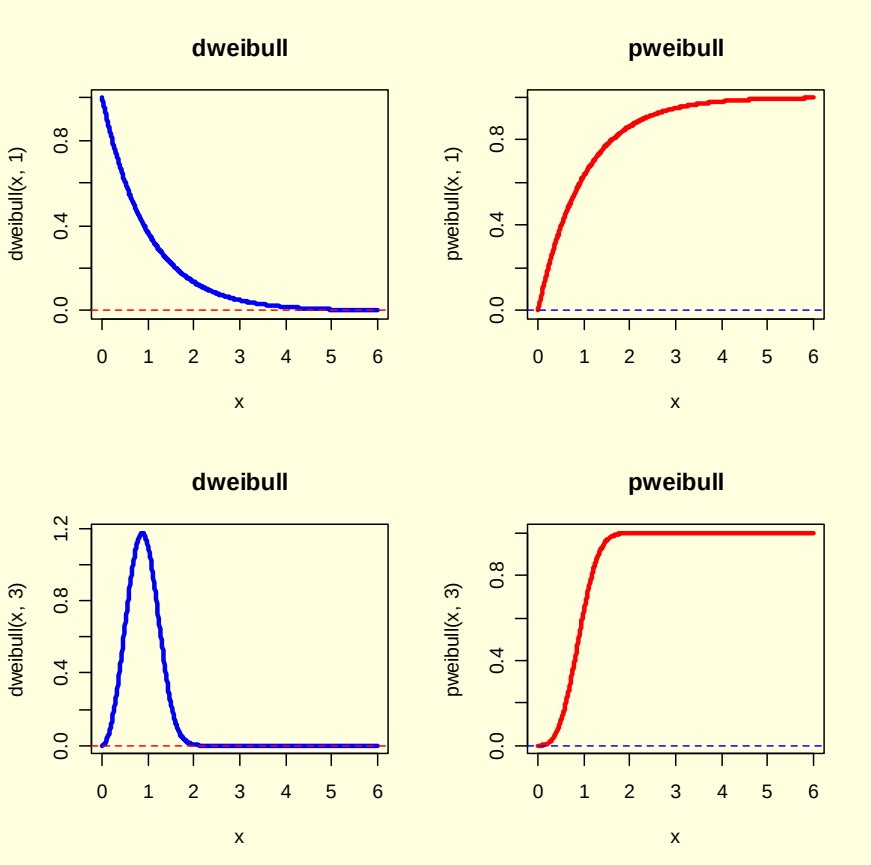
$$y = a - b \cdot e^{-(cx^d)}$$

Se R-manualen

```
 ?dweibull
 ?pweibull
```

Figurer med form a=1 og a=3

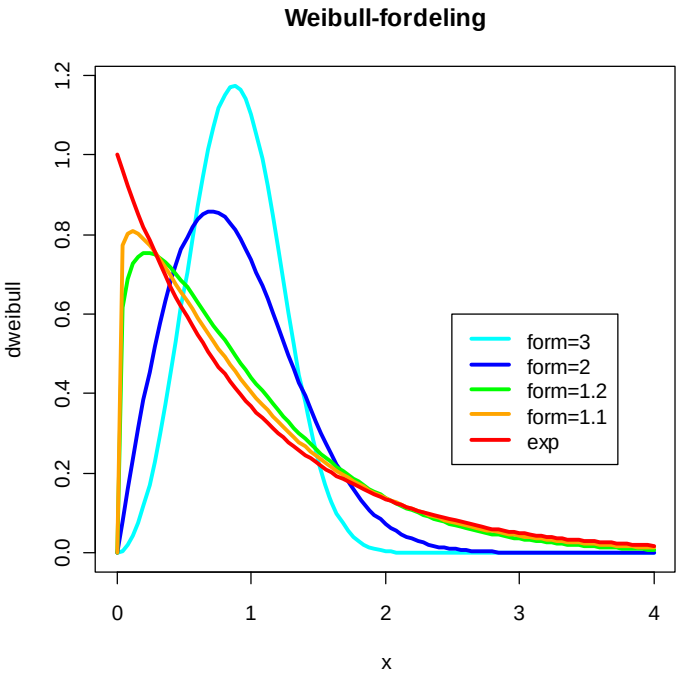
```
par(mfrow=c(2,2),bg="lightyellow")
x<-seq(0,6,0.01)
plot(x,dweibull(x,1),type="l",col="blue",lwd=3,main="dweibull"
)
abline(h=0,lty=2,col="red")
plot(x,pweibull(x,1),type="l",col="red",lwd=3,main="pweibull")
abline(h=0,lty=2,col="blue")
plot(x,dweibull(x,3),type="l",col="blue",lwd=3,main="dweibull"
)
abline(h=0,lty=2,col="red")
plot(x,pweibull(x,3),type="l",col="red",lwd=3,main="pweibull")
abline(h=0,lty=2,col="blue")
```



Weibullfunksjonen med forskjellig form og sammenlignet med eksponensialfunksjonen:

```
x<-seq(0,4,0.1)
curve(dweibull(x,3),col="cyan",lwd=3, ylab="dweibull",
xlim=c(0,4),main="Weibull-fordeling")
```

```
curve(dweibull(x,2),add=T,col="blue",lwd=3)
curve(dweibull(x,1.2),add=T,col="green",lwd=3)
curve(dweibull(x,1.1),add=T,col="orange",lwd=3)
curve(dexp(x),add=T,col="red",lwd=3)
legend(2.5,0.6,c("form=3","form=2","form=1.2","form=1.1","exp"
),
lty=c(1,1,1,1,1), lwd=c(3,3,3,3,3),
col=c("cyan","blue","green","orange","red"))
```



Uniform fordeling

Uniform fordeling er en fordeling fra min til max og er en enkel kontinuerlig sannsynlighetsfunksjon. Det er en tilfeldig variabel som har like stor sannsynlighet for å havne mellom verdiene angitt av min og max.

$$f(x) = \frac{1}{max - min}$$

Forventet verdi(gjennomsnitt) av Uniform fordeling

$$E(X) = \int_{min}^{max} x \cdot \frac{1}{max - min} dx = \frac{max + min}{2}$$

Varianse for Uniform fordeling:

$$Var(X) = \int_{min}^{max} (x - \mu)^2 \cdot f(x) dx = \int_{min}^{max} \left(x - \frac{max + min}{2}\right)^2 \cdot \frac{1}{max - min} dx$$

$$= \frac{(max - min)^2}{12}$$

Sannsynlighetstetthetsfunksjonen har en konstant høyde *h*, fordi integralet over mulige verdier blir lik 1:

$$\int_{min}^{max} h \cdot dx = h \cdot max - h \cdot min = 1$$

Derfor blir *h=1/(max-min)*
i vårt tilfelle nedenfor blir h=1 (h=1/(1-0))
Hvis vi har minimumsverdi *a* og maksimumsverdi *b* på xx-aksen så vil for a<x<b y-verdien (f(x)) bli:

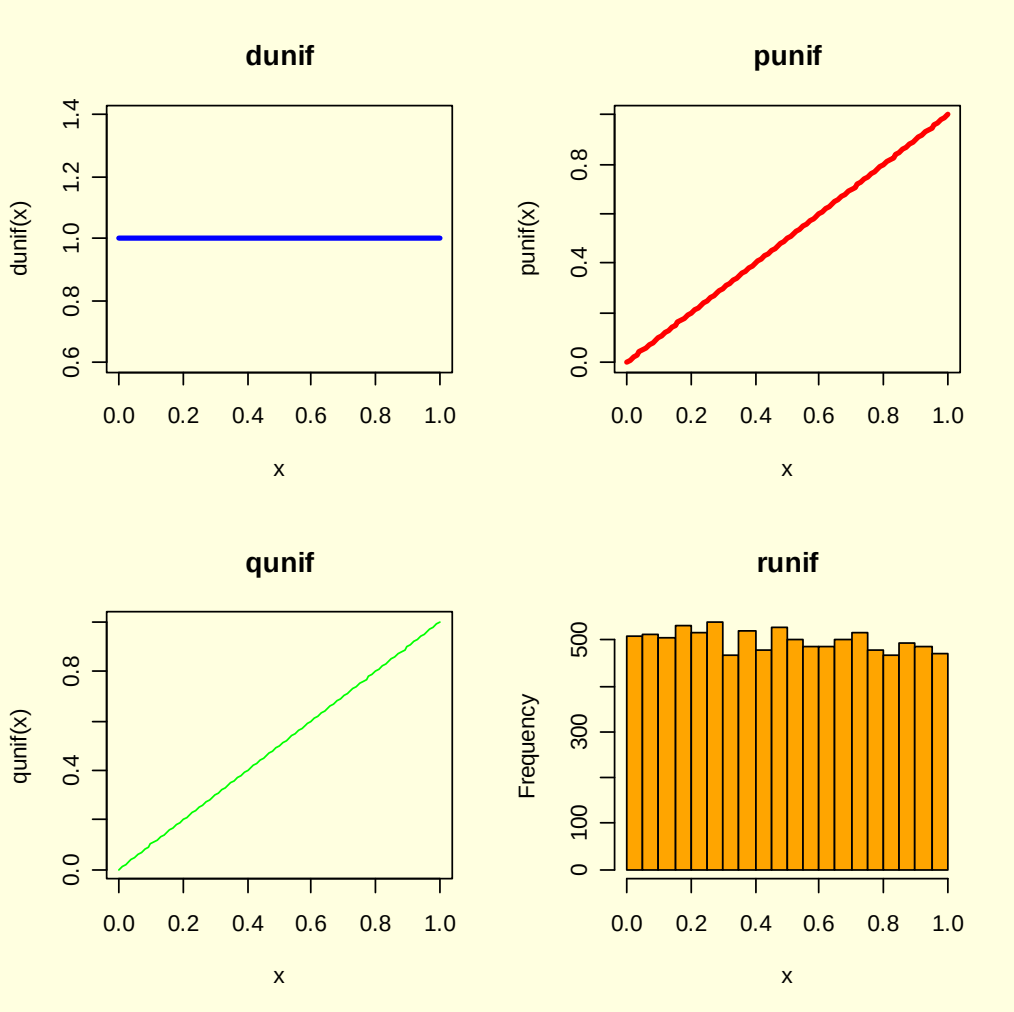
$$f(x) = \frac{1}{a-b}$$

Forventet verdi eller gjennomsnitt blir:1/2(a+b)
Variansen blir: $1/12(b-a)^2$.
For øvrig vil f(x)=0
Se R-manual:

```
?dunif
?punif
```

Uniform fordeling brukes bl.a. i produksjonen av slumptall i slumptallsgeneratorer.
Figur som viser tetthetsfunksjon, kumulativ sannsynlighet, kvantiler og histogram over 10000 tall plukket tilfeldig ut med runif, min=0 og max=1. Histogrammet viser uniform fordeling og tetthetsfunksjonen er konstant og lik 1.

```
par(mfrow=c(2,2),bg="lightyellow")
x<-seq(0,1,0.01)
plot(x,dunif(x),type="l",col="blue",lwd=3,main="dunif")
plot(x,punif(x),type="l",col="red",lwd=3,main="punif")
plot(x,qunif(x),type="l",col="green",main="qunif")
x<-runif(10000)
hist(x, col="orange",main="runif")
```



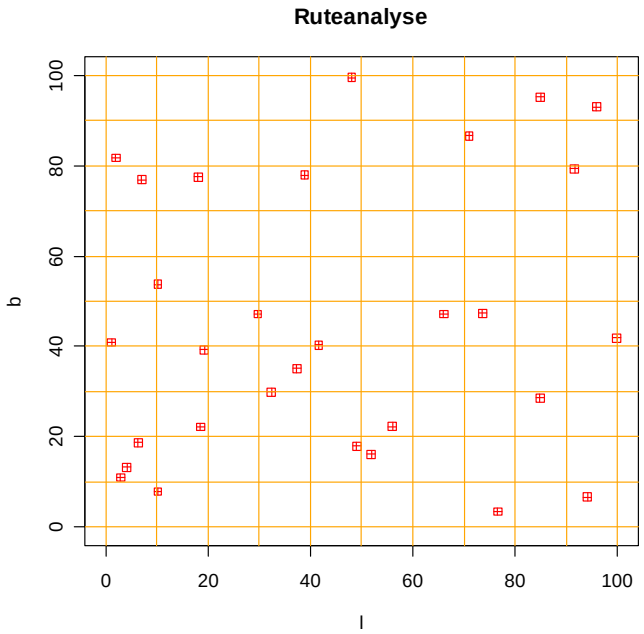
Middeltallet i en uniform fordeling faller midtveis mellom max og min, og i vårt tilfelle max=1 og min=0 blir dette 0.5.

```
x<-seq(0,1,0.01)
pu<-punif(x)
mean<-mean(pu);mean
[1] 0.5
```

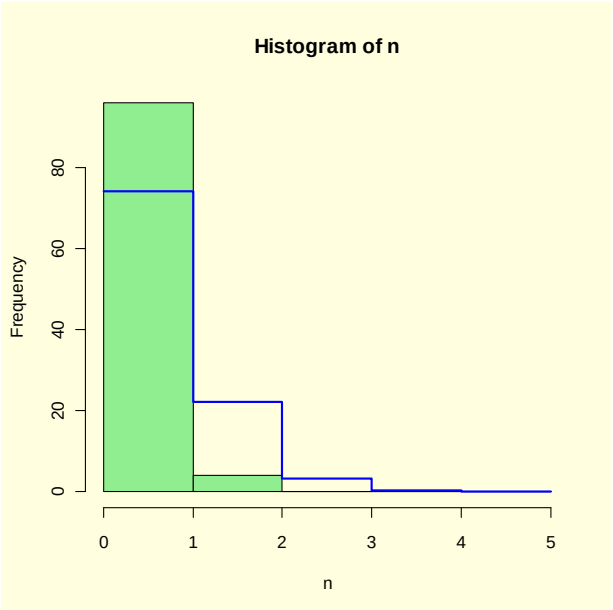
Vi kan simulere en ruteanalyse hvor man teller antall planter på en rute 100x100 cm. Vi lager 30 tilfeldige punkter (planter) mellom 0 og 100, og lager et rutenett. Med kommandoen **cut** omdannes koordinatene til tall avhengig av i hvilket intervall de faller. Sjekk i objektet telling at antallet stemmer. Vi forventer at en slik tilfeldig plassering av planter vil følge Poisson-fordeling. Med 30 individer fordelt på 100 ruter blir λ=0.3

```
l<- 100 * runif(30)
b <- 100 * runif(30)
plot(l, b, xlim=c(0,100),ylim=c(0,100), pch = 12,
col="red",main="Ruteanalyse")
abline(h = seq(0,100,10), v = seq(0,100,10),col="orange")
xl <- cut(l, seq(0,100,10))
yb <- cut(b, seq(0,100,10))
telling <- table(xl,yb);telling
```

```
table(telling)
telling
 0  1  2
74 22  4
```




```
n <- matrix(telling, ncol = 1)
hist(n, breaks=seq(0,5,1) ,col="lightyellow")
forventet<- 100 * dpois(0:5, lambda=0.3);forventet
lines(seq(0,5,1), forventet, type = "s", lwd = 2, col =
"blue")
```



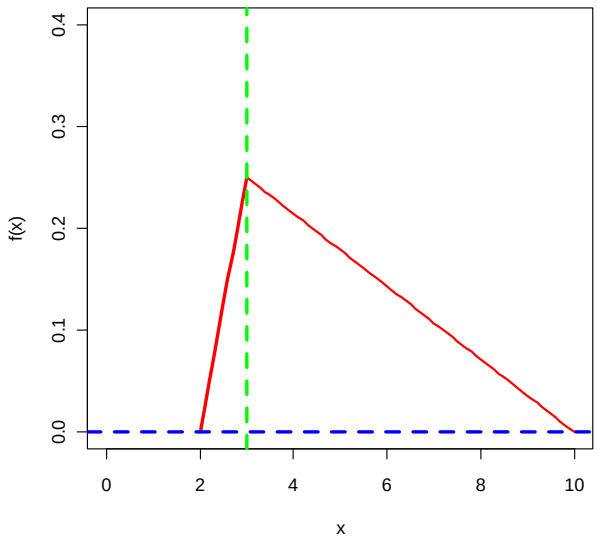
Triangulær fordeling

Hvis vi har minimumsverdi a og maksimumsverdi *b* og *m* ligger mellom *a* og *b* på x-aksen så har vi:

$$f(x) = \begin{cases} h\dfrac{x-a}{m-a} & for\ a < x \leq m \\ h\dfrac{b-x}{b-m} & for\ m \leq x < b \\ 0 & forøvrig \end{cases}$$

Sannsynlighetsfunksjonen øker i intervallet *a-m* og synker i intervallet *m* til *b*. h=2/(b-a)

```
a<-2
b<-10
m<-3
h<-2/(b-a)
x<-seq(a,m,0.1)
plot(x,h*((x-a)/(m-
a)),xlim=c(0,10),ylim=c(0,0.4),type="l",lwd=3,col="red",ylab="
f(x)")
x<-seq(m,b,0.1)
lines(x,h*((b-x)/(b-m)),type="l",lwd=2,col="red")
abline(h=0,col="blue",lty=2,lwd=3)
abline(v=m,col="green",lty=2,lwd=3)
```



Gammafunksjonen

Gammafunksjon som også kalles generalisert faktoriseringsfunksjon er definert som et integral ifølge Euler:

$$\Gamma(x) = \int_0^\infty t^{x-1}\, e^{-t} dt$$

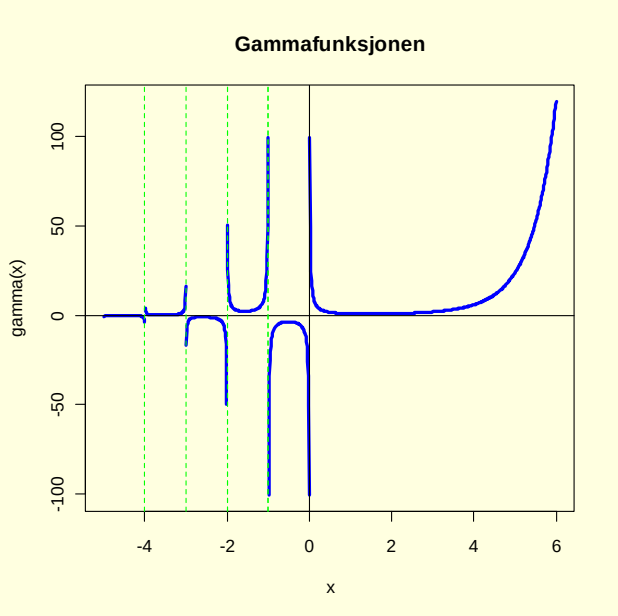
hvor x-1>-1 eller x>0.
Gir også sammenhengen:

$$\Gamma(x+1) = x \cdot \Gamma(x)$$

Når x=1 blir integralet lik 1. Vi kan deretter bruke formelen ovenfor og ser at Γ(2) = 1·Γ(1)=1. Γ(3)= 2·Γ(2) = 2·1=3 Γ(4)=3·Γ(3)=3·2·1 Det vil si

$$\Gamma(n+1) = n!$$

```
x<-seq(-5,6,0.01)
par(bg="lightyellow")
plot(x,gamma(x),type="l",lwd=3,col="blue",main="Gammafunksjonen")
abline(h=0,v=0)
abline(v=c(-1,-2,-3,-4),col="green",lty=2)
```



Figur. Gammafunksjonen nærmer seg til vertikale assymptoter for negative heltall.

Det viser seg også at gammafunksjonen til ½ er lik kvadratroten av pi

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

Bootstrap og resampling

Funksjonen **sample** stokker et datasett.
Vi lager et datasett fra 0-20, stokker dette, beregner middeltallet og plasserer resultatet i en tabell:

```
x<-0:20
sample(x)
 [1] 20  3 19  0  6  4  7 16 14 12  1  9 15 17 10  8 18  5 13  2 11
mean(x)
[1] 10
table(sample(x,replace=T))

 0  1  3  4  6  7 10 11 12 13 14 16 17 19 20
 2  1  2  2  1  2  1  1  1  2  1  1  1  2  1
```

Vi lager en vektor som vi fyller med 20000 middeltall:

```
 xmiddel<-numeric(20000)
 for(i in 1:20000)xmiddel[i]<-mean(sample(x,replace=T))

 mean(xmiddel)
[1] 9.996543
```

Vi finner 95% konfidensintervall for bootstrappingen:

```
 quantile(xmiddel,c(0.25,.975))
     25%     97.5%
 9.095238 12.571429
```

Vi kan finne standardfeilen SE:

```
 SE<-sqrt(var(x)/length(x))
 SE
[1] 1.354006
```

t-verdien fra tabell:

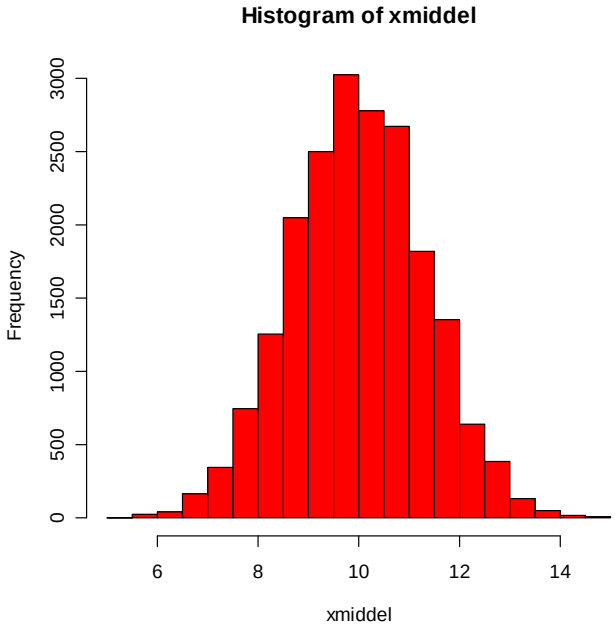
```
 tverdi<-qt(0.975,length(x))
 tverdi
[1] 2.079614
```

Konfidensintervall CI:

```
 CI<-tverdi*SE
 CI
[1] 2.815810
 mean(x)+CI
[1] 12.81581
 mean(x)-CI
[1] 7.18419
```

Lager et histogram for middeltallene, som vi ser blir normalfordelt:

```
 hist(xmiddel,col="red")
```



Vi har produksjon av hvete per dekar ifølge SSB fra 1985-2007

```
hvete<-
c(435,400,429,340,374,466,465,353,511,284,475,453,431,469,445,
460,407,410,461,477,489,416,459)
år<-c(seq(1985,2007,1)
```

Vi kan beregne konfidensintervall parametrisk (CIp)

```
 CIp<-mean(hvete)+1.96*sqrt(var(hvete)/length(hvete));CIp
[1] 452.5728
 CIp<-mean(hvete)-1.96*sqrt(var(hvete)/length(hvete));CIp
[1] 409.0793
```

95% konfidensintervall= 409.0793 til 452.5728 kg/da

Vi kan gjøre det samme regnestykket med bootstraping og resampling 10.000 ganger:

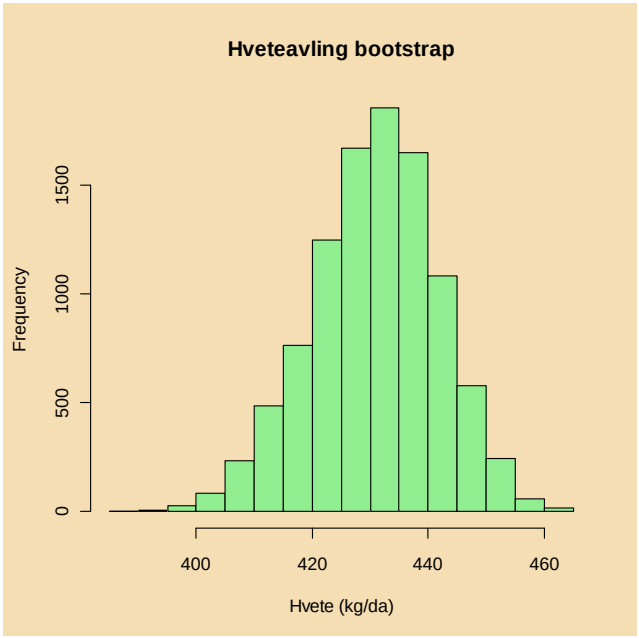
```
n<-10000
m<-numeric(n)
for (i in 1:n){
m[i]<-mean(sample(hvete,replace=T))
}
CIb<-quantile(m,c(0.025,0.975));CIb
   2.5%    97.5%
408.4337 451.0435
```

Konfidensintervall bestemt ved bootstraping:
95% konfidensintervall= 408.4337 til 451.0435 kg/da
Som er rimelig godt i overensstemmelse med CI bestemt parametrisk. Datasettet hvete er imidlertid ikke helt normalfordelt, noe som gir forskjeller.

Vi kan plotte histogrammet for middelverdiene beregnet ved bootstraping:

```
par(bg="wheat")
```

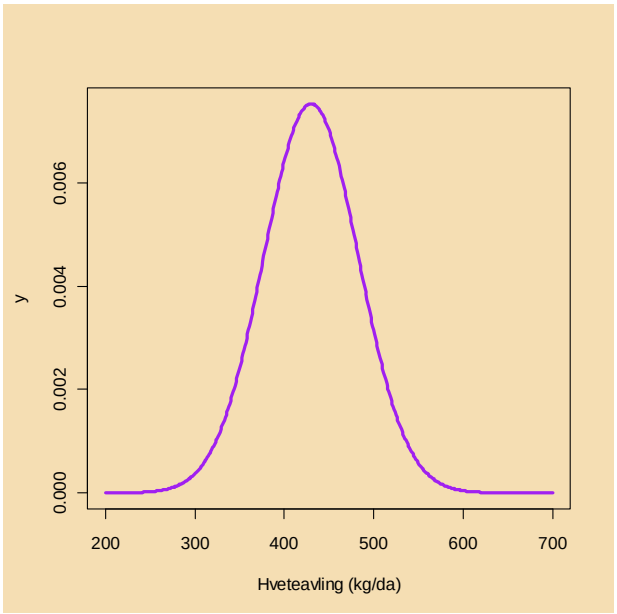
```
hist(m,col="lightgreen", xlab="Hvete (kg/da)",
main="Hveteavling bootstrap")
```




```
 mean(hvete)
[1] 430.8261
 sd(hvete)
[1] 53.21112
```

Vi lager en normalfordelingskurve med snitt=430 og standardavvik 53

```
x<-seq(200,700,1)
y<-dnorm(x,mean=430,sd=53)
par(bg="wheat")
plot(x,y,type="l",col="purple",lwd=3,xlab="Hveteavling
(kg/da)")
```



Arealet under denne kurven =1
Hvor stor andel av arealet under normalfordelingskurven tilsvarer en avling på <340 kg/da ?

Vi regner om til standard normalfordeling (z), hvor s er standardavviket:

$$z = \frac{y - \bar{y}}{s} = \frac{340 - 430}{53} = -1.698113$$

Vi får negativ verdi siden vår verdi er mindre enn gjennomsnittet. Hvor stor del av arealet under normalfordelingskurven tilsvarer dette ?

```
pnorm(-1.698113)
[1] 0.04474322
```

Det vil si 4.5%

Hvor sannsynlig er det å få en avling over 500 kg/da ?

$$z = \frac{y - \bar{y}}{s} = \frac{500 - 430}{53} = 1.320755$$

```
1-pnorm(1.320755)
[1] 0.09329153
```

Det vil si 9.3%

Hvor mye av avlingen er mellom 400-500 kg/da ?

$$z_1 = \frac{y - \bar{y}}{s} = \frac{400 - 430}{53} = -0.5660377 \; og \; z_2 = \frac{500 - 430}{53} = 1.320755$$

Vi trekker den minste sannsynligheten for den største:

```
pnorm(1.320755)-pnorm(-0.5660377)
[1] 0.6210244
```

Dvs. 63%

Den enkleste måten å lage tabeller i R er med kommandoen **table()**.

```
sml<-cut(vals,3)
table(sml)
sml
(-0.017,5.66]    (5.66,11.3]     (11.3,17]
         9623            354            23
```

Kategoriske data kan tolkes som en faktor med kommandoen **factor()**. For eksempel tallene fra 1-5 som faktorer:

```
factor(1:5)
[1] 1 2 3 4 5
Levels: 1 2 3 4 5
```

Lineær regresjon

Regresjon benyttes når man har en kontinuerlig uavhengig og kontinuerlig avhengig variabel.

	Uavhengig variabel	
Avhengig variabel	Kontinuerlig	Kategorisk
Kontinuerlig	Regresjon	ANOVA

Kategorisk	Logistisk regresjon	Tabell

ANCOVA brukes hvis det er to uavhengige variable, en kategorisk og en kontinuerlig kovariabel.

Regresjon er en funksjonell avhengighet mellom variablene, i motsetning til korrelasjon. Sikre deg at prediktorvariabel er uniform innen prøveområdet. Hvis en eller to av verdiene for prediktorvariabel er svært innflytelsesrike så vil disse punktene dominere stigningskoeffisienten i regresjonen og gi en sammenheng som ikke eksisterer i virkeligheten.

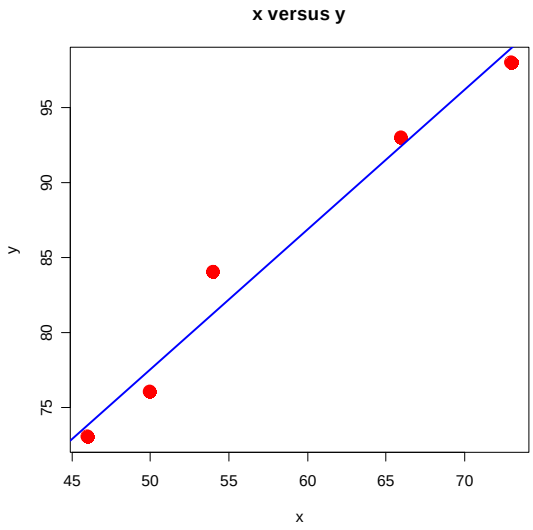
Regresjon omfatter lineær regresjon, multippel regresjon med flere prediktorer, polynomregresjon og logistisk regresjon (generaliserte lineære modeller).
En regresjonslinje viser relasjon og sammenheng mellom to variable:
responsvariabel (avhengig variabel) y og prediktorvariabel (uavhengig variabel) x. Regresjonsmodellen er $y = \alpha + \beta \cdot x + \varepsilon$. Hvor regresjonskoeffisientene α er skjæring med y-aksen ("intercept"), β er stigningskoeffisienten til linjen og ε er et ledd som angir feil. Regresjonskoeffsientene finnes med kommandoen lm(modell), hvor lm står for lineær modell.

```
x<-c(46,54,66,50,73)
y<-c(73,84,93,76,98)
lm(y~x)
```

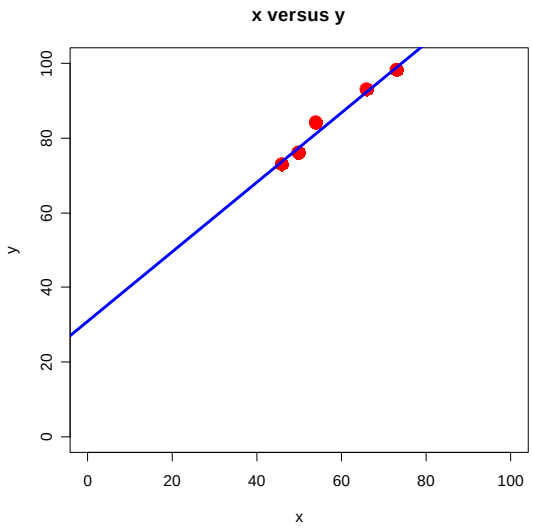
```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    30.8323       0.9337
```

Vi kan plotte regresjonslinjen på et punktskyplot med kommandoen **abline()**. I et punktskyplot er x-asken uavhengig variabel (forklaringsvariabel) og y-aksen er avhengig variabel (responsvariabel. Vil man at regresjonslinjen skal gå gjennom origo kan man sette opp modellen y~x-1.

```
modell=lm(y~x)
plot(x,y,col="red",pch=19,cex=2,main="x versus y")
abline(modell,col="blue",lwd=2)
```



Det er ikke alltid oppdelingene av aksene er slik vi ønsker og dette kan vi gjøre med kommandoene **xlim** og **ylim**:




```
plot(x,y,pch=19,cex=2,col="red",xlim=c(0,100),ylim=c(0,100),ma
in="x versus y")
 abline(modell,col="blue",lwd=3)
```

Regresjonslinjen som vi har funnet:
 y = 30.8323 + 0.9337·x
kan brukes til å prediktere (forutsi) verdier, y-verdien for x=60 er for eksempel

```
 30.8323 + 0.9337*60
[1] 86.8543
```

En annen måte å finne regresjonskoeffisientene er via kommandoen **coef()** anvendt på regresjonsmodellen y~x:

```
 coef(modell)
(Intercept)           x
 30.8322933   0.9336973
```

Med kommandoen **predict()** kan prediktere verdier og med **resid()** kan man finne residualene (restverdiene etter regresjonen, forskjellen mellom tilpasset og observert verdi). Med **fitted()** finner man tilpassete y-verdier som forventes fra en gitt x ifølge regresjonslinjen. Med **summary()** kan man få den samlede oversikt over regresjonen:

```
summary(modell)
```

```
Call:
lm(formula = y ~ x)

Residuals:
      1       2       3       4       5
-0.7824  2.7480  0.5437 -1.5172 -0.9922

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.83229    5.12742   6.013  0.00922 **
x            0.93370    0.08738  10.686  0.00175 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.979 on 3 degrees of freedom
Multiple R-Squared: 0.9744,     Adjusted R-squared: 0.9659
F-statistic: 114.2 on 1 and 3 DF,  p-value: 0.001752
```

Koeffisientene i regresjonstabellen gir regresjonslinjen $y = 30.8323 + 0.9337 \cdot x$. For hver av koeffisientene er det oppgitt standardfeilen til estimatet, t-verdien fra testen om koeffisienten er forskjellig fra 0 (nullhypotesen), samt sannsynligheten, P-verdier(`Pr(>|t|)`), for å få en t-verdi som tabellen viser. Under tabellen finner vi "R^2" (kvadratet av korrelasjonskoeffisienten) som viser at regresjonslinjen forklarer ca. 97% av variasjonen i datasettet. Helt nederst i tabellen finner vi p-verdien fra testing av signifikansen fra nullhypotesen om at responsvariabel (y) er uavhengig av prediktor (x). p=0.001752 viser at det er ca. 0.18% sannsynlig at det ikke er noen sammenheng mellom x og y, altså forkaster vi nullhypotesen og kommer fram til konklusjonen at det er en lineær sammenheng mellom x og y.

Plotting av modellen: Først lager man plass til fire plot med kommandoen **par(mfrow=c(2,2))** og deretter plotter man modellen og får ut QQ-plot, og plotting av residulane etter regresjonen

```
par(mfrow=c(2,2))
plot(modell)
```

Vi kan også beregne regresjonslinjen $y=a+bx$ på en annen måte hvor:

$$b = r \cdot \frac{s_y}{s_x}$$

og r er korrelasjonskoeffisienten og s_y og s_x er standardavviket for y og x.

Regresjonslinjen vil alltid gå igjennom middelverdien for x og y. Da vi har funnet verdien for b kan vi sette denne inn i følgende og bestemme a:

$$a = \bar{y} - b \cdot \bar{x}$$

```
mx<-mean(x)
my<-mean(y)
r<-cor(x,y)
vx<-var(x)
vy<-var(y)
sx<-sqrt(vx)
sy<-sqrt(vy)
b<-r*sy/sx;b
[1] 0.9336973
a<-my-b*mx;a
[1] 30.83229
```

Det vil si $y=30.83229 + 0.9336973 \cdot x$

Hvis $\hat{y}$ (yhatt) er y-verdien som predikteres av regresjonslinjen for en verdi av x, så vil residualen (avstand fra et punkt til regresjonslinjen) være lik $y-\hat{y}$. I et residual-plot bør punktene ligge jevnt spredt over og under 0-linjen.

Vi kan få en liste over muligheter tilknyttet objektet modell:

```
names(modell)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
```

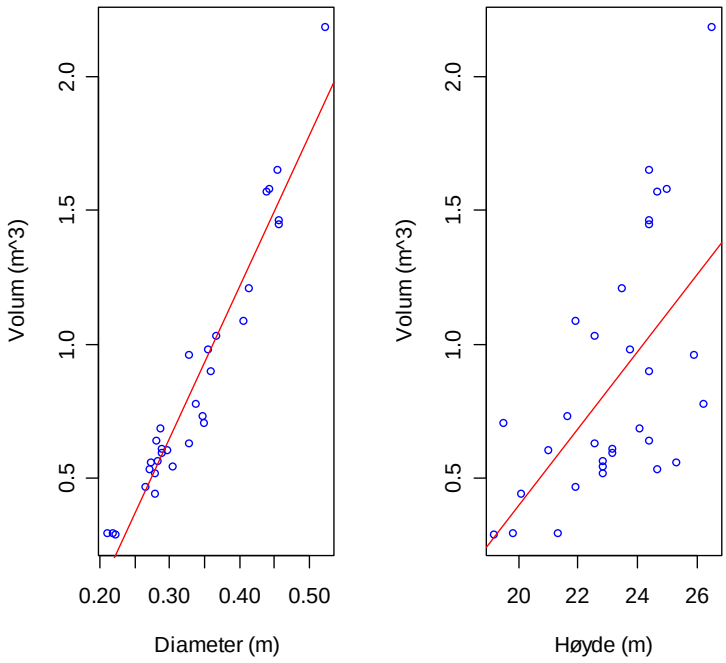
For eksempel:

```
modell$model
   y  x
1 73 46
2 84 54
3 93 66
4 76 50
5 98 73
```

Et datasett trees fra R som viser sammenhengen mellom volum (cubic ft), diameter (girth, inch) og høyde (height, ft)for 31 felte kirsebærtrær. Vi må regne om fra tommer (1 inch=2.54 cm) og fot (1 foot=30.38 cm)til meter, **help(trees)**:

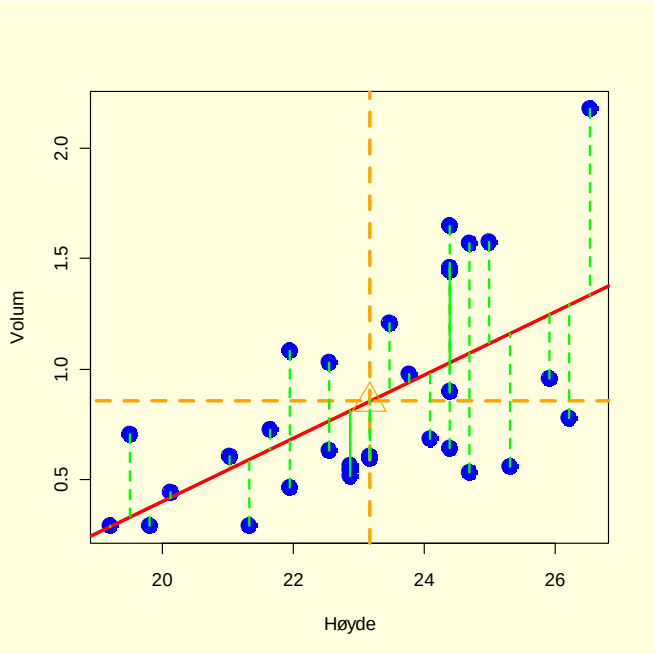
```
library(datasets)
data(trees)
attach(trees)
names(trees)
trees2<-
transform(trees,DIAM=Girth*2.54/100,HEIGHT=Height*30.48/100,VO
LUME=Volume*(30.48/100)^3)
trees2
trees3<-trees2[,c("DIAM","HEIGHT","VOLUME")]
trees3
attach(trees3)
names(trees3)
par(mfrow=c(1,2))
```

```
plot(VOLUME~DIAM, col="blue",xlab="Diameter (m)",ylab="Volum
(m^3)")
abline(lm(VOLUME~DIAM),col="red")
plot(VOLUME~HEIGHT, col="blue",xlab="Høyde (m)",ylab="Volum
(m^3)")
abline(lm(VOLUME~HEIGHT),col="red")
model1<-lm(VOLUME~DIAM)
summary(model1)
model2<-lm(VOLUME~HEIGHT)
summary(model2)
```



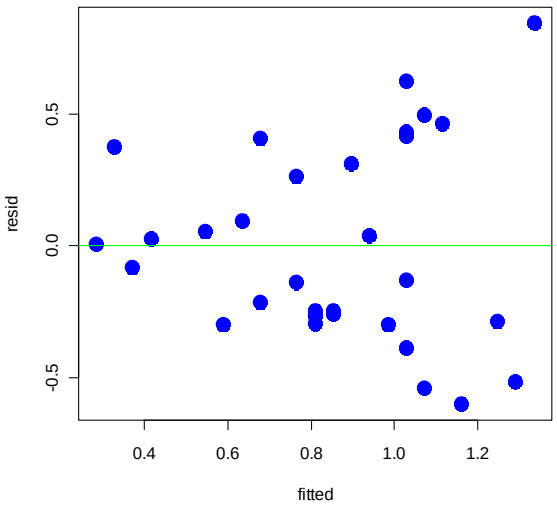
Men finner midtverdien og vipper linjen rundt dette punktet slik at de kvadrerte avvik blir minst mulig:

```
par(bg="lightyellow")
plot(HEIGHT,VOLUME,xlab="Høyde",ylab="Volum",pch=19,cex=2,col=
"blue")
abline(lm(VOLUME~HEIGHT),col="red",lwd=3)
lines(c(19,27),c(mean(VOLUME),mean(VOLUME)),lty=2,lwd=3,col="o
range")
abline(v=mean(HEIGHT),lty=2,lwd=3,col="orange")
points(mean(HEIGHT),mean(VOLUME),pch=24,col="orange",cex=3)
fitted<-predict(lm(VOLUME~HEIGHT))
for(i in
1:31)lines(c(HEIGHT[i],HEIGHT[i]),c(VOLUME[i],fitted[i]),lty=2
,lwd=2,col="green")
```



Vi plotter residualer mot tilpassete verdier. Det ser ut som residualene øker med økende verdier, noe som indikerer at dette kanskje ikke er en enkel lineær modell, noe vi skal se seinere.

```
fitted<-predict(lm(VOLUME~HEIGHT))
resid<-resid(lm(VOLUME~HEIGHT))
plot(fitted,resid,col="blue",pch=19,cex=2)
abline(h=0,col="green")
```



Lineær regresjon:
Modell 1:

```
lm(formula = VOLUME ~ DIAM)

Residuals:
      Min        1Q    Median        3Q       Max
-0.228386 -0.087972  0.004303  0.098961  0.271468
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.04612    0.09529  -10.98 7.62e-12 ***
DIAM         5.64760    0.27578   20.48  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1204 on 29 degrees of freedom
Multiple R-Squared: 0.9353,     Adjusted R-squared: 0.9331
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

En lineær modell er av typen

$$y = \alpha + \beta \cdot x + \varepsilon$$

Koeffisientene gir estimater av α og β, inkludert standardfeil, t-verdi og p-verdi og en eller flere * angir signifikansnivå.
I resultatet sies det litt om residualene og fordeling av disse, middelverdiene av residualene skal være lik 0, og medianverdien bør ligge i nærheten av dette. Man kan se på kvartilene om dataene ser balansert ut. Verdien for skjæingspunkt med y-aksen, α, avgjør om regresjonslinjen går gjennom origo eller ikke, men mest interesse knytter det seg til stigningskoeffisienten β. Nullhypotesen er β=0 og $t=\beta/SE_\beta$ for n-2. Korrelasjon er mål på assosiasjon mellom to variable og varierer fra -1 til +1, og 0 er ingen korrelasjon. Assosiasjon er ikke det samme som årsak. R^2 er kvadratet av Pearsons korrelasjonskoeffisient. Gjør F-test for hypotesen om at regresjonskoeffisienten er lik 0. Gir samme resultat som t-test for stigningen=0, F-test er kvadratet av t-test.
Estimat dividert på standardfeil (std.Error) gir t-verdien. Vi ser bort fra fortegnet for *t* siden det de absolutte forskjellene vi ser på. Deretter testes sannsynligheten for å finne en slik t-verdi sammenlignet med de kritiske tabellverdiene for *t*.

Vi finner de to regresjonslinjene (model1 og model2):
VOLUME = -1.046 + 5.648·DIAMET
VOLUME = -2.467 + 0.143·HEIGHT
R^2 angir hvor mye av variasjonen som kan forklares av regresjonslinjen, henholdsvis 93.5% og 35.8%. Modellen er signifikant og vi forkaster nullhypotesen om at β=0.
Kvadratsummene (SumSq) og F-verdi finner vi med

I avsnittet residuals sjekker man forutsetningene om medianverdien er ca. 0, samt at max- og min-verdien er omtrent like store.
To variable er **konfundert** (konfundering) hvis effekten på responsvariabel ikke kan atskilles.

```
summary(aov(model1))
            Df Sum Sq Mean Sq F value    Pr(>F)
DIAM         1 6.0794  6.0794  419.36 < 2.2e-16 ***
Residuals   29 0.4204  0.0145
```

R^2=SSR/SSY=6.0794/6.4998=93.5%

Kritisk verdi for *F* med 1 og 29 frihetsgrader (d.f.)og p=0.05:

```
qf(0.95,1,29)
[1] 4.182964
```

Sannsynligheten for å få en F-verdi lik 419.36 er omtrent lik 0 (ca. 10^{-16})

```
1-pf(419.36,1,29)
[1] 0
```

```
 trees3
      DIAM  HEIGHT    VOLUME
1  0.21082 21.3360 0.2916635
2  0.21844 19.8120 0.2916635
3  0.22352 19.2024 0.2888318
4  0.26670 21.9456 0.4643963
5  0.27178 24.6888 0.5323567
6  0.27432 25.2984 0.5578419
7  0.27940 20.1168 0.4417428
8  0.27940 22.8600 0.5153666
9  0.28194 24.3840 0.6399607
10 0.28448 22.8600 0.5635052
11 0.28702 24.0792 0.6852677
12 0.28956 23.1648 0.5946538
13 0.28956 23.1648 0.6059805
14 0.29718 21.0312 0.6031488
15 0.30480 22.8600 0.5408518
16 0.32766 22.5552 0.6286340
17 0.32766 25.9080 0.9571094
18 0.33782 26.2128 0.7758816
19 0.34798 21.6408 0.7277430
20 0.35052 19.5072 0.7050895
21 0.35560 23.7744 0.9769312
22 0.36068 24.3840 0.8976440
23 0.36830 22.5552 1.0279015
24 0.40640 21.9456 1.0845352
25 0.41402 23.4696 1.2062977
26 0.43942 24.6888 1.5687533
27 0.44450 24.9936 1.5772484
28 0.45466 24.3840 1.6508722
29 0.45720 24.3840 1.4583176
30 0.45720 24.3840 1.4441592
31 0.52324 26.5176 2.1803972
```

```
model2<-lm(VOLUME~HEIGHT)
 summary(model2)
Call:
lm(formula = VOLUME ~ HEIGHT)
Residuals:
     Min       1Q   Median       3Q      Max
-0.60242 -0.28018 -0.08195  0.34171  0.84532
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.46707    0.82892  -2.976 0.005835 **
HEIGHT       0.14338    0.03566   4.021 0.000378 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.3794 on 29 degrees of freedom
Multiple R-squared: 0.3579,     Adjusted R-squared: 0.3358
F-statistic: 16.16 on 1 and 29 DF,  p-value: 0.0003784
```

Som gir regresjonslinjen:

VOLUME ~ -2.46707 + 0.14338·HEIGHT + error

Source	df	SS	MS	F	F tabell (5%)

Regression	1	SSR	SSR	$F=SSR/s^2$	qf(0.95,1,n-2)
Error	n-2	SSE	$s^2=SSE/(n-2)$		
Total	n-1	SST			

ANOVA-tabell for regresjon, df-antall frihetsgrader, SS-kvadratsumm, MS-middelkvadratsum (varianse)

Varianse=SS/df
SST = SSR + SSE
y = a + bx
R^2= SSR/SST = 2.32631/6.499815 = 0.357904
Adjusted R^2= 1- [(SSE/(n-2))/(SST/(n-1))]= 1- [(4.173505/29))/(6.499815/30)]= 0.3357628

Fra R^2**=** 0.357904 ser vi at 35.8% av variasjonen kan forklares med regresjonslinjen.
R^2 øker med hvor mye variasjon som blir forklart.
Adjusted R^2 er lik:

$$R^2_{adj} = 1 - \left(\frac{\frac{SSE}{n-2}}{\frac{SST}{n-1}}\right) = 1 - \left(\frac{n-1}{n-p}\right)\cdot(1-R^2)$$

SSE er feilkvadratsummen, SST er totalkvadratsummen, *n* er prøvestørrelsen, *p* er antall parametere i modellen og p=2 for vanlig lineære regresjon. Adjusted R^2 minsker med antall parametere som tillegges modellen.

t-verdien får vi ved å dividere estimatene på standardfeilen.
F-verdien er:
$F=SSR/s^2$ = 2.32631/0.1439140= 16.16458
Tabellverdi for kritisk verdi av F:

```
 qf(0.95,1,29)
[1] 4.182964
```

Sannsynligheten for å få en F-verdi vi har fått:

```
 1-pf(16.16458,1,29)
[1] 0.0003783709
```

Det vil si vi forkaster nullhypotesen om et det ikke er noen lineær sammenheng mellom VOLUME og HEIGHT.
Responsvariabel VOLUME (y) og forklaringsvariabel HEIGHT (x)
Trenger følgende tall:
$\sum x$ = 718.1088
$\sum x^2$= 16748.00
$\sum y$ = 26.48475
$\sum y^2$= 29.12697
$\sum xy$= 629.7384
$(\sum y)^2$ = 701.4418
$(\sum x)^2$ = 515680.2

```
 sum(HEIGHT)
[1] 718.1088
 sum(VOLUME)
[1] 26.48475
 sum(HEIGHT^2)
```

```
[1] 16748.00
 sum(VOLUME^2)
[1] 29.12697
 sum(HEIGHT*VOLUME)
[1] 629.7384
 (sum(HEIGHT)^2)
[1] 515680.2
 (sum(VOLUME)^2)
[1] 701.4418
```

Kovarianse mellom VOLUME og HEIGHT:

```
 VOLUME*HEIGHT
 [1]  6.222933  5.778438  5.546264 10.191455 13.143248 14.112507  8.886452
 [8] 11.781281 15.604803 12.881730 16.500698 13.775036 14.037417 12.684944
[15] 12.363871 14.178965 24.796791 20.338029 15.748940 13.754322 23.225953
[22] 21.888152 23.184525 23.800776 28.311324 38.730637 39.421114 40.254867
[29] 35.559616 35.214377 57.818900
```

SST = $\sum y^2 - 1/n(\sum y)^2$

SSX = $\sum x^2 - 1/n\ (\sum x)^2$

SSXY = $\sum xy - 1/n(\sum x \sum y)$

SST = 29.12697 – 1/31*701.4418 = 6.499815

SST = SSR + SSE = 6.0794 + 0.4202 = 6.4998

SSX = 16748.00 – 1/31*515680.2 = 113.1548

SSXY = 629.7384 – 1/31* (718.1088 *26.48475) = 16.22446

Stigningskoeffisienten b:

b =SSXY/SSX = 16.22446/113.1548 = 0.1433829

Skjæring med y-aksen (linje gjennom stormiddeltallet)

a = $y_m - bx_m$ = $\sum y/31 - b*\sum x/31$ = 26.48475/31 – 0.1433829 * 718.1088 /31= -2.467089

VOLUME = -2.467089 + 0.1433829*HEIGHT

Regresjonskvadratsummen SSR:

SSR = b*SSXY = 0.1433829 *16.22446 = 2.32631

Feilkvadratsummen SSE:

SSE = SST – SSR = 6.499815 – 2.32631 = 4.173505

Standardfeilen til regresjonsstigningskoeffisienten b:

s^2 = SSE/(n-2) = 4.173505/29 = 0.1439140

SE_b=$\sqrt{s^2}$/SSX = √0.1439140/113.1548 = 0.03566277

Standardfeilen til skjæring med y-aksen (a): (x_m-gjennomsnitt av x, √ = sqrt)

SE_a= $\sqrt{s^2}[1/n+ \sum x^2/SSX]$= √ ((0.1439140*16748.00)/(31 * 113.1548))= 0.8289258

t-verdier:

t_a = a/SE = -2.467089/0.8289258= -2.976248

t_b = b/SE = 0.1433829 /0.03566277 = 4.020521

Sannsynligheten for å få en slik t-verdi:

```
2*pt(-2.976248,29)
[1] 0.005834442
```

95% konfidensintervall (CI_b) for b:

CI_b= t (α=0.025, df=(n-2))·SE_b

Kritisk verdi for t:

```
 qt(0.975,29)
[1] 2.045230
```

Vi må beregne standardfeilen til skjæringspunkt og stigningskoeffisient, og konfidensintervall for *a* og *b*:

CI_b = 0.1433829 ± 2.045230*0.03566277= 0.1433829 ± 0.07293857

CI_a = -2.467089 ± 2.045230*0.8289258= -2.467089 ± 1.695344

95% konfidensintervall for parametere kan finnes med kommandoen **confint()**.

```
 confint(model2)
                 2.5 %     97.5 %
(Intercept) -4.16240288 -0.7717291
HEIGHT       0.07044363  0.2163201
```

Som er det samme som vi har funnet over

Vi kan bruke modellen til å prediktere verdier. Hvor stort er volumet ved høyde 23 meter ?

```
 predict(model2,list(HEIGHT=23))
        1
0.8307173
```

Svar: 0.83 m^3.

Vi kan finne den totale variansen SST også på en annen måte via nullmodellen lm(VOLUME~1):

```
 deviance(lm(VOLUME~1))
[1] 6.499813
```

Feilkvadratsummen (SSE) finner vi ved:

```
 deviance(lm(VOLUME~HEIGHT))
[1] 4.173513
```

SST=SSR+SSE, dvs. SSR=6.499813-4.173513= 2.3263

Som vi også finner igjen i ANOVA-tabellen:

```
 summary(aov(lm(VOLUME~HEIGHT)))
            Df Sum Sq Mean Sq F value    Pr(>F)
HEIGHT       1 2.3263  2.3263  16.165 0.0003784 ***
Residuals   29 4.1735  0.1439
```

R^2 som angir hvor stor prosent var variasjonen som blir forklart av regresjonsmodellen:

$$R^2 = \frac{SST - SSE}{SST} = \frac{6.499813 - 4.173513}{6.499813} = 0.3579026$$

Som er det samme som:

```
 summary(lm(VOLUME~HEIGHT))[[8]]
```

```
[1] 0.3579026
```

Korrelasjonskoeffisienten r:

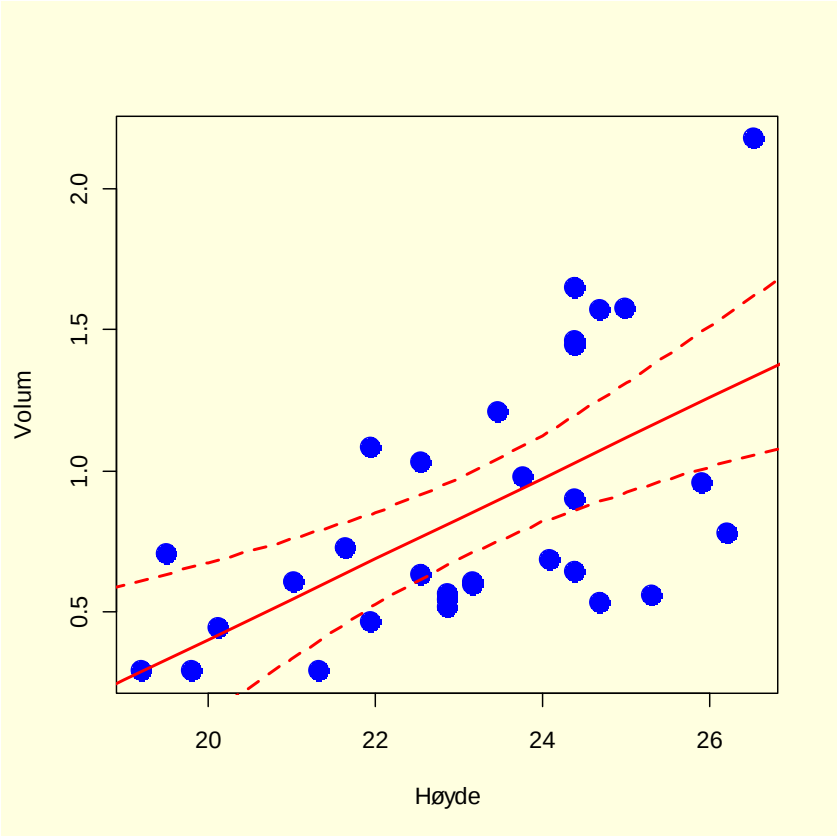
$$r = \frac{SSXY}{\sqrt{SSX \cdot SST}} = \frac{16.22446}{\sqrt{113.1548 \cdot 6.499813}} = 0.5982509$$

Som er det samme som:

```
cor(VOLUME,HEIGHT)
[1] 0.5982497
```

Vi kan plotte konfidensintervallet:

```
plot(HEIGHT,VOLUME,xlab="Høyde",ylab="Volum",pch=19,cex=2,col=
"blue")
hoyde<-data.frame(HEIGHT=18:27)
conf<-predict(lm(VOLUME~HEIGHT),int="c",newdata=hoyde)
predheight<-hoyde$HEIGHT
matlines(predheight,conf,lty=c(1,2,2),col="red",lwd=2)
```



Matriser i statistiske modeller

Skrevet som matrise er en lineær regresjonsmodell lik

$$Y = X \cdot b + e$$

$$y_i = \alpha + \beta \cdot x_i + e_i \quad for\ i = 1,2,3, \ldots n$$

hvor *e* er normalfordelte feil-ledd (error) og vi ønsker å bestemme *b*.

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{pmatrix} = \begin{pmatrix} \alpha + \beta \cdot x_1 + e_1 \\ \alpha + \beta \cdot x_2 + e_2 \\ \alpha + \beta \cdot x_3 + e_3 \\ \vdots \\ \alpha + \beta \cdot x_n + e_n \end{pmatrix}$$

Nå skal vi gjøre det samme regnestykket med datasett trees3, men nå i form av matriser:

```
 VOLUME%*%HEIGHT
         [,1]
[1,] 629.7384
 VOLUME%*%VOLUME
         [,1]
[1,] 29.12697
 HEIGHT%*%HEIGHT
         [,1]
[1,] 16748.00
```

Vi lager matrisen Y med VOLUME.

```
Y<-VOLUME
```

I matrisen X må vi lage en kolonne med 1-tall og binde disse sammen med HEIGHT.

```
X<-cbind(1,HEIGHT)
```

$\sum y^2 = 29.12697$

```
 t(Y)%*%Y
         [,1]
[1,] 29.12697
```

Vi finner en matrise som inneholder n, $\sqrt{x}$ og $\sqrt{x^2}$ ved å ta den transponerte matrisen ${}^{t}X$ og gange den med X:

```
 tXX<-t(X)%*%X;tXX
                  HEIGHT
       31.0000   718.1088
HEIGHT 718.1088 16748.0026
```

Vi finner en matrise som inneholder $\sum y$ og $\sum xy$:

```
 tXY<-t(X)%*%Y;tXY
           [,1]
       26.48475
HEIGHT 629.73836
```

Vi løser matriseligningene og finner b

```
 b<-solve(tXX,tXY);b
            [,1]
      -2.4670660
HEIGHT  0.1433819
```

Som vi ser er lik koeffisientene i modellen:

VOLUME ~ -2.46707 + 0.14338·HEIGHT + error

$$\begin{pmatrix} 31 & 718.1088 \\ 718.1088 & 16748.0026 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} 26.48475 \\ 629.73836 \end{pmatrix} \rightarrow \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} = \begin{pmatrix} -2.4670660 \\ 0.1433819 \end{pmatrix}$$

Som er det samme som de to ligningene:

$$\alpha \cdot n + \beta \cdot \sum x = \sum y$$
$$\alpha \cdot \sum x + \beta \cdot \sum x^2 = \sum xy$$

De samme regneoperasjonene kan nå gjøres i matriseform hvis vi har m>1 x-variable, noe som viser den store fleksibiliteten ved å regne med matriser.

Stigningskoeffisienten b

$$b = \frac{\sum(x - \bar{x}) \cdot (y - \bar{y})}{\sum(x - \bar{x})^2}$$

Skjæringspunktet a:

$$a = \bar{y} - b \cdot \bar{x}$$

Faktoriell design og modellseleksjon

Modeller har følgende oppbygning, for eksempel med 2 prediktorvariable (uavhengige variabel, forklaringsvariabel, faktor, prediktor).

$$Responsvariabel \sim prediktorvariabel1 + prediktorvariabel2 + error(støy)$$

Responsvariabel kalles også **avhengig variabel.**
Det er to hovedtyper kategoriske forklaringsvariabel fikserte effekter (**fixed effects**,hann-hunn, gjødsel-ikke gjødsel, lys-skygge, tørr-våt), og tilfeldige effekter (**random effects**,blokker, split plot i plot, genotype).

Det er mange typer modeller: Lineære modeller (**lm**, **aov**),multivariable modeller (**manova**), generaliserte lineære modeller med linkfunksjoner (**glm**), lineære miksete modeller (**lme**, **aov**), ikke-linære modeller (**nls**), ikke-lineære miksete modeller (**nlme**), og generaliserte additive modeller (**gam**).

Vi laster inn datasettet "trees" som ligger i R og som gir oversikt over omkrets, høyde og volum av svartkirsebærtrær:

```
library(datasets); data(); data(trees)
```

og transformerer måleverdiene til meter (se ovenfor)

```
names(trees3)
[1] "DIAM"   "HEIGHT" "VOLUME"
```

Hvis vi forutsetter at VOLUME er normalfordelt kan vi lage følgende modell:

```
model3<-glm(VOLUME~DIAM+HEIGHT)
summary(model3)
```

Gir modellen med tilhørende AIC-verdi:-44.077

VOLUME = -1.64203 + 5.24883·DIAMET + 0.03152·HEIGHT

```
glm(formula = VOLUME ~ DIAM + HEIGHT)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
```

```
-0.181411  -0.075021  -0.008143   0.062306   0.240260

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.64203    0.24461  -6.713 2.75e-07 ***
DIAM         5.24883    0.29461  17.816  < 2e-16 ***
HEIGHT       0.03152    0.01209   2.607   0.0145 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.01208268)

    Null deviance: 6.49981  on 30  degrees of freedom
Residual deviance: 0.33832  on 28  degrees of freedom
AIC: -44.077

Number of Fisher Scoring iterations: 2
```

Som vi kan plotte for å teste forutsetningene:

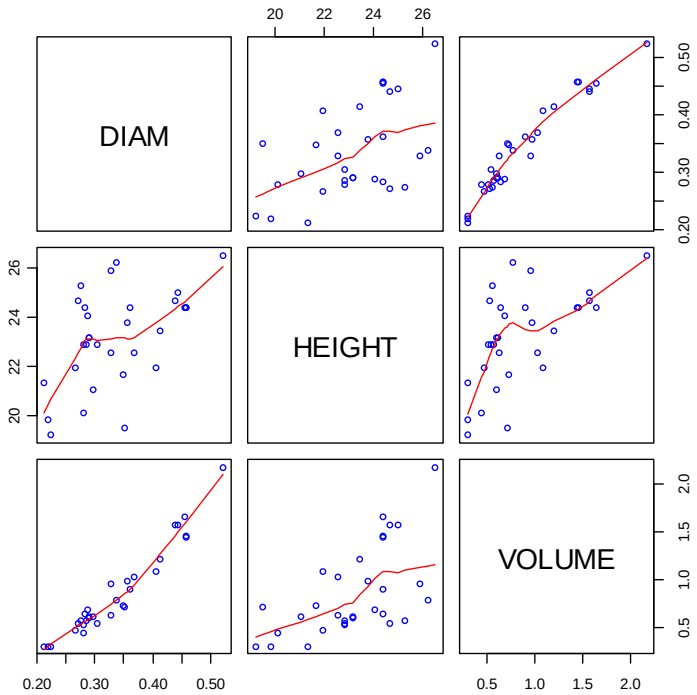
```
par(mfrow=c(2,2))
plot(model3)
```

Vi kan se at det er interaksjon mellom diameter og høyde:

```
model3B<-glm(VOLUME~DIAM*HEIGHT)
summary(model3B)
```

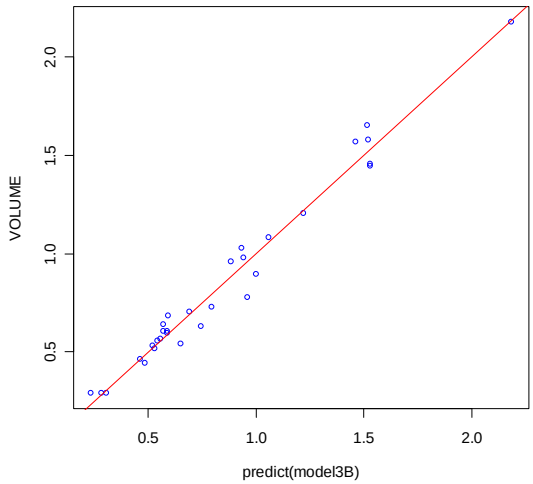
Plotting av datasettet:

```
pairs(trees3,panel=panel.smooth,col="blue")
```



Vi kan undersøke vår måleverdier mot de predikterte verdiene, og hvis modellen er god bør punktene ligge på en rett linje stigning 45°, hvilket de ser ut til å gjøre:

```
plot(predict(model3B),VOLUME,col="blue")
abline(a=0,b=1,col="red")
```



Vi kan også plotte residualene (restene) mot predikterte verdier og i dette plottet bør ikke punktene danne noe mønster:

```
plot(predict(model3B),residuals(model3B),col="blue")
```

Ved **Box-Cox transformasjon** kan vi finne den beste linkfunksjonen for modellen ved å bestemme lambda (λ). Dette er en potenstransformasjon hvor vi erstatter y med y^{λ}. Logtransformasjon tilsvarer λ=0.

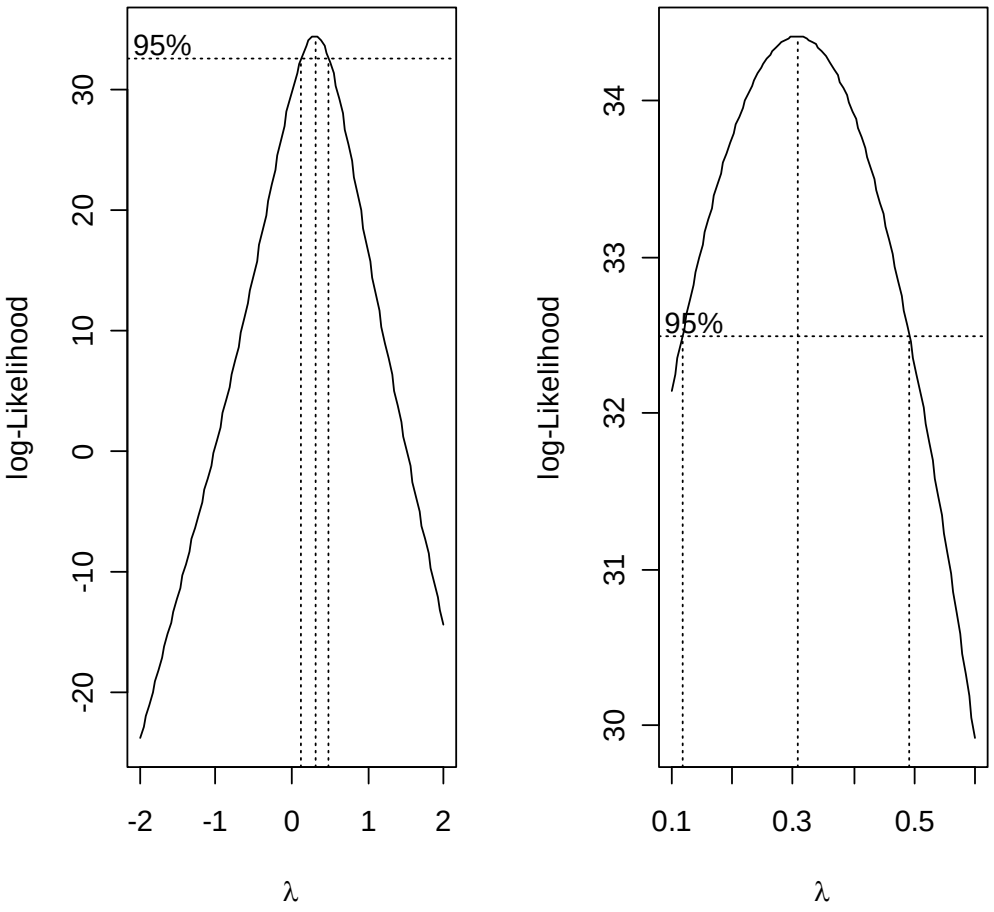
$$y(\lambda) = \frac{y^{\lambda} - 1}{\lambda} \quad hvis\ \lambda \neq 0$$
$$y(\lambda) = \log(y) \quad hvis\ \lambda = 0$$

Hvis det er de minste verdiene av variabelen som trengs å spres bruker man liten λ, og hvis de største verdiene skal spres bruker man stor λ.
λ=0 tilsvarer logtransformasjon (ln)
λ=1 lineær transformasjon
λ=1/2 kvadratrottransformasjon

Hvis data har høyreskew: prøv ln(y), 1/y, kvadratrottransformasjon (√y)
Hvis venstreskew: prøv y^2
For %-data som ikke er normalfordelt prøv arcsin-transformasjon: Divider %-verdien på 100. Ta kvadratroten av tallet. Velg $\sin^{-1}$ eller asin. R gir verdier i radianer, hvis man ønsker grader fra 0-90 multipliser med 180/π. Generelt har generaliserte lineære modeller redusert behovet for transformering av data.
Etter at man har gjort dataanalysen på transformerte data, må man **tilbaketransformere** data før de presenteres. Har man log-transformert må man foreta antilog.

Vi starter biblioteket **library(MASS)** og finner at lamda er ca. 0.3

```
library(MASS)
par(mfrow=c(1,2))
boxcox(VOLUME~DIAM+HEIGHT)
boxcox(VOLUME~DIAM+HEIGHT,lambda=seq(0.1,0.6,0.01))
```

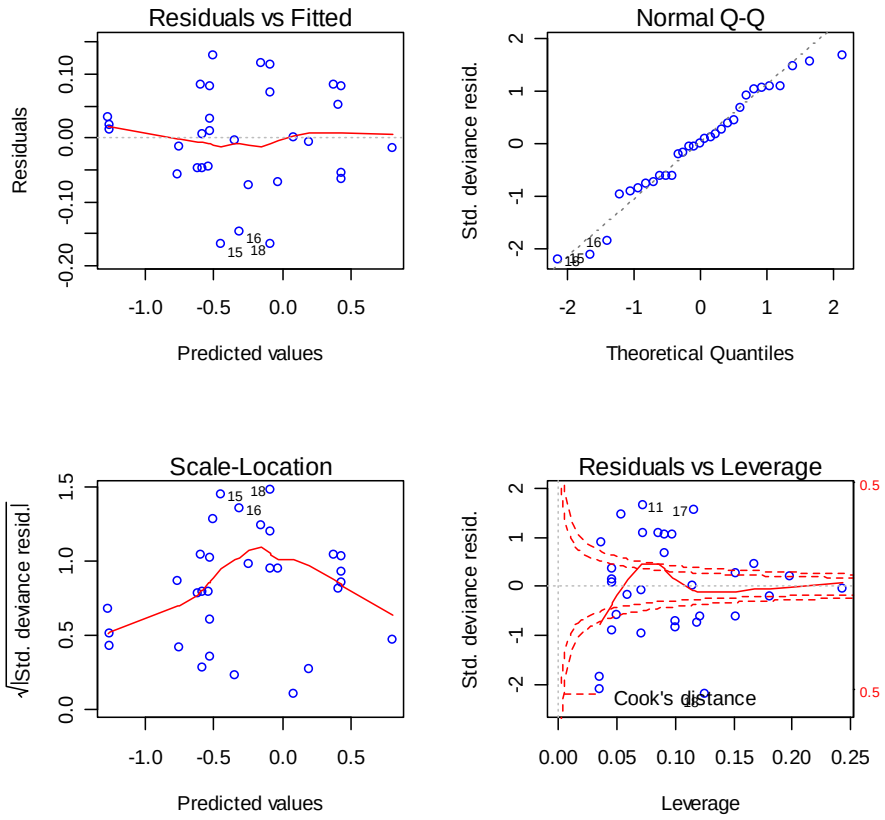


Som angir forslag til en linkfunksjon opphøyd i 1/3

```
model4<-glm(VOLUME~DIAM+HEIGHT,family=quasi(link=power(1/3)))
summary(model4)
plot(model4)
```

Eller en logaritmemodell med transformering av forklaringsvariable med Gammafordeling:

```
model5<-
glm(VOLUME~log(DIAM)+log(HEIGHT),family=Gamma(link=log))
summary(model5)
plot(model5,col="blue")
```



Man må undersøke hvor godt modellen beskriver data. Vi kan se på restene (residualene) som er forskjellen mellom de modellpredikterte dataene og de virkelige dataene

$$residualer\ (rester) = \ observerte\ y - predikterte\ y$$

Øverst t.v. undersøkes om det er **heteroskedastitet** (ikke-konstant varianse) og residualene plottes mot predikterte verdier, og i denne figuren bør punktene ligge helt tilfeldig fordelt og ikke danne noe mønster. **Homoskedastitet,** det vil si konstant varianse, er en nødvendig forutsetning for å kunne gjøre varians- eller regresjonsanalyse. Hvis forutsetningene for normalfordelingskal være tidfredsstilt bør kvantil-kvantil plot (normal kvantilplot) gi en rett linje hvor måleverdiene er plottet mot z-skår. Cook´s avstand brukes til å finne punkter (utliggerverdier) som er spesielt innflytelsesrike i bestemmelsen av parametere i regresjonsmodellen. Man fjerner en observasjon ad gangen fra regresjonen og finner hvor mye residualfeilen påvirkes når datapunktet er fjernet. Leverage (innflytelse) er et mål på viktigheten av en prøve i regresjonsmodellen relatert til Mahalanobis-avstand.
Eksempel på en generalisert lineær modell, kommando **glm()**, er logistisk regresjon: **glm(modell,family=…)** hvor **family=binomial** ved logistisk regresjon.
Ikke-lineære modeller kan undersøkes med kommandoen **nls()**: **nls(modell,start=c(…),trace=False)**. Kommandoen **predict()** kan brukes til å plotte predikterte verdier ut fra modellen.

AIC-verdier finnes med **AIC(modell)** eller ved **stepAIC()** for trinnvis.
Akaike informasjonskriterium (AIC) brukes i modellseleksjon og beregnes som:

$$AIC = -2 \cdot \ln(\text{likelihood}) + 2p$$

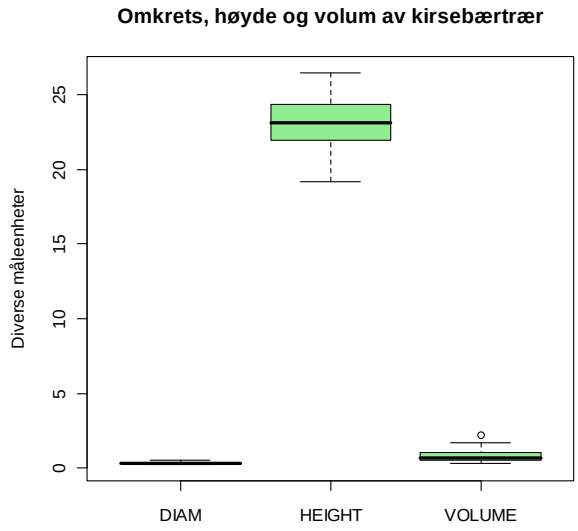
hvor p er antall parametere i modellen.
Bayesiansk informasjonskriterium (BIC) brukes også til modellseleksjon

$$BIC = -2 \cdot \ln(likelihood) - (p)\ln(n)$$

Hvor n er prøvestørrelsen, og likelihood-funksjonen er observerte data ($x_1, x_2, ..x_n$) beskrevet av parameterverdiene p (eller hypotesen H):$L(x_1, x_2, ..x_n|p$ (eller H))

Kommandoen **expand.grid** brukes til å lage en dataramme for alle kombinasjoner av faktorer.

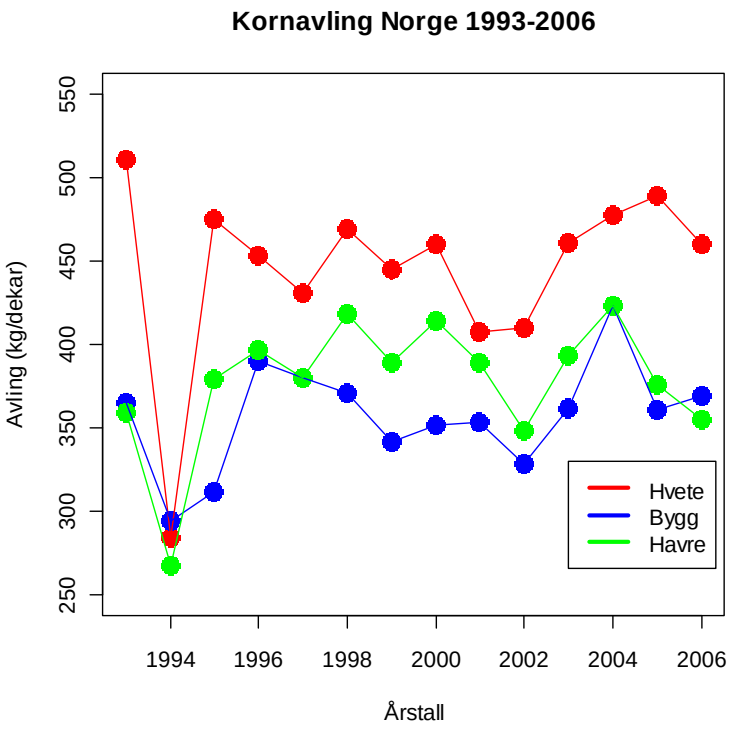
```
boxplot(trees3,col="lightgreen",ylab="Diverse måleenheter",
main="Omkrets, høyde og volum av kirsebærtrær")
```

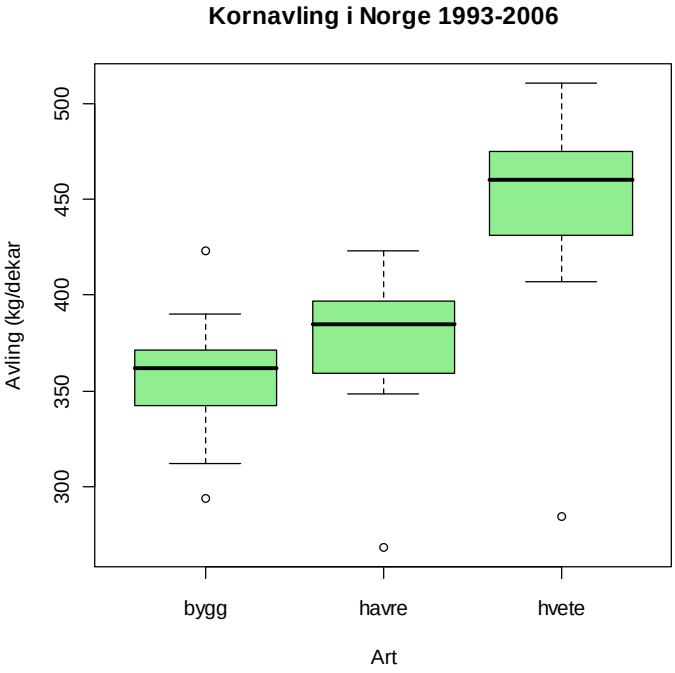


Avling av hvete, bygg og havre SSB 2006, 339:

```
år<-c(seq(1993,2006,1),seq(1993,2006,1),seq(1993,2006,1))
art<-c(rep("hvete",14),rep("bygg",14),rep("havre",14))
avling<-c(511,284,475,453,431,469,445,460,407,410,461,477,489,460,
365,294,312,390,380,371,342,352,353,328,362,423,361,369,
359,268,379,397,380,418,389,414,389,348,393,423,376,355)
biomasse<-data.frame(år,art,avling);biomasse
boxplot(avling~art,col="lightgreen",ylab="Avling (kg/dekar",xlab="Art",
main="Kornavling i Norge 1993-2006")
plot(år[art=="hvete"],avling[art=="hvete"],col="red",pch=16,cex=2,ylim=c(25
0,550),ylab="Avling (kg/dekar)",
xlab="Årstall",main="Kornavling Norge 1993-2006")
points(år[art=="bygg"],avling[art=="bygg"],col="blue",pch=16,cex=2)
points(år[art=="havre"],avling[art=="havre"],col="green",pch=16,cex=2)
lines(lowess
(år[art=="hvete"],avling[art=="hvete"],f=0.1,iter=3,),col="red")
lines(lowess
(år[art=="bygg"],avling[art=="bygg"],f=0.1,iter=3,),col="blue")
```

```
lines(lowess
(år[art=="havre"],avling[art=="havre"],f=0.1,iter=3,),col="green")
legend(2003,330,c("Hvete","Bygg","Havre"),
lty=c(1,1,1),lwd=c(3,3,3),col=c("red","blue","green"))
```





Sjekk for normalfordeling:

```
qqnorm(avling,col="blue")
qqline(avling,col="red")
```

Enveis variansanalyse. Nullhypotesen er at det er ingen forskjell i avling mellom de tre artene. Art er en faktor:

```
art<-factor(art)
```

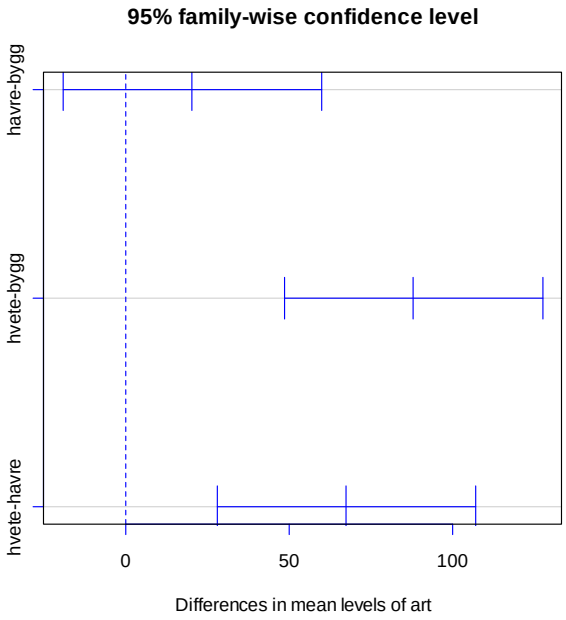
```
is.factor(art)
model<-aov(avling~art)
summary(model)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
art          2  59186   29593  16.139 7.82e-06 ***
Residuals   39  71513    1834
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nullhypotesen forkastes.Variansanalyse tester om det er signifikante forskjeller mellom prøver, men angir ikke hvem som er forskjellig fra hvem. Det finnes imidlertid en rekke post-hoc tester som gjør dette bl.a. Tukey´s Honestly Significant Difference:

```
TukeyHSD(aov(avling~art))
plot(TukeyHSD(aov(avling~art)),col="blue")
```

```
Tukey multiple comparisons of means
    95% family-wise confidence level
Fit: aov(formula = avling ~ art)
$art
                diff       lwr       upr     p adj
havre-bygg  20.42857 -19.00305  59.86019 0.4247120
hvete-bygg  87.85714  48.42552 127.28876 0.0000095
hvete-havre 67.42857  27.99695 106.86019 0.0004783
```



Horisontale linjer som krysser den prikkete er ikke signifikant forskjellige. Ingen signifikant forskjell mellom havre og bygg.

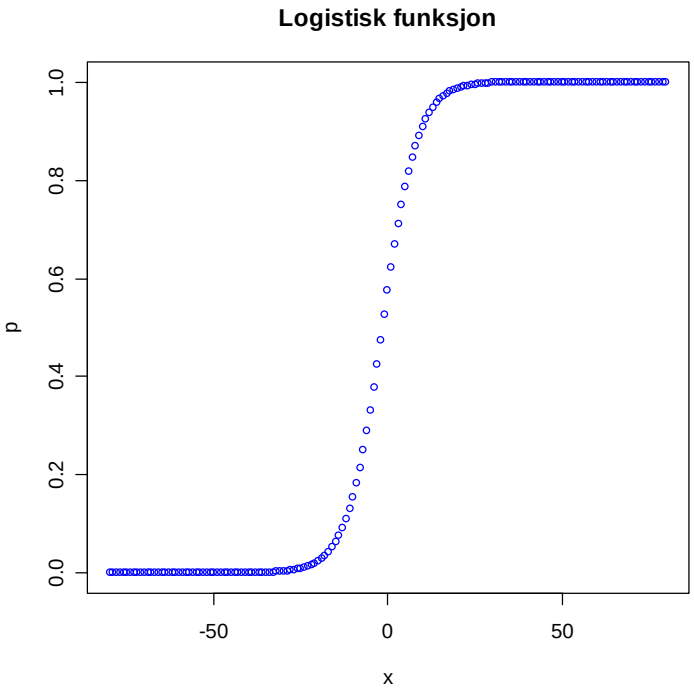
Logistisk regresjon

Binomiale data har to mulige verdier.Vi kan modellere sannsynligheter på en transformert skala.
En **logistisk ligning** eller modell er av typen:

$$p = \frac{e^y}{1+e^y}$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

```
x<-seq(-80,80,1)
a=0.3
b=0.2
p<-(exp(a+b*x))/(1+exp(a+b*x))
plot(x,p,col="blue", main="Logistisk funksjon")
```



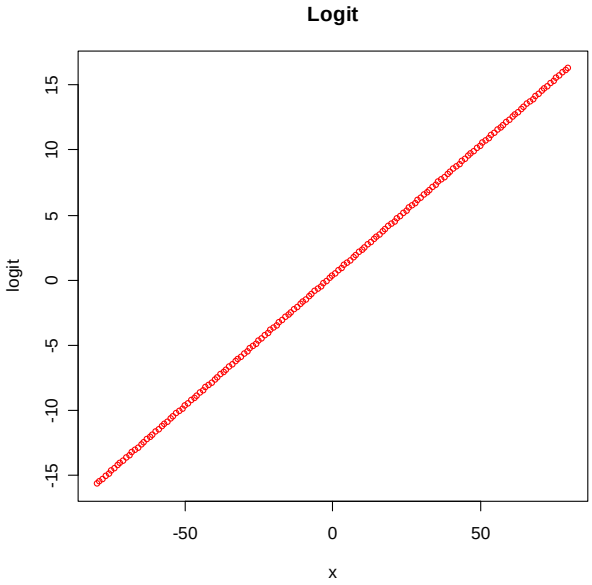
En logistisk ligning kan bli transformert til lineær form ved å bruke linkfunksjonen **logit** som er det samme som logartimen til odds (log odds). Odds (p/q), sannsynligheten for at noe skal inntreffe dividert på sannsynligheten for at noe ikke skal inntreffe brukes av bookmakere i veddemålsammenheng, mens *p* er vanlig i vitenskapelig bruk. Logaritmen til odds kalles logit. Sannsynligheten for å velge en tilfeldig ukedag er 1/6, seks til 1, 1 er sannsynlighten for å lykkes, 6 er sannsynligheten for å feile.

$$\text{logit}\ p = \ln\frac{p}{1-p} = \ln\frac{p}{q} \quad hvor\ q = 1-p\ og\ \frac{p}{q} = odds$$

Man kan da vise at logit p blir en rett linje:

$$\text{logit}\ p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

```
logit<-log(p/(1-p))
plot(x,logit,col="red",main="Logit")
```



I lineær regresjon med **minste kvadraters metode** vil man minimalisere summen av kvadrerte residualer noe som gir maksimal R^2. Ved logistisk bruker man i stedet et maksimum likelihood estimat (MLE). Man gjør en gjetning om hvor godt parameterverdiene er tilpasset dataene, beregner en log-likelihoodfunksjon som et mål på hvor god tilpasningen er. Parameterverdiene endres litt fram og tilbake ved iterasjoner og man ser om man får et bedre estimat, beholder det beste estimatet som til slutt konvergerer mot en sluttverdi. Ved MLE ønsker man å maksimere log-likelihoodfunksjonen (LL). LL blir negativ, og vanligvis multipliseres log-likelihoodfunksjonen med -2 (-2 log-likelihoodfunksjonen, -2LL) slik at den blir positiv. Ved å multiplisere med 2 følger log-likelihoodfunksjonen kji-kvadratfordeling som kan brukes for testing av signifikans. Jo lavere verdi på -2LL, desto bedre tilpasset er parameterverdiene til dataene, og optimalt er -2LL=0. Vi bruker kji-kvadrat(χ^2) i MLE og tester signifikans, etter samme prinsipp som bruk av F i ANOVA.

$$\chi^2 = (-2 \cdot LL_0) - (-2 \cdot LL_M)$$

Hvor $-2 \cdot LL_0$ er null-modellen før man har satt inn variablene og $-2 \cdot LL_M$ er modellen etter at variablene er satt inn. Dette blir også et mål på **devianse**. Antall frihetsgrader (df) er antall uavhengige variable i modellen man tester minus antall uavhengige variable i null-modellen (df=0). Ved lineær regresjon hadde vi R^2 som mål på korrelasjon mellom avhengig variabel og predikert verdi av avhengig variabel (responsvariabel)

Vi har sannsynligheten (Pr-probabilitet) for en hendelse A:

$$\Pr(A) = \frac{gunstige\ utfall}{mulige\ utfall}$$

Ved kasting av en mynt er vi to mulige utfall, kron eller mynt, og hvis vi sier kron er et gunstig utfall blir sannsynligheten Pr(kron)=1/2=0.5.
Hvis vi kaster en terning så er det 6 mulige utfall (1-6) og sannsynligheten for å få en sekser (gunstig utfall) er Pr(6)=1/6= 0.1666667. Sannsynligheten for å få 1-5 er Pr(1-5)=5/6= 0.8333333.
Sannsynlighet har den ulempen at den må ligge mellom 0-1. Vi kan også uttrykke sannsynlighet i form av odds (odds-ratio) som ikke har denne begrensningen med ikke kunne være større enn 1, hvor

$$odds = \frac{p}{q} = \frac{p}{1-p}$$

Odds for ikke å få en sekser er =0.8333333/ 0.1666667 = 5.
Et odds-ratio som er lik 0 betyr at en hendelse aldri vil skje, og for å kunne få verdier som er mindre enn 0 bruker vi i stedet ln(odds)som kan bli mindre enn 0 og logit kan gå fra ±∞. Kjenner vi sannsynligheten p kan vi regne ut odds og logit, og kjenner vi logit eller odds kan vi regne ut sannsynlighet.

$$p(hendelse) = \frac{odds(hendelse)}{\left(1 + odds\ (hendelse)\right)}$$

Sannsynligheten for ikke å få en sekser er p=5/1+5=0.8333333
Logit=ln(odds) brukes som en link-funksjon for å koble en lineær prediktor til p:

$$logit = ln(odds) = ln\left(\frac{p}{q}\right) = ln\left(\frac{1}{1-p}\right) = a + bx$$

For en logit-modell har vi generelt:

$$log\frac{p}{1-p} = logit = \ln(odds) = a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_n \cdot x_n + \varepsilon$$

Dette tilsvarer:

$$y = g(a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_n \cdot x_n + \varepsilon)$$
$$= \frac{exp(a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_n \cdot x_n + \epsilon)}{1 + \exp\left(a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_n \cdot x_n + \epsilon\right)}$$

Hvor funksjonen *g* tilsvarer en logit-transformasjon

Her er et fiktivt eksempel på et studium hvor man undersøker sammenhengen mellom angina (hjerteproblemer), røyking, spising av frukt, trimming og drikking av litt rødvein. Binære data 1 (positivt) og 0 (negativt). Datasettet har altfor liten teststyrke til så mange variable, *n* burde være mange hundre, eller best tusenvis, men her brukes det bare som et eksempel på logistisk regresjon. Man kunne også operert med variable inndelt i diskret telling for eksempel 1(lite),2,3,4,5(mye).

```
janei<-c("ja","nei");janei
A<-
expand.grid(røyking=janei,frukt=janei,jogging=janei,rødvin=jan
ei);A
#Simulerte data
A$TOTAL<-c(21,23,27,12,31,15,23,19,31,29,16,22,18,11,17,24);A
A$ANGINA<-c(4,1,5,0,3,2,12,2,4,7,2,3,0,1,10,0);A
```

```
   røyking frukt jogging rødvin TOTAL ANGINA
1       ja    ja      ja     ja    21      4
2      nei    ja      ja     ja    23      1
3       ja   nei      ja     ja    27      5
4      nei   nei      ja     ja    12      0
5       ja    ja     nei     ja    31      3
6      nei    ja     nei     ja    15      2
7       ja   nei     nei     ja    23     12
8      nei   nei     nei     ja    19      2
9       ja    ja      ja    nei    31      4
10     nei    ja      ja    nei    29      7
11      ja   nei      ja    nei    16      2
12     nei   nei      ja    nei    22      3
13      ja    ja     nei    nei    18      0
14     nei    ja     nei    nei    11      1
15      ja   nei     nei    nei    17     10
16     nei   nei     nei    nei    24      0
```

Vi kan enten utføre logistisk regresjon på ratio eller på matrisen for totalantall og antall med angina. Vi ser at modellene blir like. Ved ratio må totalantallet oppgis som weights.

```
ratio<-A$ANGINA/A$TOTAL
modell1<-
glm(ratio~røyking+frukt+jogging+rødvin,data=A,family=binomial,
weights=TOTAL)
summary(modell1)
```

```
Call:
glm(formula = ratio ~ røyking + frukt + jogging + rødvin, family =
binomial,
    data = A, weights = TOTAL)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8498  -1.2999  -0.3580   0.6391   3.0232

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.79936    0.32335  -5.565 2.63e-08 ***
røykingnei  -0.93637    0.32560  -2.876  0.00403 **
fruktnei     0.69896    0.30468   2.294  0.02179 *
joggingnei   0.27335    0.30071   0.909  0.36334
rødvinnei    0.07252    0.30257   0.240  0.81057
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 57.978  on 15  degrees of freedom
Residual deviance: 43.220  on 11  degrees of freedom
AIC: 90.162

Number of Fisher Scoring iterations: 5
```

```
modell2<-glm(cbind(ANGINA,TOTAL-
ANGINA)~røyking+frukt+jogging+rødvin,data=A,family=binomial)
summary(modell2)
#konfidensintervall for parameterverdier
confint(modell1)
Waiting for profiling to be done...
                 2.5 %     97.5 %
(Intercept) -2.4592010 -1.1876443
røykingnei  -1.5993445 -0.3158746
fruktnei     0.1085207  1.3079138
joggingnei  -0.3158536  0.8675678
rødvinnei   -0.5225249  0.6681882
```

Vi finner antydning til signifikans for røyking og ikke spise frukt. Koeffisienttabellen over betyr følgende modell siden vi bruker linkfunksjonen for binomial:

Logit(ratio)~ -1.79936 -0.93637·røykingnei + 0.69896·fruktnei + 0.27335 ·joggingnei + 0.07252·rødvinnei

Det er noe overdispersjon slik at man kan prøve family=quasibinomial

For å komme fra logit til odds ratio:
Sannsynligheten p
P=odds/(1+odds)

Ikke-lineære funksjoner

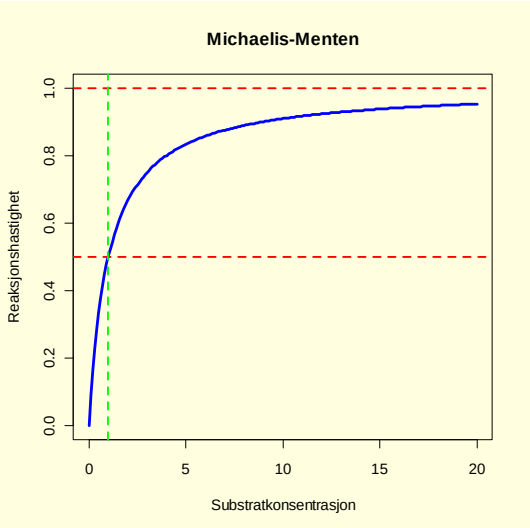
Michaelis-Menten funksjon

Michaelis-Menten ligningen beskriver enzymkinetikk hvor v er initialhastigheten på reaksjonen og S er substratkonsentrasjonen, V_{max} er maksimal reaksjonshastighet og K_m er substratkonsentrasjonen som gir halvparten av maksimal reaksjonshastighet. Funksjonen nærmer seg asymptotisk til en maksimalverdi.

$$v = \frac{S \cdot V_{max}}{S + K_m}$$

Vi setter V_{max}=1 og K_m=1 i vårt tilfelle:

```
Vmax=1
S<-seq(0,20,0.1)
Km=1
v<-S*Vmax/(S+Km)
par(bg="lightyellow")
plot(S,v,type="l",lwd=3,col="blue",ylim=c(0,1),xlab="Substratkonsentrasjon"
,ylab="Reaksjonshastighet",main="Michaelis-Menten")
abline(h=1,lty=2,col="red",lwd=2)
abline(h=0.5,lty=2,col="red",lwd=2)
abline(v=1,lty=2,col="green",lwd=2)
```

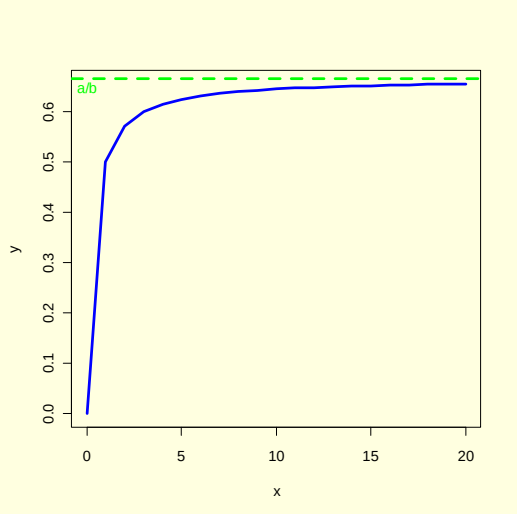


Mange funksjoner i biologi og biokjemi har samme form som Michaelis-Menten-kurven, men disse har fått sine egne navn, selv om de i prinsippet er like. Monodkurve (1942) har substratkonsentrasjon på x-aksen og spesifikk vekstrate på y-aksen, og har samme form som Michaelis-Menten kurven. Funksjonell responskurve (Nicholson-Bailay-modell) beskriver relasjonen mellom antall verter og tettheten av parasitter.

En tilsvarende funksjon er :

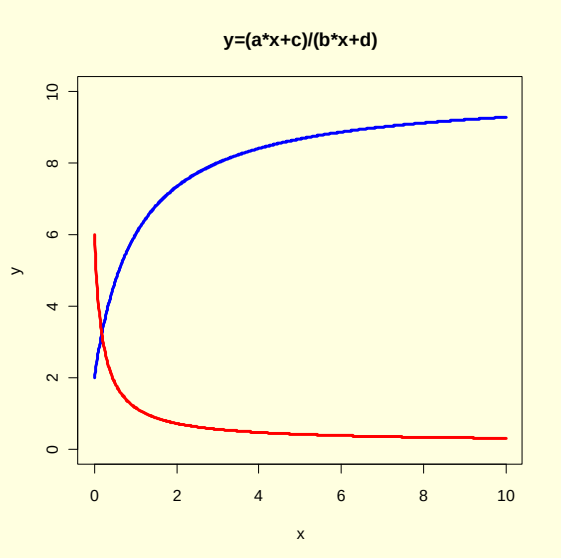
$$y = \frac{a \cdot x}{1 + b \cdot x}$$

```
x<-0:20
a=2
b=3
y<-a*x/(1+b*x)
par(bg="lightyellow")
plot(x,y,col="blue",type="l",lwd=3)
abline(h=a/b,col="green",lty=2,lwd=3)
text(0,0.65,"a/b",col="green")
```



Metningsfunksjon hvor a=2, b=3. Grafen nærmer seg asymptotisk til y=a/b, i dette tilfellet y=2/3

```
x<-seq(0,10,0.01)
a=10;b=1;c=2;d=1
y<-(a*x+c)/(b*x+d)
par(bg="lightyellow")
plot(x,y,col="blue",type="l",ylim=c(0,10),lwd=3,
main="y=(a*x+c)/(b*x+d)")
a=1;b=5;c=6;d=1
y<-(a*x+c)/(b*x+d)
points(x,y,col="red",type="l",ylim=c(0,10),lwd=3)
```



Asymptotisk funksjon med konstantene a=10;b=1;c=2;d=1 for blå stigende kurve og a=1;b=5;c=6;d=1 for rød synkende kurve. For x=0 starter kurven ved c/d og nærmer seg asymptotisk a/b.

Ikke-lineære funksjoner

Generelt hvis vi har den rasjonale funksjonen:

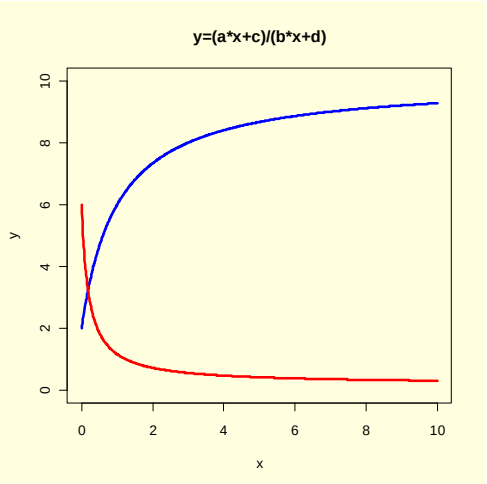
$$y = \frac{a \cdot x + c}{b \cdot x + d}$$

Så vil den starte ved c/d når x=0 med stigning eller synkning $(ad-bc)/d^2$ og nærmer seg asymptotisk a/b.

En vanlig utgave av denne er Michaelis-Menten-ligningen

$$y = \frac{a \cdot x}{1 + b \cdot x}$$


```
x<-seq(0,10,0.01)
a=10;b=1;c=2;d=1
y<-(a*x+c)/(b*x+d)
par(bg="lightyellow")
plot(x,y,col="blue",type="l",ylim=c(0,10),lwd=3,
main="y=(a*x+c)/(b*x+d)")
a=1;b=5;c=6;d=1
y<-(a*x+c)/(b*x+d)
points(x,y,col="red",type="l",ylim=c(0,10),lwd=3)
```



Asymptotisk funksjon med konstantene a=10;b=1;c=2;d=1 for blå stigende kurve og a=1;b=5;c=6;d=1 for rød synkende kurve. For x=0 starter kurven ved c/d og nærmer seg asymptotisk a/b.

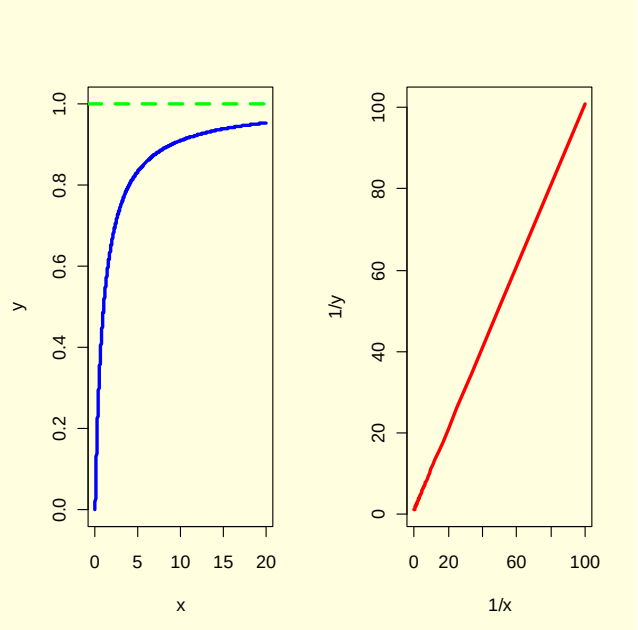
Vi kan lage den resiproke funksjonen:

$$\frac{1}{y} = \frac{1 + b \cdot x}{a \cdot x}$$

$$\frac{1}{y} = \frac{1 + b \cdot x}{a \cdot x} = \frac{1}{a \cdot x} + \frac{b}{a}$$

Hvis man plotter 1/x versus 1/y så kan man bestemme *a* og *b*.

```
x<-seq(0,20,0.01)
a=1;b=1
par(mfrow=c(1,2),bg="lightyellow")
y<-a*x/(1+b*x)
plot(x,y,col="blue",ylim=c(0,1),type="l",lwd=3)
abline(h=a/b,col="green",lty=2,lwd=3)
y2<-1/y
x2<-1/x
plot(x2,y2,col="red",type="l",lwd=3, xlab="1/x",ylab="1/y")
```



Potensligninger av typen (som også beskriver allometrisk vekst):

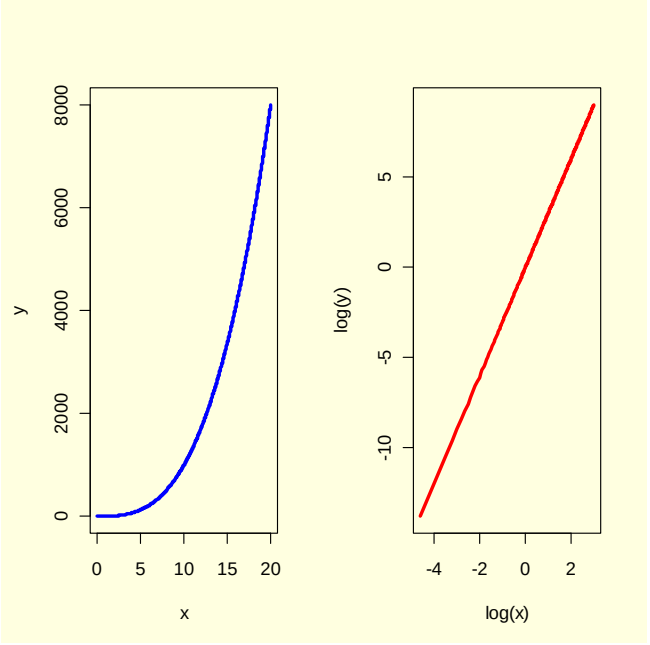
$$y = a \cdot x^b$$

hvor *a* og *b* er konstanter kan omgjøres til en lineær funksjon ved å ta logaritmen på begge sider av likhetstegnet:

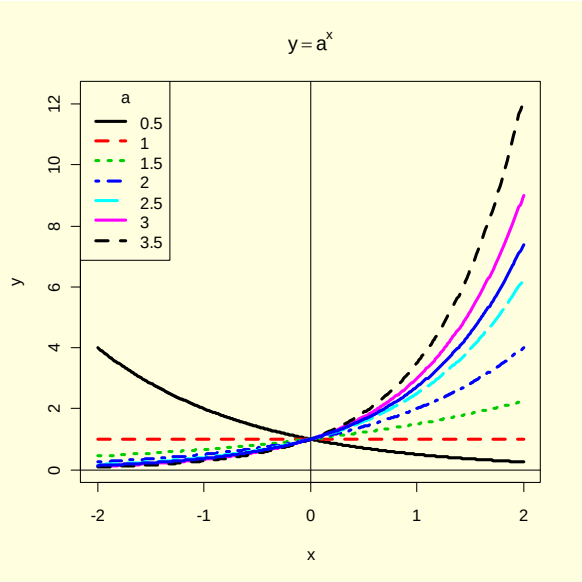
$$\log y = \log(a \cdot x^b) = \log a + b \cdot \log x$$

som får stigningskoeffisient *b* og skjæring ved *log a*.

```
x<-seq(0,20,0.01)
a=1;b=3
par(mfrow=c(1,2),bg="lightyellow")
y<-a*x^b
plot(x,y,col="blue",type="l",lwd=3)
y2<-log(y)
x2<-log(x)
plot(x2,y2,col="red",type="l",lwd=3, xlab="log(x)",ylab="log(y)")
```




```
par(bg="lightyellow")
x<-seq(-2,2,0.01)
a<-seq(0.5,3.5,0.5)
eksponent<-outer(x,a, function(x,a)a^x)
par(bg="lightyellow")
matplot(x,eksponent,type="l",ylab="y",lwd=3,
main=expression(y==a^x))
abline(h=0,v=0)
lines(x,exp(x),col="blue",lwd=3)
legend("topleft",as.character(a),title="a",lty=1:5,col=1:6,lwd
=3)
#Det naturlige tallet e
exp(1)
```



Figur. $y=a^x$ for forskjellige verdier av grunntallet a. Alle kurvene går gjennom (x,y)=(0,1). For a=1 er en rett linje siden $1^x=1$. Hvis a er lik det naturlige tallet e= 2.718282 har vi eksponentialfunksjonen $y= e^x$ som er angitt med den blå heltrukne linjen, som er lik den deriverte funksjonen.

$a^x \cdot a^y = a^{x+y}$	$0^x = 0$	$a^0 = 1$	$1^x = 1$
$(a \cdot c)^x = a^x \cdot c^x$	$a^{\frac{1}{x}} = \sqrt[x]{a}$	$a^{\frac{1}{2}} = \sqrt{a}$	$(a^x)^y = a^{x \cdot y}$
$\frac{1}{a^y} = a^{-y}$	$\frac{a^x}{a^y} = a^{x-y}$	$\left(\frac{a}{c}\right)^x = \frac{a^x}{c^x}$	$(a^x)^{\frac{1}{x}} = a$

Noen potenser hvor x og y er eksponenter, a er grunntall og c er en konstant.

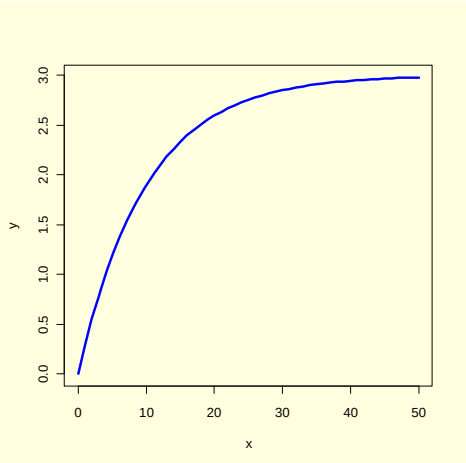
Eksponentialfunksjon med asymptote og to parametre

Funksjonen

$$y = a \cdot (1 - e^{-b \cdot x})$$

er en asymptotisk eksponentialfuksjon med to parametere a og b og *e.g.* a=3 og b=0.1

```
x<-0:50
y<-3*(1-exp(-0.1*x))
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=3)
```



Asymptotisk eksponentialfunksjon med a=3 og b=0.1

Potensligninger av typen:

$$y = a \cdot x^b$$

hvor *a* og *b* er konstanter kan omgjøres til en lineær funksjon ved å ta logaritmen på begge sider av likhetstegnet:

$$\log y = \log(a \cdot x^b) = \log a + b \cdot \log x$$

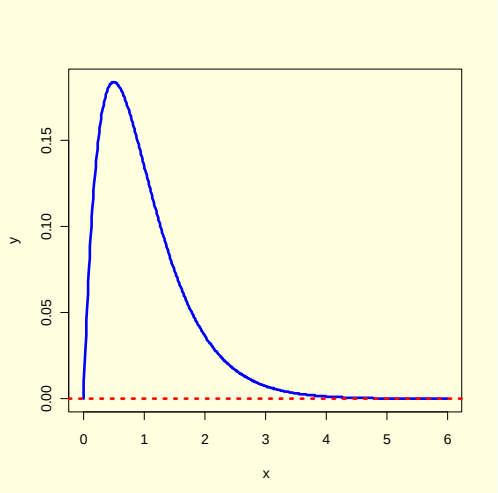
som får stigningskoeffisient *b* og skjæring ved *log a*.

Funksjonen:

$$y = x \cdot e^{-2 \cdot x}$$

Går mot 0 når x går mot uendelig:

```
x<-seq(0,6,0.01)
y<-x*(exp(-2*x))
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=3)
abline(h=0,lty=3,col="red",lwd=3)
```



Utforsk funksjonen når *r* endrer verdi. For figuren over er r=2

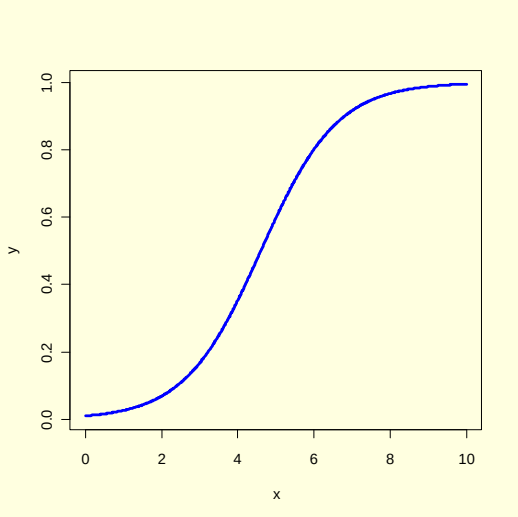
```
x<-seq(0,6,0.01)
r<- 0.5
y<-x*(exp(-r*x))
plot(x,y,type="l",col="blue",lwd=3)
abline(h=0,lty=2,col="red")
```

Sigmoide logistisk funksjon med 3 parametre

Eksempel på en sigmoid eller S-formet funksjon med tre parametere *a*, *b* og *c*:

$$y = \frac{a}{1 + b \cdot e^{-c \cdot x}}$$

```
x<-seq(0,10,0.01)
a=1;b=100;c=1
y<-a/(1+b*exp(-c*x))
par(bg="lightyellow")
plot(x,y,col="blue",type="l",lwd=3)
```



Treparametrisk sigmoid vekstkurve med a=1, b=100 og c=1.

Gammafunksjonen

Gammafunksjonen blir betegnet med den store greske bokstaven for gamma (Γ)introdusert av Euler. Gammafunksjonen kan defineres på flere måter bl.a. som et integral:

$$\Gamma(x) = \int_0^\infty t^{x-1} \cdot e^{-t} dt$$

Gammafunksjonen er også en løsning av:

$$\Gamma(x+1) = x \cdot \Gamma(x)$$

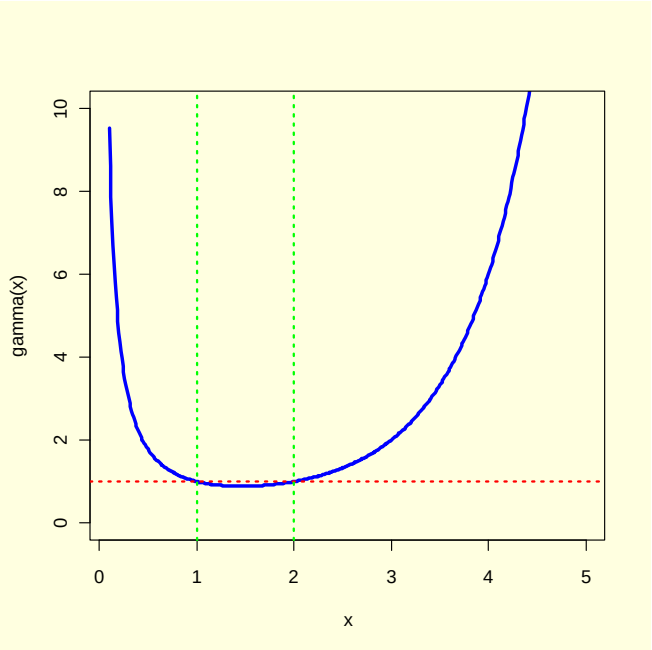
For hele positive tall av x er det tilknytning til begrepet fakultet (!):

$$\Gamma(x) = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (x-1) = (x-1)$$

Gammafunksjonen er lik 1 for x=1 og x=2. For heltall av *x* blir *Γ(x+1)=x!* Gammafunksjonen inngår bl.a. i gammafordelingen.

```
x<-seq(0.1,5,0.01)
par(bg="lightyellow")
plot(x,gamma(x),type="l",ylim=c(0,10),lwd=3,col="blue")
abline(h=1,lty=3,lwd=2,col="red")
abline(v=1,lty=3,lwd=2,col="green")
```

```
abline(v=2,lty=3,lwd=2,col="green")
```



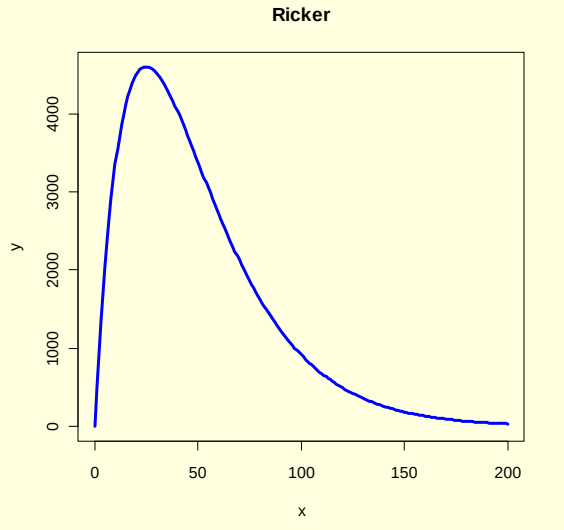
Gammafunksjonen

Ricker-kurve

$$y = a \cdot x \cdot e^{-b \cdot x}$$

Etter en kanadisk fiskebiolog Ricker (1954), brukt i fiskeforvaltning hvor y er rekruttering i bestanden ut fra foreldrebestanden x.

```
x<-seq(0,200,1)
a<-500
b<-0.04
y<-a*x*exp(-b*x)
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=3,main="Ricker")
```

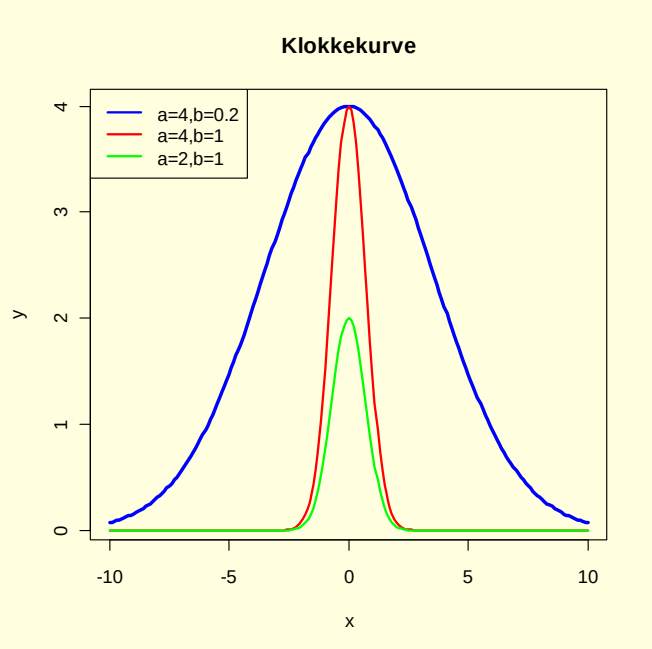


Klokkekurve

$$y = a \cdot e^{(-|b \cdot x|^2)}$$

```
x<-seq(-10,10,0.1)
a<-2
b<-1
y<-a*exp(-abs(b*x)^2)
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=3,main="Klokkekurve")
lines(x,y,col="green",lwd=3)
legend("topleft",c("a=4,b=0.2","a=4,b=1","a=2,b=1"),lty=c(1,1,
1),lwd=c(2,2,2),col=c("blue","red","green"))
```

Når x går fra ±∞ og integralet under kurven er lik 1 har vi tetthetsfunksjonen for normalfordelingskurven.

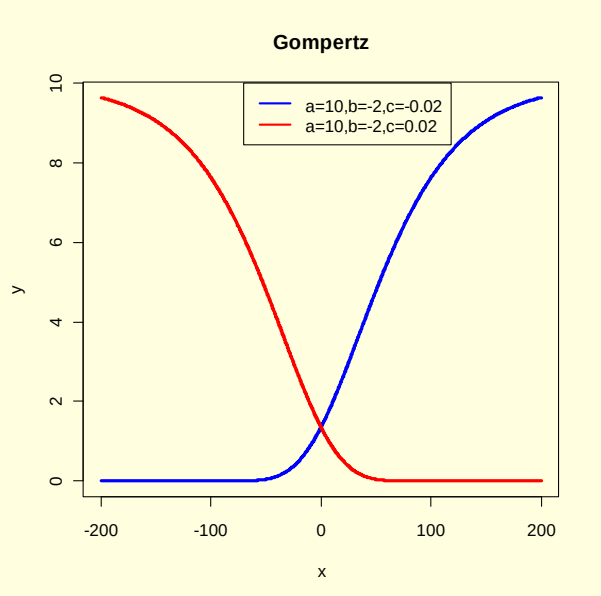


Gompertz

Gompertz-funksjonen blir brukt i forsikringsmatematikk og demografiske studier.

$$y = a \cdot e^{b \cdot e^{c \cdot x}}$$

```
x<-seq(-200,200,0.1)
a<-10
b<--2
c<-0.02
y<-a*exp(b*exp(c*x))
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=3,main="Gompertz")
#Sett inn c=0.02 og kjør på nytt, men bruk lines
lines(x,y,col="red",lwd=3)
legend(-70,10,c("a=10,b=-2,c=-0.02","a=10,b=-2,c=0.02"),
lty=c(1,1),lwd=c(2,2),col=c("blue","red"))
```



Bieksponensiell

$$y = a \cdot e^{b \cdot x} + c \cdot e^{d \cdot x}$$

Endrer form ved forskjellig fortegn og verdier av a,b,c og d

Ikke-lineær regresjon

R inneholder en rekke selvstartende SSmodeller som brukes til å estimere parametere i modell. **SSasymp** for en asymptotisk regresjonsmodell,**SSmicmen** (Michalis-Menten) og **SSlogis** (logistisk), **SSfol** (første ordens kompartment), **SSweibull** (Weibull), **SSgomperts** (Gompertz) osv.
Vi har et datasett med areal av flytebladplanten andemat som funksjon av tid:

```
  tid<-c(0,2,4,6,8,10,12)
 areal<-c(2,6,19,35,48,65,72)
 vekst<-cbind(tid,areal);vekst
 plot(areal~tid,pch=16,cex=2,col="blue",ylim=c(0,80),xlab="Tid
(dager)",ylab="Areal (cm2)")
```

Det lages estimat av asymptoten (Asym), x-verdi ved infleksjonspunkt (xmid) og en skaleringsparamter (scal).

```
  modell<-nls(areal~SSlogis(tid,Asym,xmid,scal))
 summary(modell)
```

```
Formula: areal ~ SSlogis(tid, Asym, xmid, scal)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
Asym  77.8752     3.9940  19.498 4.08e-05 ***
xmid   6.6596     0.3389  19.650 3.96e-05 ***
scal   2.1615     0.2243   9.636 0.000649 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.224 on 4 degrees of freedom
```

```
Number of iterations to convergence: 1
Achieved convergence tolerance: 1.524e-06
```

Vi predikterer data:

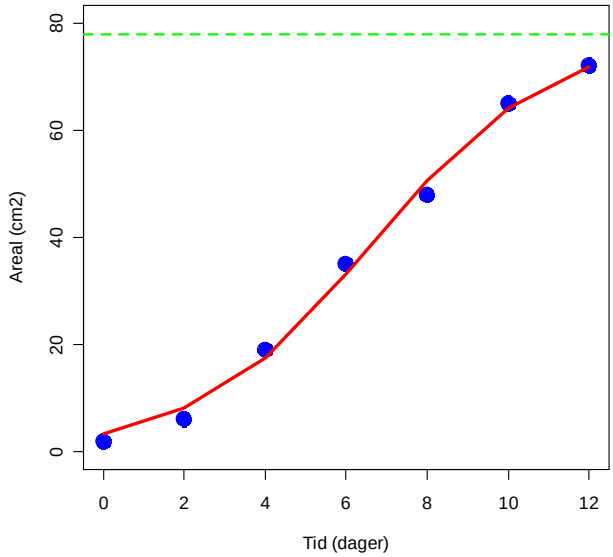
```
 predikt<-predict(modell)
 lines(tid, predikt, col = "red", lwd =3)
```

Vi trekker en horisontal (h) linje ved den estimerte asymptoten med abline(h=):

```
 abline(h = coef(modell)["Asym"], col = "green", lty = 2, lwd
=2)
```

Maksimal veksthastighet 46%:

```
 rgr <- 1 / coef(modell)["scal"];rgr
    scal
0.4626321
```



Er det mange punkter på et punktsky-plot kan enkeltpunkter identifiseres med kommanoden **identify()**. Med kommandoen **locator()** kan man finne x,y-koordinater til utvalgte punkter på et plot ved å peke på dem med markøren. For eksempel to punkter som man vil finne koordinatene til **locator(2)**.

Funksjonen **split** lager en liste med vektorer med basis til nivåene til en faktor. Viktig når man skal lage plot

Med kommandoen **data.frame** kan man koble sammen kolonnedata og lage en større matrise som inneholder alle dataene. Omdanner data.frame til grupperte data med kommandoen **groupedData**:

Multipanel plot velegnet for grupperte data

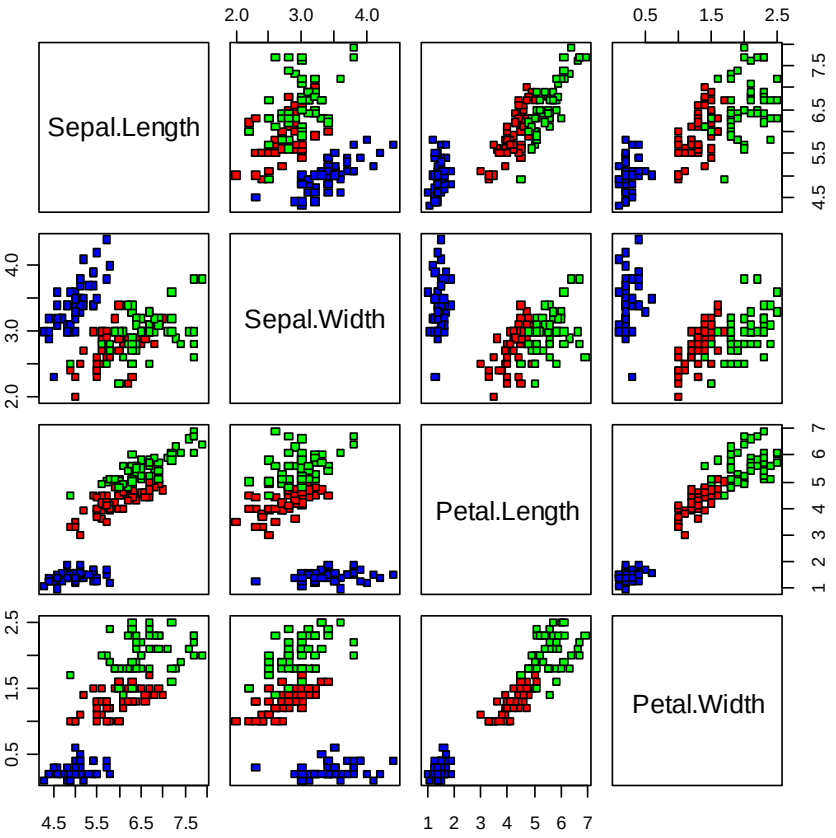
Trellis plot

Trellis-plot er nyttige for å vise hvordan responsvariabel avhenger av nivået av andre kontinuerlige forklaringsvariable. De hentes fra et bibliotek kalt **lattice.**

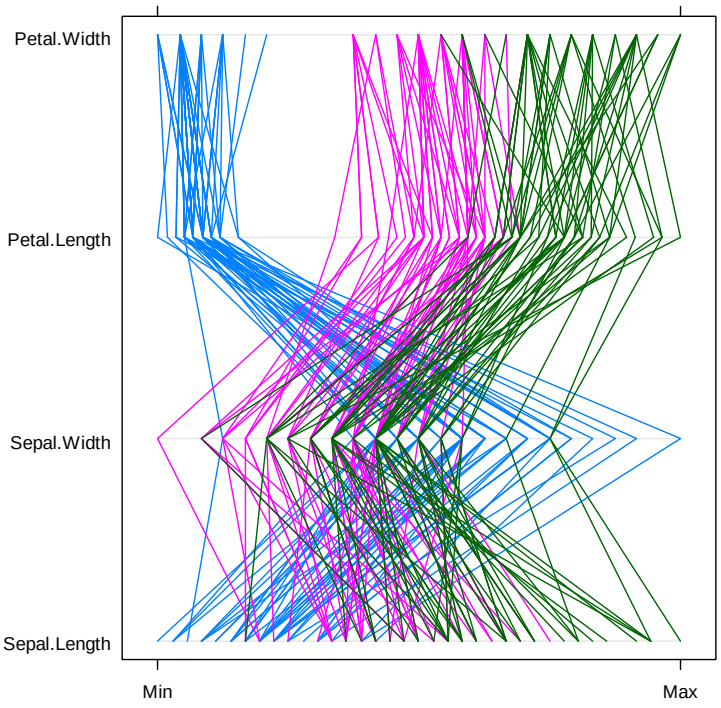
```
data(iris)
attach(iris)
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
plot(iris[1:4],pch=22,bg=c("blue","red","green")[as.numeric(iris$Species)])
```



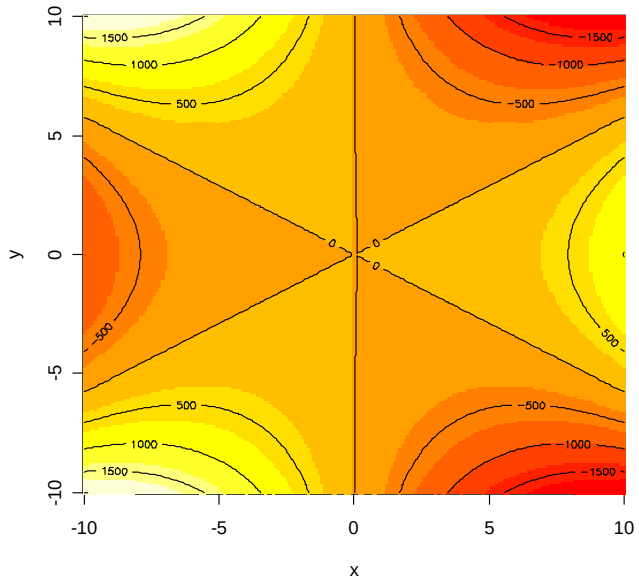

```
library(lattice)
parallel(~iris[1:4],groups=Species, iris)
```



Plot

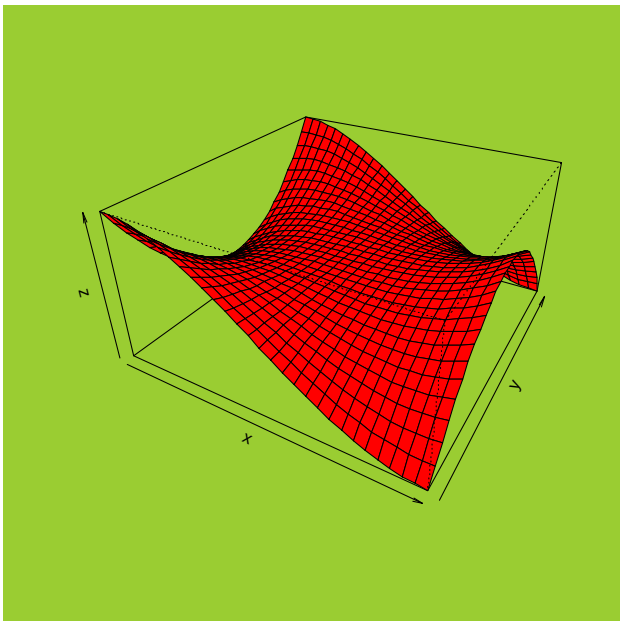
For å lage plot **image** og **contour** er funksjonen **outer** nyttig. Den evaluerer en funksjon, **func**, ved hver kombinasjon av x og y innen et kvadrat eller retangulært array, og dette lager data som kreves av **image** og **contour**. Kommandoen **add=T** lager kontur på toppen av et farget bilde.
$z=x^3-3xy^2$

```
x<-seq(-10,10,0.1)
y<-seq(-10,10,0.1)
z<-function(x,y) x^3-3*x*y^2
image(x,y,outer(x,y,z))
contour(x,y,outer(x,y,z),add=T)
```



Scriptfil for funksjonen $z=x^3-3xy^2$ vist over

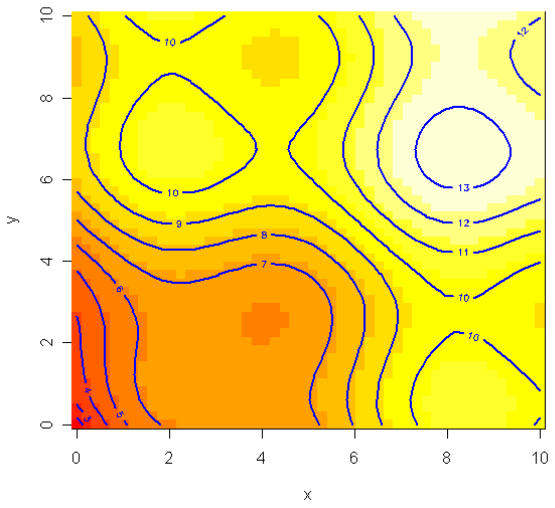
```
x <- seq(-10, 10, length= 30)
y <- x
f <- function(x,y) { x^3-3*x*y^2
}
 z <- outer(x, y, f)
 z[is.na(z)] <- 1
 op <- par(bg = "yellowgreen")
 persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col =
"red")
```



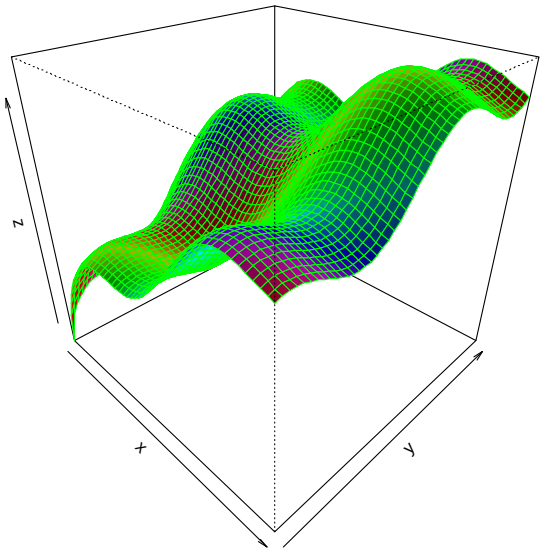
Hvor det er et sadelpunkt ved origo

Funksjonen z=sqrt(x+y)+cos(y)+sin(x):

```
n <- 50
x <- seq(0,10, length=n)
y <- seq(0,10, length=n)
xm <- matrix(x, nrow=n, ncol=n)
ym <- matrix(y, nrow=n, ncol=n,byrow=TRUE)
z <- 3*sqrt(xm+ym)+cos(ym)+sin(xm)
image(x, y, z)
contour(x, y, z, col="blue",lwd=2, add=TRUE)
```




```
persp(x, y, z, theta = 45, phi = 30,shade = .5,col =
rainbow(n),border = "green")
```



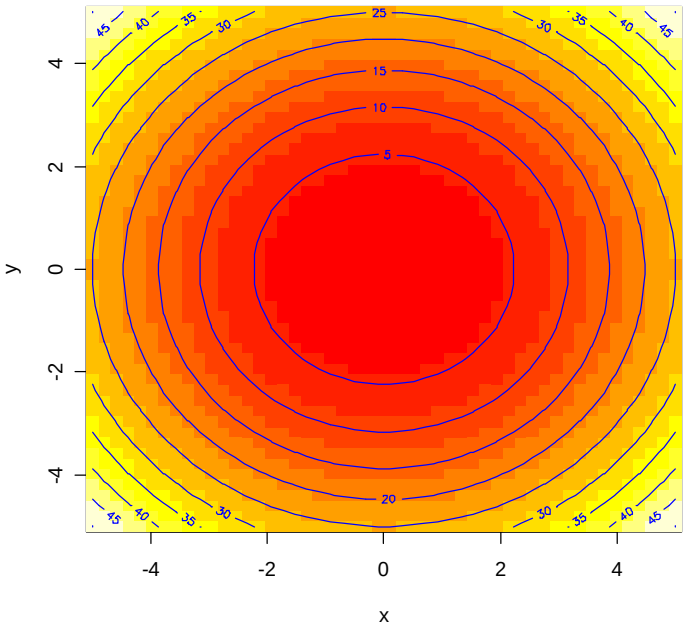
Funksjonen:

$$z = x^2 + y^2$$

med følgende konturplot

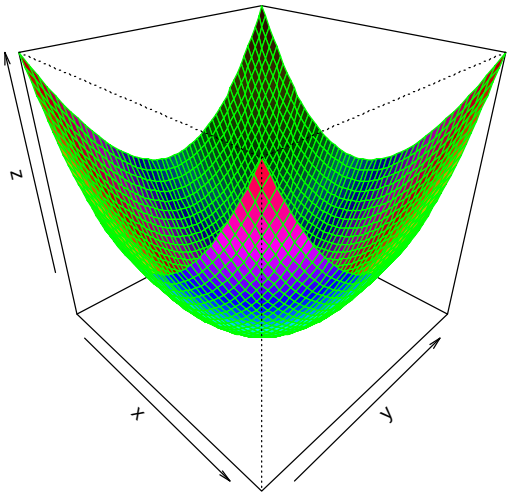
```
n <- 50
```

```
x <- seq(-5,5, length=n)
y <- seq(-5,5, length=n)
xm <- matrix(x, nrow=n, ncol=n)
ym <- matrix(y, nrow=n, ncol=n,byrow=TRUE)
z <- xm^2 + ym^2
image(x, y, z)
contour(x, y, z, col="blue", add=TRUE)
```



Funksjonen **par** brukes til å sette grafiske parametere. Kommandoen **bg** gir bakgrunnsfargen på plot, **mar** angir marger rundt figuren c(bottom, left, top, right) hvor defaultverdi er c(5, 4, 4, 2) + 0.1.
Tredimensjonalt plot av funksjonen ovenfor:

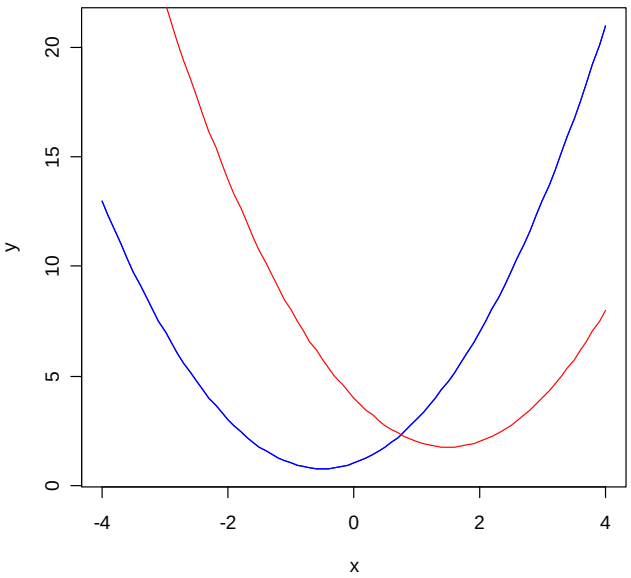
```
persp(x, y, z, theta = 45, phi = 30,shade = .5,col =
rainbow(n),border = "green")
```



Et annengradspolynom får utseende som en parabel:

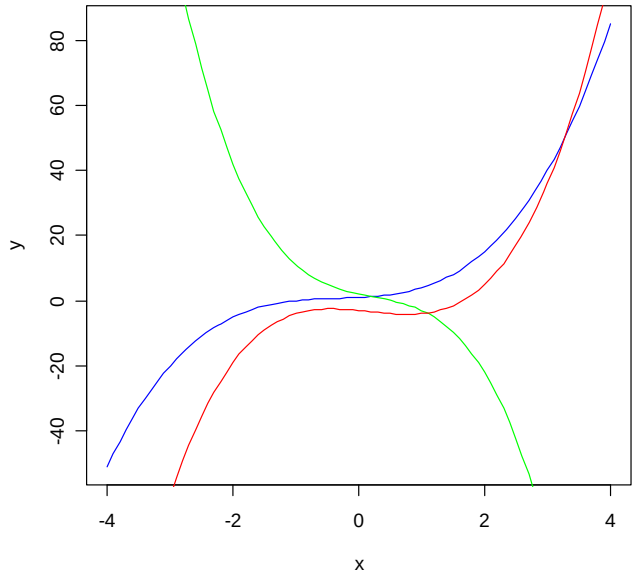
$y = x^2+bx+c$

```
a<-1
b<-1
c<-1
plot(x,a*x^2+b*x+c,col="blue",type="l",xlab="x",ylab="y")
a<-1
b<--3
c<-4
points(x,a*x^2+b*x+c,col="red",type="l")
```



Hvis vi har et tredjegradspolynom:
$y= ax^3+bx^2+cx+d$

```
x<-seq(-4,4,0.1)
a<-1
b<-1
c<-1
d<-1

plot(x,a*x^3+b*x^2+c*x+d,col="blue",type="l",xlab="x",ylab="y"
)
a<--3
b<-2
c<--4
d<-2
points(x,a*x^3+b*x^2+c*x+d,col="green",type="l")

a<-2
b<--1
c<--2
d<--3
points(x,a*x^3+b*x^2+c*x+d,col="red",type="l")
```

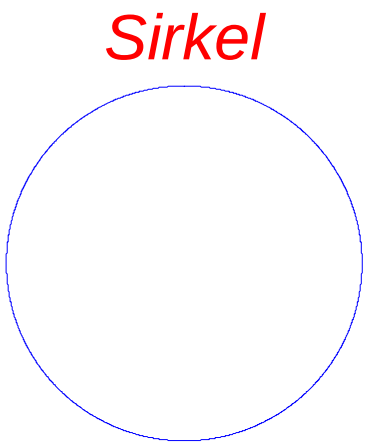


Sirkelen har generell formel:

$x^2+y^2+ax+by+c=0$ og en utgave av denne er $x^2+y^2=1$. Hvis vi har enhetssirkelen med radius 1 vil sinx og cosx ligge på sirkelen fordi $\sin^2x+\cos^2x=1$. Vi lager en rekke fra 0-2π og plotter sinx mot cosx. **pin** setter størrelsen i tommer (inches). **axes=F** undertrykker alle akser.

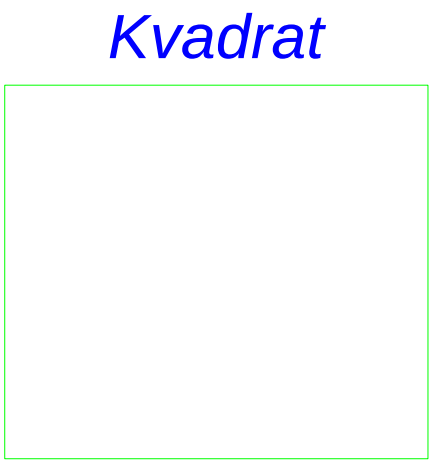
```
o<- seq(0,2*pi,length=2000)
```

```
x<-sin(o)
y<-cos(o)
par(pin=c(4,4))
plot(x,y,type="l",col="blue",axes=F,xlab="",ylab="")
title(main=list("Sirkel",cex=4,col="red",font=3))
```



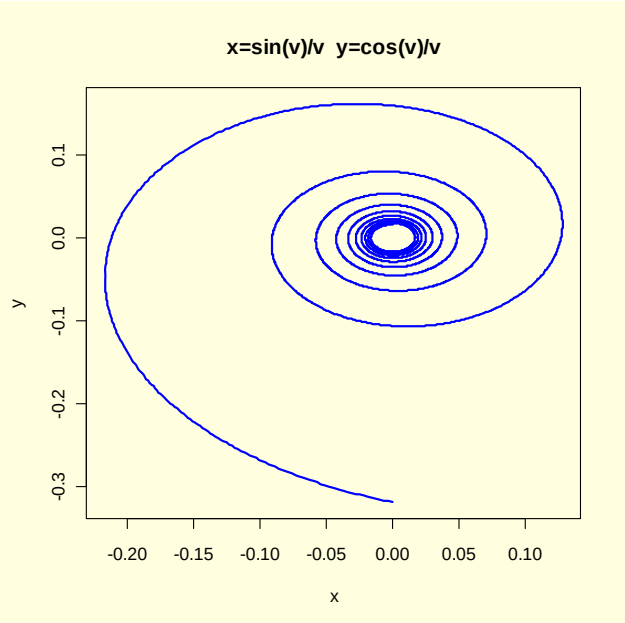
Et rektangel trenger to vektorer x og y med koordinater for alle fire hjørner, men for at linjene skal komme tilbake til utgangspunktet trengs fem koordinater:

```
x<-c(1,-1,-1,1,1)
y<-c(1,1,-1,-1,1)
plot(x,y,type="l",col="green",axes=F,xlab="",ylab="")
title(main=list("Kvadrat",cex=4,col="blue",font=3))
```

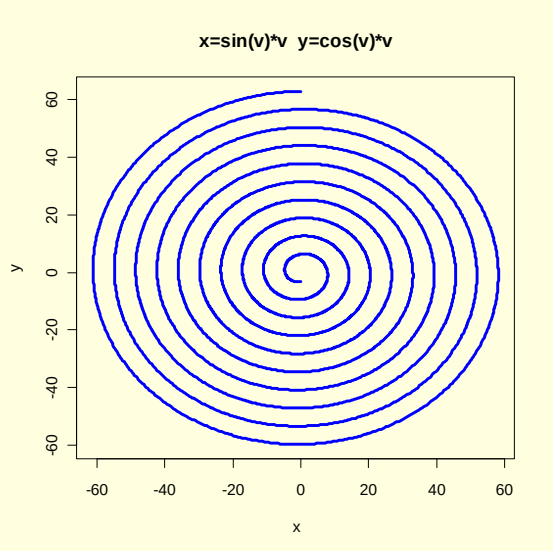


Man kan lage en spiral ved å endre radius kontinuerlig:

```
  v<-seq(pi,20*pi,0.01)
x<-sin(v)/v
y<-cos(v)/v
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=2,main="x=sin(v)/v
y=cos(v)/v")
```




```
v<-seq(pi,20*pi,0.01)
x<-sin(v)*v
y<-cos(v)*v
par(bg="lightyellow")

plot(x,y,type="l",col="blue",lwd=3,main="x=sin(v)*v
y=cos(v)*v")
```

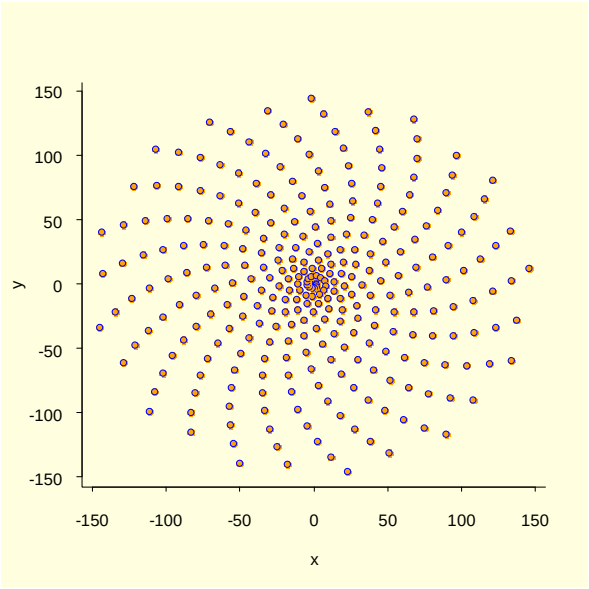



```
v<-seq(pi,20*pi,0.01)
x<-sin(v)*v
y<-cos(v)
par(bg="lightyellow")
plot(x,y,type="l",col="blue",lwd=3,main="x=sin(v)*v
y=cos(v)")
```

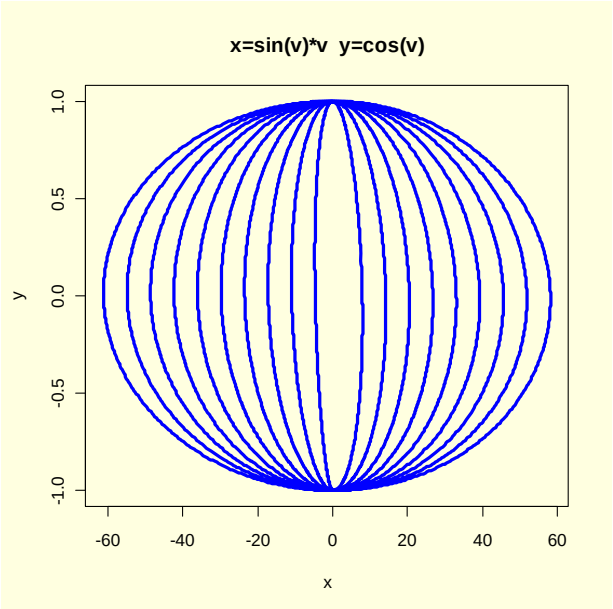
Lager punktplot av det samme:

```
o<-seq(0,150,0.5)
x<-sin(o)*o
y<-cos(o)*o
```

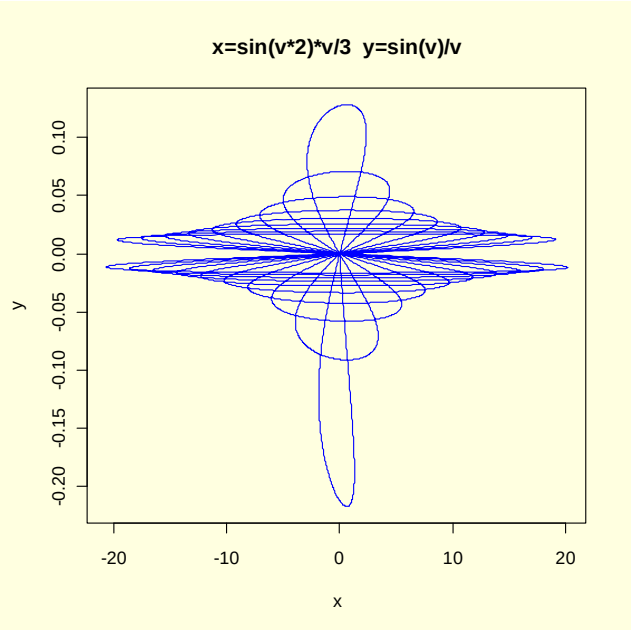
```
par(bg="lightyellow")
plot(x,y,pch=21, col="blue", bg="orange",bty="l", tcl=-.25,
las=1)
```



Figur. Punktplot for x=sin(o)·o og y=cos(o)·o minner om spiralfyllotaksis

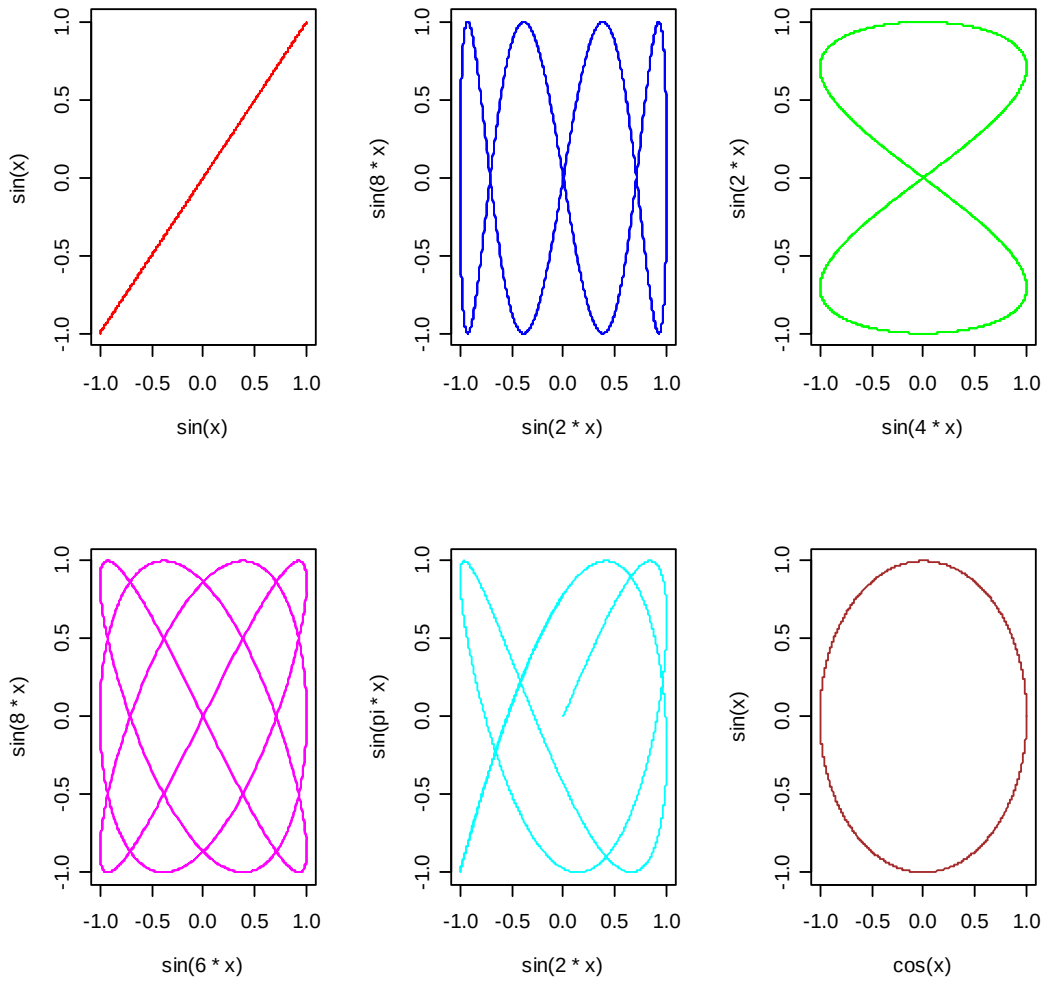



```
v<-seq(pi,20*pi,0.01)
x<-sin(v*2)*v/3
y<-sin(v)/v
par(bg="lightyellow")
plot(x,y,type="l",col="blue",main="x=sin(v*2)*v/3
y=sin(v)/v")
```



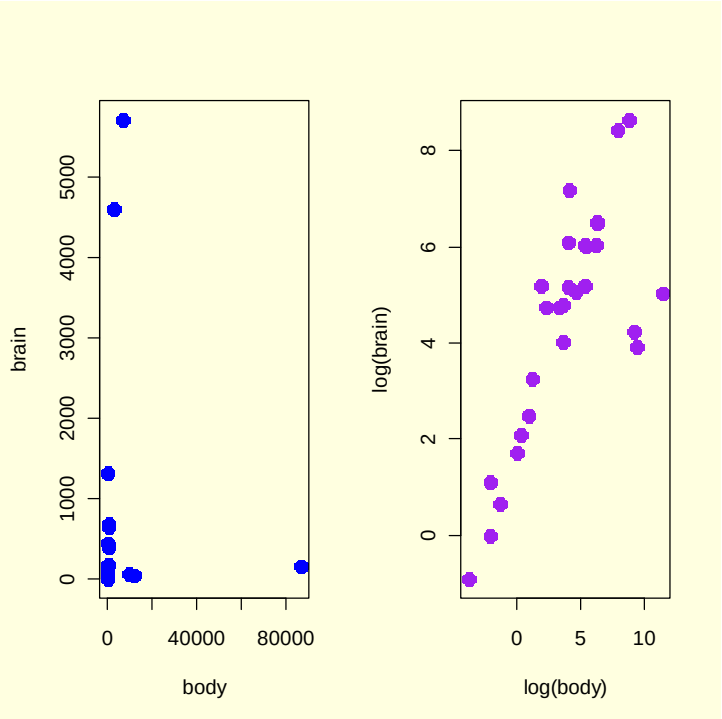
Lissajousfigurer består av to svingninger som står vinkelrett på hverandre. Finnes bl.a. i katodestråleoscilloskoper hvor man kan kontrollere forskjell i fase og frekvens for svingninger.
$y_1(x)=\sin(ax)$ og $y_2=\sin(bx)$
a og *b* er positive tall og $0<x<2\pi$
Hvis *b* er et irrasjonalt tall *e.g.* 5/3, √2 (**sqrt(2)**), pi (π) eller det naturlige tallet *e* (**exp(1)**) blir det aperiodiske svingninger. Bare pi er vist på figuren.

```
par(mfrow=c(2,3))
x<-seq(0,2*pi,length=2000)
plot(sin(x),sin(x),col="red",type="l")
plot(sin(2*x),sin(8*x),col="blue",type="l")
plot(sin(4*x),sin(2*x),col="green",type="l")
plot(sin(6*x),sin(8*x),col="magenta",type="l")
plot(sin(2*x),sin(pi*x),col="cyan",type="l")
plot(cos(x),sin(x),col="brown",type="l")
```



Sammenhengen mellom gjennomsnittsvekt av kropp(kg) og hjerne (gram) hos 28 terrestriske dyr:

```
library(MASS)
library(help=MASS)
attach(Animals)
Animals
par(mfrow=c(1,2),pch=16,bg="lightyellow")
plot(body, brain,col="blue",pch=16,cex=1.5)
plot(log(body),log(brain),col="purple",pch=16,cex=1.5)
```



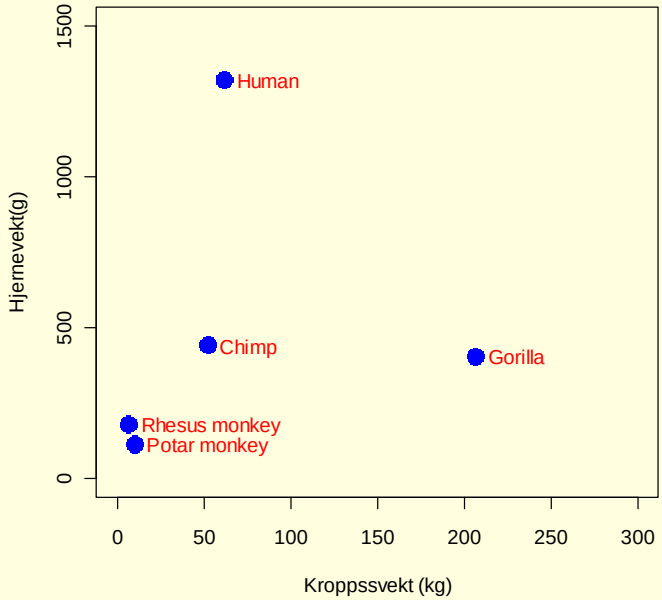
Datasettet primates er et utvalg av primater fra Animals, men som også kan finnes i pakken DAAG:

```
library(DAAG)
library(help=DAAG)
attach(primates)
names(primates)
primates
```

```
              Bodywt Brainwt
Potar monkey    10.0     115
Gorilla        207.0     406
Human           62.0    1320
Rhesus monkey    6.8     179
Chimp           52.2     440
```

```
par(bg="lightyellow")
plot(Bodywt, Brainwt, pch=16,cex=2,col="blue",xlim=c(0,300),
ylim=c(0,1500),
xlab="Kroppssvekt (kg)", ylab="Hjernevekt(g)")
text(Bodywt,Brainwt, labels=row.names(primates),col="red",
pos=4)
```

Skal navnet skrives på venstre side av symbolet bruk **pos=2.**



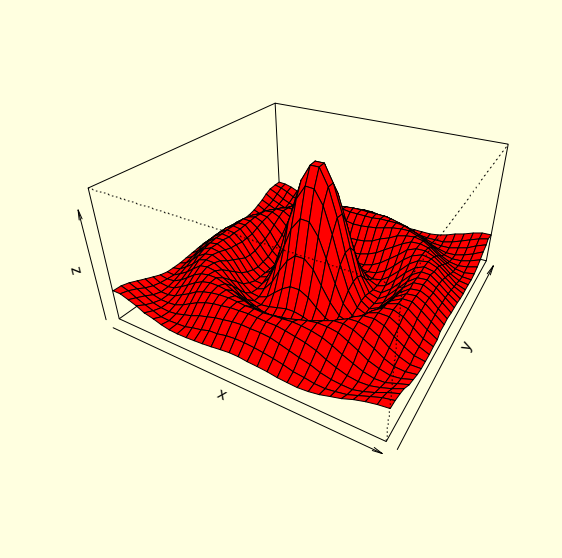
3D-plot

Det finnes en rekke muligheter for å lage 3D-plot. Se eksempler med kommandoen:

```
 demo(persp)
```

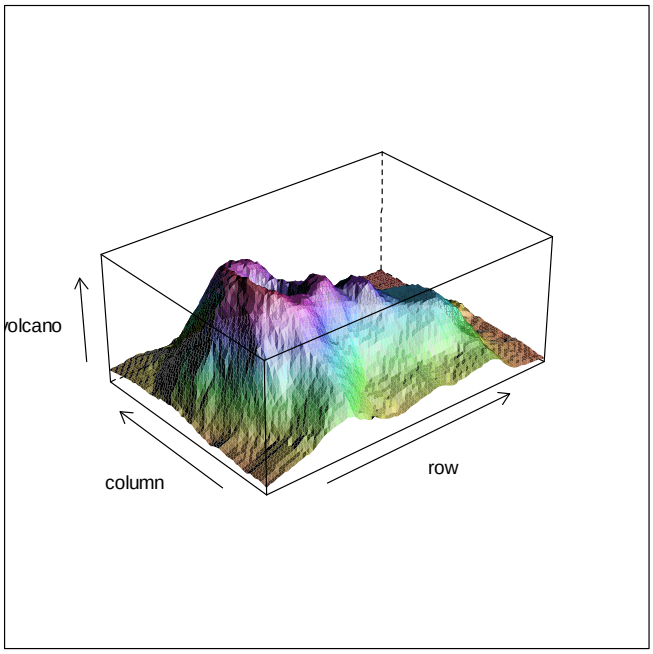
Her er ett av dem:

```
  x <- seq(-10, 10, length= 30)
  y <- x
  f <- function(x,y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
  z <- outer(x, y, f)
  z[is.na(z)] <- 1
  op <- par(bg = "lightyellow")
  persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col =
"red")
```



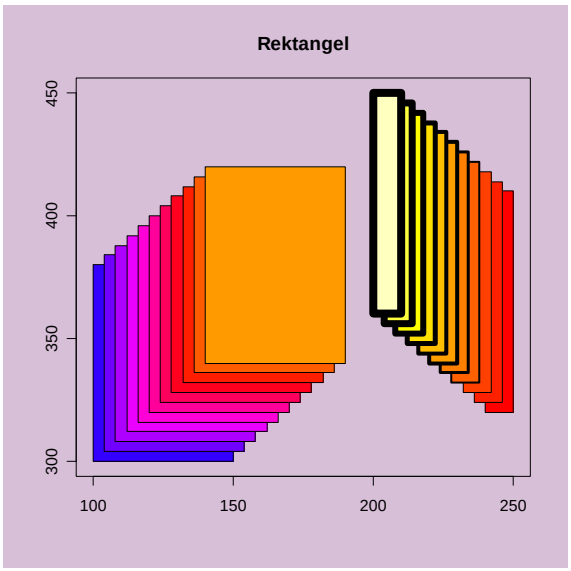
I library (lattice)ligger funksjonen **wireframe** som kan brukes til å lage 3D-plot. Eks. datasett fra R

```
 library(lattice)
 wireframe(volcano, shade = TRUE,
+           aspect = c(61/87, 0.4),
+           light.source = c(10,0,10))
```



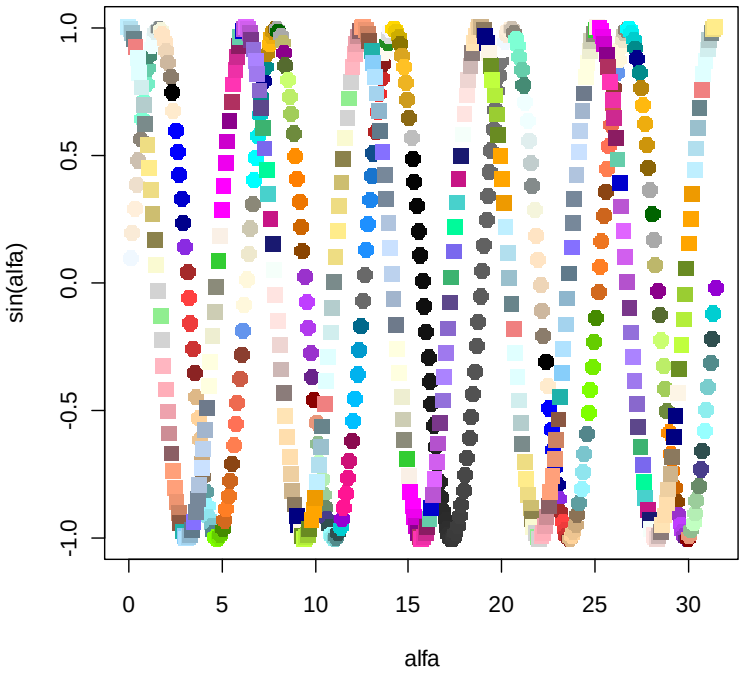
Tilføye et rektangel med **rect,** kan også fylles med farger:

```
 par(bg = "thistle")
 plot(c(100, 250), c(300, 450), type = "n", xlab="", ylab="",
main = "Rektangel",col="blue")
 rect(100+i, 300+i, 150+i, 380+i, col=rainbow(11,
start=.7,end=.1))
 rect(240-i, 320+i, 250-i, 410+i, col=heat.colors(11),
lwd=i/5)
```



Oversikt over de 657 fargenavnene man kan velge I finnes med **colors()**. Her er et lite utvalg:

```
alfa<-seq(0,10*pi,0.1)
plot(alfa,sin(alfa),col=colors()[1:200],pch=16,cex=1.5)
points(alfa,cos(alfa),col=colors()[400:500],pch=15,cex=1.5)
```



Greske tegn og matematikse operatorer kan plottes via **expression()**. Et utvalg vises med **demo(plotmath)**.

Oversikt over plottefunksjoner

abline – trekker en linje i et eksisterende plot
arrows – tilføyer piler på et plot
axis – tilføyer en akse i et plot, punkter på aksen **lab=**, logakse for eksempel y-aksen **log="y"**, orienteringen av aksetekst **las=**, 0=parallelt med aksen, 1= horisontalt, 2= vertikalt, akseplassering **msg=c(x,y)**,tick-merke **tck=**,
barplot – stolpediagram
biplot – viser rader og kolonner i en datamatrise
contour – konturplot
dotchart – lager punktskydiagram
frame – starter en ny plotramme
hist – lager histogram
image – viser et fargebilde
interaction.plot – interaskjonsplot for to-faktoreksperiment
legend – føyer på aksetekst på et plot
lines – tilføyer linjer på et plot
lines(x,y) – tilføyer linjer på et eksisterende plot. Linjetype settes med **lty=** og linjebredden (linjetykkelsen) med **lwd** Når du har startet programmet får du fram vinduet nedenfor.

= (lwd større enn 1 øker default-verdien); farge **col=**; forstørrelsesfaktor **cex=** (default=1)
matplot – plotter kolonner i en matrise
mtext – skriver tekst i margene på et plot
pairs – parvise plot mellom multiple variable
par – setter inn grafiske parametere
persp – lager perspektivplot
pie – lager kakediagram
plot – lager plot
polygon – tilføyer polygon på eksisterende plot
points – tilføyer punkter på et plot
points(x,y)- Tilføyer punkter på et eksisterende plot og symbol settes med **pch=** hvor tallene 0-25 gir forskjellige symboler
, fonttype med karakterutvidelse **cex=** , høyden på fontene i tommer **csi=**, strengrotasjon i grader **srt=**,
qqplot – kvantil-kvantil plot
qqnorm – normalfordelt qqplot
rect – tilføyer et rektangel på et plot, kan være farget
rug – Skriver punktlinjer på x-aksen (default). Skal man ha punktlinjer på y-aksen tilføy **use side=2.**
scatter.smooth – punktskyplot med kurveglatting med loess
smooth.spline – trendlinjer i punktskyplot trukkes med kubisk spline
supsmu – Kurveglatting med Friedmans superglattings algoritme
segments – tilføyer linjesegmenter til et plot
stars – lager et stjerneplot av multivariable data
symbols – skriver forskjellige symboler på et plot

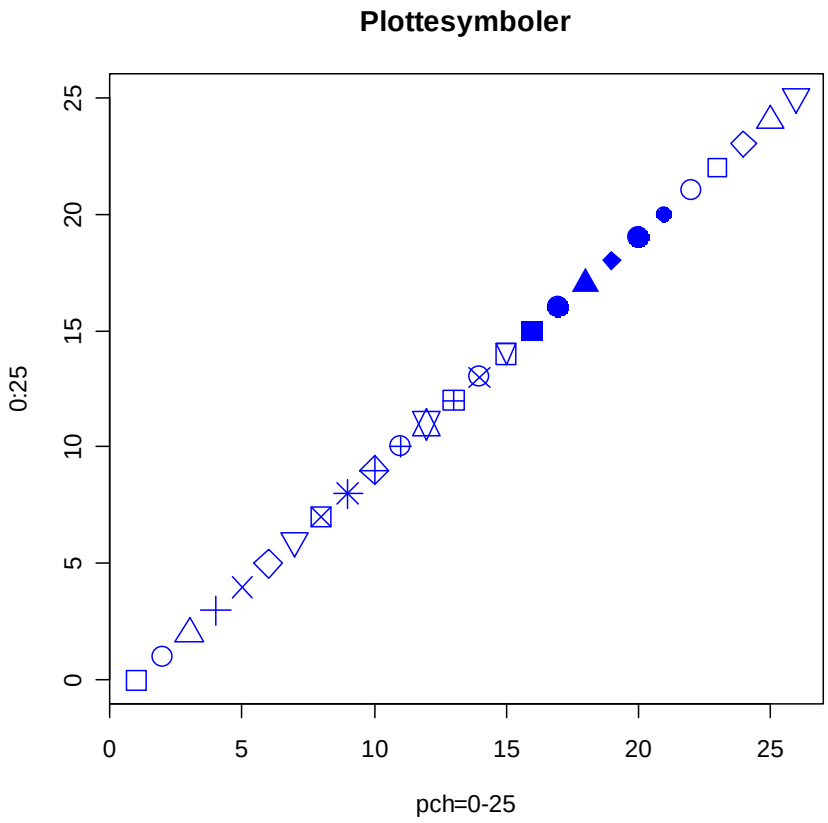
text – tilføyer tekstsymboler på et plot
text(x,y,label) – setter inn tekst ved punkt x,y, label er en tall- eller bokstavvektor.
title – tilføyer tittel på et plot

Kommandoene etterfølges av () med innhold for eksempel **abline()**.

Akser, plottsymboler og akseenheter blir satt automatisk man kan også kontrolleres av brukeren:
type="x"
p - for punkter
l – for linjer
n – ingen plot, men akser blir satt opp
b – for både punkter og linjer
o – for overliggende punkter og liner
h – høytetthets vertikale linjer
s – trappefunksjon, s for venstre side
S – Trappefunksjon, S for høyre Side

xlab="xxxx" – x-akse tekst
ylab="xxxx" – y-akse tekst
xlim=c(lo,hi) – omtrentlig x-akselengde
ylim=c(lo,hi) – omtrentlig y-akselengde
main="xxxx" – hovedtittel plot
sub="xxxx" – undertittel plot

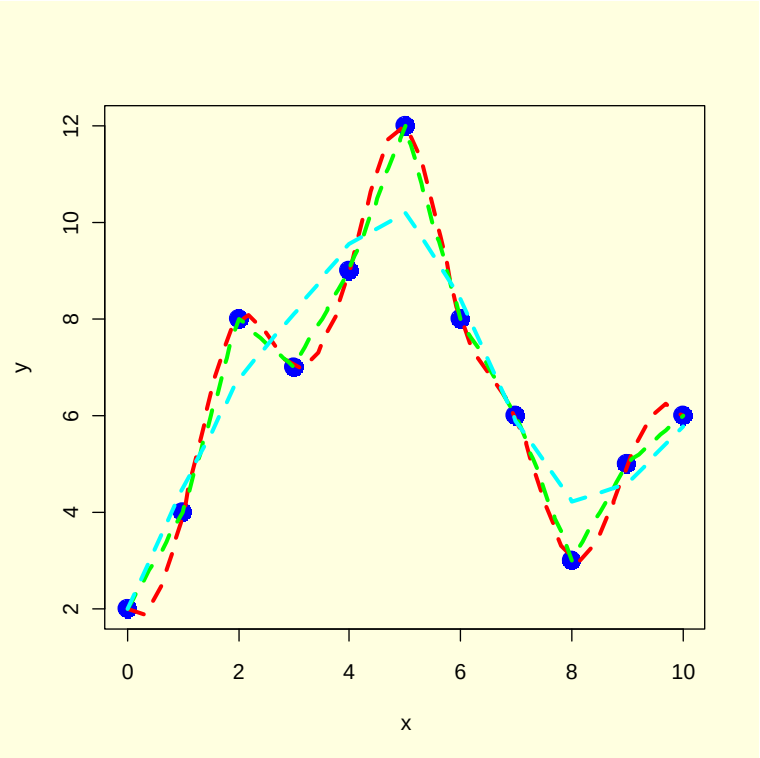
pch= Forskjellige plottesymboler fra pch=0:25. pch=19:25 og pch=21:25 kan fylles med farger.
Oversikt over plottesymboler:
plot(0:25,pch=0:25,col="blue",cex=2,xlab="pch=0-25",main="Plottesymboler")



Splines og kurveglatting

Forskjellige former for kurveglatting via funksjonene **?spline,?approx** og **?smooth.spline:**

```
x<-seq(0,10,1)
y<-c(2,4,8,7,9,12,8,6,3,5,6)
par(bg="lightyellow")
plot(x,y,pch=19,cex=2,col="blue")
lines(spline(x,y),lty=2,col="red",lwd=3)
lines(approx(x,y, xout=seq(0,10,0.1)),lty=2,lwd=3,col="green")
lines (smooth.spline(x,y),lty=2,col="cyan",lwd=3)
```



Kurveglatting

Statistiske modeller i R

En modellformel angir variable og faktorer som inngår i modellen.
Modellkoeffisientene er skjæring og stigning på den tilpassete lineære modell-linjen
En modell har følgende form:

Observert verdi ~modell prediktor + statistisk støy (ε)

Modeller skal gi inferens (implikasjon, logisk slutning, konklusjon), og vi bruker begrepet statistikk inferens.
Modellen kan utprøves med kjente data fra lignende tilfeller.
Modellene bygger på forutsetninger som må undersøkes om de er oppfylt bl.a. homogen varianse og normalfordeling.

Modell	Modellformel
Null-modell	Y~1
Regresjon	Y~X
Regresjon gjennom origo	Y~X-1
En-veis ANOVA	Y~Kat
En-veis ANOVA uten skjæring	Y~Kat-1
To-veis ANOVA	Y~Kat1+kat2
Faktoriell ANOVA	Y~Kat1*Kat2*Kat3
Tre-veis ANOVA	Y~Kat1*Kat2*Kat3+kat1:kat2:kat3
ANCOVA (kovarianse)	Y~X +Kat
ANCOVA (kovarianse)	Y~X*Kat
Nestet ANOVA	Y~Kat1/Kat2/Kat3
Split-plot ANOVA	Y~Kat1*Kat2*Kat3+Error(kat1/kat2/kat3
Multippel regresjon	Y~X+Z
Multippel regresjon	Y~X*Z
Multippel regresjon	Y~poly(X,2)+Z
Ikke-parametrisk (gam)	Y~s(X)+s(Y)
Transformerte variable	Log(Y)~sqrt(X)

Kat er kategorisk variabel, X og Z er kontinuerlige variable

Linkfunksjoner:
Identitets link (normalfordeling):

$$y = a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_2 \cdot x_2$$

Log link (Poissonfordeling):

$$log(\lambda) = a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_2 \cdot x_2$$

Logit link (Binomial fordeling):

$$logit\,(y) = a + b_1 \cdot x_1 + b_2 \cdot x_2 + \cdots b_2 \cdot x_2$$

lm(y~x-1) er en regresjonsmodell som går gjennom origo. Hvis x er et polynom blir modellen **lm(y~poly(x,degree=n)**, n=2 for et andregradspolynom, **I(x^n)**vil si et n-gradspolynom av x.

	Normal	Poisson	Binomial	Gamma
Link	Identitet	log	logit	resiprok
Varianse	1	μ	μ(n-μ)/n	μ^2
Devianse	$\sum(x-\mu)^2$	$2\sum y ln(y/\mu)-(y-\mu)$	$2\sum y ln(y/\mu)+(n-)\cdot ln(n-y)/(n-\mu)$	$2\sum(y-\mu)/y -ln(y/\mu)$

Poisson-fordelingen er for tellinger, binomial fordeling for binære tall 0 og 1, samt normalfordeling for kontinuerlige data. Linkfunksjonen transformerer de ikke-lineære deler til et sett med lineære parametere.
I generaliserte lineære modeller og logistisk regresjon bruker vi begrepet **devianse** (avvik) fra modellen. Deviansen bestemmes på forskjellige måter i de forskjellige fordelingene. Mens man i lineær regresjon bruker kvadratsum og middelkvadratsum

tilsvarer dette henholdsvis devianse og **middeldevianse** i logistisk regresjon.

Hvis residual devianse er by større enn antall frihetsgrader indikerer dette overdispersering. Likelihood (L) er sannsynligheten for å få gitte avhengige variable i modellen gitt parameterverdier α og $\beta_1,\ldots\beta_n$. Maksimum likelihood estimat er gitt ved minimalisering av -log(L).

Her er eksempler på noen av kommandoene som kan brukes til statistisk modellering:

aov – variansanalyse (ANOVA)
factanal – faktoranalyse
glm – generaliserte lineære modeller
gam – generaliserte additive modeller (hentes fra **library(mgcv)**)
loess – lokal polynom regresjonstilpasning
lm – tilpasse lineære regresjonsmodeller
lme – lineære mixed-effekt modeller (**library(nlme)**)
nlme – ikke-lineære mixed-effekt modeller (**library(nlme)**)
manova – multivariabel variansanalyse
nls – ikke-lineære minste kvadrat modeller
princomp – prinsipalkomponentanalyse

I tillegg er det en rekke bibliotek (library)som kan lastes ned som pakker (packages) i R:
library(boot) – bootstrapping
library(multiv) – multivariabel analyse
library (cluster) – klyngeanalyse
library(survival) - overlevelsesanalyse

Generaliserte additive modeller er gunstig å bruke hvor man ikke har kjennskap til formen på relasjonen mellom x og y, inkludert ikke-parametriske glattingsfunksjoner med forskjellige typer linkfunksjoner. En slik modell kan være y~s(x) + s(z) hvor s er en glattingsfunksjon, eller y~x + s(z) og andre kombinasjoner. Rutiner for gam er laget av Simon Wood i pakken **mgcv** (**library(help=mgcv)**).

Null-modellen

Den enkleste modellen er null-modellen y~1 som har bare en parameter, middelverdien.
La x være tallene 2,4,6 og 8,n=4, vi lar y=x og ser på lm(y~1), hvor vi finner at skjæringspunkt (intercept) er lik middeltallet:

```
x<-c(2,4,6,8)
y<-x
model<-lm(y~1)
summary(model)
```

```
Call:
lm(formula = y ~ 1)

Residuals:
 1  2  3  4
-3 -1  1  3

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    5.000      1.291   3.873   0.0305 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.582 on 3 degrees of freedom
```

```
m<-mean(x);m
[1] 5
```

$$\bar{x} = \frac{\sum_{i=0}^{n} x_i}{n} = \frac{2+4+6+8}{4} = \frac{20}{4} = 5$$

Vi ønsker å gjøre kvadratsummen til residualene (SSR) så liten som mulig ved å finne en verdi *a* slik at de kvadrerte avvik blir minimalisert:

$$SSR = (a-2)^2 + (a-4)^2 + (a-6)^2 + (a-8)^2$$

Vi har et maksimums eller minimumspunkt når den deriverte dSSR/da=0

$$\frac{dSSR}{da} = 2(a-2) + 2(a-4) + 2(a-6) + 2(a-8) = 0$$

Dette uttrykket blir lik 0 når:

$$(a-2) + (a-4) + (a-6) + (a-8) = 0$$
$$4a = 20 \rightarrow a = 5$$

Som er snittet og det vi finner i null modellen som skjæringspunkt.

Variansen til x:

```
var(x)
[1] 6.666667
```

$$var(x) = s^2 = \frac{\sum_{i=0}^{n}(x_i - \bar{x})^2}{n-1} = \frac{(2-5)^2 + (4-5)^2 + (6-5)^2 + (8-5)^2}{4-1}$$
$$= \frac{(-3)^2 + (-1)^2 + (1)^2 + 3^2}{4-1} = \frac{9+1+1+9}{3} = \frac{20}{3} = 6.666667$$

Vi ser her i telleren de kvadrerte residualene -3,-1,1 og 3 som vi finner i null-modellen over som residuals.
Standardavviket SD:

$$SD = s = \sqrt{var(x)} = \sqrt{s^2}$$

```
SD<-sqrt(var(x));SD
[1] 2.581989
```

Det finner vi igjen i null-modellen som Residual standard error

Standardfeilen (SE)

$$SE = \frac{SD}{\sqrt{n}} = \frac{s}{\sqrt{n}}$$

```
 SE<-SD/sqrt(4);SE
[1] 1.290994
```
Den finner vi som standardfeilen til estimatet.
Og t-verdien er estimat/SE:
```
 t<-5.000/1.291;t
[1] 3.872967
```
Tabell-verdien og kritisk verdi for t med df=3
```
 qt(0.975,3)
[1] 3.182446
 pt<-pt(3.872967,3);pt
[1] 0.9847667
 2*(1-pt)
[1] 0.03046663
```

Mikset effektmodeller

Det er forskjellige typer kategoriske variable: **fikserte effekter** (lys-skygge, hann-hunn, ung-gammel, sprøytet-ikke sprøytet, gjødselet-ikke gjødslet) og **randome effekter** (genotype, blokk, foreldre, vanning, tetthet). Randomiserte effekter baserer seg på en stor populasjon. Fikserte effekter påvirker bare middeltallet til y, randome effekter påvirker bare variansen til y

Lineære mikset effekt modeller (lme) kan undersøkes med lme elelr **lmer** i pakken **lme4**. Hvis alle de kategoriske variablene er randome effekter (ingen fikserte) så blir de fikserte effektene et estimat av skjæringspunktet i modellen (fixed=y~1). De randome effektene (a,b,c) angis i hierkisk rekkefølge (nestet i hverandre) som random=~1|a/b/c.
Dette blir en modell av type:
```
model<-lme(y~1,random=~1|a/b/c)
```
Hvis for eksempel rotlengen (responsvariabel) av utvalgte planter (plante) måles i gjødslete og ikke-gjødslete områder hver uke i noen uker (temporær pseudoreplikasjon) så uttrykkes dette i modellen som fikserte effekter (fixed=rotlengde~gjødsel) og randome effekter (random=~uke|plante).
Replikater må være uavhengige av hverandre. Gjentatte målinger på samme individ eller lokalitet er ikke ekte replikater, men pseudoreplikater, henholdsvis **temporal pseudoreplikasjon** og **romlig pseudoreplikasjon**. Longitudinelle data hvor samme individ måles gjentatte ganger over tid gir temporal pseudoreplikasjon

Split-plot eksperiment og mikset effektmodeller

Ikke-lineære og lineære mikset effektmodeller undersøkes med pakken **nlme** (**library(help=nlme)**)

Datasettet **Oats** (?Oats)i nlme er resultat av et 4x72 faktorielt split-plot eksperiment: Avling (yield) med 3 varieteter (Variety) av havre og 3 nitrogenkonsentrasjoner (nitro) i et hierarkisk blokkdesign med 6 blokker(Block, VI < V < III < IV < II < I), hver blokk med 3 hele plot, hver inndelt i 4 sub-plot.

```
 attach(Oats)
 names(Oats)
[1] "Block"   "Variety" "nitro"   "yield"
 Oats
Grouped Data: yield ~ nitro | Block
   Block     Variety nitro yield
1      I     Victory   0.0   111
2      I     Victory   0.2   130
```

Osv.

Analysert med nlme og Block som random faktor, og metode ML-maksimum likelihood. Som defålt-verdi er REML, og som egentlig er best og tar hensyn til antall df som blir brukt av fikserte effekter. Vi bruker allikevel metode=ML, siden da kan vi sammenligne modeller med anova.

```
library(nlme)
model1<-
lme(yield~nitro*Variety,random=~1|Block/Variety,method="ML",da
ta=Oats)
summary(model1)

Linear mixed-effects model fit by maximum likelihood
 Data: Oats
       AIC      BIC    logLik
  618.0616 638.5516 -300.0308

Random effects:
 Formula: ~1 | Block
        (Intercept)
StdDev:    13.36901

 Formula: ~1 | Variety %in% Block
        (Intercept) Residual
StdDev:     9.24302 12.62438

Fixed effects: yield ~ nitro * Variety
                           Value Std.Error DF   t-value p-value
(Intercept)              81.90000  8.265221 51  9.908991  0.0000
nitro                    75.33333 12.036877 51  6.258545  0.0000
VarietyMarvellous         8.51667  8.463740 10  1.006253  0.3380
VarietyVictory           -8.60000  8.463740 10 -1.016099  0.3335
nitro:VarietyMarvellous -10.75000 17.022715 51 -0.631509  0.5305
nitro:VarietyVictory      5.75000 17.022715 51  0.337784  0.7369
 Correlation:
                        (Intr) nitro  VrtyMr VrtyVc ntr:VM
nitro                   -0.437
VarietyMarvellous       -0.512  0.427
VarietyVictory          -0.512  0.427  0.500
nitro:VarietyMarvellous  0.309 -0.707 -0.603 -0.302
nitro:VarietyVictory     0.309 -0.707 -0.302 -0.603  0.500

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.86612124 -0.68166416 -0.07901088  0.60390090  1.72719507

Number of Observations: 72
Number of Groups:
             Block Variety %in% Block
                 6                18
```

Vi ser at interaksjonsleddet er ikke signifikant og vi kan forenkle modellen:

```
model2<-
lme(yield~nitro+Variety,random=~1|Block/Variety,method="ML",da
ta=Oats)
summary(model2)
```

```
Linear mixed-effects model fit by maximum likelihood
 Data: Oats
       AIC      BIC    logLik
  615.1077 631.0444 -300.5539

Random effects:
 Formula: ~1 | Block
        (Intercept)
StdDev:      13.369

 Formula: ~1 | Variety %in% Block
        (Intercept) Residual
StdDev:    9.200769 12.74726

Fixed effects: yield ~ nitro + Variety
                   Value Std.Error DF   t-value p-value
(Intercept)       82.40000  7.612467 53 10.824349  0.0000
nitro             73.66667  6.913172 53 10.655986  0.0000
VarietyMarvellous  5.29167  6.649475 10  0.795802  0.4446
VarietyVictory    -6.87500  6.649475 10 -1.033916  0.3255
 Correlation:
                  (Intr) nitro  VrtyMr
nitro             -0.272
VarietyMarvellous -0.437  0.000
VarietyVictory    -0.437  0.000  0.500

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-1.67778032 -0.66417570 -0.04535348  0.56284561  1.78923973

Number of Observations: 72
Number of Groups:
             Block Variety %in% Block
                 6                 18
```

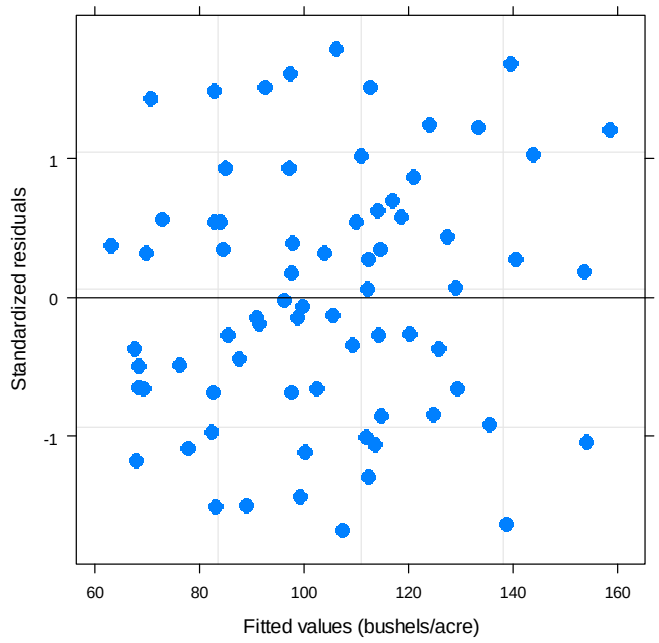
Vi ser av tabellen også informasjonskriteriene (AIC og BIC) hvor vi velger den med lavest verdi, samt log-likelihood (LogLik). Siden vi har brukt metode maksimum likelihood kan vi også sammenligne modellene med anova.

```
anova(model1,model2)
```

```
       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
model1     1  9 618.0616 638.5516 -300.0308
model2     2  7 615.1077 631.0444 -300.5539 1 vs 2 1.046115  0.5927
```

Vi ser at det ikke er noen signifikant forskjell mellom modellene og vi beholder da den enkleste, model.2
Vi plotter modell 2 og ser at residualene er i orden.

```
plot(model2,pch=16,cex=1.5)
```



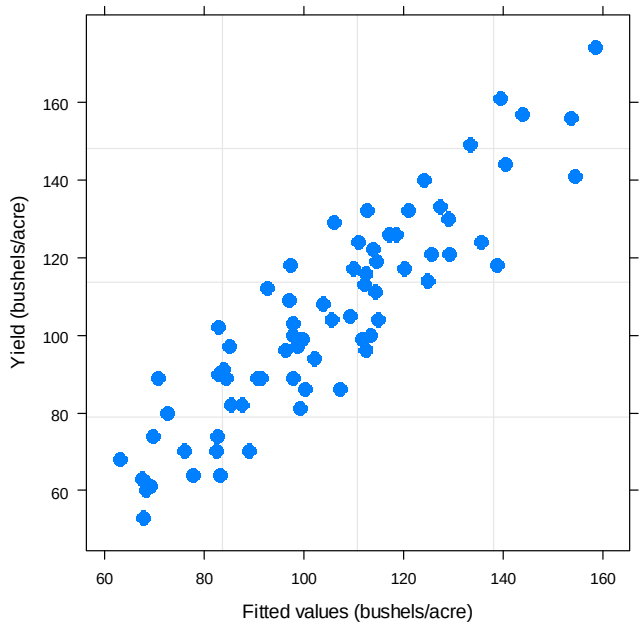
100 bushels/acre tilsvarer ca. 358 kg/da.
Yield er målt i bushels/acre (1 bushel havre=14.5 kg og 1 acre er 4.04686 dekar) og skal man regne om til kg/da kan man bruke

```
Oats2<-transform(Oats,yield=yield*14.5/4.04686)
```

Deretter kunne man i stedet gjort modellundersøkelser med datasett Oats2.

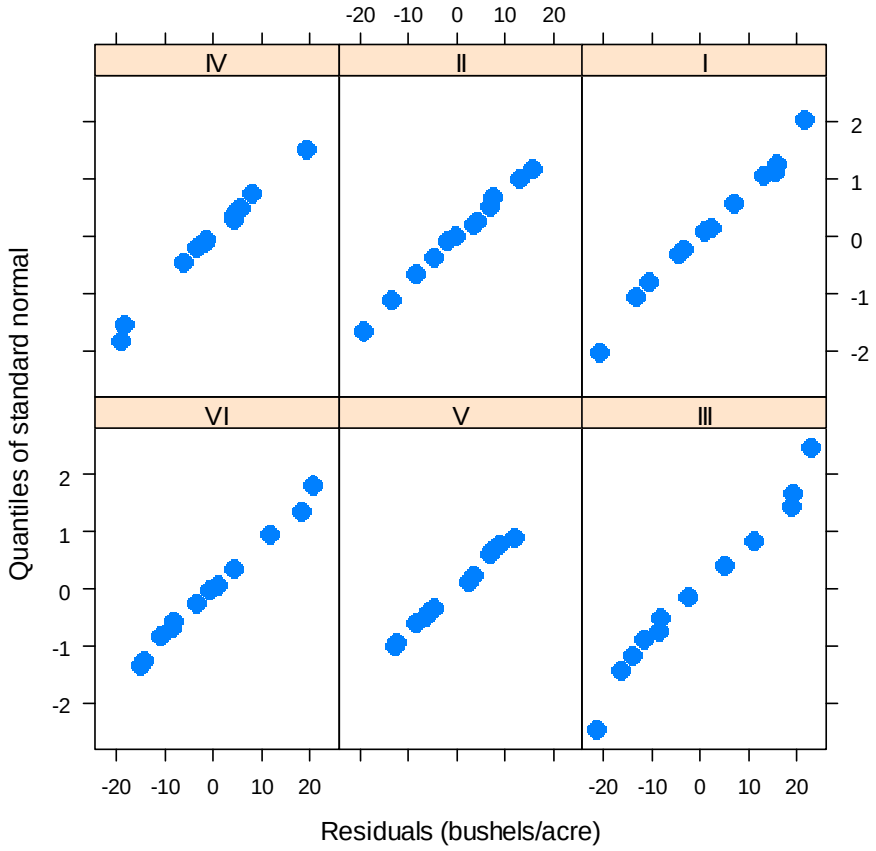
Vi ser om responsvariabel er lineær med tilpassete verdier, og det ser rimelig ut:

```
plot(model2,yield~fitted(.),pch=16,cex=1.5)
```

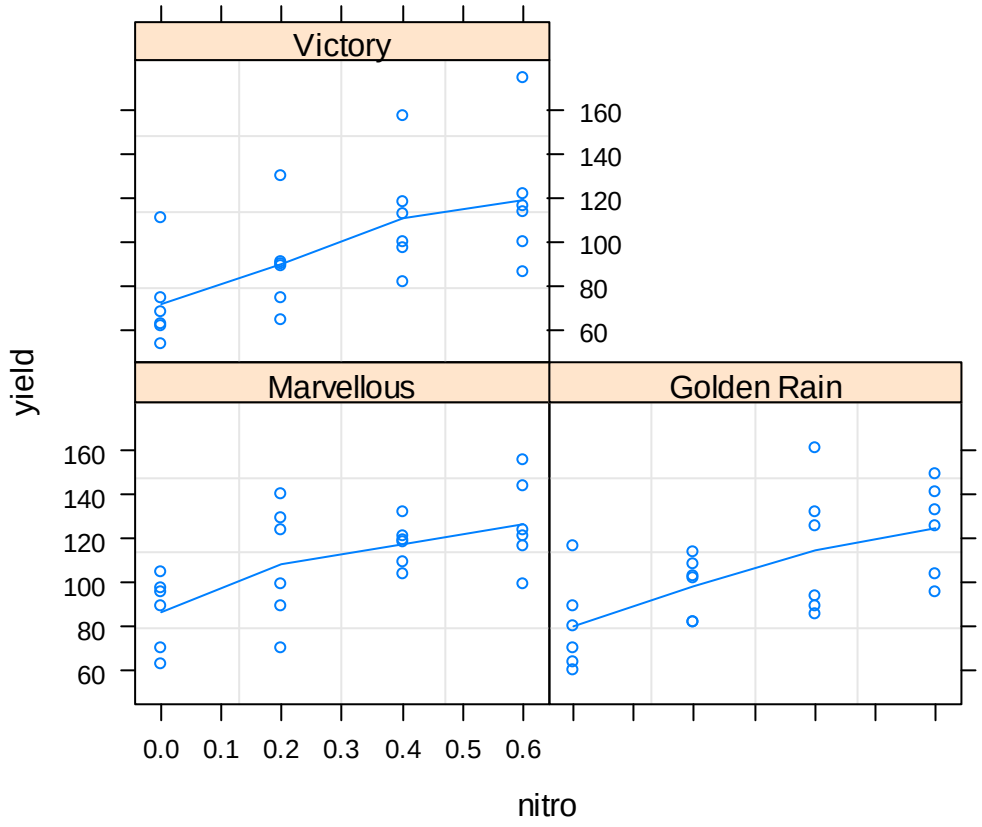


Vi kan deretter se på residualene i hver av de 6 blokkene, om de ser normalfordelte ut:

```
qqnorm(model2,~resid(.)|Block,pch=16,cex=1.5)
```




```
library(lattice)
samlet<-groupedData(yield~nitro|Variety,outer=~Block)
plot(samlet)
```



Hierarkisk datainnsamling og mikset effekt modeller

I observasjonsstudier hvor man samler inn data fra forskjellige geografiske områder med forskjellig regional skala hus, gater, distrikter, byer, regioner og land er det romlig pseudoreplikasjon og alle disse kategoriske variablene er randome effekter. Med **varianse komponent analyse** kan man undersøke om hvor er denne regionale skalaen er det mest variasjon.

Generelt hvis man har data med pseudoreplikasjon og ikke-normalte fordelte error kan man vurdere bruk av generaliserte lineære mikset effektmodeller og **lmer**, hvor lmer kan bruke de samme error-familer som generaliserte lineære modeller, altså family=binomial, familiy=poisson osv.
Split-plot eksperiment på en 96-brønners plate med 8 rader A-H, 12 kolonner 1-12. Bruker bare de 60 brønnene i midten (rad B-G) og kolonne 2-11. Blokk 1 er kolonne 2-6 og blokk 2 kolonne 7-11. I hver blokk er det 6 prøver plassert tilfeldige i radene og 5 faktornivåer plassert tilfeldig i hver kolonne. Måler absorbanse som responsvariabel. Som eksempel kan man bruke datasett Assays i nlme.

Programmering i R

Programspråket i R er basert på objekter, grunnlaget for objektorientert programmering.

I R er alt objekter og hvert objekt tilhører en klasse. Objektorienterte programmering behandler objekter, klasser og metoder. Klassene kan være for eksempel en vektor med heltall (integer), en vektor med reelle numeriske tall (numeric), vektor med komplekse tall (complex), matriser (matrix), funksjoner (function), faktorer (factor), logiske verdier (logical, TRUE/FALSE), og datarammer (data.frame). I R finnes det mange forhåndsdefinerte klasser med plottemetoder som brukes til å plotte objekter:

```
 methods("plot")
```

Vi kan se hva slags klasse et objekt tilhører med kommandoen **class()**. Klassene er av type numeric, factor, logical etc.

```
x<-1:10
 class(x)
[1] "integer"

 korn<-list(c("bygg","rug","havre"))
 class(korn)
[1] "list"

 class(plot)
[1] "function"

 NPK<-factor(c("nitrogen","fosfor","kalium"))
 class(NPK)
[1] "factor"
 NPK
[1] nitrogen fosfor   kalium
Levels: fosfor kalium nitrogen
```

Med **setClass** kan man lage en klasse, og elementene i en klasse kalles spor (slots) som kan identifiseres med **getSlots**.
Med funksjonen **new** kan man lage objekter fra en klasse.
cat – lenker sammen (konkatenerer) og printer
paste - konkatenerer strenger.

For eksempel vis dato og klokkeslett på skjermen:

```
 paste("I dag er datoen", date())
```

show – viser et objekt
setMethod – lager og lagrer metoder
dumpMethod – behandler metoder
slot – objekt i en klasse
source – leser R-kode fra en fil

traceback – hvilke funksjoner som ble kalt ved feil (feilsøking)
trace – brukes i feilsøking

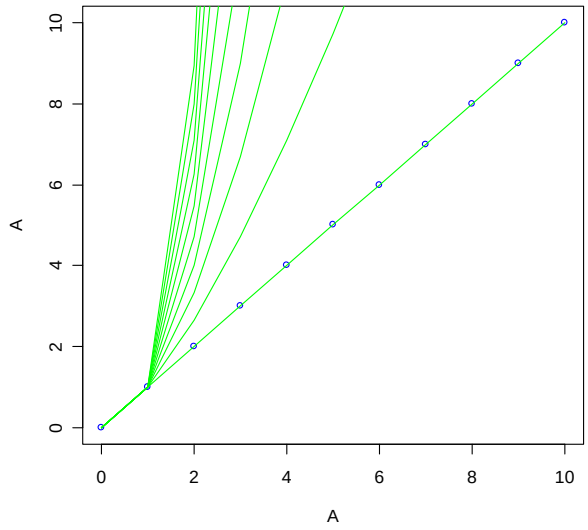
Vi kan bruke **for**-løkker (for-loops)for å gjenta prosesser (iterasjoner), som har følgende oppsett:

```
for (vektorvariabel){
   Kommandoer
}
```

Andre kommandoer er **while**, **repeat**, og **if**.
For eksempel:

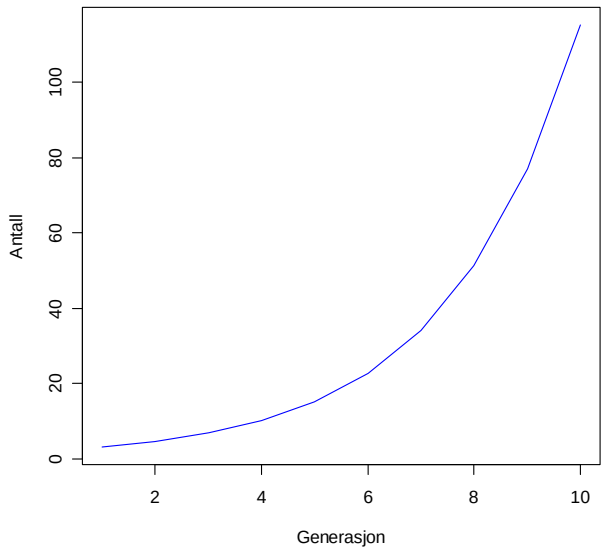
```
if(betingelser){
      Utfør følgende kommandoer
}else{
      Utfør andre kommandoer
}
```

```
A<-seq(0,10,1)
plot(A,A,col="blue")
for (i in 1:10)lines(A,A^sqrt(i),col="green")
```



Vi kan beregne økning i en populasjon som starter med 3 individer, vekstøkning 1.5 Vi lager en vektor kalt "vekst" som kan lagre dataene, lager en løkke og deretter lager vi et plot i følgende script-fil. **"\n"** gir linjeskift.

```
start<-3
vekst<-rep(0,10);vekst[1]<-start
for (n in 2:10) {
vekst[n]=1.5*vekst[n-1];
x<-log(vekst[n]);
cat(n,x,"\n")
}
plot(1:10,vekst,col="blue",type="l",xlab=
"Generasjon",ylab="Antall")
```



Logiske operatorer

I R brukes **TRUE** om sann og **FALSK** om falsk:

```
 8<10
[1] TRUE
 z<-1:10
 z<5
 [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Manglende tall i et datasett beskrives med **NA** (not available).

Vi kan for eksempel se om tre vektorer x, y og z er like eller forskjellige:

```
 x=(1:10);y=(1:10);z=(2:20);identical(x,y);identical(x,z);
[1] TRUE
[1] FALSE
```

Andre logiske operatorer er:

Symboler	Logisk betydning
& eller &&	Logisk og (AND)
!	Logisk ikke (NOT)
| eller ||	Logisk eller (OR)
==	Logisk lik (=)
!=	Logisk ikke lik
<	Mindre enn
>.	Større enn
>=	Større enn eller lik
&&	Logisk og (AND) sammen med hvis (IF)
||	Logisk eller (OR) sammen med hvis (IF)
TRUE	Sann
FALSE	Falsk, usann
isTRUE(x)	

Funksjoner

Bruk av funksjoner (**function(x)**) for eksempel beregning av areal av en sirkel via en funksjon y=f(x):
Overflaten (areal) av en sirkel er gitt ved:

$$areal = \pi \cdot r^2$$

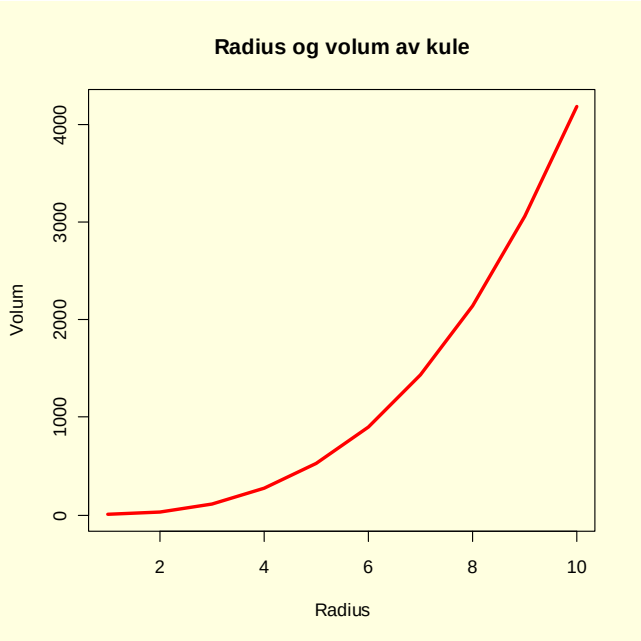
Vi kan lage en funksjon som regner ut overflaten av sirkelen ved forskjellige verdier av radius *r*:

```
areal<-function(r){pi*r^2}
r<-(1:5)
overflate<-areal(r);overflate
[1]  3.141593 12.566371 28.274334 50.265482 78.539816
```

Volumet av en kule er gitt ved formelen nedenfor og vi kan finne volumet ved *e.g.* radius fra 1-10.

$$volum = \frac{4}{3} \cdot \pi \cdot r^3$$

```
kule<-function(x){4/3*pi*r^3}
r<-seq(1,10,1)
volum<-kule(r);volum
[1]    4.18879   33.51032  113.09734  268.08257  523.59878  904.77868
 [7] 1436.75504 2144.66058 3053.62806 4188.79020
par(bg="lightyellow")
plot(r,volum,type="l",col="red",lwd=3,
xlab="Radius",ylab="Volum",main="Radius og volum av kule")
```



Figur. Sammenheng mellom radius og volum av en kule.

Overflaten av en kule er gitt ved:

$$overflate = 4\pi \cdot r^2$$

Vi kan nå lage en funksjon som regner ut både overflate og volum, *e.g.* av jordkloden som har radius 6380 km. Svar i henholdsvis km^2 og km^3.

```
kule<-function(r){
vol=4/3*pi*r^3
overfl=4*pi*r^2
return(list(vol=vol,overfl=overfl))
}
kule(6380/2)
$vol
[1] 1.087804e+12

$overfl
[1] 511506576
```

Vi kan lage funksjoner:

```
funksjon<- function(x) {
y <- cos(2*x) + sin(4*x)
return (y)
}
```

Vi kan kalle på funksjonen for å finne en tilsvarende y-verdi:

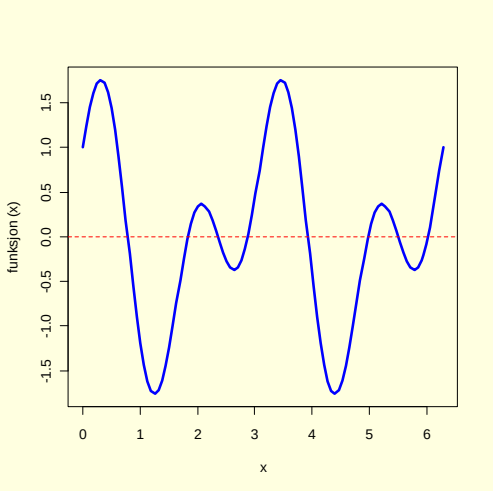
```
funksjon(0.5)
[1] 1.449600
```

Eller for et imaginært tall

```
funksjon(2+4*1i)
[1] 4394799-645337i
```

Og vi kan plotte funksjonen fra 0-2π:

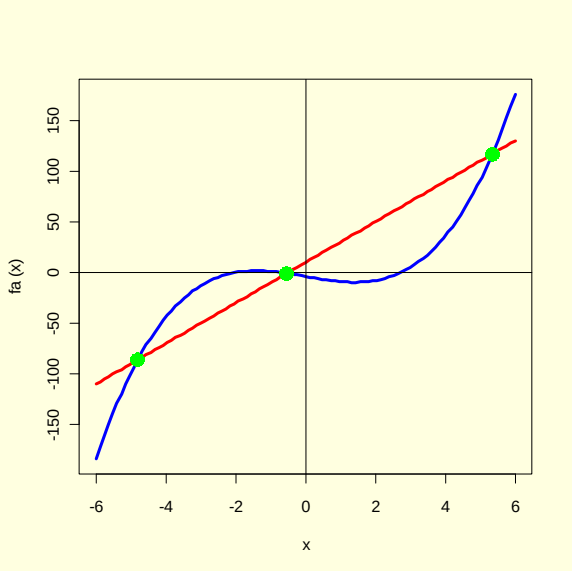
```
par(bg="lightyellow")
plot(funksjon,0,2*pi,col="blue",lwd=3)
abline(h=0,col="red",lty=2)
```



Hvis man skal lage flere plot på hverandre tilføyes **,add=TRUE** i plottekommandoen.

Vi har to funksjoner $f(x)=x^3-6x-4$ og $f(x)=10-20x$.

```
fa<-function(x) {return(x^3-6*x-4)}
fb<-function (x) {return(10+20*x)}
plot(fa,-6,6,col="blue",lwd=3)
plot(fb,-6,6,col="red", lwd=3,add=TRUE)
abline(h=0)
abline(v=0)
```



Vi ser at de to funksjonene har 3 skjæringspunkter, mellom 5 og 6, mellom -1 og 0 og mellom -5 og - 6.

Numerisk bestemmelse av røtter

Vi kan numerisk estimere **røttene** av en ny funksjon
$f(c)= f(a)-f(b)$
og bruke kommandoen **uniroot**. Rotverdien, antall iterasjoner for å finne den og estimert presisjon finnes i svaret.

```
fc<-function(x) {return(fa(x)-fb(x))}
rot<-uniroot(fc,c(5,6));rot
$root
[1] 5.349469
```

```
$f.root
[1] -0.001433008
$iter
[1] 4
$estim.prec
[1] 6.103516e-05
```

Vi gjør det tilsvarende for intervallene c(-1,0) og c(-5,-4) og finner røttene:

```
$root
[1] -0.5446767
[1] -4.804825
```

Med uniroot finner man bare en av røttene ad gangen. Ved å bruke library **rootSolve** og kommandoen **uniroot.all** så kan man finne alle røttene samtidig:

```
library(rootSolve)
roter<- uniroot.all(fc,c(-6,6));roter
[1] -4.8048160 -0.5446765  5.3494799
```

Eller hvis man vil plotte røttene på x-aksen:

```
points(roter,fa(roter),pch=16,cex=2,col="green")
```

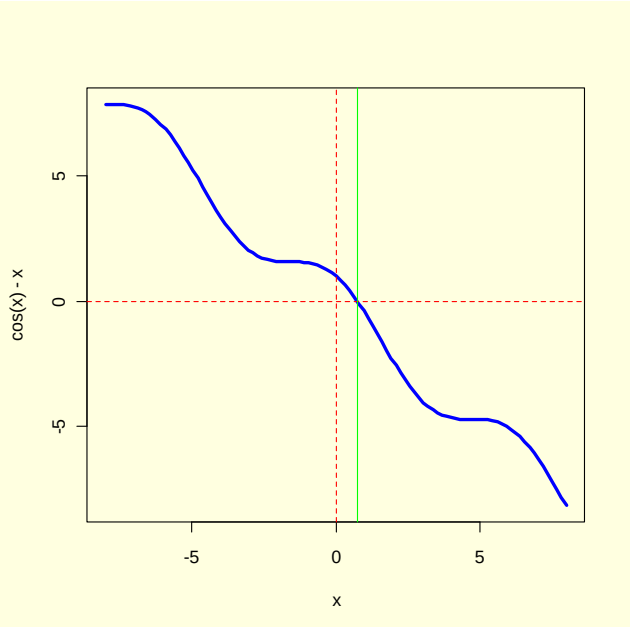
Vi kan generelt finne røttene i en ligning (skjæring med y-aksen) med kommandoen **uniroot** (se **?uniroot**). For eksempel funksjonen *y=cos(x)-x=0*

```
par(bg="lightyellow")
curve(cos(x)-x,-8,8,col="blue",lwd=3)
abline(h=0,v=0,lty=2,col="red")
rot<-uniroot(f = function(x) cos(x)-x, interval=c(-8,8));rot
$root
[1] 0.7390978
$f.root
[1] -2.120884e-05
$iter
[1] 6
$estim.prec
[1] 6.103516e-05
```

Den estimerte roten er 0.739 (ingen eksakt verdi) og den tilsvarende funksjonsverdi, antall iterasjoner og presisjonsnivå er angitt. Hvis det er flere røtter i en funksjon velger man egnet intervall etter først å ha plottet funksjonen.

Vi trekker en grønn vertikal linje for den estimerte roten:

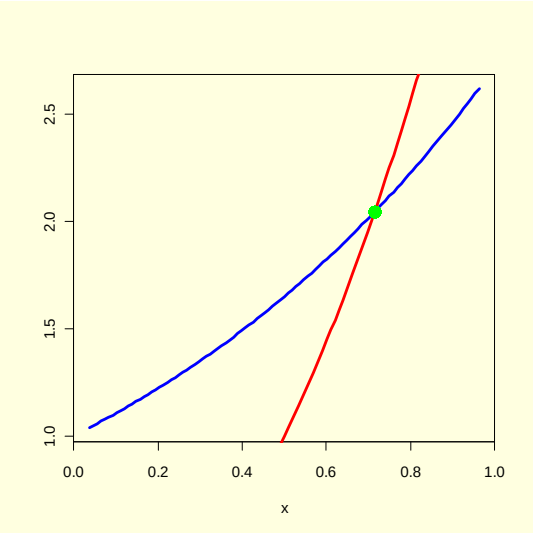
```
abline(v=rot$root,col="green")
```



Figur. Bestemmesle av rot for funksjonen $y=cos(x)-x$.

Hvis vi skal finne røttene fra funksjonen $y=e^x-4x^2$. Dette betyr at $y=0$ og det vil si at $e^x=4x^2$ som er skjæringspunktet mellom de to kurvene. Først plotter vi de to funksjonene e^x og $4x^2$. Vi ser i hvilket intervall kurvene skjærer hverandre, og setter inn det numeriske estimatet av roten som en grønn prikket linje og ser at det stemmer:

```
x<-seq(0,1,0.01)
curve(exp(x),col="blue",lwd=3,ylab="")
curve(4*x^2,add=T,col="red",lwd=3)
f<-function(x) exp(x)-4*x^2
rot<-uniroot(f, interval=c(0.6,0.8));rot
$root
[1] 0.7148064
$f.root
[1] -1.711383e-06
$iter
[1] 4
$estim.prec
[1] 6.103516e-05
points(rot$root,exp(rot$root),pch=16,cex=2,col="green")
```

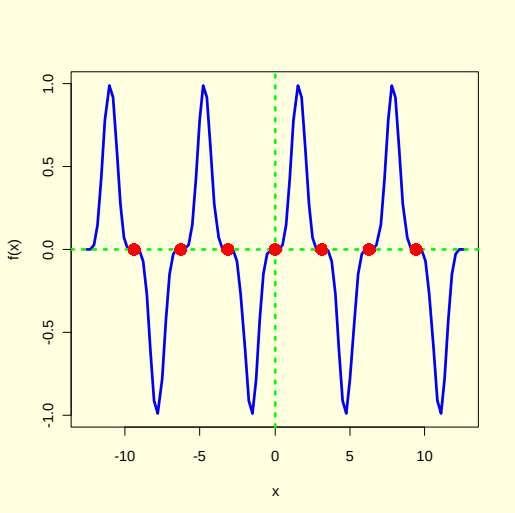


Figur: $y=e^x$ og $y=4x^2$ med tilhørende rot

Vi plotter funksjonen $sin(x)^5$ og finner røttene med uniroot.all ifølge R-manualen Soetaert,K.*Package rootSolve:roots gradients and steady-states in R.*

```
f<-function (x)sin(x)^5
par(bg="lightyellow")
curve(f(x),-4*pi,4*pi,col="blue",lwd=3)
abline(h=0,v=0,lty=3,lwd=3,col="green")
uniall <- uniroot.all(f,c(-4*pi,4*pi))
points(uniall,f(uniall),pch=16,cex=2,col="red")
uniall
```

```
[1] -9.424778e+00 -6.283185e+00 -3.141593e+00  1.776357e-15  3.141593e+00
[6]  6.283185e+00  9.424778e+00
```



Figur: $f(x)=sin(x)^5$ med tilhørende røtter for $f(x)=0$.

Klassifisering og klyngeanalyse

Ved **klassifisering** søker man etter mønstre og klassemedlemskap for biologiske data, og plasserer objekter i på forhånd definerte grupper. Dette danner basis for mønstergjenkjenning. Innen taksonomi er arter plassert i grupper basert på likheter og forskjeller.

En kortstokk kan klassifiseres på forskjellige måter som gir forskjellig antall klasser:

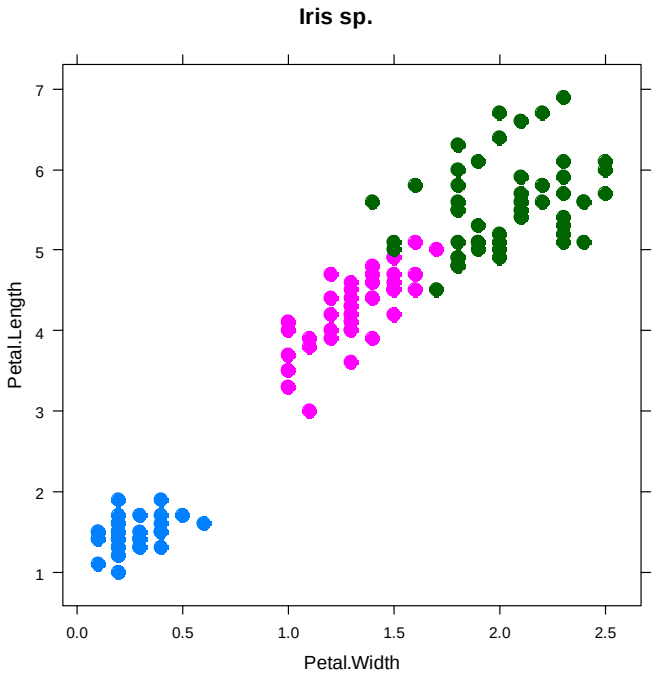
4 : ♠♣♥♦

2 : ♠♣ og ♥♦

13:1,2,3,4,5,6,7,8,9,10,Kn,Ko,Da
3: 2,4,6,8,10 og 1,3,5,7,9 og Kn,Ko,Da

Ved **klyngeanalyse** plasseres objekter eller grupper i ikke forhåndsdefinerte klasser. Naturlige grupper blir avhengig av startbetingelsene for klyngeanalysen.
Ved klyngeanalyse har man et avstandsmål (likhetsmål) hvor objekter i samme klynge er nærmere hverandre enn objekter i andre klynger. Forskjellige avstandsmål benyttes. Euklids avstand (aritmetisk avstand) er en rett linje i et euklidsk rom. Mahalanobis avstand er en annen. Algoritmen BLAST (Basic Local Alignment Search Tool) brukes til å klassifisere sekvensdata (nukletidsekvens, aminosyresekvens). Ved Bayesiansk klassifisering har man a priori (på forhånd) definert egenskaper ved klassen og man kan deretter beregne sannsynligheten for at et objekt tilhører en av de forhåndsdefinerte klassene. Bayesiansk klassifisering brukes til å definere hva som er søppelpost (spam).
Et mye brukt datasett er bladform hos iris.

```
library(datasets); data(iris); attach(iris);names(iris)
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
"Species"
 library(grid)
  library(lattice)

xyplot(Petal.Length~Petal.Width,data=iris,groups=Species,pch=1
9,cex=1.5,main="Iris sp.")
```



>

Vi ser på biblioteket **cluster** og datafilen **flower** som ligger i R:

```
library(cluster)
library(help=cluster)
flower
```

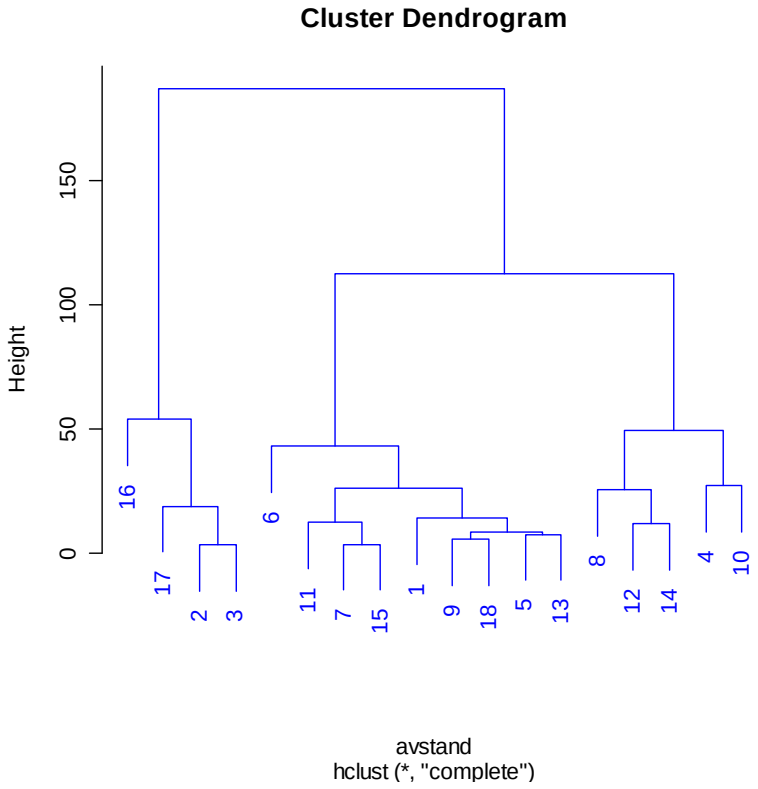
```
   V1 V2 V3 V4 V5 V6  V7 V8
1   0  1  1  4  3 15  25 15
2   1  0  0  2  1  3 150 50
3   0  1  0  3  3  1 150 50
4   0  0  1  4  2 16 125 50
5   0  1  0  5  2  2  20 15
6   0  1  0  4  3 12  50 40
7   0  0  0  4  3 13  40 20
8   0  0  1  2  2  7 100 15
9   1  1  0  3  1  4  25 15
10  1  1  0  5  2 14 100 60
11  1  1  1  5  3  8  45 10
12  1  1  1  1  2  9  90 25
13  1  1  0  1  2  6  20 10
14  1  1  1  4  2 11  80 30
15  1  0  0  3  2 10  40 20
16  1  0  0  4  2 18 200 60
17  1  0  0  2  2 17 150 60
18  0  0  1  2  1  5  25 10
```

```
?flower
```

8 egenskaper for 18 forskjellige blomster

V1 Vinterherdighet binær 0 eller 1
V2 Skyggetoleranse binær 0 eller 1
V3 Planter med rotstokk binær 0 eller 1
V4 Blomsterfarge 1=hvit, 2=gul,3=rosa, 4=rød, 5=blå
V5 Jordtypepreferanse 1=tørr, 2=normal, 3= våt
V6 Preferanse verdier fra 1-18
V7 Høyde i cm
V8 Planteavstand i cm

```
attach(flower)
names(flower)
[1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8"
```
Ønsker man å editere datafilen flower:
```
fix(flower)
```
Dissimilaritetsmatrise finner man med:
```
daisy(flower)
```
Avstandsmatrise med Euklids avstand
```
avstand<-dist(flower,method="euclidean")
?hclust
klynge<-hclust(avstand)
plot(klynge,col="blue")
```



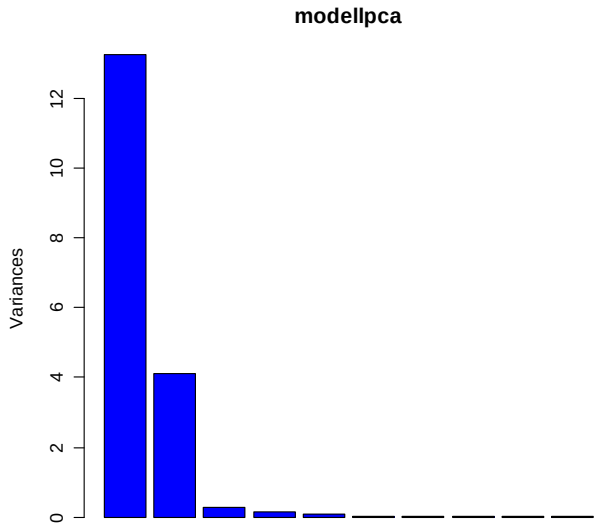
Klyngeanalyse med kommandoene **hclust** og **kmeans** er en form for multivariabel statistikk for å studere struktur.

Ved prinsipalkomponentanalyse med kommandoen prcomp ønsker man å trekke to prinsipalkompoenenter gjennom datasettet som forklarer mest mulig av variasjonen.
Det er ikke spesielt velegnet, men for enkelhet bruker vi det samme datasettet flower og matrisen med euklids avstander i en prinsipalkomponentanalyse:
```
modellpca<-prcomp(avstand,scale=TRUE)
summary(modellpca)
```

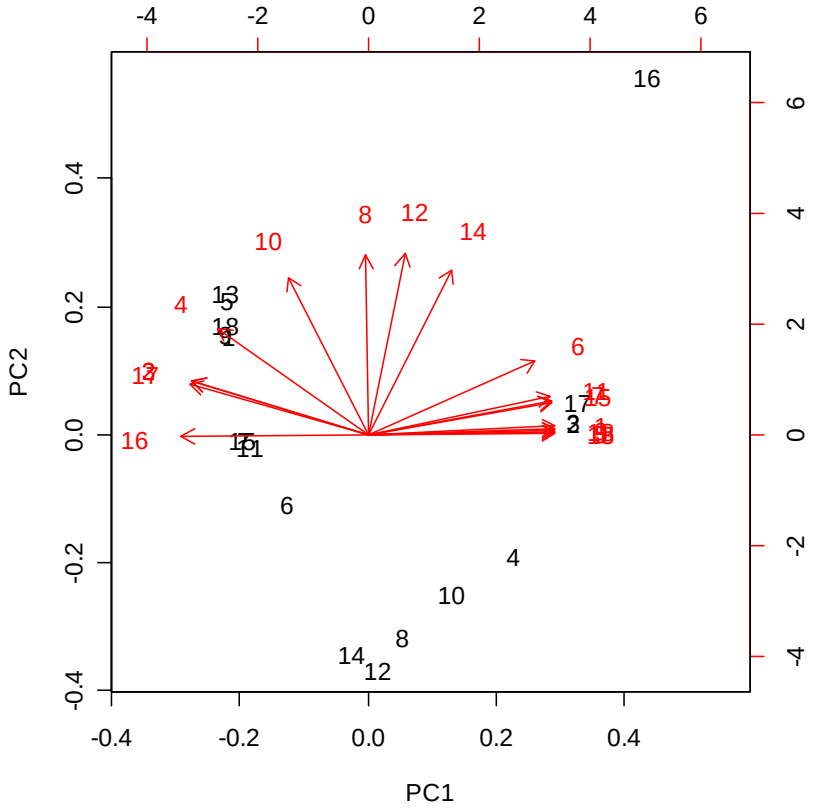
```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5    PC6     PC7
Standard deviation     3.641 2.028 0.5402 0.39376 0.29043 0.1798 0.14894
Proportion of Variance 0.736 0.228 0.0162 0.00861 0.00469 0.0018 0.00123
```

```
Cumulative Proportion  0.736 0.965 0.9811 0.98973 0.99442 0.9962 0.99745
```

Her er bare vist de 7 første prinsipalkomponentene hvor prinsipalkomponent 1 (PC1) forklarer 73.6% av variasjonen og prinsipalkomponent 2 (PC2) de neste 22.8% av variasjonen. Vi kan lage et plot som viser betydningen av de forskjellige prinsipalkomponentene:



I et biplot vises det 18 variablene (artene) med piler og i hvilken grad de påvirker positivt eller negativt på prinsipalkomponent 1 og 2.



Faktoranalyse kan utføre med kommandoen **factanal.**
Studier av neurale nettverk med

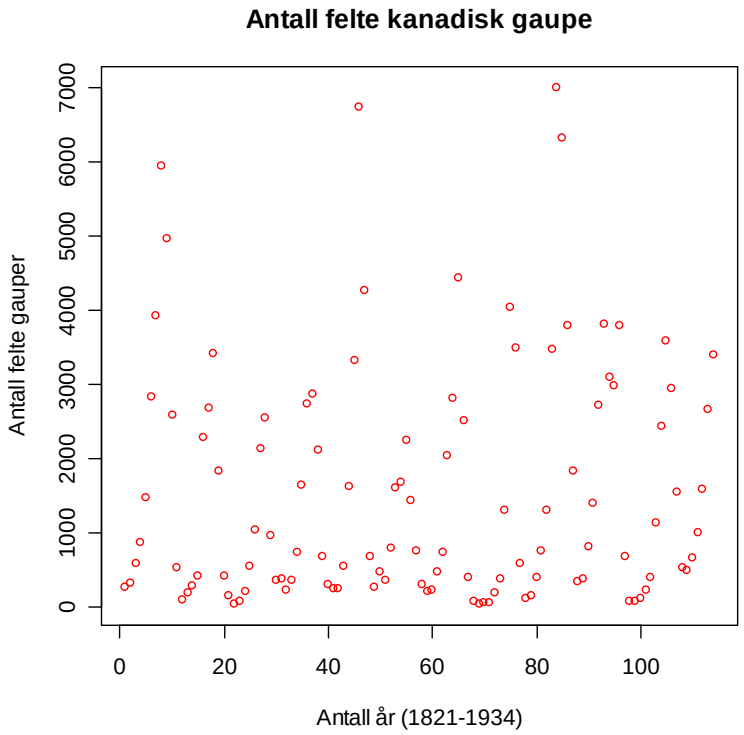
```
library(nnet)
library(help=nnet)
```

Tidsserier og trender

Tidsserier består av en tallrekke med tid på x-aksen og på y-aksen kan det være middeltemperatur, aksjekurser, årlige tellinger av dyr etc. . Tidsskalaen kan variere fra minutter til mange år. En tidsserie kan være syklisk, men behøver ikke å være det. Det kan utvikles modeller for å beskrive tidsserien.

Vi laster ned datasettet "lynx" fra R som viser årlig fangst av Kanadisk gaupe i tidsperioden 1821-1934, i alt 114 år.

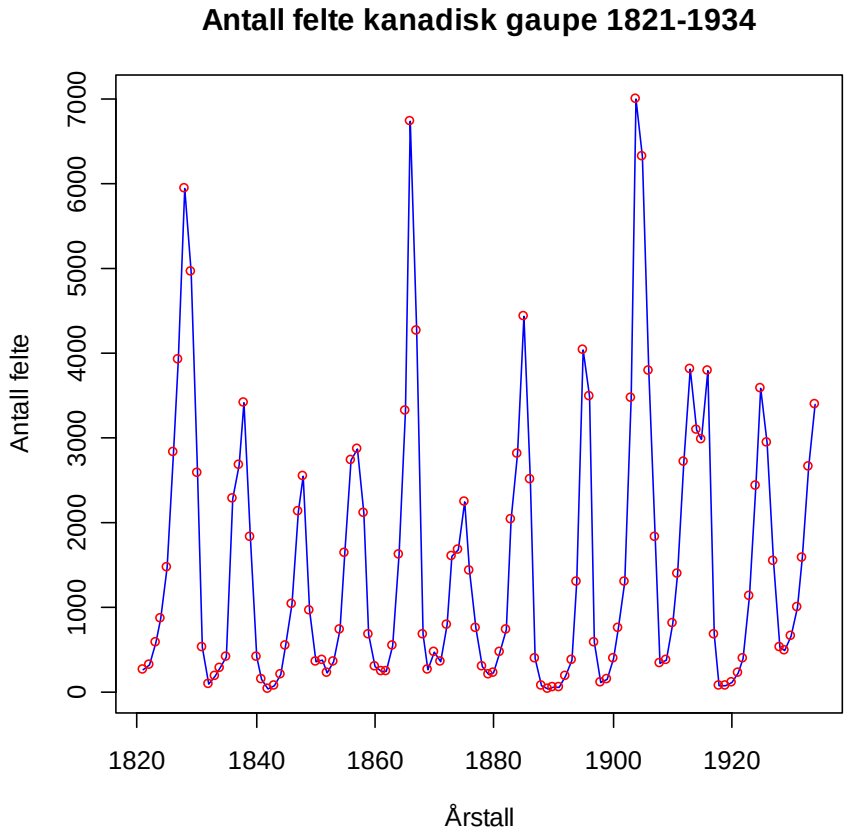
```
library(datasets)
data()
data(lynx)
index<-1:114
plot(index,lynx,col="red",xlab="Antall år (1821-1934)",ylab="Antall felte gauper",main="Antall felte kanadisk gaupe")
```



Nå kan vi utnytte mulighetene til tidsserieanalyse i R i stedet. Vi kan omforme "lynx" til et tidsserieobjekt med kommandoen **ts.** Generelt omdanner **ts()** en numerisk vektor til en tidsserie og plotting av multiple tidsserier skjer med

kommandoen **ts.plot()**. En annen type tidsserieplot via kommandoen **plot.ts()**:

```
 gaupe<-ts(lynx,start=c(1821,1))
 ts.plot(gaupe,col="blue",xlab="Årstall",ylab="Antall
felte",main="Antall felte kanadisk gaupe 1821-1934")
 points(lynx,col="red")
```

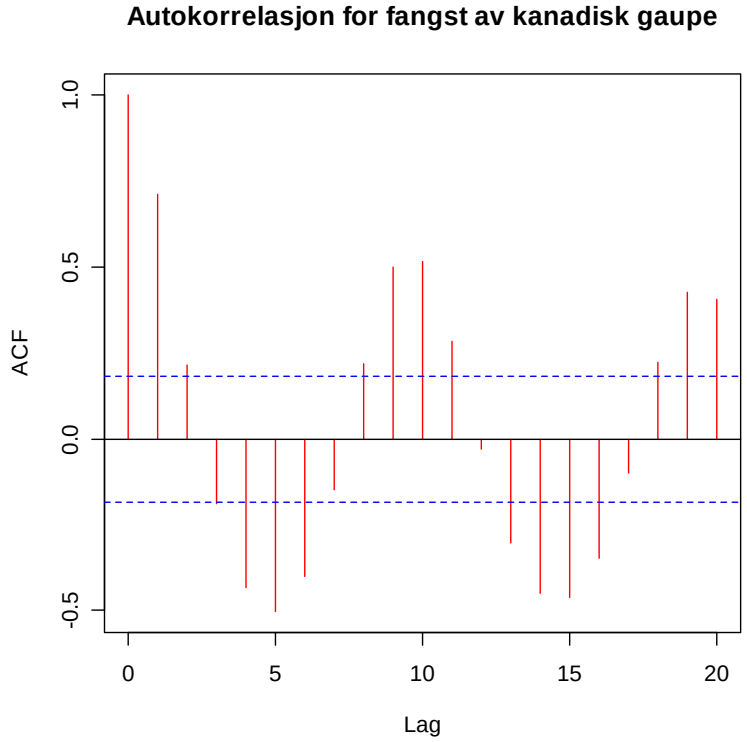


Autokorrelasjon

Ofte er det spørsmål om det er **trender** i utviklingen og seriell avhengighet. Det vil være **autokorrelasjon** mellom temperaturen i en måned og den foregående måneden, autokorrelasjon lag 1. Autokorrelasjon lag 2 sier noe om korrelasjonen mellom temperaturen i en måned sammenlignet med måneden før forrige måned osv. Graden av autokorrelasjon minsker etter som lag øker. Hvis det er en syklisk trend vil korrelasjonen gå mot 0 for deretter å bli negativ, og så tilbake igjen og bli positiv. Funksjonen **acf** bestemmer grad av autokorrelasjon og angir korrelasjon mellom lags. Det er tre typer mulige autokoorelasjonsplot a. Kovarianse beregnet ved forskjellige lags (type="covariance");b. korrelasjon beregnet ved forskjellige lags (type="correlation"); c: partiell korrelasjon ved forskjellige lags (type="partial"). Funksjonen **pacf** bestemmer **partiell autokorrelasjon**.

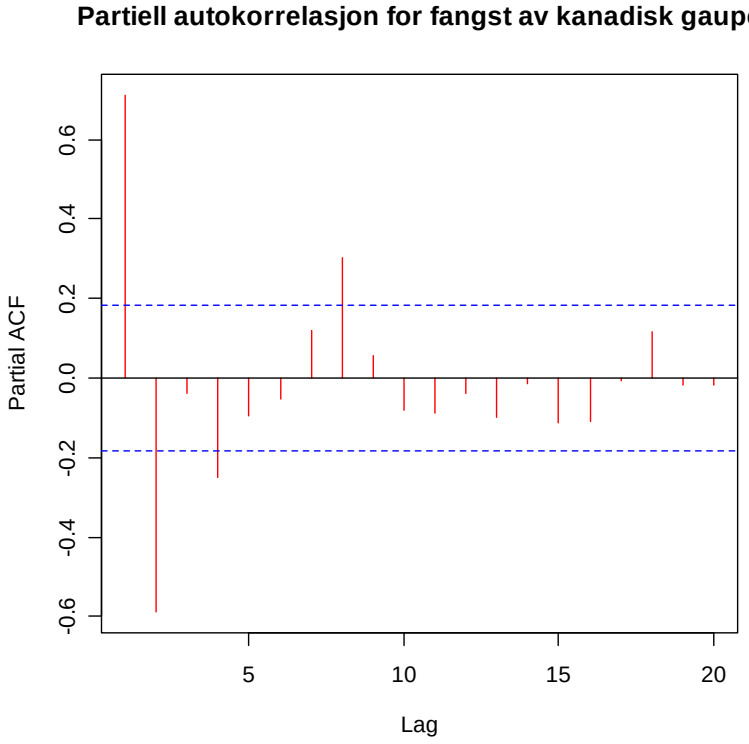
Nedenfor ser vi den sykliske variasjonen i fangst av gaupe. Denne metoden kan brukes til å studere sykliske variasjoner i biologiske prosesser. Vi bruker kommandoen **acf**.

```
acf(gaupe)
acf(gaupe,col="red",main="Autokorrelasjon for fangst av kanadisk gaupe")
```



For partiell autokorrelasjon brukes kommandoen pacf

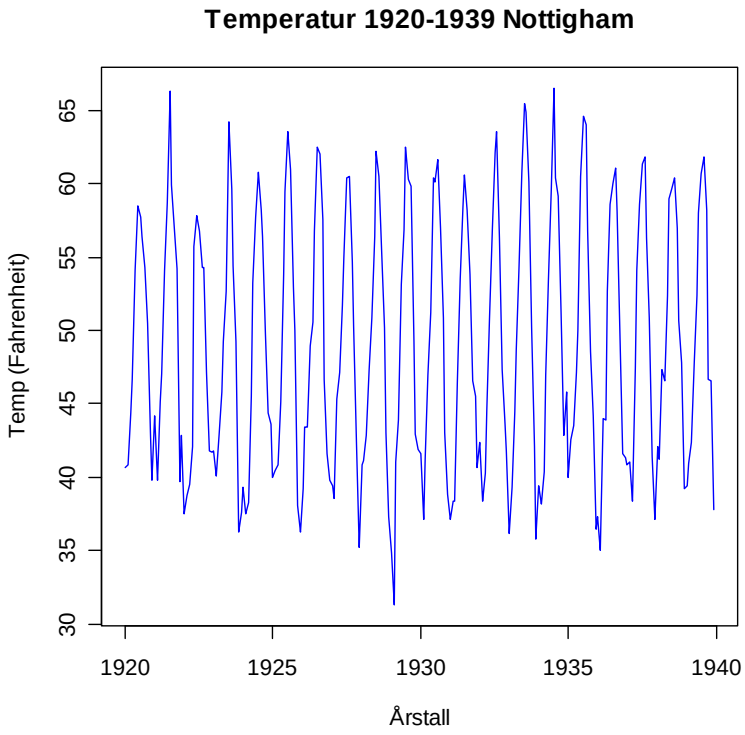
```
pacf(gaupe,col="red",main="Partiell autokorrelasjon for fangst av kanadisk gaupe")
```



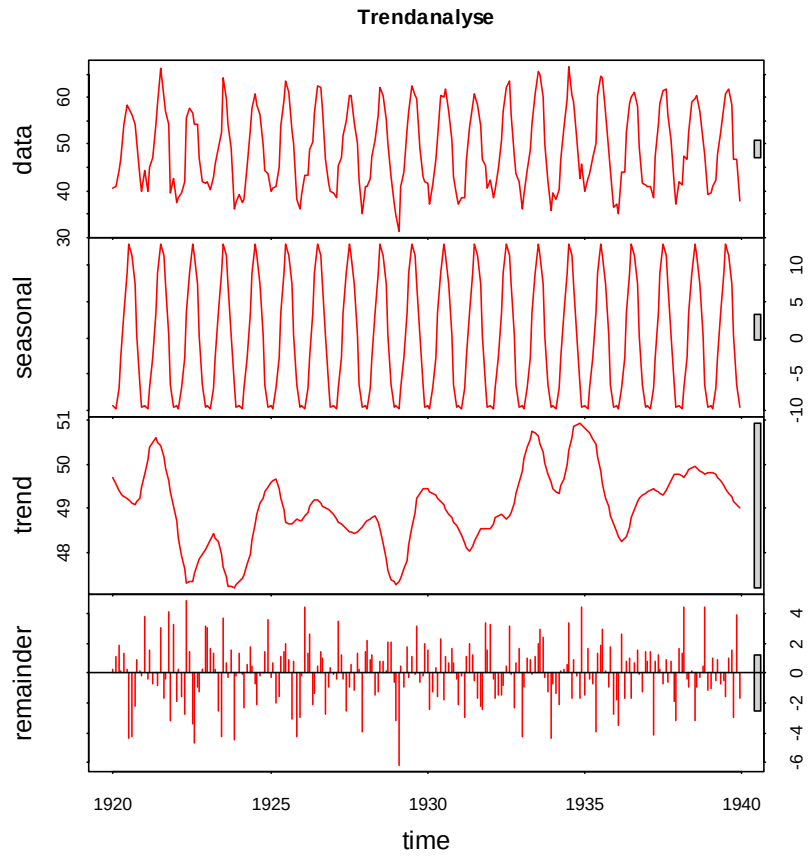
Trendanalyse

Vi kan dele opp tidsserien i komponenter og se på trender, svingninger i løpet av året og restene.
Vi bruker funksjonen **stl** for å dekomponere sesong-variasjonen med Loess-glattingsfunksjon.
Vi henter datasettet nottem som inneholder gjennomsnittlig middels månedstemperatur (Fahrenheit)i Nottingham 1920-1939

```
data()
data(nottem)
temp<-ts(nottem,start=c(1920,1),frequency=12)
 ts.plot(temp,col="blue",xlab="Årstall",ylab="Temp
(Fahrenheit)",main="Temperatur 1920-1939 Nottigham")
```

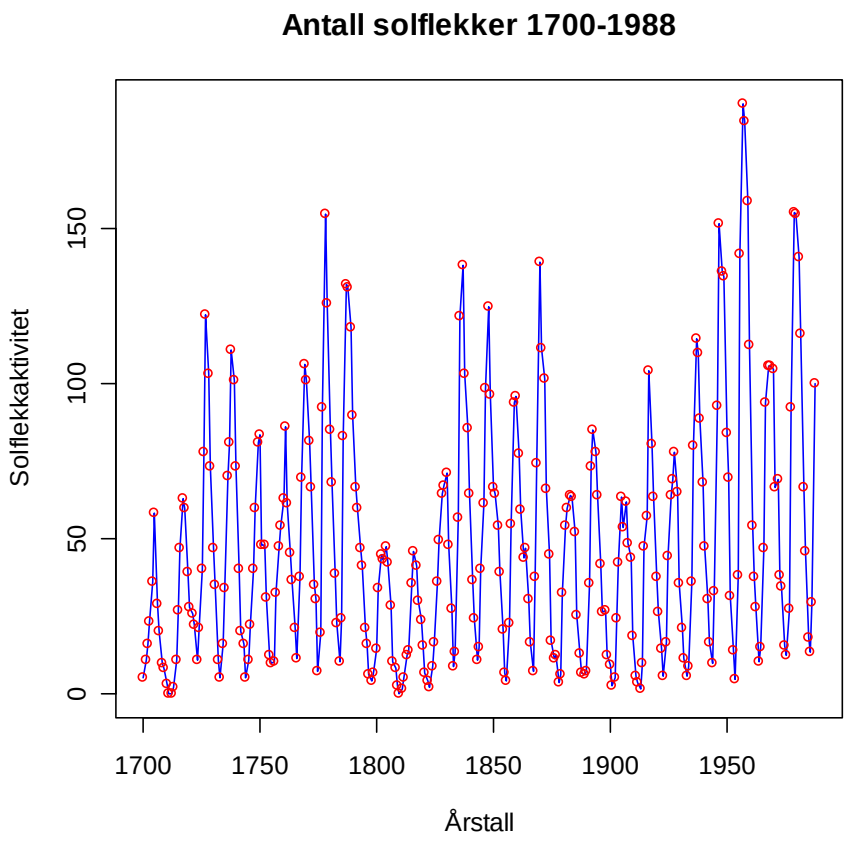



```
middeltemp<-stl(temp,"period")
plot(middeltemp,col="red",main="Trendanalyse")
```

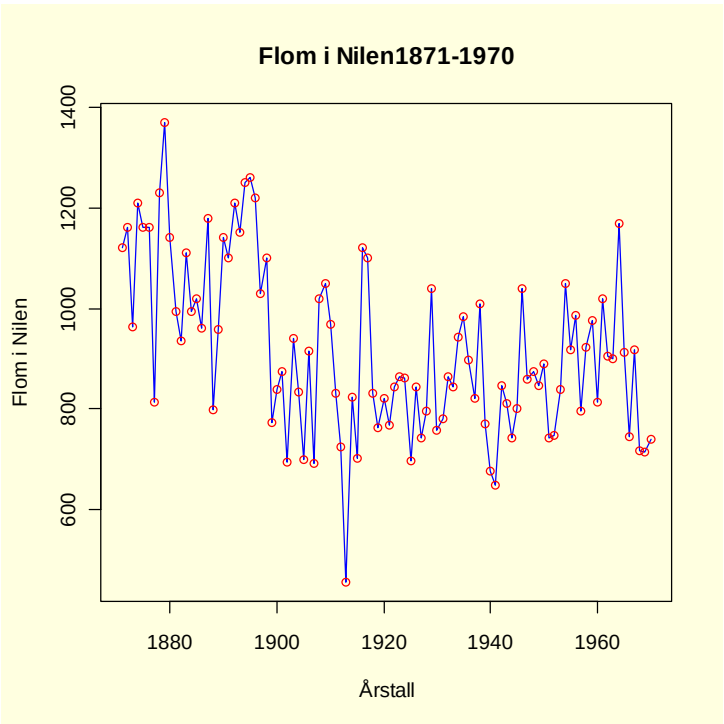


Solflekkaktiviteten har stor effekt på livet på jorda.
Vi henter datasettet sunspot.year som ligger i R.

```
  library(datasets)
 data()
 data(sunspot.year)
  solflekker<-ts(sunspot.year,start=c(1700,1))

ts.plot(solflekker,col="blue",xlab="Årstall",ylab="Solflekkakt
ivitet",main="Antall solflekker 1700-1988")
  points(sunspot.year,col="red")
```



Flom i Nilen ved Ashwan hentet fra library(datasets)




```
data(Nile)
nilen<-ts(Nile,start=c(1871,1))
par(bg="lightyellow")
```

```
ts.plot(nilen,col="blue",xlab="Årstall",ylab="Flom i
Nilen",main="Flom i Nilen1871-1970")
points(Nile,col="red")
acf(Nile)
arima(Nile)
```

Soldeklinasjon

Lag en matrise med data for soldeklinasjon hentet fra Alamanakken

	DATO	DEKLIN	DAG		DATO	DEKLIN	DAG		DATO	DEKLIN	DAG
1	1.01	-23.0	0	13	5.01	15.1	120	25	8.29	9.4	240
2	1.11	-21.8	10	14	5.11	17.9	130	26	9.08	5.7	250
3	1.21	-19.9	20	15	5.21	20.2	140	27	9.18	1.9	260
4	1.31	-17.4	30	16	5.31	21.9	150	28	9.28	-2.0	270
5	2.10	-14.4	40	17	6.10	23.0	160	29	10.08	-5.8	280
6	2.20	-10.9	50	18	6.20	23.4	170	30	10.18	-9.6	290
7	3.02	-7.2	60	19	6.30	23.2	180	31	10.28	-13.1	300
8	3.12	-3.4	70	20	7.10	22.2	190	32	11.07	-16.2	310
9	3.22	0.6	80	21	7.20	20.7	200	33	11.17	-19.0	320
10	4.01	4.5	90	22	7.30	18.5	210	34	11.27	-21.1	330
11	4.11	8.3	100	23	8.09	15.9	220	35	12.07	-22.6	340
12	4.21	11.8	110	24	8.19	12.8	230	36	12.17	-23.3	350
								37	12.27	-23.3	360

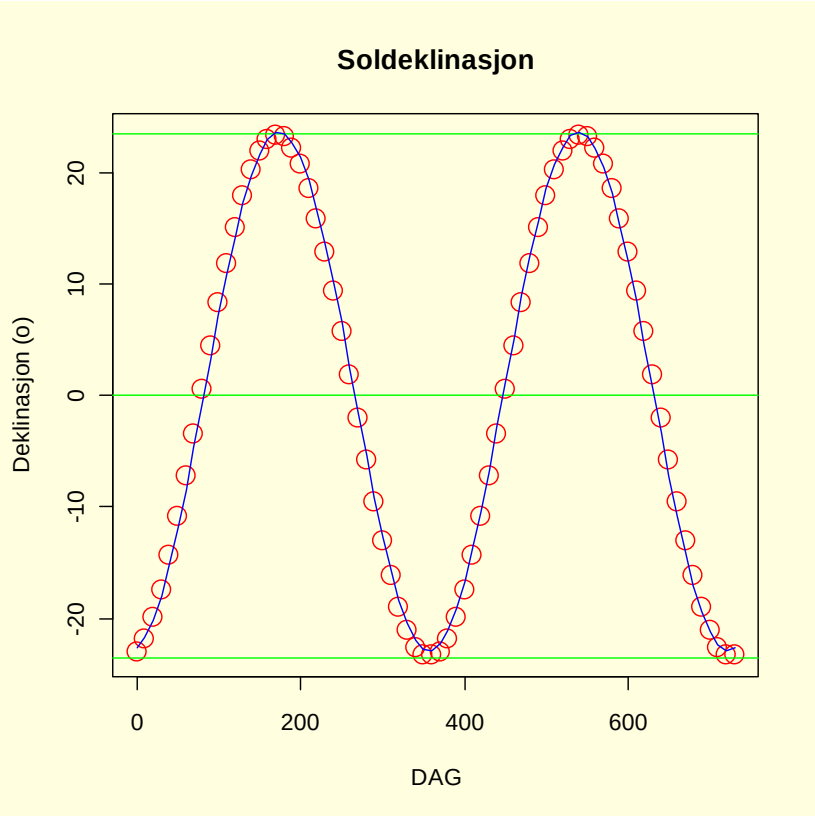
Fra *Almanakk for Norge*. UiO 196(2009)45

Ligning for sesongvariasjon er en lineær modell:

$$y = a + b \cdot \sin(2\pi t) + c \cdot cos(2\pi t) + e$$

Vi kan bestemme koeffisientene i modellen og bruke den til å prediktere

```
plot(DAG,DEKLIN, col="red",cex=2,ylab="Deklinasjon (o)",
main="Soldeklinasjon")
abline(h=0,col="green")
abline(h=23.5,col="green")
abline(h=-23.5,col="green")

day<-seq(0,730,10)
t<-day/365.2632
model<-lm(DEKLIN~sin(2*pi*t)+cos(2*pi*t))
summary(model)
lines(DAG,predict(model),col="blue")
```



Soldeklinasjon ved sommersolverv 21. juni: +23.5°, vintersolverv 21.desember:-23.5°, vårjevndøgn 20.mars og høstjevndøgn 22.september: 0°.

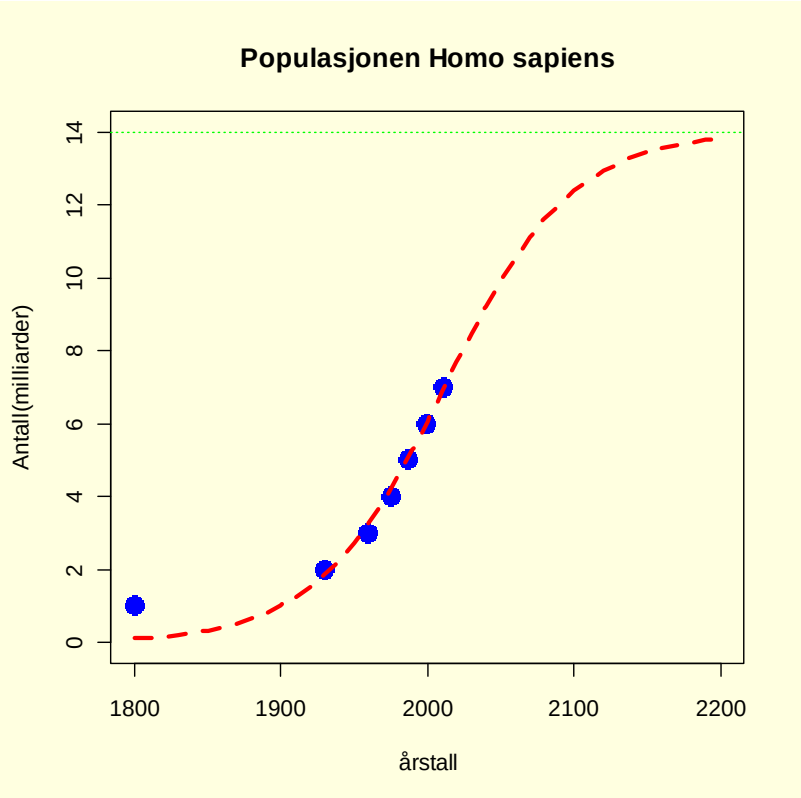
Menneskehetens største utfordring: menneskepopulasjonens vekst. Med begrensede jordbruksarealer vil dette med tiden bety økte matvarepriser, matmangel og økt spredning av sykdommer, samt mulig populasjonskollaps.
Vi plotter de kjente tallene for populasjonen, og modellerer deretter utviklingen:

```
årstall<-c(1800,1930,1960,1975,1987,1999,2011)
antall<-c(1,2,3,4,5,6,7)
par(bg="lightyellow")
plot(årstall,antall,
ylim=c(0,14),xlim=c(1800,2200),pch=16,cex=2,col="blue",ylab="A
ntall(milliarder)",
main="Populasjonen Homo sapiens")
hsap<-cbind(årstall,antall);hsap
```

```
     årstall antall
[1,]    1800      1
[2,]    1930      2
[3,]    1960      3
[4,]    1975      4
[5,]    1987      5
[6,]    1999      6
[7,]    2011      7
```

```
#Verhulst-ligningen med K=14
r<-0.023
```

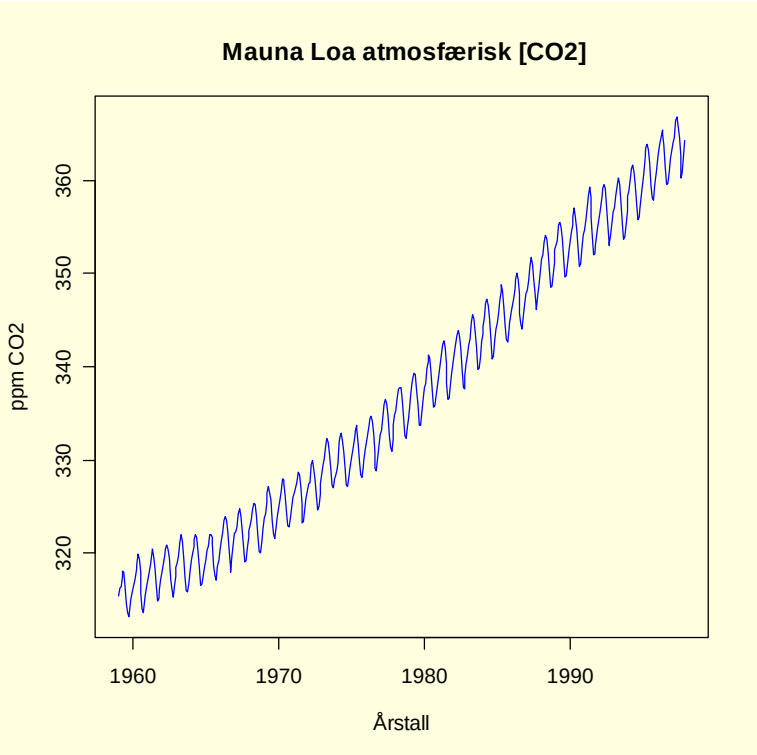
```
x0<-1
K<-14
t0<-1900
t<-seq(1800,2200,10)
lines(t,K/(1+((K-x0)/x0)*exp(-r*(t-
t0))),col="red",lwd=3,lty=2)abline(h=14,lty=3,col="green")
```



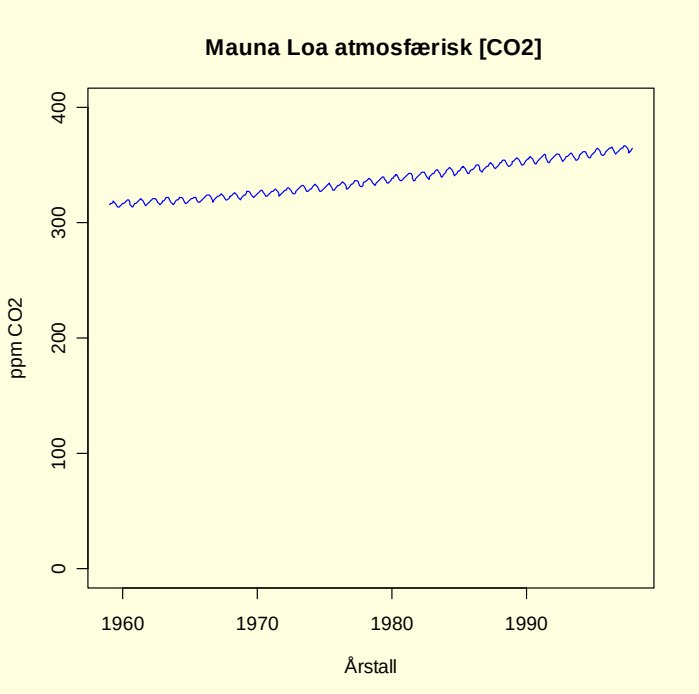
Figur. Utvikling av menneskepopulasjonen med K=14 og veksthastighet r=0.023(2.3%). Brukes Verhulsts (1838) logistiske differensialligning (rød stiplet linje) kan vi forvente en populasjon på ca. 14 milliarder i år 2200, mer enn en dobling i forhold til dagens populasjon.

Vi hører stadig om økt konsentrasjon av CO_2 i atmosfæren og data fra Hawaii blir ofte brukt som eksempel:

```
library(datasets)
data(co2)
CO2<-ts(co2,start=c(1959,1),frequency=12)
par(bg="lightyellow")
ts.plot(CO2,col="blue",lwd=1.5,xlab="Årstall",ylab="ppm
CO2",main=" Mauna Loa atmosfærisk [CO2]")
```



Svingningene i CO_2-konsentrasjon skyldes forskjell i fotosyntese sommer og vinter. Imidlertid hvis vi endrer y-aksen ylim=c(0,400)og plotter de samme dataene ser kurven mindre skremmende ut:



CO_2 er en forutsetning for alt liv på jorda, uten CO_2 ikke noe liv. Store deler av CO_2-mengden i atmosfæren blir hvert år assimilert av terrestrisk og marin vegetasjon, samt planteplankton og går inn i biosfærens kretsløp. Plantebiomassen brukes til oppbygning av andre organismer, og

noe CO_2 går via respirasjon tilbake til atmosfæren. Store deler av vanninnholdet i atmosfæren har kommet dit via vegetasjonen, og dagens oksygenkonsentrasjon (ca. 20%) er også et resultat av plantenes fotosyntese. Biosfæren følger tilsynelatende Le Chateliers prinsipp, noe som gir tilnærmet stabil likevekt (homeostase) over lange tidsperioder. Økt konsentrasjon av CO_2 i atmosfæren gir økt forsuring av vann og hav.

Overlevelse-/feil-analyse

Overlevelsedata angir tiden før en hendelse skjer. Dette kan være tiden før det oppstår feil på komponenter eller utstyr, tiden det tar før en organisme dør, eller tiden det tar før et frø spirer. Hendelsene blir da feil, død eller spiring. Det kan dreie seg om å overleve vinter eller tørketid. Evnen til å overleve avhenger av alder og mengde opplagsnæring. Ved overlevelseanalyse hender det at individer overlever slik at man får ikke noe tidsangivelse for hendelsen død innen undersøkelsen er avsluttet, og man sier at overlevelsestiden er **sensorert** ("censored"). Hadde man ventet lenge nok ville alle ha dødd. Det er flere måter å analysere slike data bl.a. Kaplan-Meier-kurver. Etter analysen får man et mål på tiden til hendelsen skjer, eller om det er en sensorert tid.
I R kan man gjøre overlevelseanalyse med programpakken *survival* som ble laget av Terry Therneau for S-PLUS og omformet til R av Thomas Lumley. Error-fordelingen for data som angir tiden for en hendelse er ikke normalfordelt, og ofte kjenner man ikke til error-fordelingen, men den kan være av typen Weibull, Gamma, eksponensiell eller log normal eller lignende.
I overlevelseskurver plotter man den naturlige logaritmen til andelen av en kohort med individer fra starttid 0 og som ennå lever ved tiden t. Dødsraten er vanligvis ikke lineær med alderen, oftest er dødsraten størst blant de yngste og de eldste. Eksempler på slike aldersspesifikke hasardmodeller er Rayleigh, Makeham, Gomperts, Weibull og eksponensiell.
Overlevelsesfunksjonen $S(t)$ viser andelen av individer fra den opprinnelige kohorten som ennå lever ved tiden t. Alle i kohorten lever ved tid=0 slik at overlevelsesfunksjonen har en skjæring ved 1. **Tetthetsfunksjonen $f(t)$** angir sannsynligheten for å dø i tidsintervallet t og $t+dt$. **Hasardfunksjonen $h(t)$** er forholdet mellom tetthetsfunksjonen og overlevelsesfunksjonen.

$$f(t) = \frac{e^{\frac{-t}{\mu}}}{\mu}$$

$$S(t) = e^{\frac{-t}{\mu}}$$

$$h(t) = \frac{f(t)}{S(t)} = \frac{1}{\mu}$$

Kaplan-Meier overlevelsefordeling er en trappetrinnskurve som angir når død skjer og hentes med kommandoen **survfit.** Cox proporsjonal hasardmodell er den mest brukte regresjonsmodellen for overlevelsedata, brukt på samme måte som lineære modeller.

Opplasting av programpakken:

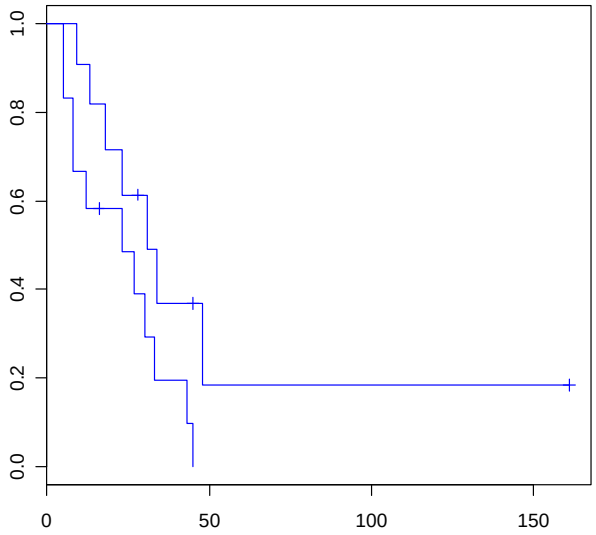
```
 library(survival)
```

Vi bruker datasettet *aml* som ligger i R som viser overlevelse av pasienter med akutt myelogen leukemi med og uten behandling.

```
data(aml)
aml
 ?aml
```

Tilpass Kaplan-Meier og plot. Merkene på kurvene viser sensoring:

```
  tilpass <- survfit(Surv(time, status) ~ x, data=aml)
 plot(tilpass,col="blue")
```




```
 summary(tilpass)
```

```
Call: survfit(formula = Surv(time, status) ~ x, data = aml)

                x=Maintained
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    9     11       1    0.909  0.0867       0.7541        1.000
   13     10       1    0.818  0.1163       0.6192        1.000
   18      8       1    0.716  0.1397       0.4884        1.000
   23      7       1    0.614  0.1526       0.3769        0.999
   31      5       1    0.491  0.1642       0.2549        0.946
   34      4       1    0.368  0.1627       0.1549        0.875
```

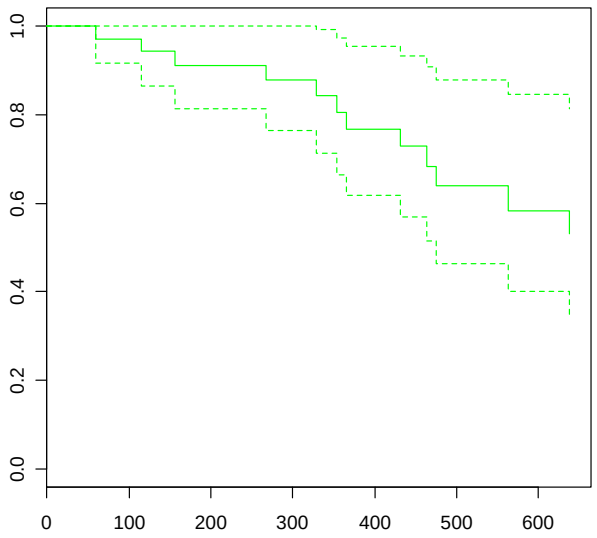
```
   48      2       1    0.184  0.1535       0.0359        0.944

               x=Nonmaintained
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    5     12       2   0.8333  0.1076       0.6470        1.000
    8     10       2   0.6667  0.1361       0.4468        0.995
   12      8       1   0.5833  0.1423       0.3616        0.941
   23      6       1   0.4861  0.1481       0.2675        0.883
   27      5       1   0.3889  0.1470       0.1854        0.816
   30      4       1   0.2917  0.1387       0.1148        0.741
   33      3       1   0.1944  0.1219       0.0569        0.664
   43      2       1   0.0972  0.0919       0.0153        0.620
   45      1       1   0.0000      NA           NA           NA
```

Datasettet ovarian for overlevelse av eggstokk-kreft

```
data(ovarian)
ovarian
?ovarian
```

Tilpass Cox proporsjonal hasardmodell og plot prediktert overlevelse:

```
tilpass2 <- coxph(
Surv(futime,fustat)~resid.ds+rx+ecog.ps,data=ovarian)
plot(survfit(tilpass2),col="green")
```



Numerisk løsning av differensialligninger

Modeller

En dynamisk modell er en forenklet framstilling av den virkelige verden i form av ligninger. Den kalles dynamisk fordi modellen viser hvordan systemet oppfører seg over tid. Man kan lage en matematisk modell av et økosystem, en

populasjon eller av været brukt i meteorologenes klimamodeller. I arbeidet med modellen samles og systematiseres kunnskap. Modellen brukes til å forstå systemet, forutsi hendelser og trekke konklusjoner fra modellen, og se om det modellen predikterer stemmer med virkeligheten. Modellen har et sett med tilstandsvariable som oppsummerer egenskapene til systemet. Dynamiske ligninger viser hvordan tilstandsvariable endrer seg over tid. Måleenhetene på venstre side av likhetstegnet i differensialligningene må være lik måleenhetene på høyre side.

På siste del av 1600-tallet studerte Newton av bevegelsen av sola og jorda, et tolegemesystem i klassisk mekanikk karakterisert av bevegelse, posisjon og hastighet som funksjon av tiden, og som kunne beskrives med differensialligninger. Det viste seg at trelegemesystemet med jord, måne og sol var vanskelig å løse. Kong Oskar II utlovet en pris på 2500 kroner til den som kunne løse *n*-legemeproblemet. Den franske matematikeren Jules Henri Poincaré (1854-1912) fikk prisen i 1889, men han hadde egentlig ikke løst problemet. I 1890 publiserte imidlertid Poincaré den riktige løsningen: *Sur le probléme des trois corps et et les equations de la dynamique* i Acta Matematica 13 (1890)1-270. Poincaré valgte en geometrisk tilnærming for løsning av differensialligninger, en geometrisk analyse av stabilitet. Løsningene av differensialligningene kan presenteres geometrisk som **trajektorier** i et **faserom**. **Bifurkasjoner** er endringer i et dynamisk system når en eller flere parametere passerer kritiske verdier. Noen systemer viser hysterse, hvor bifurkasjoner får forskjellige verdier avhengig av retningen på endringen av parameterverdiene. Poincaré betraktes som grunnlegger av teorien om bifurkasjoner. I 1963 kunne Lorenz vise at tre differensialligninger som beskrev værfenomener oscillerte irregulært og nådde ikke likevekt, Lorenz sommfugleffekt: Et vingeslag i Brasil kan gi en tornado i Texas. På 1970-tallet kunne biologen Robert May vise at det oppstod kaos i den logistiske vekstligningen ved høye veksthastigheter. Kaos (gr.) betyr fravær av orden, og kaos er aperiodisk oppførsel følsom for initialbetingelsene. Meget små forandringer i utgangsbetingelsene gir meget forskjellig resultat, selv i samme ligning. Væske kan strømme laminært eller turbulent, og turbulent strøm gir kaos. Mens planetene går i stabile elipsebaner rundt sola, går den lille månen Hyperion i en kaotisk bane rundt Saturn. Belousov-Zhabotinsky ringer er eksempler på kaotiske kjemiske oscillasjoner (spiralbølger) ved oksidasjon av malonsyre i en sur løsning med bromationer med små mengder metallioner, cesium eller jern som katalysator. Kaotisk leketøy er satt sammen av to pendler som svinger. Mandelbrot var en av pionerene innen fraktalgeometri, et resultat av interasjonsprosesser i enkle ligninger. Feigenbaum studerte av faseoverganger og kaos. Datamaskiner gjorde det mulig å visualisere deterministisk

kaos fra ikke-lineære ligninger og synliggjort strukturer og mønstre som gjentar seg.

Det er svært vanskelig å forutsi utviklingen av komplekse systemer. På noen stadier kan de begynne å oppføre seg kaotiske og uforutsigbare, men det som skjer er allikevel ikke tilfeldig. På visse steder svinger systemet mellom to rytmer, og det skjer en todeling (**bifurkasjon)**. Deretter kan det skje helt uregelmessige og uforutsigbare reaksjoner uten noen fast rytme dvs. systemet oppfører seg **kaotisk**. For økologene har kaosmodellene og fraktale mønstere vært med på å forklare utviklingen av populasjoner og hvordan formen på en plante oppstår. Ofte kan populasjoner svinge uten å nå en likevektssituasjon og variasjonen synes å være usystematisk. Det fantes tidligere ingen fullgod matematisk beskrivelse av denne situasjonen. Det er så mange faktorer ute i naturen som påvirker et system at det er vanskelig å finne noe mønster i virvaret, men det er en struktur. Det kan plutselig finnes "vinduer" i kaoset hvor populasjonen faller til ro omkring en likevektsverdi. Seinere går det på nytt inn i kaos. Forgreiningene i fikentreet oppstår med stadig kortere mellomrom, og intervallene mellom periodefordoblingen minsker med faktoren 4.669201.., kalt Feigenbaums tall. Det er også en gjentakelse av mønsteret for periodefordoblinger og overgang til kaos. Man sier at systemet viser **selvsimilaritet**. Det er i den kaotiske delen mønsterstrukturer gjentar seg i miniatyr om og om igjen. Benoit Mandelbrot har vist dette via Mandelbrot-mengden: Han gjentok en enkel beregning en rekke ganger og resultatet av denne danner utgangspunktet for den neste (iterasjoner). Det er altså en uendelighet i dette mønsteret og man kan zoome inn på stadig mindre deler av helheten. Lengden av kysten er avhengig av målestokken man bruker. En mikrostruktur går igjen. Helhetens form er identisk med delenes form. Hvis en rett linje har dimensjon en, flaten har dimensjon to og rommet dimensjon tre, vil krøllete linjer har en dimensjon som ligger mellom disse heltallene, kalt fraktaler. Kaosforskningen er istand til å gi mønstre og lovmessighet bak dynamiske systemers utvikling. Mønstre oppstår når delene i et system plasserer seg energimessig mest gunstig i forhold til hverandre for eksempel vannmolekyler i en snøkrystall.

Differensialligninger har en kontinuerlig tidsakse, mens differensligninger har diskrete tidsintervall.

I det følgende skal vi først se på systemer med en variabel: et lineært system i form av eksponensiell vekst eller nedbrytning, og det ikke-lineære systemet med logistisk vekst som under visse betingelser viser bifurkasjoner og kaos. Newtons system med sol og måne var et lineært system med to variable, men vi skal se på det ikke-lineære systemet Lotka-Volterra-modellen for predator-byttedyr. Det lineære Poincarés trelegemeproblemet har tre variable, og vi skal studere de

ikke-lineær Lorenzligningene, og samt se litt på fraktaler, diffusjon, rute-systemer og cellulære automater *e.g.* Conways game of life. Schrödingers bølgligninger innen kvantemekanikk, Maxwells ligninger for elektromagnetisme, Ficks diffusjonsligninger, Fouriers varmeligninger, Laplaces ligning for harmoniske funksjoner,Navier-Stokes ligningene for bevegelse i viskøse ikke-sammenpressbare væsker, samt Black-Scholes ligninger for økonomi er alle eksempler på differensialligninger.

Biologiske systemer er selvorganiserende dissipative energiforbrukende systemer langt fra likevekt. Det er **flukser** av stoff og energi gjennom de biologiske systemene, beskrevet av **irreversibel termodynamikk** av bl.a. Prigogine og norskamerikaneren og nobelprisvinneren Lars Onsager.

Naturforskeren studerer ikke naturen fordi det er nyttig. Han/hun studerer den fordi han fryder seg, og han fryder seg fordi det er vakkert. Hvis naturen ikke var vakker, så ville det ikke være verdt å vite, og hvis naturen ikke var verd å vite, så ville ikke livet være verd å leve. J.H.Poincaré.

Differensialligninger

Gottfired Leibniz (1646-1716), **Isaac Newton** (1642-1727), **Leonhard Euler** (1707-1782) og **Johann Bernoulli** (1667-1748) var de første som på begynnelsen av 1700-tallet løste enkle differensialligninger. **J.L Lagrange** (1736-1813) og **Pierre Simon Marquis de Laplace** (1749-1827) videreutviklet på 1800-tallet løsningsmetodene for differensialligningene. Laplace er også kjent for studiene av himmellegemenes bevegelser, og Laplace differensialligning beskriver en harmonisk funksjon. Etter hvert ble man klar over at mange differensialligninger ikke kan løses med tradisjonelle metoder, og den franske matematikeren **Augustin-Louis Cauchy** (1789-1857) kunne i 1820 forutsi at under gitte forutsetninger hadde en differensialligning løsninger. Det viser seg at ikke-lineære systemer ikke kan løses analyttisk. Lineære systemer har alle x på høyre side av ligningen og i første potens.
Lagrange var den første som innførte betegnelsen f' som **den deriverte** av funksjonen f.
Leibniz introduserte en **differenskvotient**:

$$\frac{\Delta y}{\Delta x}$$

Hvor Δ er den greske bokstaven Delta som ble brukt til å uttrykke bittesmå forskjeller. Leibniz tenkte at når Δx ble uendelig liten og Δy tilsvarende liten så ble Δx og Δy uendelige små, **infinitesimale**, og disse infinitesimale betegnet han dx og dy. Mengdene dx og dy kalte han **differensialer**. Derved ble den deriverte en **differensialkvotsient** dy/dx.

$$f'(x) = \frac{dy}{dx} = \lim \frac{\Delta y}{\Delta x} \quad når\ \Delta x \to 0$$

Grenseverdien betegnes *lim* (limes).
Leibniz hadde derved skapt differensialkvotienten *dy/dx* som symbol for den deriverte. Leibniz mente at infinitesimale var uendelige små tall, men allikevel større enn 0. Etter hvert oppga man tanken om infinitesimale som egne uendelige små tall, men fremdeles kalles derivasjon og integrasjon for infinitesimalregning. Imidlertid førte Leibniz tenkemåte til en geometrisk tolkning: den deriverte er en **tangentlinje** til en kurve. Differenskvotienten blir en tangentlinje, og den deriverte sier noe om hvor bratt kurven stiger og i hvilken retning den går. Differensialligningen omhandler tangentlinjer, mens integralregningen tar for seg arealer under kurver. Differensialligninger har kontinuerlig tidsakse, differensligninger opererer med diskrete tidsrom. Den franske matematikeren **Pierre de Fermat** (1601-1665) fant at når tangentlinjene var horisontale dvs. ingen stigning på tangenten så hadde han maksimums- eller minimumspunkter på en grafisk figur. Har den deriverte en positiv verdi stiger kurven, har den deriverte en negativ verdi synker kurven, og er den deriverte lik 0 verken stiger eller synker kurven. Når den deriverte er lik 0 har man nådd et maksimums- eller minimumspunkt på funksjonen, en form for likevektspunkt

$$\frac{dx}{dt} = 0$$

Det er mulig å studere hvor stabil løsningen er rundt likevektspunktet
I funksjonen som danner en sirkel vil tangentlinjen alltid stå normalt (vinkelrett) på radius i ethvert punkt på sirkelen. Disse tangentlinjene kunne brukes til å beregne endringer i hastighet i fysikken, eller generelt hvor rask hastighetsendring en funksjon har. Betegnelsen *f'* brukes på den **førstederiverte** og *f''* på den **andrederiverte**. Ofte skriver man *y'* i stedet for *f'(x)*.

I en todimensjonal figur har man et aksesystem med en horisontal linje kalt **x-akse** og en vertikal akse kalt **y-akse**. Der disse to aksene skjærer hverandre har man **origo** (0). Ethvert punkt i dette planet (**xy-planet**) kan angis med et par med tall (x,y) kalt **koordinater**, hvor x-koordinaten ofte kalles **abscisse** og y-koordinaten kalles **ordinat**.
Leibniz introduserte ordet **funksjon** om en type matematiske formler. Disse funksjonene kan tegnes som en **grafisk framstilling** i et plan med x- og y-koordinater. Sinus (*sin*), cosinus (*cos*) og tangens (*tan*) er eksempler på **trigonometriske funksjoner** som spesielle periodiske egenskaper. Studerer man

tidsavhengige prosesser er x-aksen en **tidsakse** med tidsenheter t og tidsfunksjonen blir av typen $f(t)$. Vi skal etter hvert se at prosesser som endrer seg over tid kan beskrives med differensialligninger.

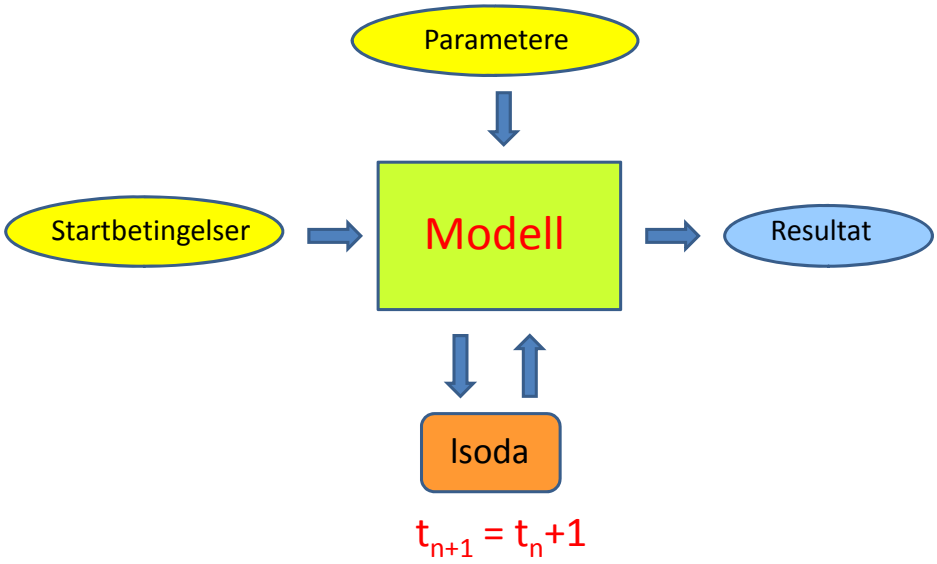
En differensialligning viser i hvilken retning en tangent til en funksjon peker til enhver tid. Moderne datamaskiner har gjort det mulig å løse differensialligninger numerisk hvis man har gitt utgangsparameterne for ligningen.

Differensialligninger er av to hovedtyper:
Ordinære differensialligninger hvor den ukjente er en funksjon av en variabel og **partielle differensialligninger** som har to eller flere variable.
I en differensialligning inngår funksjonen og den deriverte funksjonen. Det finnes **første ordens differensialligninger** og **høyere ordens differensialligninger**.

Generelt har en differensialligning formen:

$$y' = f(x, y)$$

Retningsfeltet til differensialligning er korte linjestykker som tangenter til en integralkurve. Trajektoriene (integralkurvene) er løsningene til differensialligningene. Gjennom et punkt (x,y) i planet går det bare ett trajektorium (integralkurve).



Figur. Gitt et modellsystem beskrevet av differensialligninger. Ved gitte startbetingelser, parameterverdier og et bestemt antall tidstrinn *t* så vil diffrenensialligningene kunne løses numerisk via deSolveren lsoda.

odesolver i R

For løsing av første ordens ordinære differensialligninger (ODE) har R i **library(odesolve)** en funksjon *lsoda* som danner interface til en Fortran ODE solver skrevet av Petzold og Hindmarsh. Vi skal løse differensialligningene numerisk når vi har gitt utgangsparamtere. For å kunne løse differensialligninger må man laste ned programpakke odesolve.

library(odesolve)

Denne inneholder løsningsmetoden **rk4** som bruker 4.ordens Runge Kutta metoden, men en kraftigere løsningsmetode er **lsoda.** Funksjonen må inneholde parameterene **function(t,x,parms)**, og x må inngå selv om den ikke brukes for å beregne den deriverte. Objektet **parms** er en vektor som inneholder alle parameterverdiene, og gjør det lett å endre disse og se hvordan dette påvirker løsningen av ligningene. De deriverte plasseres i **list.**

lsoda(y, times, func, parms, rtol=1e-06, atol=1e-06,tcrit=NULL, jacfunc=NULL, verbose=FALSE, dllname=NULL,hmin=0,hmax=Inf)

rtol (relativ feiltoleranse) og **atol** (absolutt feiltoleranse)angir nøyaktigheten som solver forsøker å oppnå. Små verdier gir mer nøyaktig løsning, men det tar lenger tid. **hmin** og **hmax** styrer bredden av tidstrinn. Ved å sette inn en verdi for hmax overstyrer man lsoda fra å ta for store tidstrinn.

Generelt er det vanskelig, for i noen tilfeller umulig, å finne analytiske løsninger av differensialligninger, men vi kan finne en tilnærmet numerisk løsning gitt bestemte utgangsbetingelser. Eulers metode, lsoda eller fjerde ordens Runge Kutta metode kan benyttes til numerisk løsning av differensialligninger.
Den deriverte av x ved tid t har grenseverdi:

$$\frac{dx}{dt} = \lim_{h \to 0} \frac{x(t+h) - x(t)}{h}$$

Hvis vi benytter numerisk løsning på den enkle ligningen:

$$\frac{dx}{dt} = rx$$

og sammenligner med den kjente analytiske løsningen:

$$x = x_0 e^{rt}$$

Eksponensialfunksjon og eksponensiell vekst

Et eksempel på en differensialligning hvor den deriverte av x ($f'(x)$) er lik funksjonen selv ($f(x)$)er:

$$f'(x) = f(x)$$

Denne differensialligningen har en løsning hvis $f(x)=e^x$, hvor C er en **integrasjonskonstant**:

$$f(x) = C \cdot e^x$$

Eksponensialfunksjonen e^x er et eksempel på en funksjon som er lik sin deriverte. For forskjellige verdier av C kan man tegne integralkurver for differensialligningen. Det betyr at eksponensiell vekst for en populasjon med organismer kan beskrives som en første ordens differensialligning.

Vekstraten for en populasjon kan beskrives som endring i antall individer N over tiden t (eller endring i biomasse over tiden t) dN/dt. $N(t)$ er antall individer (eller biomasse) ved tid t og $N(0)$ er antall individer (eller biomasse) ved t=0.
Den **relative vekstraten** r blir:

$$\frac{1}{N} \cdot \frac{dN}{dt} = r$$

Denne kalles også per capita vekstrate. Denne er den samme som:

$$\frac{dN}{dt} = r \cdot N$$

Denne kalles også Malthusligningen (Malthus 1798). Dette er en lineær (proporsjonal med N), ordinær (ingen partiellderiverte) differensialligning (deriverte eller differensial). En løsning av denne differensialligningen blir lik en eksponensialfunksjon som beskriver eksponensiell vekst:

$$N = N_0 \cdot e^{r \cdot t}$$

Løsningen går mot uendelig når $r>0$ (eksponensiell vekst, og går mot 0 når $r<0$ (eksponensiell nedbrytning).
Man må skille mellom **relativ vekstrate** r (antall antall^{-1} dag^{-1} eller g g^{-1} dag^{-1})som er et mål på hvor effektivt hvert individ (eller enhet biomasse) produserer nye individer (eller biomasse) og **absolutt vekstrate** (antall dag^{-1} eller g dag^{-1}) som angir økning i antall (eller biomasse) per dag. For mange organismer er den relative vekstraten høyest når N er rundt halvmetning. Det blir vanskelig å finne parringspartner når N er liten, og når N blir stor gir det konkurranse om mat og andre ressurser
I virkeligheten kan ingen populasjon vokse eksponensielt i det uendelige. Ligningen over kan også brukes til å beskrive desintegrasjonen av en radioaktiv isotop, men hvor fortegnet på høyre side av ligningen blir negativ.

Eksponensialfunksjonen e^x er en funksjon som er lik sin deriverte. For forskjellige verdier av C kan man tegne integralkurver for differensialligningen.

Vi benytter numerisk løsning på den enkle første ordens differensialligningen:

$$\frac{dx}{dt} = rx$$

Denne er enkel å løse ved å flytte x over på venstre side og dt over til høyre og ta integralet på hver side av likhetstegnet:

$$\int \frac{dx}{x} = r \cdot \int dt$$

Som er lik:

$$lnx = rt + C$$

Ved tiden $t=0$ har vi, hvor C er en integrasjonskonstant:

$$lnx_0 = C$$

$$lnx = rt + lnx_0$$

$$ln\frac{x}{x_0} = rt$$

og vi finner da den analytiske løsningen:

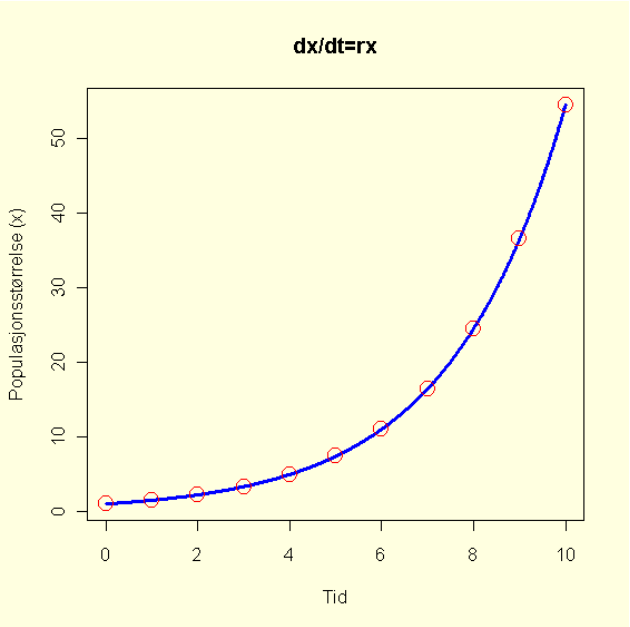
$$x(t) = x_0 e^{rt}$$

Dette er det samme som den lineære funksjonen hvor veksthastigheten r er stigningskoeffisienten:

$$\ln\big(x(t)\big) = \ln x_0 + r \cdot t$$

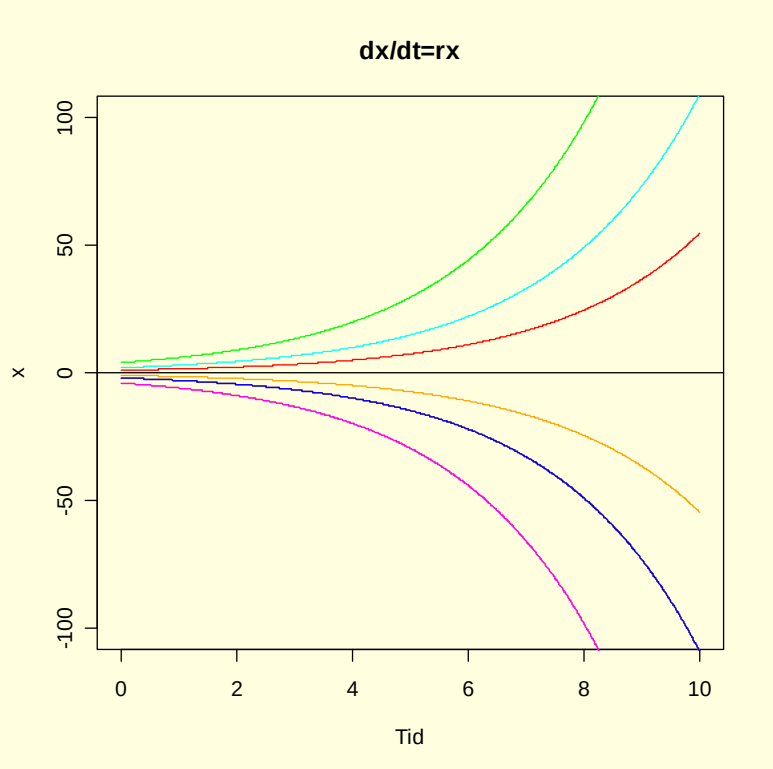
Først løser vi differensialligningen numerisk med lsoda(blå) og deretter setter vi inn verdier for en analytisk løsning og finner at det god overensstemmelse mellom de to løsningsmetodene. Legg merke til at måleenhetene på hver side av likhetstegnet er like, antall per tidsenhet = 1/tidsenhet ·antall.

```
#Laster inn pakken deSolve som bl.a. inneholder lsoda
library(deSolve)
#Setter inn parameterverdien r for veksthastighet. Kan endres.
params<-c(r=0.4)
#Definerer en funksjon som inneholder tid t, x og parameter p
eksp<-function(t,x,p)
{
#Definerer differensialligningen, legg merke til parameter
dxdt<-p["r"]*x[1]
#Lager en liste som inneholder dx/dt
list(dxdt)
}
require(deSolve)
#Definerer tiden
tid<-seq(0,10,0.01)
#Definerer startverdi
start<-1
#Overfører verdier til lsoda og lagrer resultat i dataramme
resultat<-as.data.frame(lsoda(start,tid,eksp,params,rtol=1e-
4))
#Plotter grafisk den numeriske løsningen av ligningen
plot(resultat[,1],resultat[,2],type="n",col="blue",xlab="Tid",
ylab="Populasjonsstørrelse (x)",main="dx/dt=rx")
lines(resultat[,1],resultat[,2],col="blue",lwd=3)
#Sammenligner den numeriske løsningen med den analytiske
r<-0.4
t<-seq(0,10,1)
points(t,exp(r*t),col="red",cex=2)
```

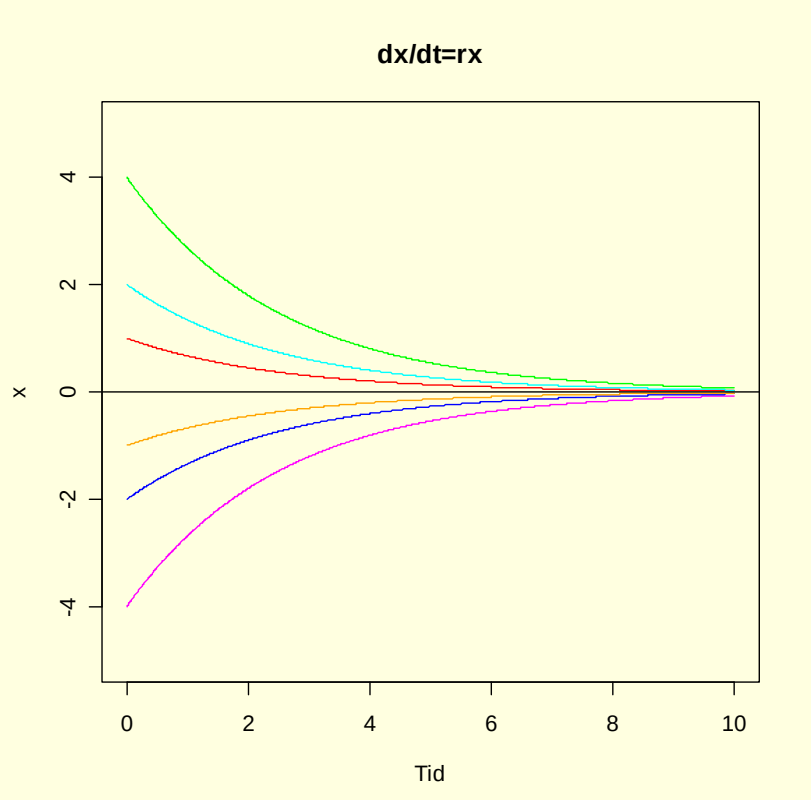


Figur. Viser numerisk løsning av *dx/dt=rx* med Fortran subrutingen lsoda (blå linje) sammenlignet med analyttisk løsning (røde punkter) for r=0.4. Initialverdi x_0=1

```
params<-c(r=0.4)
eksp2<-function(t,x,p)
{
with(as.list(p), {
dxdt<-r*x[1]
list(dxdt)
})
}
tid2<-seq(0,10,0.01)
start2<-1
resultat2<-
as.data.frame(lsoda(start2,tid2,eksp2,params,rtol=1e-4))
matplot(resultat2[,1],resultat2[,2],type="l",lwd=3,col=2,xlab=
"Tid",ylab="x",main="dx/dt=rx")
```



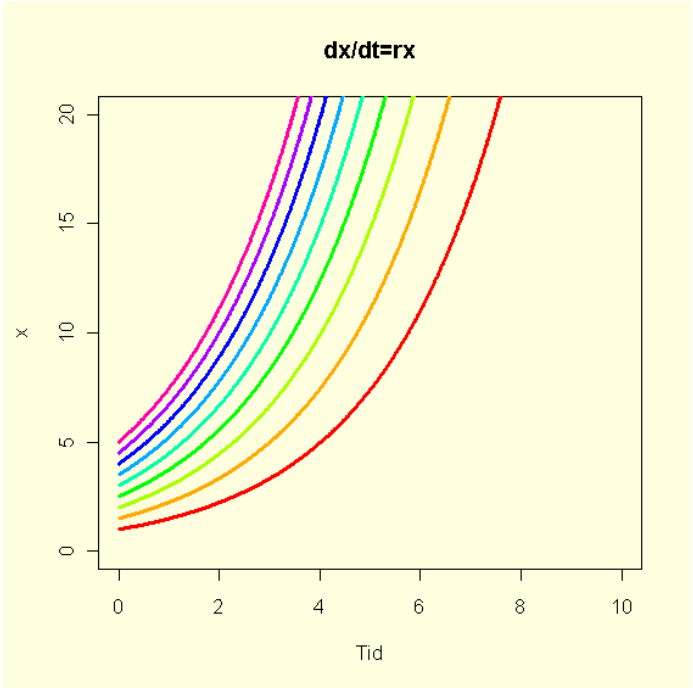
Figur. Trajektorier (integralkurver) for *dx/dt=rx* for *r*=0.4 og forskjellige startverdier: -4 (magenta),-2 (blå),-1 (oransje), 1(rød), 2(cyan), og 3 (grønn)



Figur. Tilsvarende trajektorier (integralkurver) for *dx/dt=rx* men hvor r<0, for *r*=-0.4 og forskjellige startverdier: -4 (magenta),-2 (blå),-1 (oransje), 1(rød), 2(cyan), og 3 (grønn)

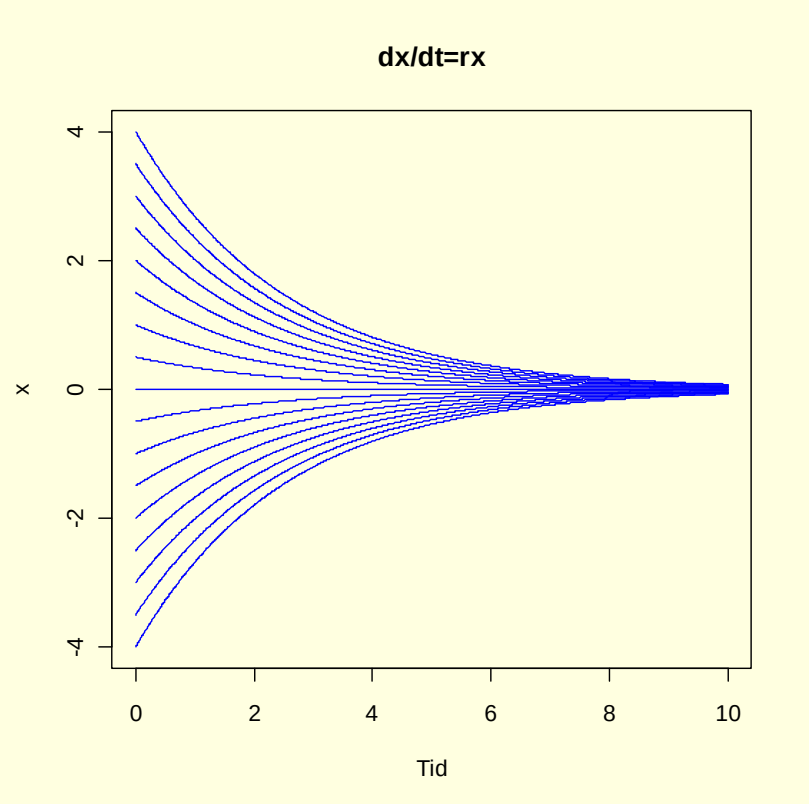
Eller man kan automatisk sette flere initialverdier, her seq (-4,4,0.5), samt out.state<-as.data.frame definert som objekt fra tidligere kjøring foran

```
#Forskjellige initialverdier
params<-c(r=0.4)
eksp3<-function(t,x,p)
{
with(as.list(p), {
dxdt<-r*x[1]
list(dxdt)
})
}
tid3<-seq(0,10,0.01)
#lager et tomt plot
y<-seq(0,10,0.01)
par(bg="lightyellow")
plot(tid3,y,ylim=c(0,20),type="n",col=4,xlab="Tid",ylab="x",ma
in="dx/dt=rx")
start3<-1
#Lar initialverdiene s variere fra 1 til 5, kan endres
s<-seq(1,5,0.5)
#antall initialverdier
n<-length(s)
for (i in 1:n) {
start3<-s[i]
resultat3<-
as.data.frame(lsoda(start3,tid3,eksp,params,rtol=1e-4))
lines(resultat3[,1],resultat3[,2],lwd=3,col=rainbow(n)[i])
}
```



Figur. r=0.4 og forskjellige initialverdier
Eller forskjellige initialverdier -4 til 4, samt out.state<-as.data.frame definert som objekt fra tidligere kjøring foran

```
#Halvor Aarnes
library(deSolve)
params<-c(r=-0.4)
eksp<-function(t,x,p)
{
dxdt<-p["r"]*x[1]
list(dxdt)
}
require(deSolve)
out.time<-seq(0,10,0.01)
s<-seq(-4,4,0.5)
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],ylim=c(-
4,4),type="n",col="blue",xlab="Tid",ylab="x",main="dx/dt=rx")
for (i in 1:17) {
init.state<-s[i]
out.state<-
as.data.frame(lsoda(init.state,out.time,eksp,params,rtol=1e-
4))
lines(out.state[,1],out.state[,2],col="blue")
}
```



Logistisk ligning og logistisk vekst

Ingen populasjon kan fortsette med eksponensiell vekst til evig tid. Populasjoner kan imidlertid følge en **logistisk vekstligning** brukt av P. F. Verhulst i 1838 for å beskrive vekst av menneskepopulasjonen

$$\frac{dN}{dt} = r \cdot N \cdot \left(1 - \frac{N}{K}\right)$$

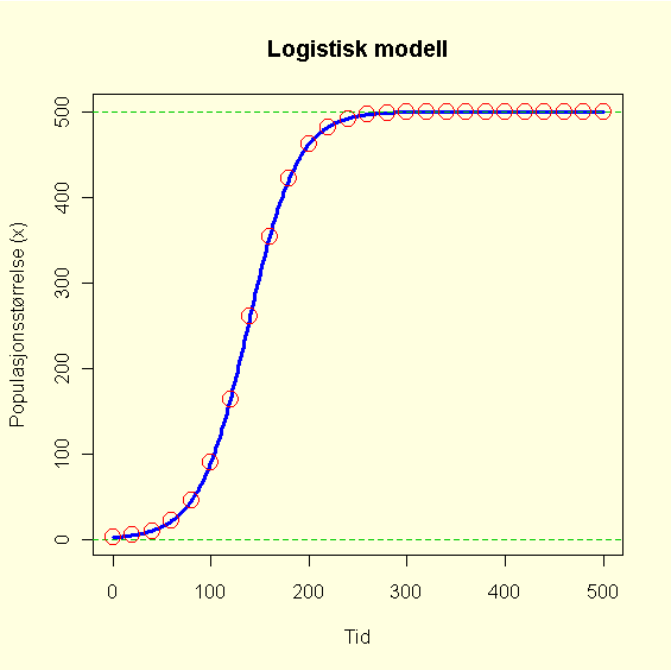
hvor populasjonen vil nærme seg bærekapasiteten K, $N(t) \rightarrow K$ når $t \rightarrow \infty$, unntatt når $N_0=0$. Faste punkter er $N=0$ og $N=K$. Den logistiske ligningen er ganske god til å beskrive veksten av enkle organismer som bakterier, gjær og encellete alger, men er lite egnet hvis organismen har flere livsstadier *e.g.* insekter.

$$\frac{dx}{dt} = r \cdot x \cdot \left(1 - \frac{x}{K}\right)$$

Hvor r er veksthastighet og K er bærekapasitet. Det er vanligvis meget vanskelig å finne analyttiske løsninger av en ikke-lineær differensialligning, men i dette tilfelle kan vi finne den analyttiske løsningen:

$$x(t) = \frac{K}{\left[1 + \left(\frac{K - x_0}{x_0}\right) \cdot e^{-rt}\right]}$$

```
#Logistisk modell
#Logistisk vekstkurve
params<-c(r=0.04,K=500)
logistisk<-function(t,x,p)
{
with(as.list(p), {
dxdt<-r*x*(1-(x/K))
list(dxdt)
})
}
require(deSolve)
time<-0:500
init<-2 #startverdi
results<-
as.data.frame(lsoda(init,time,logistisk,params,rtol=1e-4))
plot(results[,1],results[,2],type="n",xlab="Tid",
ylab="Populasjonsstørrelse (x)",
main="Logistisk modell")
lines(results[,1],results[,2],col=4,lwd=3)
abline(h=c(0,500),col=3,lty=2)
#Sammenligner med den analytiske løsningen
r<-0.04
x0<-2
K<-500
t<-seq(0,500,20)
points(t,K/(1+((K-x0)/x0)*exp(-r*t)),col="red",cex=2)
```



Figur Viser numerisk løsning av en logistisk vekstkurve for r=0.04,K=500, initialverdi x_0=2 (blå). Kurven vil assymptotisk nærme seg bærekapasiteten K=500. Analytisk løsning som røde punkter, og vi ser at det er god overensstemmelse mellom numerisk og analytisk løsning av differensialligningen.

Ved grafisk framstilling av løsningene av en differensialligning så vil løsningene (punktene) bevege seg i et abstrakt faserom og følge **trajektorier**. Disse trajektoriene vil etter hvert nærme seg grenser eller assymptoter.

Før vi går videre skal vi se på en enklere utgave av den **logistiske differensialligningen:**

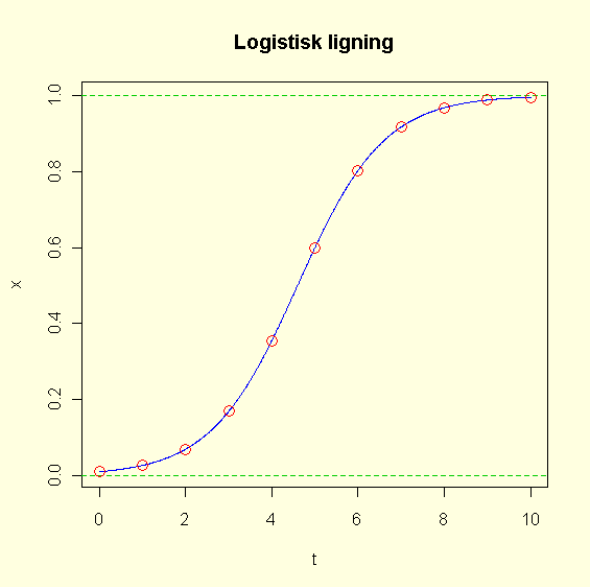
$$\frac{dx}{dt} = r * x(1-x) = r * x - r * x^2$$

Vi setter initialverdi x_0=0.01 og r=1.
Den analyttiske løsningen er:

$$x(t) = \frac{1}{1 + \left(\frac{1}{x_0} - 1\right) * e^{-r*t}}$$

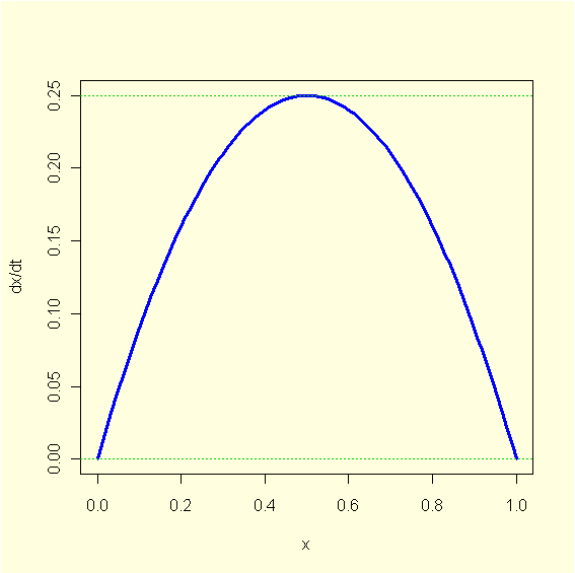
```
params<-c(r=1)
logistisk2<-function(t,x,p)
{
with(as.list(p), {
dxdt<-r*x*(1-x)

list(dxdt)
})
}
require(deSolve)
time2<-seq(0,10,0.01)
init2<-0.01
```

```
results2<-
as.data.frame(lsoda(init2,time2,logistisk2,params,rtol=1e-4))
par(bg="lightyellow")
plot(results2[,1],results2[,2],type="n",col=4,xlab="t",ylab="x
",main="Logistisk ligning")
lines(results2[,1],results2[,2],col=4)
abline(h=c(0,1),lty=2,col=3)
#Se at numerisk og analytisk løsning stemmer overens
r<-1
x0<-0.01
t<-seq(0,10,1)
points(t,1/(1+(1/x0-1)*exp(-r*t)),col=2,cex=1.5)
```

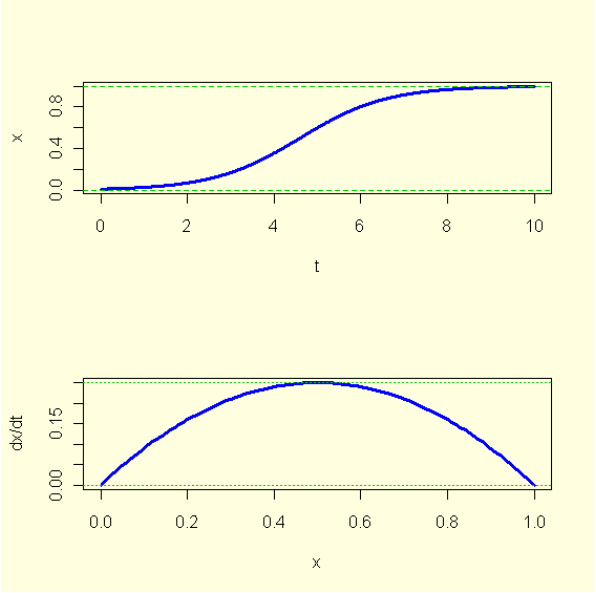


Figur. Løsning av den logistiske differensialligningen for x_0=0.01 og r=1.Den analyttiske løsningen er markert med rød punkter.

```
#Vi kan plotte den deriverte dx/dt mot x
x<-seq(0,1,0.01)
y<-r*x*(1-x)
par(bg="lightyellow")
plot (x,y,col=4,type="l",lwd=3,xlab="x",ylab="dx/dt")
abline(h=c(0,0.25),lty=3,col=3)
#For å se hva som skjer med den deriverte se på begge figurene
par(mfrow=c(2,1))
plot(results2[,1],results2[,2],type="n",col=4,xlab="t",ylab="x
")
lines(results2[,1],results2[,2],lwd=3,col=4)
abline(h=c(0,1),lty=2,col=3)
plot (x,y,col=4,type="l",lwd=3,xlab="x",ylab="dx/dt")
abline(h=c(0,0.25),lty=3,col=3)
```

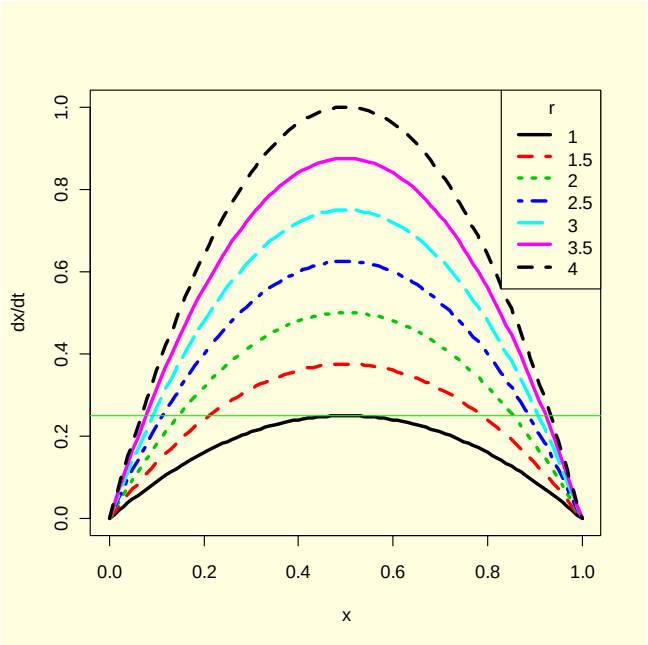


Figur. Logistisk differensialligning for x_0=0.01 og r=1. Den deriverte dx/dt=0 ved x=0 og x=1. Den deriverte har et vendepunkt ved x=0.5 og dx/dt har maksimalpunkt ved r/4. For mange organismer er den effektive vekstraten høyest når N er intermediær.

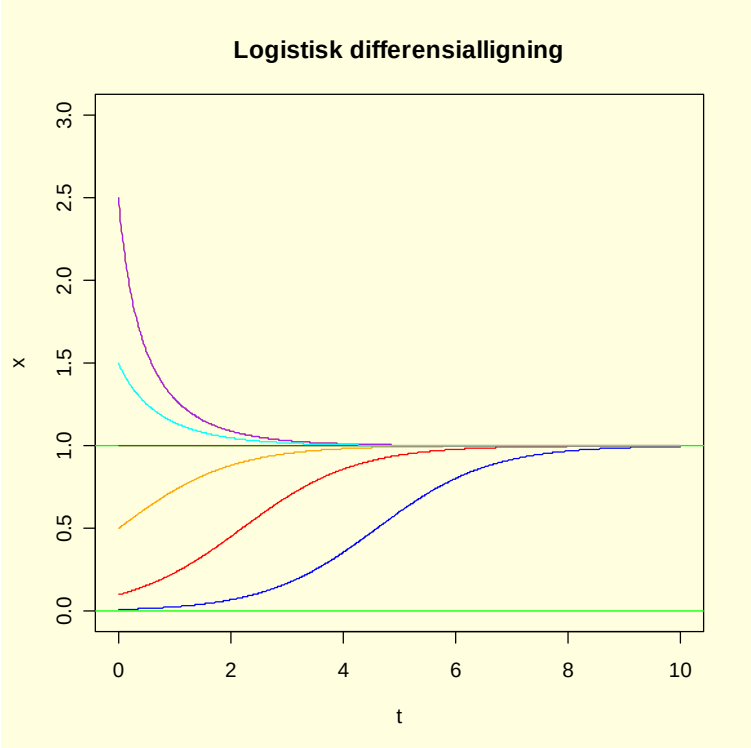


Eller for flere verdier av r:

```
x<-seq(0,1,0.01)
r<-seq(1,4,0.5)
logistisk<-outer(x,r, function(x,r)-r*x^2+r*x)
par(bg="lightyellow")
matplot(x,logistisk,type="l",lwd=3,ylab="dx/dt")
legend("topright",as.character(r),title="r",lty=1:5,col=1:6,lwd=3)
abline(h=0.25,col="green")
```



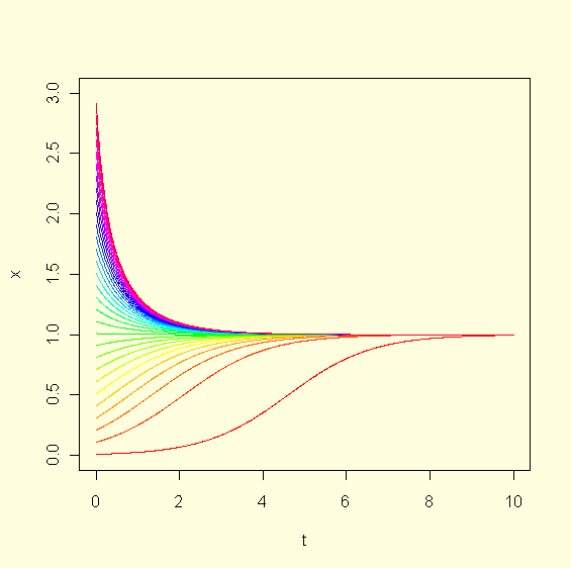
Ved likevekt er den førstederiverte lik null, dx/dt=0, dvs. $rx-rx^2=0$ som har løsninger x=0 og x=1. Den logistiske differensialligningen er et ikke-lineært flyt mellom to likevekter.



Figur. Viser logistisk differensialligning ved forskjellige initialverdier. Konjugerer mot 1 når t → ∞ og nærmer seg 0 når t → - ∞. Vektorfelter vil være tangenter til løsningene på figuren.

Eller automatisk med flere initialverdier, out.state<- as.data.frame må være definert som objekt fra tidligere:

```
#Logistisk vekstkurve med forskjellige initialverdier
params<-c(r=1)
logistisk4<-function(t,x,p){
with(as.list(p), {
dxdt<-r*x*(1-x)
list(dxdt)
})
}
require(deSolve)
time3<-seq(0,10,0.01)
#forskjellige initialverdier s
s<-seq(0.01,3,0.1)
n2<-length(s) #antall initialverdier
#tomt plot
y<-seq(0,10,0.01)
plot(time3,y,ylim=c(0,3),type="n",xlab="t",ylab="x")
for (i in 1:n2) {
init3<-s[i]
results3<-
as.data.frame(lsoda(init3,time3,logistisk4,params,rtol=1e-4))
lines(results3[,1],results3[,2],col=rainbow(n2)[i])
}
```



Figur. Logistisk kurve ved flere initialverdier seq(0.01,3,0.1).

Sirkulær bevegelse

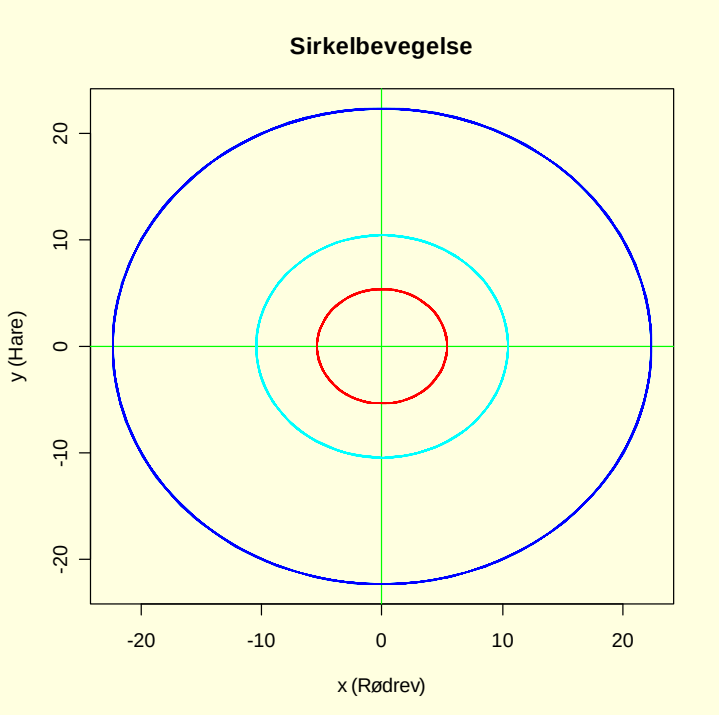
Anta i et enkelt system at vi har en stabil bestand av predatoren rødrev og byttedyret hare. Ved tid=0 blir det satt ut et overskudd av hare, $y=y_0$. Revene får mer mat og revebestanden øker, og disse dreper flere harer slik at harebestanden begynner å synke. Endring i revebestanden *dx/dt*

og harebestanden *dy/dt* kan beskrives av differensialligningene:

$$\frac{dx}{dt} = y$$

$$\frac{dy}{dt} = -x$$

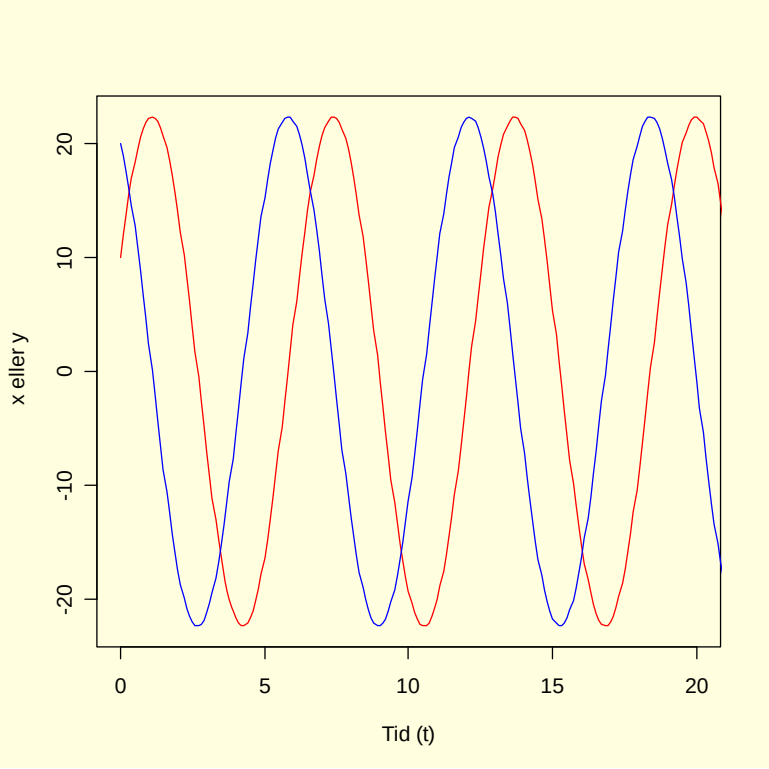
```
library(deSolve)
params<-
sirkel <-function(t,x,p)
{
dxdt<-x[2]
dydt<--x[1]
list(c(dxdt,dydt))
}
require(deSolve)
out.time<-seq(0,100,0.1)
init.state<-c(10,20)
out.state<-lsoda(init.state,out.time,sirkel,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],type="n",ylab="y
(Hare)",xlab="x (Rødrev)",main="Sirkelbevegelse")
lines(out.state[,2],out.state[,3],col="blue")
abline(h=0,col="green")
abline(v=0,col="green")
```



Figur. Fasediagram som viser lineær flyt omkring origo. Startbetingelser (10,20),blå), (2,5,rød) og (3,10,cyan).

```
library(deSolve)
params<-
```

```
sirkel <-function(t,x,p)
{
dxdt<-x[2]
dydt<--x[1]
list(c(dxdt,dydt))
}
require(deSolve)
out.time<-seq(0,100,0.1)
init.state<-c(10,20)
out.state<-lsoda(init.state,out.time,sirkel,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],type="n",xlim=c(0,20),ylab="
x eller y",xlab="Tid (t)")
lines(out.state[,1],out.state[,2],col="red")
lines(out.state[,1],out.state[,3],col="blue")
```



Figur. Tidsseriedigram for rødrev (x, rød) og hare (y, blå). Populasjonene gjennomgår sinusoscillasjoner med 90° ut av fase fra hverandre.

Lotka-Volterra predator-byttedyrmodell

Lotka-Volterra ligningen er et par første ordens ikke-linære differensialligninger som beskriver dynamikken når to arter interagerer med hverandre: predator (P) og byttedyr (B). Modellen er en forenkling av virkeligheten, men gir et innblikk i bruk av modeller.

Tankegangen ble presentert av både **Alfred J. Lotka** (1925) og **Vito Volterra** (1926). Endring i byttedyrbestand over tid er dB/dt og kan beskrives som:

$$\frac{dB}{dt} = rB\left(\frac{K-B}{K}\right) - bBP$$

Endring i bestanden av predatorer er dP/dt og kan beskrives som

$$\frac{dP}{dt} = cBP - dP$$

Ligningene har en rekke konstanter:
r – vekstraten for byttedyr uten predator tilstede
b – effekten av predator på byttedyr
d – dødsrate for predator i fravær av byttedyr
c – effektivitet og formeringsrate hos predator

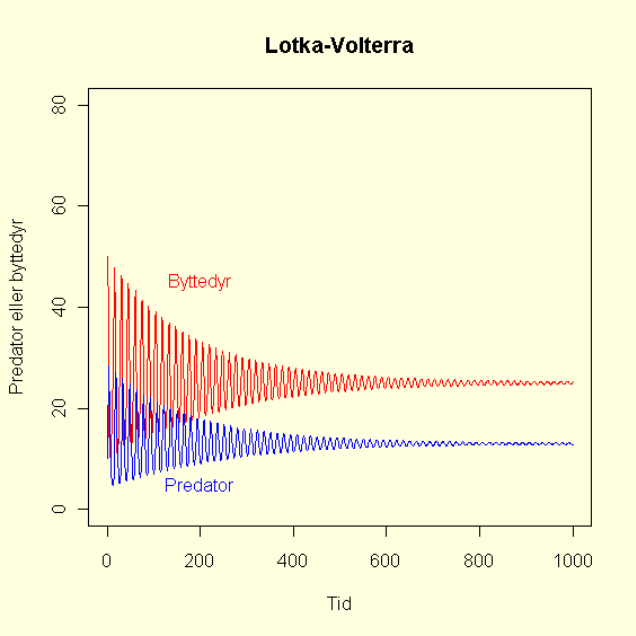
I ligningen for byttedyr har vi satt inn et ledd *(K-B)/K* hvor vi antar at byttedyrbestanden ikke vokser uendelig, men når et metningsnivå med en **bærekapasitet** *K*.

Følgende modelleksempel kan klippes ut og limes inn som script-fil i R.

Tidsseriediagram

```
#Lotka-Volterra tidsserie
#Parameterverdier (params)
params<-c(r=0.4,K=1000,b=0.03,c=0.02,d=0.5)
lotkavolterra<-function(t,x,p)
{
with(as.list(p), {
dbdt<-r*x[1]*(K-x[1])/K-b*x[1]*x[2]
dpdt<-c*x[1]*x[2]-d*x[2]
list(c(dbdt,dpdt))
})
}
require(deSolve)
tid<-0:1000
start<-c(50,10)
results<-
as.data.frame(lsoda(start,tid,lotkavolterra,params,rtol=1e-4))
plot(results[,1],results[,2],xlim=c(0,1000),ylim=c(0,80),type=
"n",ylab="Predator eller byttedyr",
xlab="Tid",main="Lotka-Volterra")
```

```
lines(results[,1],results[,2],col=2)
lines(results[,1],results[,3],col=4)
text(200,45,"Byttedyr",col=2)
text(200,5,"Predator",col=4)
```

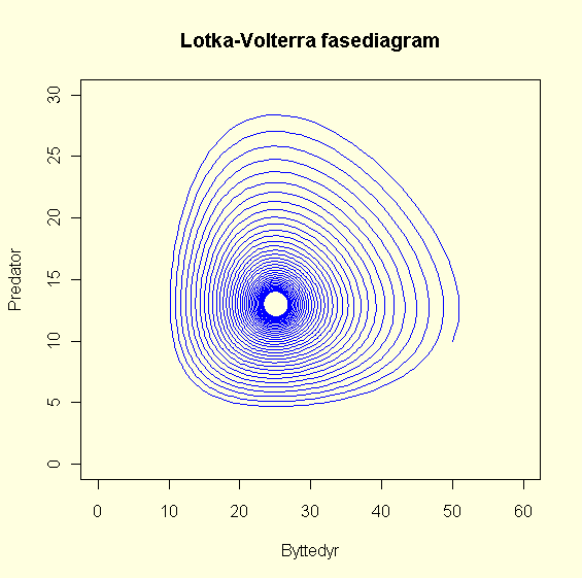


Figur. Tidsseriekurve for byttedyrvekstrate r=0.4, bærekapasitet K=1000, b=0.03, c=0.02, d=0.5, starter med 50 byttedyr og 10 predatorer (init.state<-c(50,10)). Etter en del svingninger vil etter hvert bestanden av byttedyr og predatorer stabilisere seg.

Fasediagram

```
#Lotka-Volterra fasediagram
#Parameterverdier (params)
params<-c(r=0.4,K=1000,b=0.03,c=0.02,d=0.5)
lotkavolterra<-function(t,x,p)
{
with(as.list(p), {
dbdt<-r*x[1]*(K-x[1])/K-b*x[1]*x[2]
dpdt<-c*x[1]*x[2]-d*x[2]
list(c(dbdt,dpdt))
})
}
require(deSolve)
tid<-seq(0,500,0.1)
start<-c(50,10)
results<-
as.data.frame(lsoda(start,tid,lotkavolterra,params,rtol=1e-4))
plot(results[,2],results[,3],xlim=c(0,60),ylim=c(0,30),type="n
",ylab="Predator",
xlab="Byttedyr",main="Lotka-Volterra fasediagram")
```

```
lines(results[,2],results[,3],col=4)
```



Figur. Fasediagram for **r**=0.4,**K**=1000,b=0.03,c=0.02,d=0.5. Starter med 50 byttedyr og 10 predatorer (init.state<-c(50,10)).Ved likevekt er(B,P)=(d/c,r/b)= (0.5/0.02, 0.4/0.03)=(25,13),som vi kan se av figuren er likevektspunktet med 25 byttedyr og 13 predatorer

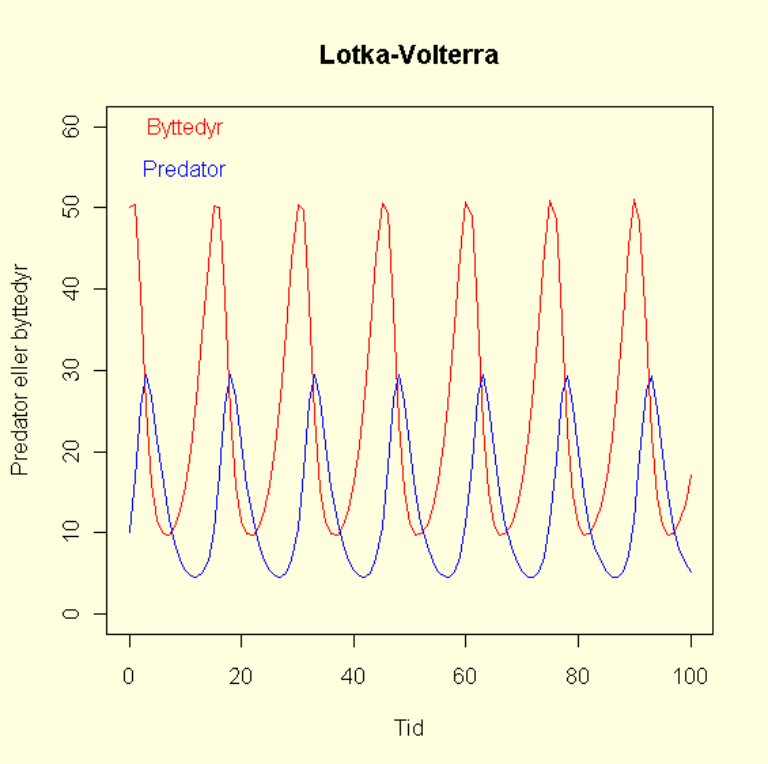
Hvis vi utelater leddet med bærekapasitet:

$$\frac{dB}{dt} = rB - bBP$$

$$\frac{dP}{dt} = cBP - dP$$

```
#Lotka-Volterra tidsserie uten K
#Parameterverdier (params)
params<-c(r=0.4,b=0.03,c=0.02,d=0.5)
lotkavolterra<-function(t,x,p)
{
with(as.list(p), {
dbdt<-r*x[1]-b*x[1]*x[2]
dpdt<-c*x[1]*x[2]-d*x[2]
list(c(dbdt,dpdt))
})
}
require(deSolve)
tid<-0:100
start<-c(50,10)
results<-
as.data.frame(lsoda(start,tid,lotkavolterra,params,rtol=1e-4))
plot(results[,1],results[,2],xlim=c(0,100),ylim=c(0,60),type="
n",ylab="Predator eller byttedyr",
```

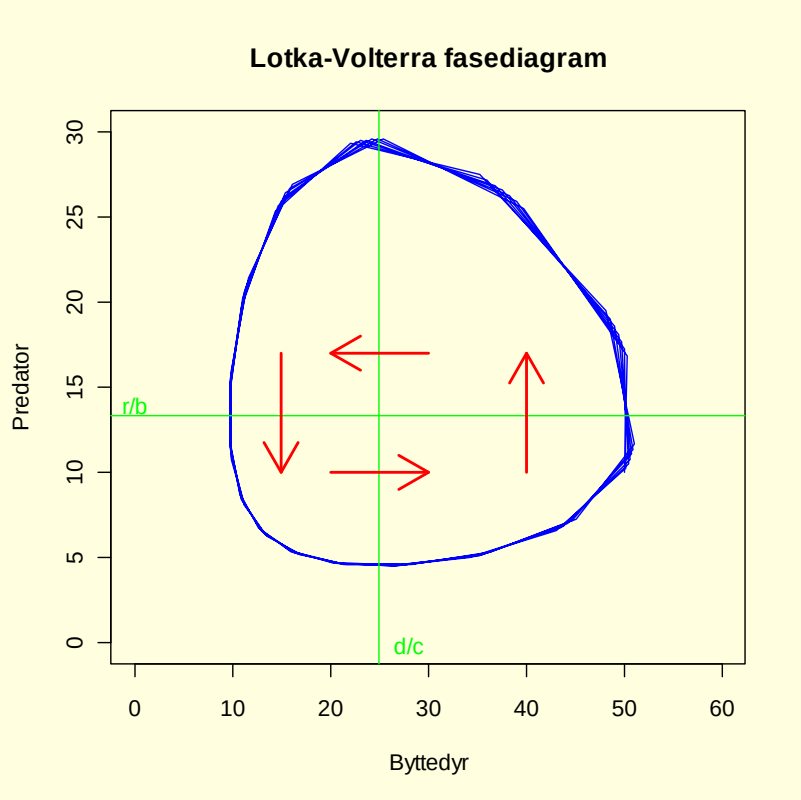
```
xlab="Tid",main="Lotka-Volterra")
lines(results[,1],results[,2],col=2)
lines(results[,1],results[,3],col=4)
text(10,55,"Predator",col=4)
text(10,60,"Byttedyr",col=2)
```



Figur. Tidsseriediagram Lotka-Volterra uten ledd for bærekapasitet. Byttedyrbestanden svinger mellom 10 og 50,og predatorbestanden svinger mellom 5 og 30.

```
#Lotka-Volterra predator-byttdyrmodell
#H. Aarnes
#Parameterverdier (params) og initialverdier init.state
params<-c(r=0.4,b=0.03,c=0.02,d=0.5)
lovol.model <-function(t,x,p)
{
dbdt<-p["r"]*x[1]-p["b"]*x[1]*x[2]
dpdt<-p["c"]*x[1]*x[2]-p["d"]*x[2]
list(c(dbdt,dpdt))
}
require(deSolve)
out.time<-0:100
init.state<-c(50,10)
out.state<-
as.data.frame(lsoda(init.state,out.time,lovol.model,params,rto
l=1e-4))
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlim=c(0,60),ylim=c(0,30),typ
e="n",ylab="Predator",xlab="Byttedyr",main="Lotka-Volterra
fasediagram")
```

```
lines(out.state[,2],out.state[,3],col="blue")
r<-0.4;b<-0.03;c<-0.02;d<-0.5
abline(h=r/b, col="green")
abline(v=d/c,col="green")
arrows(20,10,30,10,col="red",lwd=2)
arrows(30,17,20,17,col="red",lwd=2)
arrows(40,10,40,17,col="red",lwd=2)
arrows(15,17,15,10,col="red",lwd=2)
text(0,14,"r/b",col="green")
text(28,0,"d/c",col="green")
```



Figur. Lotka-Volterra fasediagram. I hvert punkt på trajektoriene (blå) kan det trekkes tangenter som viser i hvilken vei systemet beveger seg. Ved d/c og r/b er de deriverte lik 0 (d/c=25 byttedyr, og r/b=13.3) og deretter skifter systemet retning og beveger seg i retningen som de røde pilene viser, horisontale nullkliner for byttedyr og vertikale nullkliner for predatorer.

Vi må først finne maksimums- og minimumspunktene til systemet hvor de deriverte er lik 0 og vi har likevektspunkter, nullvekstisokliner for predator og byttedyr

$$\frac{dB}{dt} = \frac{dP}{dt} = 0$$

Deretter undersøker man hvordan systemet oppfører seg rundt likevekstpunktnene maks og min.

$$\frac{dB}{dt} = B \cdot (r - b \cdot P) = 0$$

$$\frac{dP}{dt} = P \cdot (c \cdot B - d) = 0$$

Som gir:

$$(B,P) = \left(\frac{d}{c}, \frac{r}{b}\right)$$

Jacobi-matrisen (*J(B,P)*)med de partiellderiverte i predator-byttedyrligningen kan brukes til å undersøke stabiliteten ved et fast punkt. For ligningene:

$$\frac{dB}{dt} = r \cdot B - b \cdot B \cdot P$$

$$\frac{dP}{dt} = c \cdot B \cdot P - d \cdot P$$

Blir Jacobi-matrisen:

$$J(B,P) = \begin{bmatrix} r - b \cdot P & -b \cdot P \\ c \cdot B & c \cdot B - d \end{bmatrix}$$

Ved likevekt *J(B,P)= J(0,0)* blir denne:

$$J(0,0) = \begin{bmatrix} r & 0 \\ 0 & -d \end{bmatrix}$$

Vi finner egenverdiene (λ)ved å løse:

$$\begin{vmatrix} r - \lambda & 0 \\ 0 & -d - \lambda \end{vmatrix} = 0$$

Og finner egenverdiene λ_1= r og λ_2=-d
De to tilsvarende egenvektorene(ν) danner to akser:

$$J \cdot \nu = \lambda \cdot \nu$$

Generelt om stabilitet ved likevektspunkt og Jacobimatriser:
Hvis vi har et system med to differensialligninger med initialverdier (x_0,y_0) ved tid t=0:

$$\frac{dx}{dt} = f(x,y)$$
$$\frac{dy}{dt} = g(x,y)$$

Så kalles matrisen med de partiellderiverte for Jacobimatrise *A*.

$$A = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}$$

Jacobimatrise, egenverdier og egenvektorer benyttes i studiet av stabiliteten av likevektspunkter i differensialligningene. Det kan være stabil eller ustabil likevekt. Systemet vokser eller synker i retning bestemt av egenvektorene og med hastighet bestemt av egenverdiene.

Lotka Volterra konkurransemodell

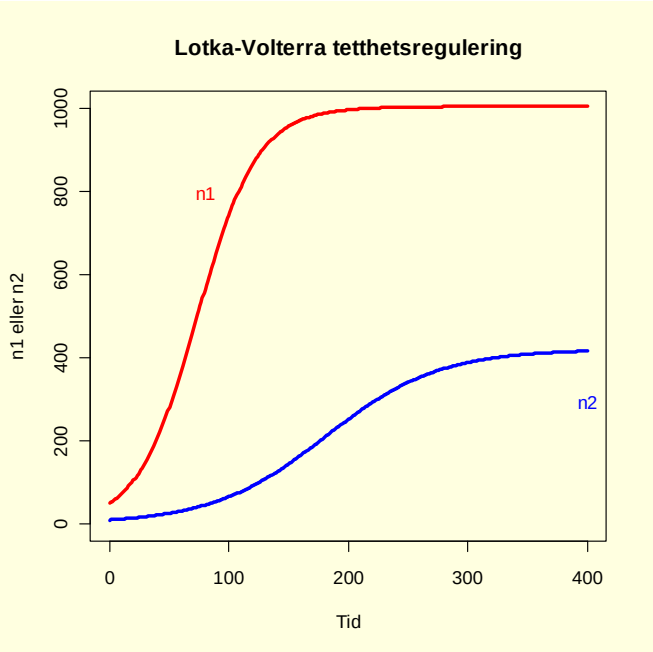
Logistisk modell som gir mer intraspesifikk konkurranse ettersom populasjonsstørrelsen øker. Vi har to arter med endring i antall arter over tid (n_1 og n_2)med veksthastigheter r_1 og r_2 og bærekapasitet K_1 og K_2, hvor a_{12} og a_{21} er konkurransekoeffisienter, a12 er per capita effekten av art2 på art1. Dette er en symmetrisk modell:

$$\frac{dn_1}{dt} = r_1 \cdot n_1 \left(\frac{K_1 - n_1 + a_{12} \cdot n_2}{K_1}\right)$$

$$\frac{dn_2}{dt} = r_2 \cdot n_2 \left(\frac{K_2 - n_2 + a_{21} \cdot n_1}{K_2}\right)$$

Hvis artene ikke interagerer dvs. a_{12} og a_{21} begge er lik 0 blir det en logistisk ligning. Hvis koeffisientene a_{12} og a_{21} begge er negative har vi mutualisme, hvis en er negativ og den andre er lik 0 er det kommensalisme. Hvis a_{12} og a_{21} begge er postive er det konkurranse mellom artene, og har de motsatt fortegn er det parasittisme.

```
##Lotka-Volterra konkurransemodell
#H. Aarnes
#Parameterverdier (params) og initialverdier init.state
params<-c(r1=0.04,r2=0.02,K1=1000,K2=400,a12=0.01,a21=0.02)
lvkonkurr<-function(t,x,p)
{
dn1dt<-p["r1"]*x[1]*(( p["K1"]-x[1]+p["a12"]*x[2])/p["K1"])
dn2dt<-p["r2"]*x[2]*(( p["K2"]-x[2]+p["a21"]*x[1])/p["K2"])
list(c(dn1dt,dn2dt))
}
require(deSolve)
out.time<-0:400
init.state<-c(50,10)
out.state<-
as.data.frame(lsoda(init.state,out.time,lvkonkurr,params,rtol=
1e-4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,400),ylim=c(0,1000),
type="n",ylab="n1 eller n2",xlab="Tid",main="Lotka-Volterra
tetthetsregulering")
lines(out.state[,1],out.state[,2],col="red",lwd=3)
lines(out.state[,1],out.state[,3],col="blue",lwd=3)
text(80,800,"n1",col="red")
text(400,300,"n2",col="blue")
```



Figur Lotka-Volterra tetthetsregulert tidsseriemodell med parameterverdier params<-c(r1=0.04,r2=0.02,K1=1000,K2=400,a12=0.01,a21=0.02)og initialverdier n_1=50 individer og n_2=10 individer, init.state<-c(50,10).

Det finnes mange modeller for sammenhengen mellom endring i antall byttedyr over tid (dy_1/dt) og antall predator over tid (dy_2/dt) hvor K (bærekapasitet), a (dødsrate for predator i fravær av byttedyr, b, og c (effektivitet og formeringsrate hos predator) er konstanter:

$$\frac{dy_1}{dt} = y_1\left(1-\frac{y_1}{K}\right) - y_2 \cdot {y_1}^b$$

$$\frac{dy_2}{dt} = c \cdot y_2 \cdot {y_1}^b - a \cdot y_2$$

Rosenzweig-MacArthur predator byttedyrmodell hvor dy_1/dt er endring bestanden av byttdyr og dy_2/dt er endringen i predatorbestanden:

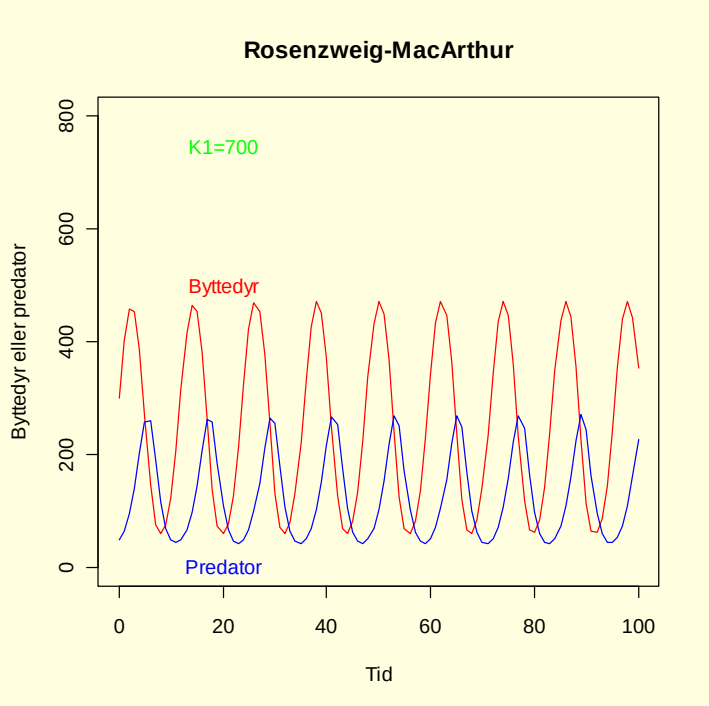
$$\frac{dy_1}{dt} = r_1 \cdot y_1\left(1-\frac{y_1}{K_1}\right) - y_2\frac{a_1 y_1}{B+y_1}$$

$$\frac{dy_2}{dt} = y_2\frac{a_2 y_1}{B+y_1} - r_2 y_2$$

Med konstanter *r1*, *r2*, *a1*, *a2*, *B* og *K1*.

```
#Parameterverdier (params)
params<-c(a1=2,a2=2,r1=1,r2=1, K1=700,B=200)
rosenarthur<-function(t,y,p)
{
dy1dt<-p["r1"]*y[1]*(1-y[1]/p["K1"])-
y[2]*((p["a1"]*y[1])/(p["B"]+y[1]))
dy2dt<-y[2]*((p["a2"]*y[1])/(p["B"]+y[1]))-p["r2"]*y[2]
list(c(dy1dt,dy2dt))
```

```
}
require(deSolve)
out.time<-0:100
init.state<-c(300,50)
out.state<-
lsoda(init.state,out.time,rosenarthur,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,100),ylim=c(0,800),t
ype="n",ylab="Byttedyr eller
predator",xlab="Tid",main="Rosenzweig-MacArthur")
lines(out.state[,1],out.state[,2],col="red")
lines(out.state[,1],out.state[,3],col="blue")
text(20,500,"Byttedyr",col="red")
text(20,5,"Predator",col="blue")
text(20,750,"K1=700",col="green")
```



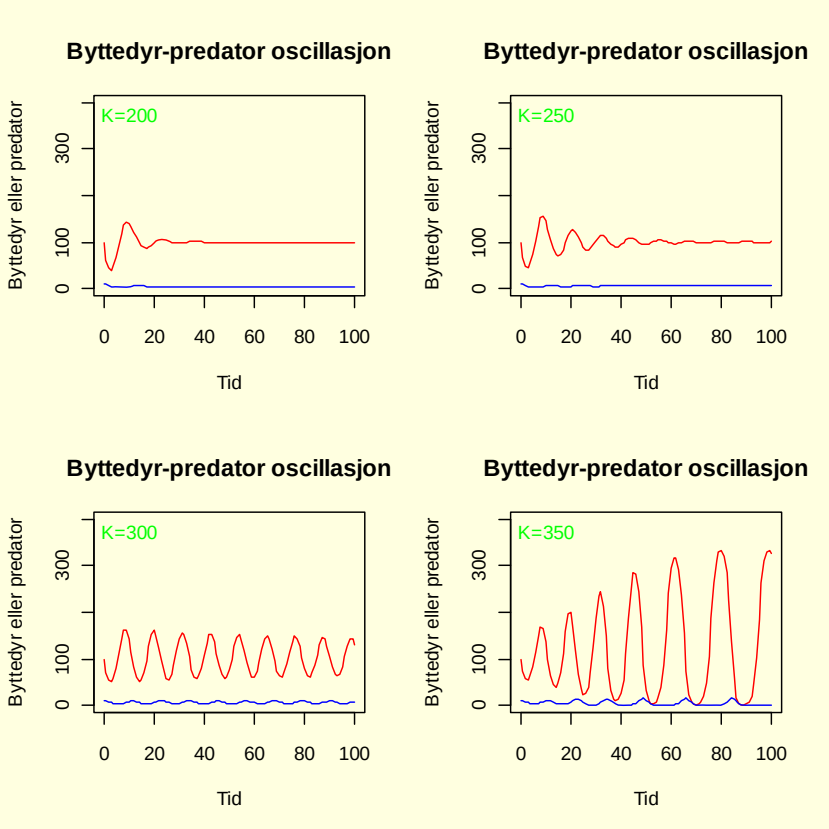
Eller:
Endring i antall byttedyr over tid (dy_1/dt) og antall predator over tid (dy_2/dt) hvor K (bærekapasitet), a (dødsrate for predator i fravær av byttedyr, b, og c (effektivitet og formeringsrate hos predator) er konstanter:

$$\frac{dy_1}{dt} = y_1\left(1-\frac{y_1}{K}\right) - y_2 \cdot {y_1}^b$$

$$\frac{dy_2}{dt} = c \cdot y_2 \cdot {y_1}^b - a \cdot y_2$$

```
Predator-Byttdyrmodell
#H. Aarnes
#Parameterverdier (params)
params<-c(b=0.5,K=350,c=0.1,a=1)
```

```
lovol.model <-function(t,y,p)
{
dy1dt<-y[1]*(1-y[1]/p["K"])-y[2]*y[1]^p["b"]
dy2dt<-p["c"]*y[2]*y[1]^p["b"]-p["a"]*y[2]
list(c(dy1dt,dy2dt))
}
require(deSolve)
out.time<-0:100
init.state<-c(100,10)
out.state<-
lsoda(init.state,out.time,lovol.model,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,100),ylim=c(0,400),t
ype="n",ylab="Byttedyr eller
predator",xlab="Tid",main="Byttedyr-predator oscillasjon")
lines(out.state[,1],out.state[,2],col="red")
lines(out.state[,1],out.state[,3],col="blue")
text(10,380,"K=350",col="green")
```



Kaos og Lorenz-ligningene

Meteorologen Edvard N. Lorenz ved MIT viste i 1963 hvordan konveksjonsstrømmer i atmosfæren kunne bli beskrevet av 12 ligninger som han forenklet til tre ligninger, hvor *x* er hastighet på rotasjonen, positiv med klokka, negativ mot klokka; *y* er temperaturforskjell mellom stigende og synkende væske, z er avvik fra linearitet i temperaturprofilen, *r* er

Rayleightall (Reynoldstall ?), σ er Prandtl tall, *b* aspekt ratio i en konveksjonssylinder (*Deterministic nonperiodic flow* J. Atmos.Sci. 1963,20,130). Startverdiene for Lorenz var $\sigma = 10$, $b = 8/3$ og $r = 28$.

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = -x \cdot z + r \cdot x - y$$

$$\frac{dz}{dt} = x \cdot y - b \cdot z$$

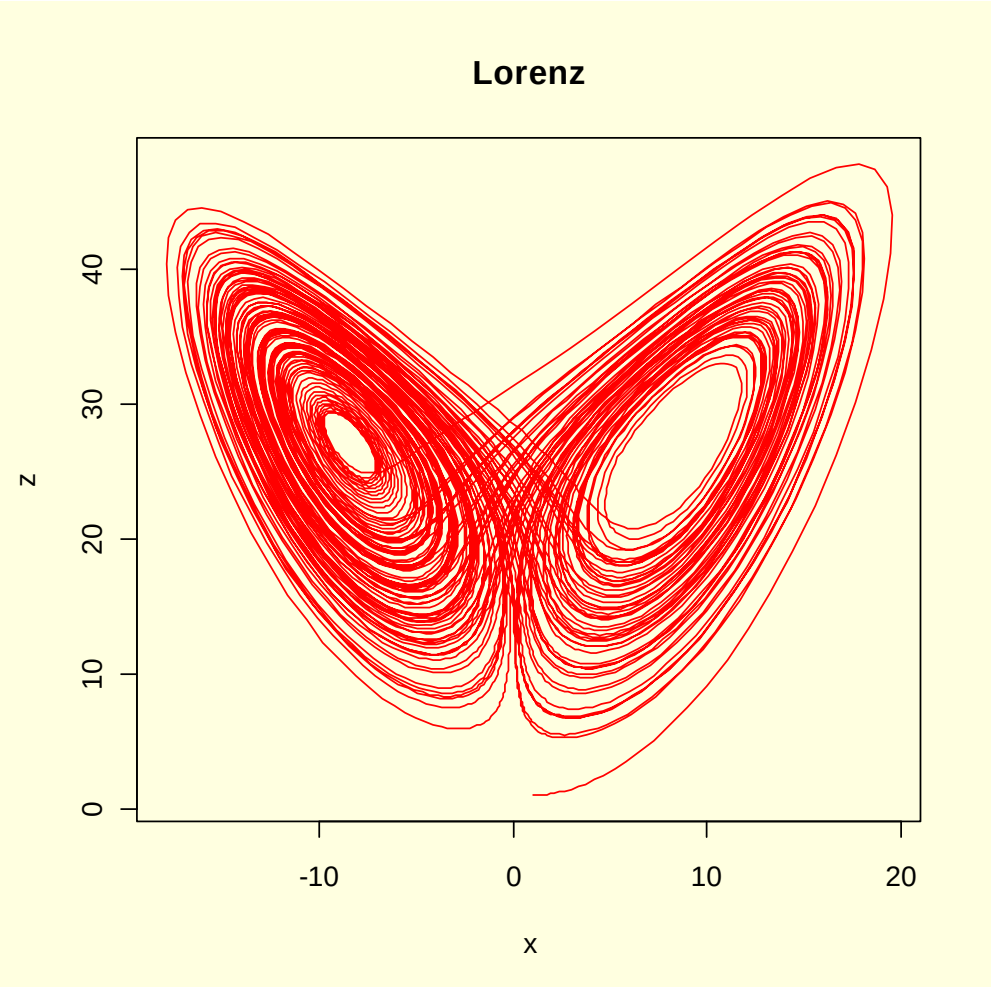
Løsningene oscillerer, men inne visse grenser, i form av en fraktal med dimensjon to til tre. Ligningene inneholder to kvadratledd *xy* og *xz*. Hvis løsningene x(t) blir plottet mot z(t) gir dette en figur som ligner på en sommerfugl. Todimensjonalt ser linjer ut til å krysse hverandre, men det skjer ikke i et tredimensjonalt plot. I et punkt på en linje kan det bare trekkes en tangent. Trajektoriet starter nær 0, svinger til høyre og inn i sentrum av en spiral på venstre side. Trajektoriet forlater spiralen etter å ha kommet en kritisk verdi vekk fra sentrum, og etter å ha gått i spiral spretter den over til høyre side, går litt i spiral, så over til venstre side, går litt i spiral osv. Det blir seende ut som to sommerfuglvinger. Lorenz viste at det å kommer med en sikker værmelding langt fram i tid er vanskelig, for ikke å si umulig. Framtiden er uforutsigbar. Resultatet er avhengig av utgangsbetingelsene, men selv om startbetingelsene er målt nøyaktig vil det alltid bli en feil mellom estimert verdi og sann verdi. Det er bare en viss tid fremover det er mulig å prediktere, det vil si komme med en statistisk sannsynlighet for en utviklingsretning. Det er dette som gjenspeiles i den berømte: *Does a flap of a butterfly´s wings in Brazil set off a tornado in Texas* (1972)

To trajektorier som ligger nær hverandre atskilles raskt og eksponensielt og kan beskrives via **Lyapunoveksponenten λ**. For et *n*-dimensjonalt rom er det *n* Lyapunoveksponenter hvor λ er størst, og λ avhenger av hvilke trajektorier man studerer. Hvis vi plotter den naturlige logaritmen *ln* til vektoren som funksjon av tiden som mål på atskillelsen av trajektoriene blir dette en tilnærmet rett linje med topper og daler, og som etter hvert når metning. Stigningen av denne linjen er gitt ved λ. Man kan bare prediktere fremtiden bare noen få multipler av 1/λ, hvilket betyr at det er vanskelig å prediktere oppførselen av kaotiske systemer jfr. værmeldinger. For Lorenz er Lyapunoveksponeten λ≈ 0.9056,0,-14,5723
Hvis et lokalt maksimum på tidsseriediagrammet *z* versus tiden *t* måles ved Z_n og neste ved z_{n+1}, og man deretter plotter lokale maksimum Z_n på abscissen mot z_{n+1} på ordinaten, $Z_{n+1} = f(z_0)$ kalles en **Lorenz graf**) så faller dataene fra en kaotisk tidsserie på en kurve. Dette ligner på **Poincaré avbildninger** og denne og Lorenz graf forsøker å forenkle analysen av differensialligninger. Poincaré avbildninger gir punkter på en

flate med to koordinater og viser hvordan koordinatene endrer seg når trajektoriet går tilbake til flaten.
Kaos er avhengig av initialbetingelsene og svært følsomme for disse. **Feigenbaum** sammenlignet kaos og fasetransisjoner May viste at det for høye veksthastigheter oppstår kaos i den **logistiske ligning**.

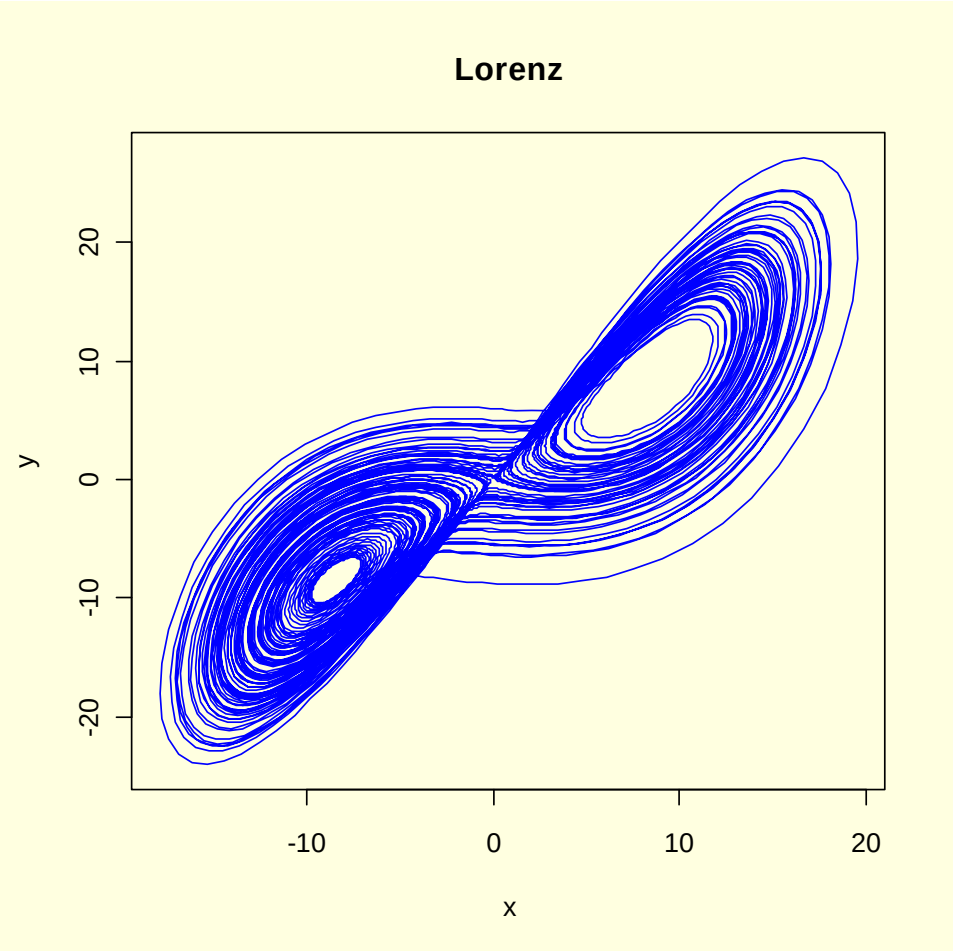
Fraktaler er geometriske framstillinger av kaosdynamikk, fingeravtrykkene til kaos. Fraktaler viser selvsimilaritet hvor et lite mønster gjenfinnes i det store. Skyformasjoner, bregnefronder, greining i trær og Romanesco-kål (*Brassica oleracea* cv. Romanesco) er eksempler på naturlige fraktaler. Andre eksempler på fraktaler er banefraktaler: Mandelbrotmengden, Julia-mengden, Lyapunov-mengden. Itererte fraktaler: von Koch´s snøflak,Cantor-mengde, Peanokurver, Sierpinskitriangel og Mengersvamp, samt tilfeldige fraktaler: Brownske fraktaler.

```
#Halvor Aarnes
library(deSolve)
params<-c(s=10,b=8/3,r=28)
lorenz<-function(t,y,p)
{
dx<-p["s"]*(y[2]-y[1])
dy<--y[1]*y[3]+ p["r"]*y[1]-y[2]
dz<-y[1]*y[2]-p["b"]*y[3]
list(c(dx,dy,dz))
}
require(deSolve)
out.time<-seq(0,100,0.01)
init.state<-c(1,1,1)
out.state<-lsoda(init.state,out.time,lorenz,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,4],xlab="x",ylab="z",type="l",co
l="red",main="Lorenz")
```



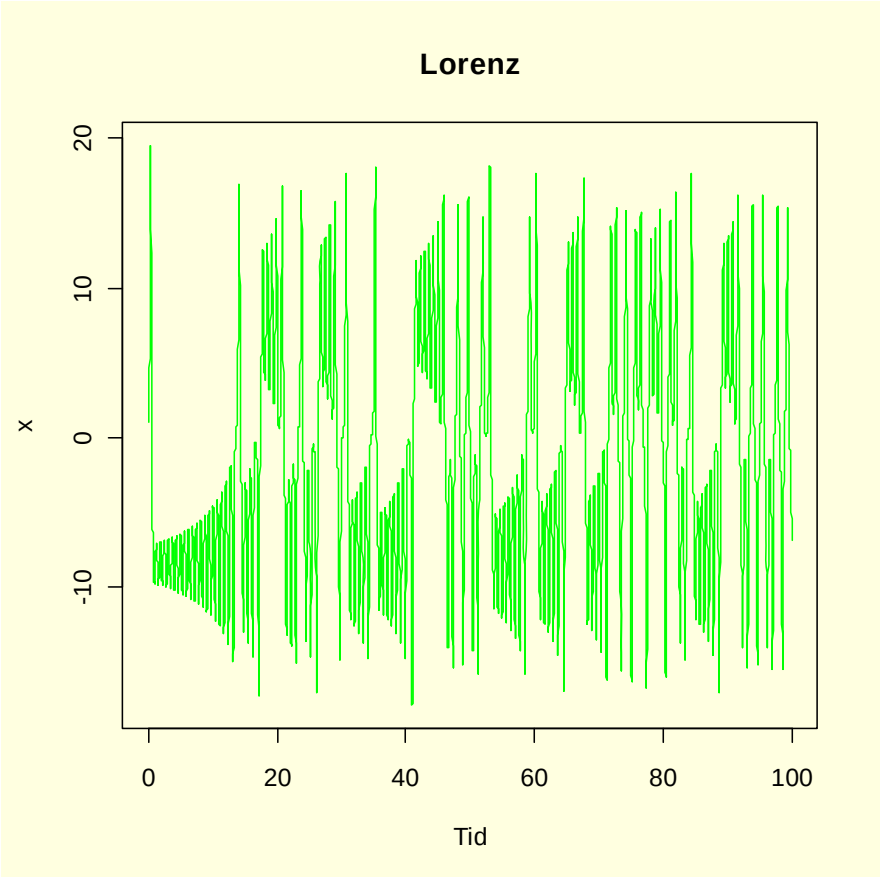
Figur. Fasediagram for Lorenz-ligningene. Figuren viser grafisk løsningen av differensialligningene. Det ser ut som linjene krysser hverandre i 2D-plot, men hadde det vært plottet i 3D ville vi ha sett at dette ikke stemmer. Det er bare en unik løsning i hvert punkt.

```
#Halvor Aarnes
library(deSolve)
params<-c(s=10,b=8/3,r=28)
lorenz<-function(t,y,p)
{
dx<-p["s"]*(y[2]-y[1])
dy<--y[1]*y[3]+ p["r"]*y[1]-y[2]
dz<-y[1]*y[2]-p["b"]*y[3]
list(c(dx,dy,dz))
}
require(deSolve)
out.time<-seq(0,100,0.01)
init.state<-c(1,1,1)
out.state<-lsoda(init.state,out.time,lorenz,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",co
l="blue",main="Lorenz")
```

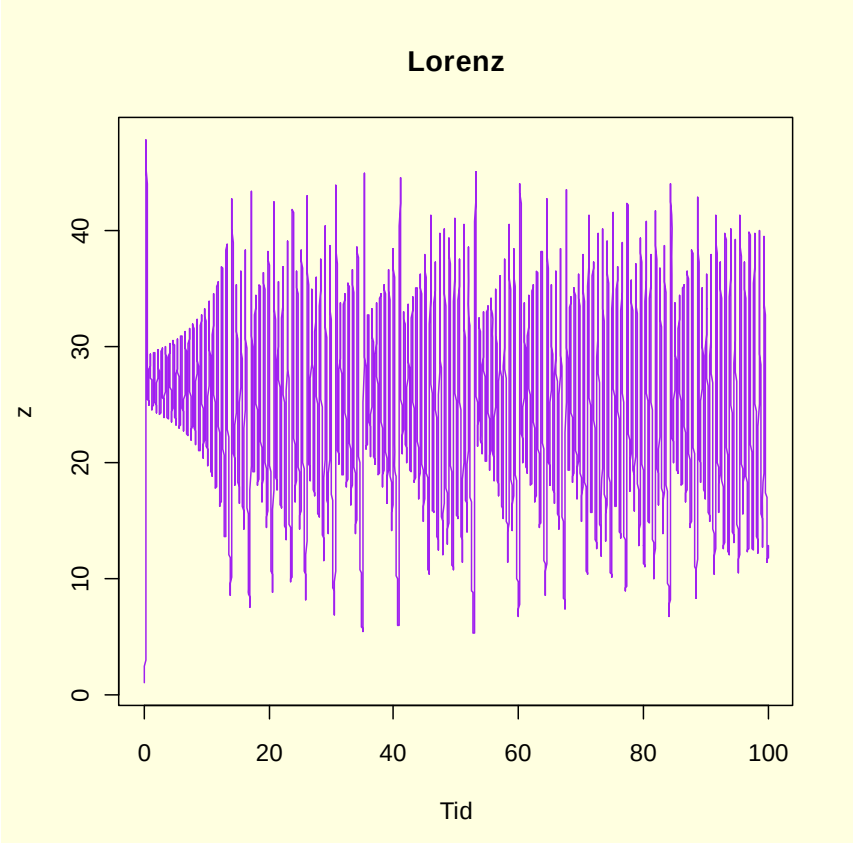


Figur. Fasediagram for Lorenz-ligningene.

```
library(deSolve)
params<-c(s=10,b=8/3,r=28)
lorenz<-function(t,y,p)
{
dx<-p["s"]*(y[2]-y[1])
dy<--y[1]*y[3]+ p["r"]*y[1]-y[2]
dz<-y[1]*y[2]-p["b"]*y[3]
list(c(dx,dy,dz))
}
require(deSolve)
out.time<-seq(0,100,0.01)
init.state<-c(1,1,1)
out.state<-lsoda(init.state,out.time,lorenz,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlab="Tid",ylab="x",type="l",
col="green",main="Lorenz")
```



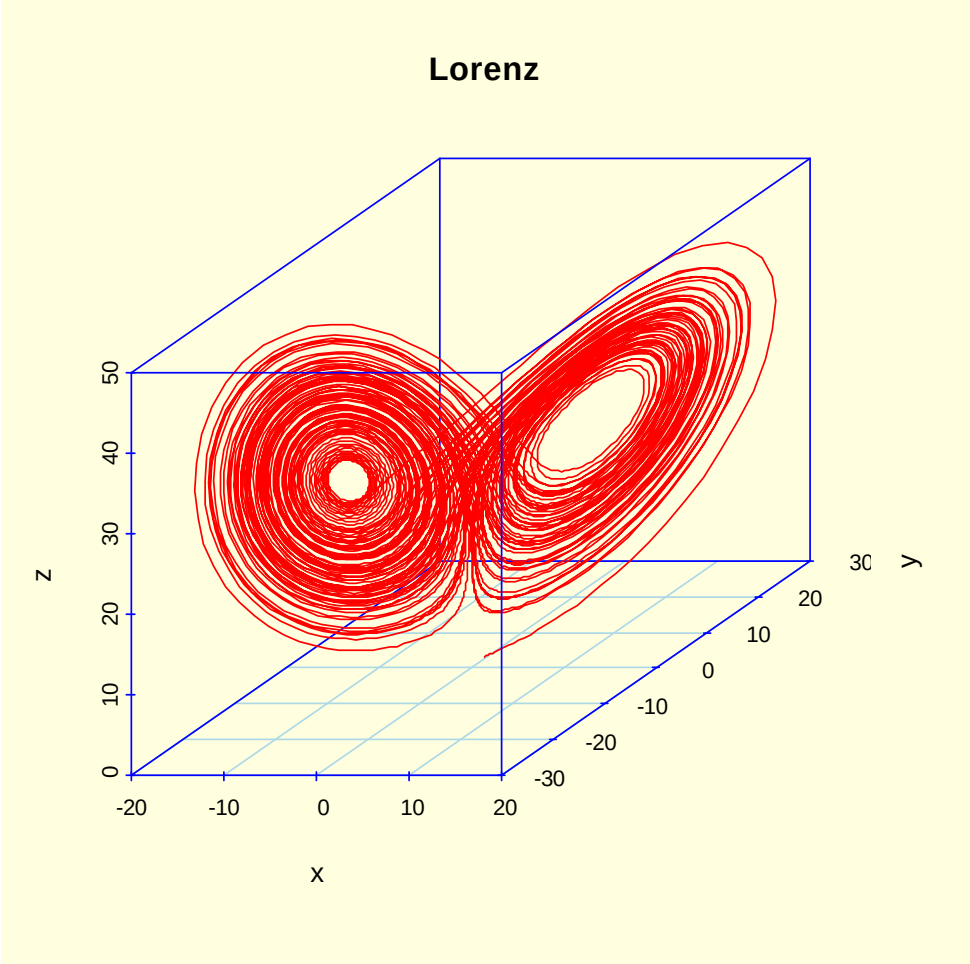
Figur. Tidsserieplot av Lorenz-ligningene som viser kaos.



Figur. Tidsserieplot av Lorenz-ligningene som viser kaos.

Hvis man ønsker et 3D plot last inn **library(scatterplot3D):**

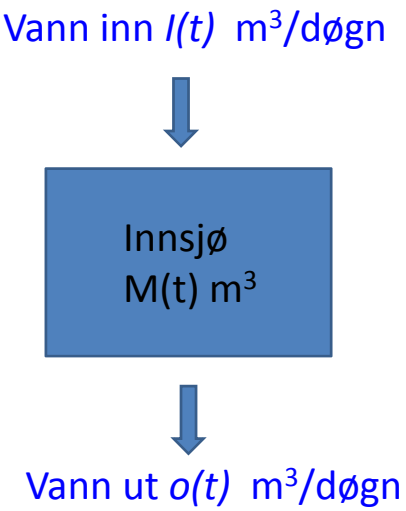
```
#Halvor Aarnes
library(deSolve)
params<-c(s=10,b=8/3,r=28)
lorenz<-function(t,y,p)
{
dx<-p["s"]*(y[2]-y[1])
dy<--y[1]*y[3]+ p["r"]*y[1]-y[2]
dz<-y[1]*y[2]-p["b"]*y[3]
list(c(dx,dy,dz))
}
require(deSolve)
out.time<-seq(0,100,0.01)
init.state<-c(1,1,1)
out.state<-lsoda(init.state,out.time,lorenz,params,rtol=1e-4)
par(bg="lightyellow")
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l",
xlab="x",ylab="y",zlab="z",main="Lorenz",color="red",col.axis=
"blue",
      col.grid="lightblue",)
```



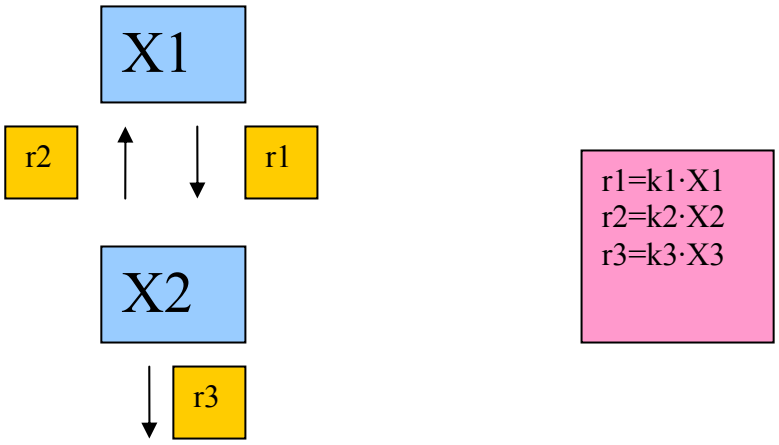
Boksmodell

Den enkleste typen boksmodell er eksempel på en innsjø hvor det strømmer vann inn og ut.
Endring i mengden vann i innsjøen *dM/dt* er lik mengden vann som renner inn i sjøen minus vann som renner ut:

$$\frac{dM}{dt} = I(t) - o(t)$$



Anta at vi har et stoff X1 som omdannes til X2 med reaksjonsrate r1; og X2 omdannes til X1 med rate r2, samt X2 forlater systemet med rate r3. Anta at vi har tre hastighetskonstanter *k1*, *k2* og *k3*.



Dette kan formuleres som følgen de differensialligninger:

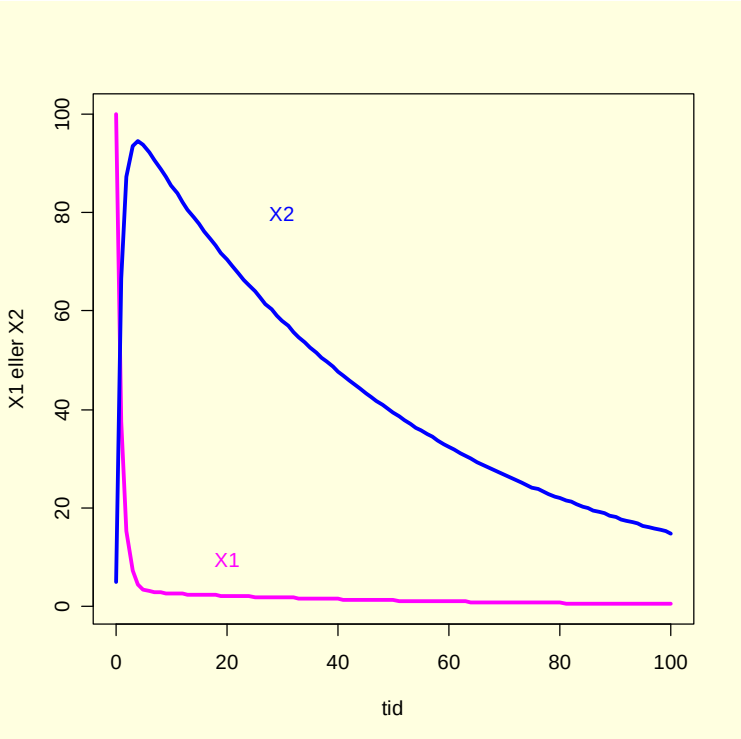
$$\frac{dx1}{dt} = k2 \cdot x2 - k1 \cdot x1$$

$$\frac{dx2}{dt} = k1 \cdot x1 - k2 \cdot x2 - k3 \cdot x2$$

Som kan løses numerisk med lsoda med gitte parameter- og utgangsverdier:

```
#eksempel
params<-c(k1=1,k2=0.03,k3=0.02)
eks2<-function(t,x,p)
{
dx1dt<-p["k2"]*x[2]-p["k1"]*x[1]
dx2dt<-p["k1"]*x[1]-p["k2"]*x[2]-p["k3"]*x[2]
list(c(dx1dt,dx2dt))
}
require(deSolve)
out.time<-0:100
init.state<-c(100,5)
out.state<-
as.data.frame(lsoda(init.state,out.time,eks2,params,rtol=1e-
4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],type="n",xlab="tid",ylab="X1
eller X2")
```

```
lines(out.state[,1],out.state[,2],col="magenta",lwd=3)
lines(out.state[,1],out.state[,3],col="blue",lwd=3)
text(20,10,"X1",col="magenta")
text(30,80,"X2",col="blue")
```



Parameterverdier k1=1,k2=0.03,k3=0.02 (params<-c(k1=1,k2=0.03,k3=0.02))og startverdi 100 X1 og 5 X2 (init.state<-c(100,5)).

Benytter man konstanter med måleenheter er det viktig at måleenhetene for modellen blir riktig.
Denne boksmodellen kan utvides til å gjelde parasitter-vertsdyr.

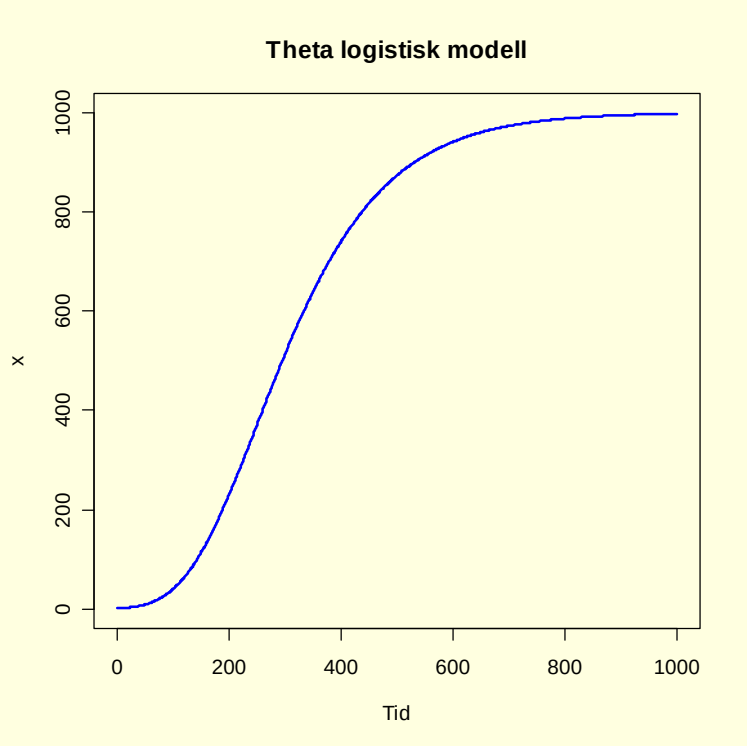
Theta-logistisk ligning

Vi har følgende theta(θ)-logistisk ligning:

$$\frac{dx}{dt} = r \cdot x\left(1 - \left(\frac{x}{K}\right)^{\theta}\right)$$

Løses numerisk i lsoda med gitte parametere (params<-c(r=0.4,K=1000,theta=0.02)og initialverdi (init.state<-1) som kan endres ettersom man tester modellen:

```
#Theta logistisk modell
params<-c(r=0.4,K=1000,theta=0.02)
logistisk<-function(t,x,p)
{
dxdt<-p["r"]*x*(1-(x/p["K"])^p["theta"])
list(dxdt)
}
require(deSolve)
out.time<-0:1000
init.state<-1
out.state<-
as.data.frame(lsoda(init.state,out.time,logistisk,params,rtol=
1e-4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],type="n",col="blue",xlab="Tid
",ylab="x",main="Theta logistisk modell")
lines(out.state[,1],out.state[,2],col="blue",lwd=2)
```



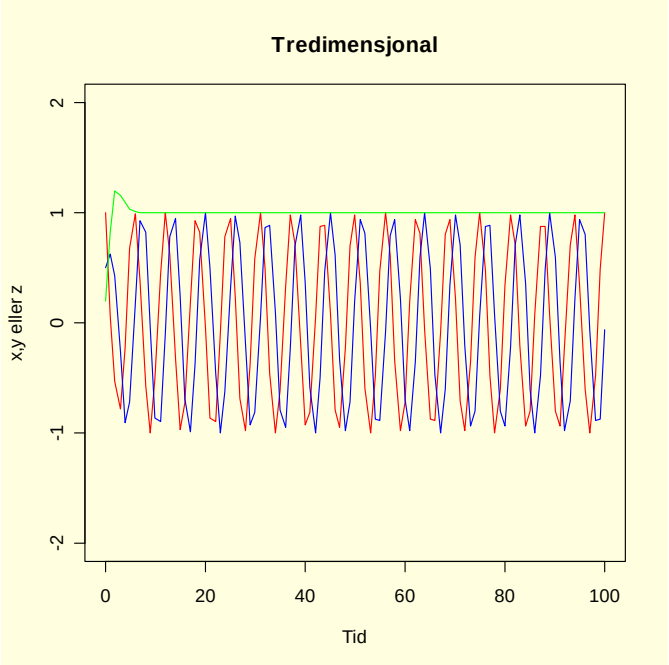
Numerisk løsning av Theta-logistisk modell med følgende verdier for konstanter: r=0.4,K=1000,theta=0.02

Vi har 3 differensialligninger som gir periodiske løsninger for x og y. Startverdier c(1,0.5,0.2)

$$\frac{dx}{dt} = -x - y + x \cdot z$$

$$\frac{dy}{dt} = x - y - y \cdot z$$
$$\frac{dz}{dt} = 1 + z - x^2 - y^2 - z^3$$

```
params<-
tredim<-function(t,x,p)
{
dxdt<--x[1]-x[2]+x[1]*x[3]
dydt<-x[1]-x[2]+x[2]*x[3]
dzdt<-1+x[3]-x[1]^2-x[2]^2-x[3]^3
list(c(dxdt,dydt,dzdt))
}
require(deSolve)
out.time<-seq(0,100,0.01)
init.state<-c(1,0.5,0.2)
out.state<-lsoda(init.state,out.time,tredim,params,rtol=1e-4)
par(bg="lightyellow")
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,100),ylim=c(-
2,2),type="n",ylab="x,y eller
z",xlab="Tid",main="Tredimensjonal")
lines(out.state[,1],out.state[,2],col="red")
lines(out.state[,1],out.state[,3],col="blue")
lines(out.state[,1],out.state[,4],col="green")
```



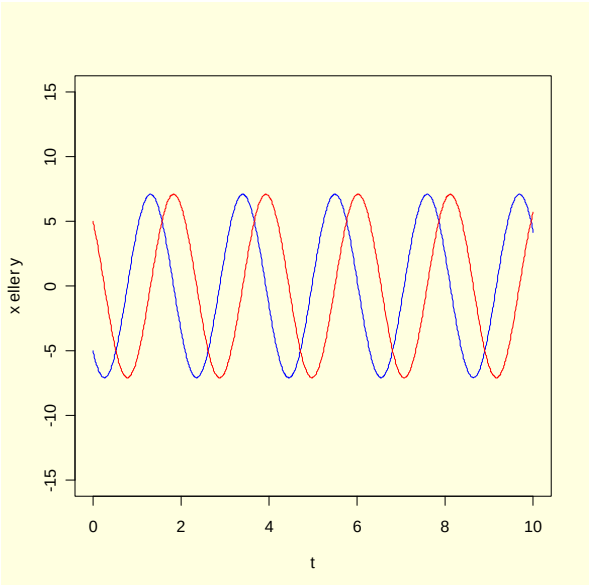
Figur. Tidsserieplot av 3 differensialligninger over.

Ligningssystemet:

$$\frac{dx}{dt} = ax + by$$
$$\frac{dy}{dt} = cx + dy$$

med a=0, b=-3, c=3, d=0, og startverdi (-5,5):

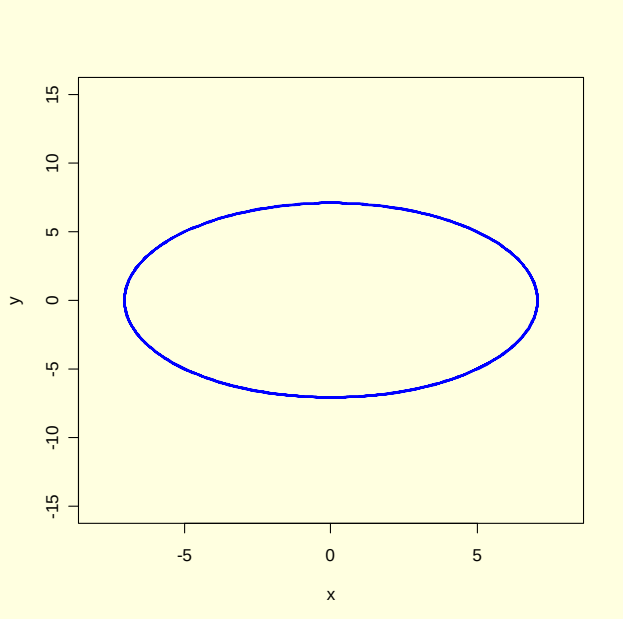
```
library(deSolve)
params<-c(a=0,b=-3,c=3,d=0)
diff<-function(t,y,p)
{
dy1dt<-p["a"]*y[1]+p["b"]*y[2]
dy2dt<-p["c"]*y[1]+p["d"]*y[2]
list(c(dy1dt,dy2dt))
}
require(deSolve)
out.time<- seq(0,10,0.01)
init.state<-c(-5,5)
out.state<-
as.data.frame(lsoda(init.state,out.time,diff,params,rtol=1e-
4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,10),ylim=c(-
15,15),type="n",ylab="x eller y",xlab="t")
lines(out.state[,1],out.state[,2],col="blue")
lines(out.state[,1],out.state[,3],col="red")
```



Figur

Hvis vi lager et fasediagram og plotter x mot y:

```
plot(out.state[,2],out.state[,3],xlim=c(-5,8),ylim=c(-
15,15),type="n",ylab="y",xlab="x")
lines(out.state[,2],out.state[,3],col="blue",lwd=2)
```

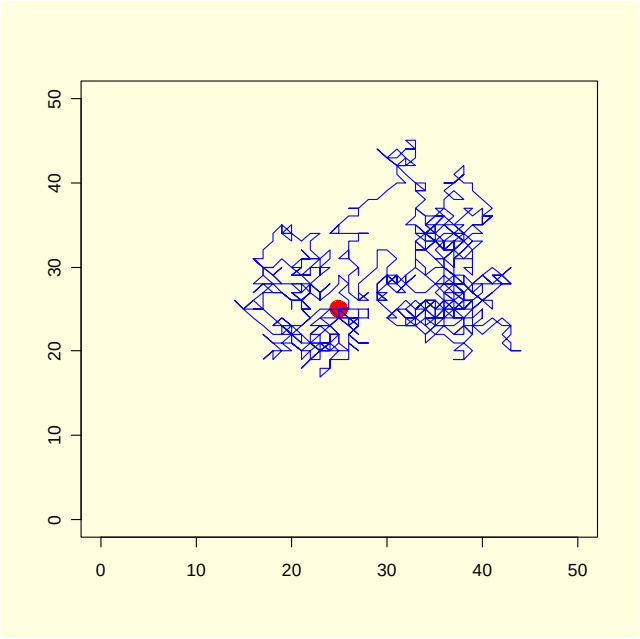


Figur

Diffusjon og random walk

Diffusjon, kinetisk gassteori og Brownske bevegelser er eksempler på tilfeldige prosesser.

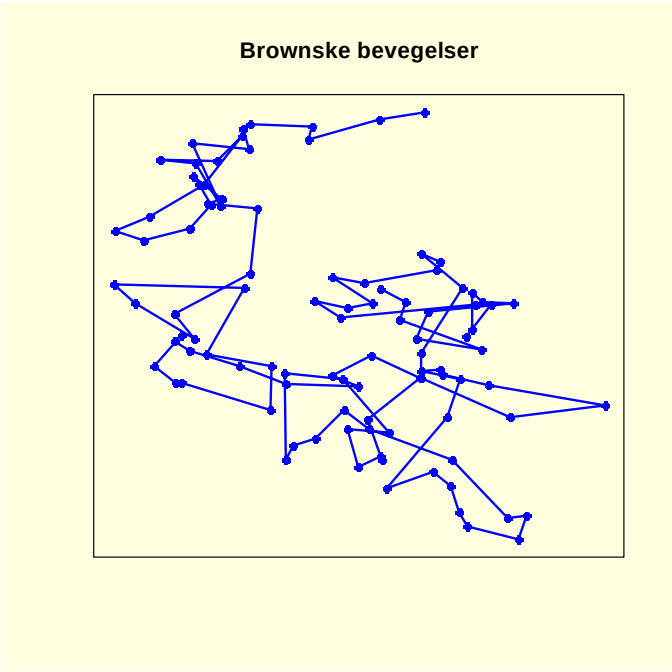
```
par(bg="lightyellow")
plot(0:50,0:50,type="n",xlab=" ",ylab=" ")
x<-25;y<-25
points(25,25,pch=16,col="red",cex=2.5)
for(i in 1:1000){
xi<-sample(c(1,0,-1),1)
yi<-sample(c(1,0,-1),1)
lines(c(x,x+xi),c(y,y+yi),col="blue")
x<-x+xi
y<-y+yi
if(x>50|x<0|y>50|y<0)break
}
```



Figur. Brownske bevegelser som gir forskjellig resultat hver gang scriptet kjøres

Vi kan lage Brownske bevegelser ved å lage en kumulativ sum (cumsum) av 100 normalfordelte tall:

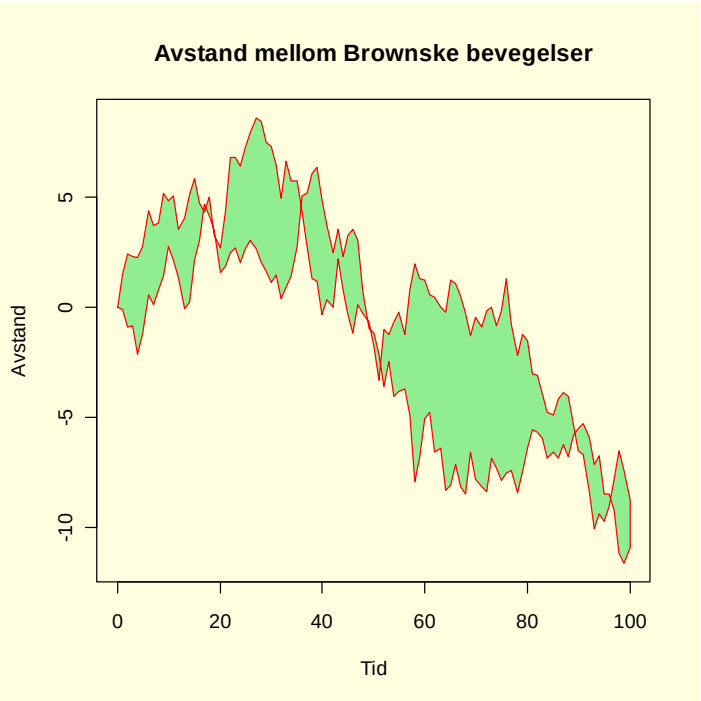
```
n<-100
x<- cumsum(rnorm(n))
y<- cumsum(rnorm(n))
par(bg="lightyellow")
plot(x,y,type = "o",col="blue",pch=16,lwd = 2,xlab=
"",ylab="",axes=FALSE,main="Brownske bevegelser")
box()
```



Vi kan også se på avstanden mellom de Brownske bevegelsene (etter R-manualen):

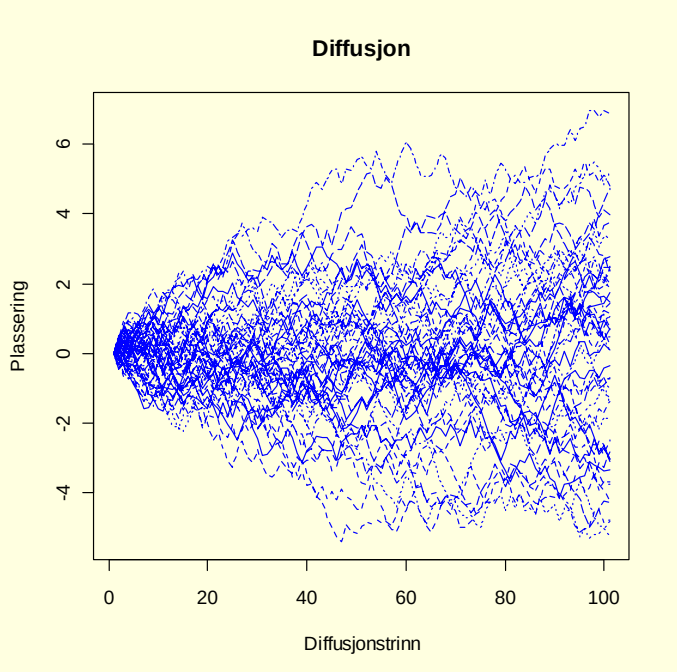
```
n<-100
```

```
xx <- c(0:n, n:0)
yy <- c(c(0,cumsum(rnorm(n))), rev(c(0,cumsum(rnorm(n)))))
par(bg="lightyellow")
plot   (xx, yy, type="n", xlab="Tid", ylab="Avstand")
polygon(xx, yy, col="lightgreen", border = "red")
title("Avstand mellom Brownske bevegelser")
```



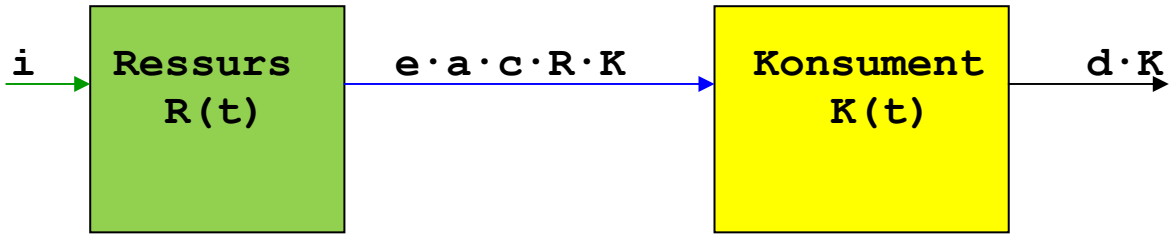
Vi kan simulere en diffusjon ved å plukke ut *n* tall mellom 0-1 med runif, trekker fra 0.5 for å unngå å få en retningsbestemt diffusjon, gjør dette i *m* trinn og samler verdiene i en matrise *A* hvor første rekke settes lik 0, som deretter plottes med matplot:

```
n<-50
m<-100
A<-matrix(nrow=m+1,ncol=n)
A[1,]<-0
for(i in 1:m) {
A[i+1,] <-A[i,] + runif (n)-0.5
}
par(bg="lightyellow")
matplot(A,type="l",col="blue",xlab="Diffusjonstrinn",ylab="Plassering",main
="Diffusjon")
```



Ressurs-konsumentmodell

Vi har ressurser (*R*) som blir konsumert (*K*) og disse endrer seg over tid (*dR/dt* og *dK/dt*). Vi lar i være konstant tilførsel av ressurser og *d* angir død av konsumentene (*dK*).



e er en omdanningsfaktor fra ressurs til konsument
a er sannsynligheten for at konsument bruker ressurs
c er kontaktrate mellom ressurs og konsument
d er død av konsumenten, *i.e.* konstant per capita dødsrate
i er immigrasjon eller tilførsel av ressurs, konstant ressurstilførsel.
Endring i ressursmengde over tid (*dR/dt*) er lik funksjonen av tilførsel av R (*f(R)*) og i dette tilfelle er *f(R)=i*. Tap av ressurs er en funksjon av både ressurs og konsument (*f(R,K)*) og i dette tilfelle er *f(R,K)=a·c·R·K*

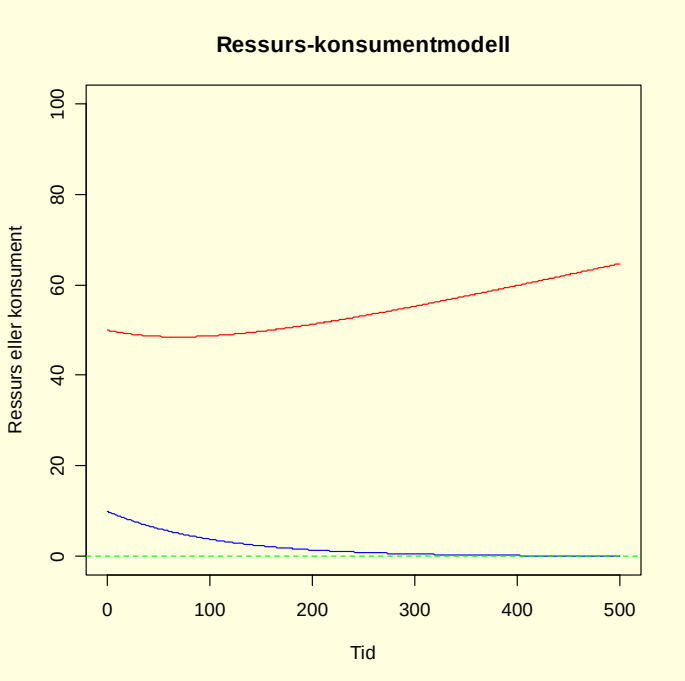
$$\frac{dR}{dt} = f(R) - f(R,K)$$

$$\frac{dR}{dt} = i - a \cdot c \cdot R \cdot K$$

Hvis f(R)=r·R har vi en konstant vekst av ressurs og dette tilsvarer Lotka-Volterra predator-byttedyrmodell. Hvis f(R)=r·R(1-R/K) vil veksten følge en logistisk funksjon. Endring i konsumentmengde (*dK/dt*) er tilførsel av ressurser (*e·a·c·R·K*) (type I ressurskonsumering) minus dødsrate (*d·K*):

$$\frac{dK}{dt} = e \cdot a \cdot c \cdot R \cdot K - d \cdot K$$

```
#Ressurs-konsumentmodell
#Parameterverdier (params) og initialverdier init.state
params<-c(e=0.04,a=0.02,c=0.01,d=0.01,i=0.05)
ressurs<-function(t,x,p)
{
dRdt<-p["i"]-p["a"]*p["c"]*x[1]*x[2]
dKdt<-p["e"]*p["a"]*p["c"]*x[1]*x[2]-p["d"]*x[2]
list(c(dRdt,dKdt))
}
require(deSolve)
out.time<-0:500
init.state<-c(50,10)
out.state<-
as.data.frame(lsoda(init.state,out.time,ressurs,params,rtol=1e
-4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,500),ylim=c(0,100),t
ype="n",ylab="Ressurs eller
konsument",xlab="Tid",main="Ressurs-konsumentmodell")
lines(out.state[,1],out.state[,2],col="red")
lines(out.state[,1],out.state[,3],col="blue")
abline(h=0,lty=2,col="green")
```

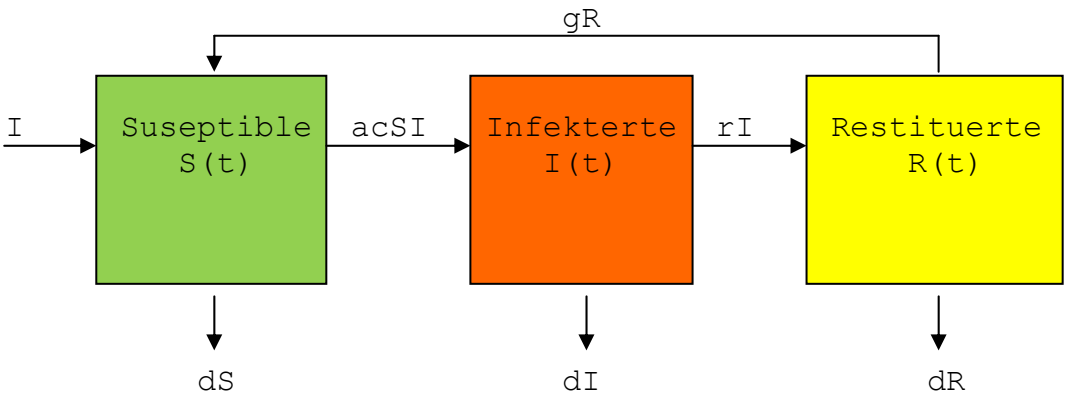


Figur ressurs-konsumentmodell med økning i ressurser (rød) og ekstinksjon av konsumenten (blå) med følgende

parameterverdier: c(e=0.04,a=0.02,c=0.01,d=0.01,i=0.05) og startverdi for ressurs=50 og konsument=10

Epidemiologisk modell

En epidemiologisk modell (SIR-modell) for spredning av sykdom



hvor
S er suseptible (mottakelige) indvider som kan bli syke
I er infekterte individer
R er restituerte individer som har blitt friske, hvorav noen kan bli syke på nytt med rate *g*
a er sannsynligheten for overføring av sykdom. Infekterte individer kommer i kontakt med suseptible med rate *c* per friskt individ
d er sannsynligheten for å dø, dødsrate
i er konstant tilførsel av suseptible individer bl.a. ved fødsel og immigrasjon.
Endring i suseptible, infekterte og restituerte per tidsenhet blir henholdsvis *dS/dt*, *dI/dt* og *dR/dt*.

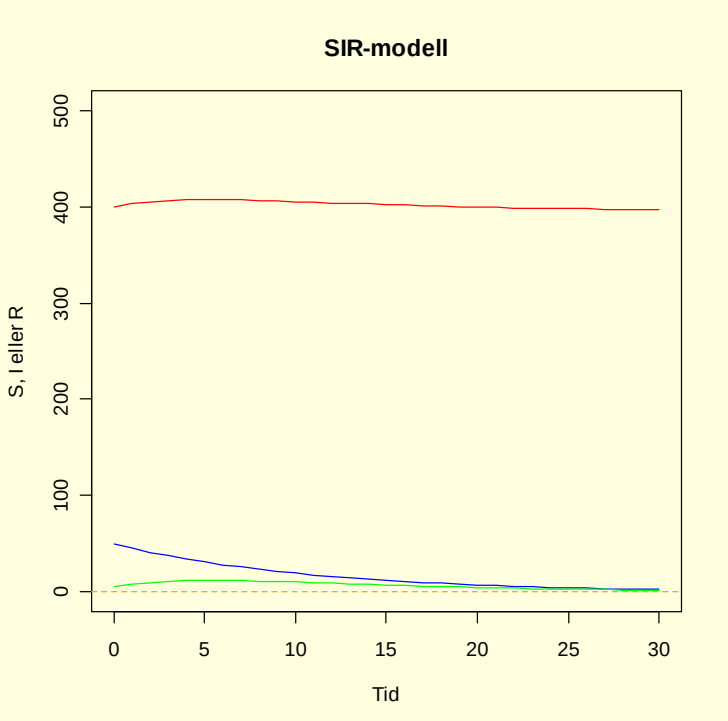
$$\frac{dS}{dt} = I - d \cdot S - a \cdot c \cdot S \cdot I + g \cdot R$$

$$\frac{dI}{dt} = a \cdot c \cdot S \cdot I - d \cdot I - r \cdot I$$

$$\frac{dR}{dt} = r \cdot I - g \cdot R - d \cdot R$$

```
#Epidemiologisk SIR-modell
#Parameterverdier (params) og initialverdier init.state
params<-
c(a=0.05,c=0.02,d1=0.001,d2=0.4,d3=0.2,i=0.05,g=0.05,r=0.1)
sir<-function(t,x,p)
{
dSdt<-p["i"]-p["d1"]*x[1]-p["a"]*p["c"]*x[1]*x[2]+p["g"]*x[3]
```

```
dIdt<-p["a"]*p["c"]*x[1]*x[2]-p["d2"]*x[2]-p["r"]*x[2]
dRdt<-p["r"]*x[2]-p["g"]*x[3]-p["d3"]*x[3]
list(c(dRdt,dIdt,dRdt))
}
require(deSolve)
out.time<-0:30
init.state<-c(400,50,5)
out.state<-
as.data.frame(lsoda(init.state,out.time,sir,params,rtol=1e-4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],xlim=c(0,30),ylim=c(0,500),ty
pe="n",ylab="S, I eller R",xlab="Tid",main="SIR-modell")
lines(out.state[,1],out.state[,2],col="red")
lines(out.state[,1],out.state[,3],col="blue")
lines(out.state[,1],out.state[,4],col="green")
abline(h=0,lty=2,col="orange")
```

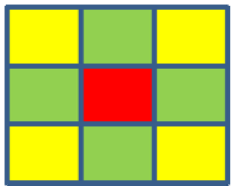


Figur SIR-modell med parametere c(a=0.05,c=0.02,d1=0.001,d2=0.4,d3=0.2,i=0.05,g=0.05,r=0.1) og initialverdier 400 suseptible individer (S, rød), 50 infekterte (I, blå) og 5 restituerte (R, grønn), init.state<-c(400,50,5)

Conways game of life

Den engelske matematikeren John Horton Conway utforsket simulering av en populasjon med celler og dens utvikling over tid,kalt Conways game of life, men er verken et spill eller liv, men emulerer prinsipper for utvikling av liv og er et eksempel på selvorganisering. Man starter med et tilfeldig sett av levende celler i et rektangulært rutenett som kan ha

to stadier ”død” eller ”levende”. Antall celler fra en runde til den neste endrer seg.
Hver celle har 8 naboceller,



4 på siden og 4 på diagonalen. Dette gir 2^9 = 512 muligheter for celler og naboer. Ytterst i rutenettet blir det færre naboer.Dette er eksempel på et Moore naboskap med r=1.
Prinsippet kan være følgende:
En levende celle med færre enn to naboer dør av ensomhet.
En levende celle med flere enn 3 levende naboer dør av trengsel.
En levende celle med 2 eller 3 levende naboer lever og overlever til neste generasjon.
En død celle med eksakt 3 levende naboer blir levende.

Dette danner en todimensjonal boolsk matrise, og det enkleste spillet er endimensjonalt hvor hver celle har to muligheter. Startmønsteret utvikles til kolonner, noen periodiske og andre statiske. Med dette settet av regler oppstår bl.a. glidere, en samling av celler som glir og forflytter seg over nettverket, blinkere, en ”organisme” som skyter ut 5 levende celler som beveger seg diagonalt. Spillet er uforutsigbart. Den eneste måten å få vite resultatet er å spille det.
http://www.youtube.com/watch?v=FdMzngWchDk&feature=related
Conways game of life er et eksempel på **cellulære automater** som kan utvides til flere dimensjoner. Andre eksempler på selvorganiserende cellulære automater er Ising-modellen (todimensjonal løsning via Lars Onsager) og neurale nettverk.

Markovkjeder og Markovmodeller

Sannsynligheten for at et system er i et spesielt stadium ved tid t avhenger bare av stadiet til systemet ved tiden $t-1$, og avhenger ikke av hva som var før $t-1$.
Tostadiemodeller og multistadiemodeller
Transisjonssannynlighetsmatriser (stokastiske matriser) for bevegelse mellom to stadier.
Markov-matriser er kvadratmatriser:

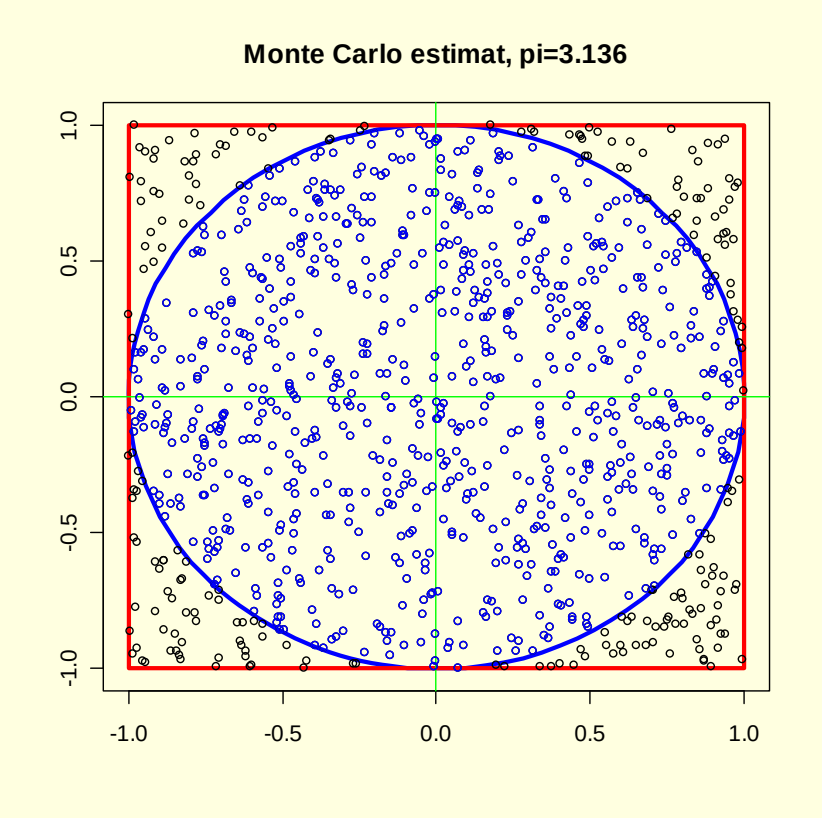
Sannsynlighetene i en rad blir lik 1 i sum.

Markov og Monte Carlo

Vi skal beregne arealet av en sjø eller beregne størrelsen av pi (π).
Anta at av 1000 punkter som havner inne i det røde kvadratet treffer 830 inne i den blå sjøen. Vi finner et estimat av hvor ofte punkter i vannet.
P=(830 treffer sjøen/1000 treffer kvadrat) = (areal sjø/1)
Kryssmultipliserer:
Areal av sjø= 830x1/1000= 0.83.
Tegner sirkel, kvadrater, velger ut tilfeldige tall fra en uniform fordeling og plotter disse. Deretter beregnes et estimat for pi (π). Velger radius r=1. Arealet av en sirkel er πr^2. Arealet av en firkant er $2r \cdot 2r = 4r^2$. Forholdet mellom dem er π/4. Trekk tilfeldige tall av punkter med x,y koordinater på innsiden av kvadratet, generert fra en uniform fordeling. Punkter hvor x2 + y2 >1 gir punkter på innsiden av sirkelen. Forholdet mellom dem gir et estimat av pi, og estimatet blir mer nøyaktig når *n* øker *e.g.* n=10.000. Monte Carlo er en stokastisk teknikk basert på tilfeldige tall. En annen måte å beregne et estimat av pi er Buffon´s nål.

```
o<- seq(0,2*pi,length=100)
x<-sin(o)
y<-cos(o)
par(bg="lightyellow")
plot(x,y,type="l",col="blue",xlab="",ylab="",lwd=3)
x<-c(1,-1,-1,1,1)
y<-c(1,1,-1,-1,1)
points(x,y,type="l",col="red",xlab="",ylab="",lwd=3)
abline(h=0,col="green")
abline(v=0,col="green")
n=1000
a<-runif(n,-1,1)
b<-runif(n,-1,1)
points(a,b)
sirkel<-a^2+b^2
i<-which(sirkel<1)
points(a[i],b[i],col="blue")
p<-4*sum(sirkel<1)/n
```

```
title(main=paste("Monte Carlo estimat, pi=",p,sep=''))
```



Figur. Viser et estimat for bestemmelse av π, basert på Monte-Carlo metode for n=1000. Vi får forskjellig estimat for pi for hver gang modellen kjøres, men nøyaktigheten øker når n øker.

Rössler, Nosé-Hoover, Rucklidge

Nedenfor er det vist en del eksempler på løsning av sett med differensialligninger:

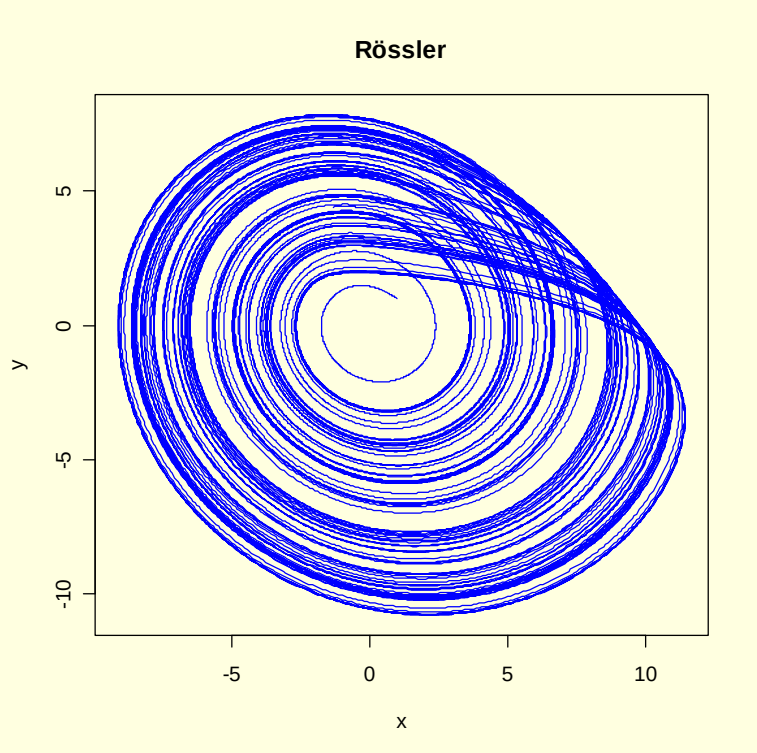
Rössler

Et enklere system enn Lorenz uten kvadratledd, med kaotisk løsning for a=0.2, b=0.2 og c=5.7, et av de enkleste kaotiske systemene.
Rössler, O.E.: *An equation for continous chaos*. Phys.Lett A57 (1976)397.

$$\frac{dx}{dt} = -y - z$$
$$\frac{dy}{dt} = x + a \cdot y$$
$$\frac{dz}{dt} = b + z \cdot (x - c)$$

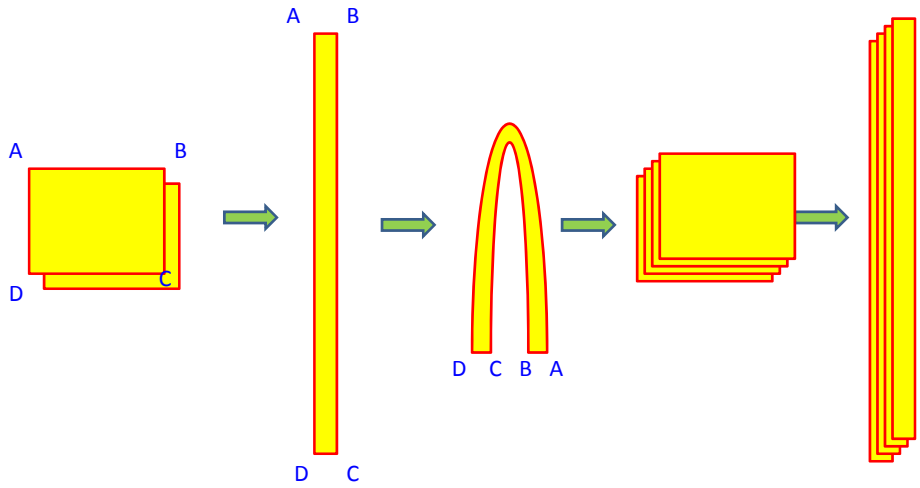
```
#Halvor Aarnes
library(deSolve)
params<-c(a=0.2,b=0.2,c=5.7)
```

```
roessler<-function(t,y,p)
{
dx<--y[2]-y[3]
dy<-y[1]+ p["a"]*y[2]
dz<-p["b"] +y[3]*(y[1]-p["c"])
list(c(dx,dy,dz))
}
require(deSolve)
out.time<-seq(0,100,0.01)
init.state<-c(1,1,1)
out.state<-lsoda(init.state,out.time,roessler,params,rtol=1e-
4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",co
l="blue",main="Rössler")
```

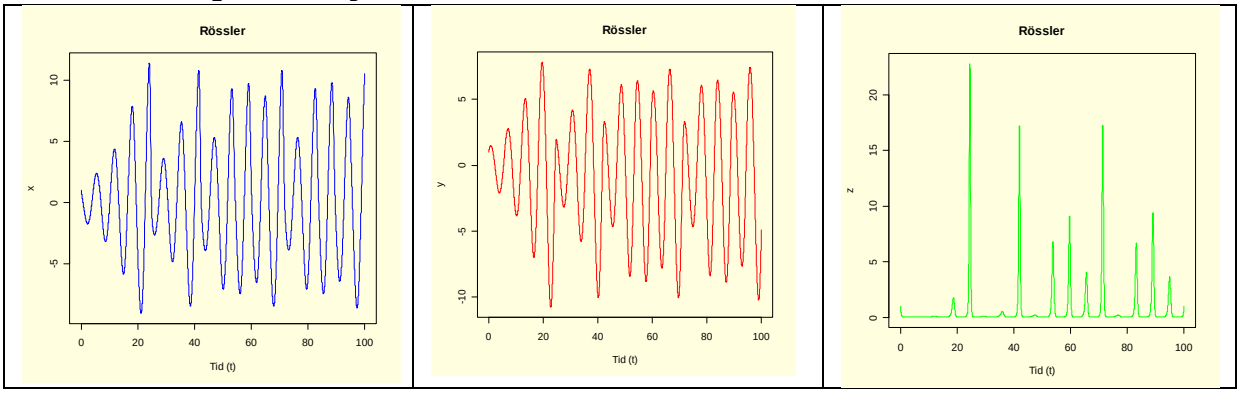


Figur. Fasediagram for Rössler ligningene og initialverdier (1,1,1).

Når trajektorier atskilles eksponensielt og divergerer uendelig hvordan er det mulig at de er begresnet ? Svaret er strekning og folding.



Figur. Som en firkantet deig som kjevles ut til en flat deig, foldes og kjevles ut på nytt, slik at den til slutt består av en rekke tynne lag. Jfr. et wienerbrød

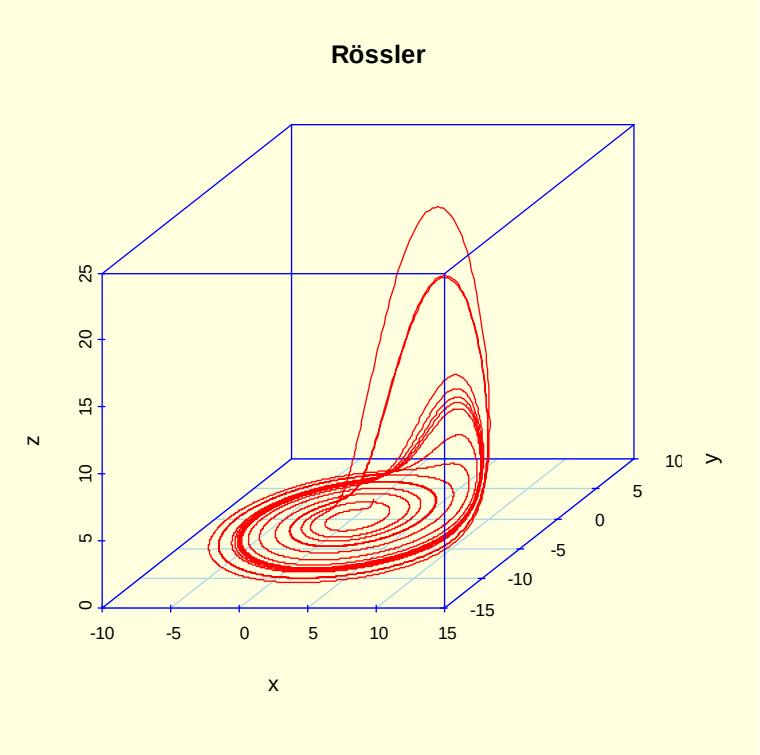


Figur. Tidsseriediagram for Rössler ligningene x(blå),y(rød),z(grønn) og initialverdier (1,1,1). Plot maksimumspunktene z_n versus z_{n+1}som gir en Lorenzavbildning

3D Rössler:

```
#Halvor Aarnes
library(deSolve)
params<-c(a=0.2,b=0.2,c=5.7)
roessler<-function(t,y,p)
{
dx<--y[2]-y[3]
dy<-y[1]+ p["a"]*y[2]
dz<-p["b"] +y[3]*(y[1]-p["c"])
list(c(dx,dy,dz))
}
require(deSolve)
out.time<-seq(0,100,0.01)
```

```
init.state<-c(1,1,1)
out.state<-lsoda(init.state,out.time,roessler,params,rtol=1e-
4)
par(bg="lightyellow")
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l",
xlab="x",ylab="y",zlab="z",main="Rössler",color="red",col.axis
="blue",
      col.grid="lightblue",)
```



Figur. 3D fasediagram for Rössler ligningene og initialverdier (1,1,1). Benytt anledning til å se på det optiske bedraget hvor den blå boksen skifter mellom hvilken side som vender fram.

Nosé-Hoover oscillator

$$\frac{dx}{dt} = y$$

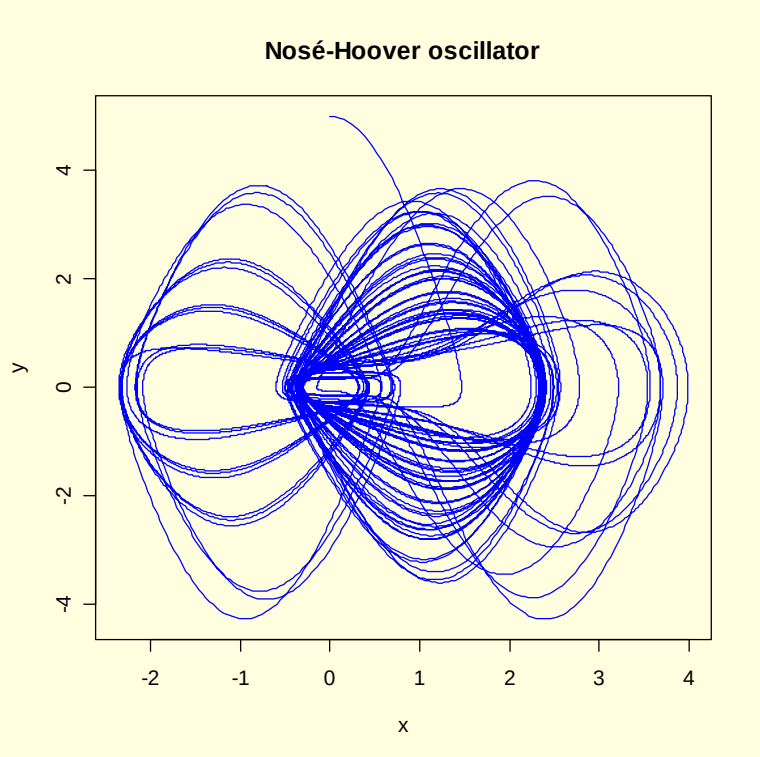
$$\frac{dy}{dt} = -x + y * z$$

$$\frac{dz}{dt} = a - y^2$$

Nosé, S. *Constant temperatur molecular dynamics methods*. Progress of theoretical Physics Supplement 103 (1991)1-46.

Hoover, G. *Remark on "Some simple chatoic flows*. Physical Review E 51 (1995)759-760.

```
#Halvor Aarnes
library(deSolve)
params<-c(a=1)
nosehoover<-function(t,y,p)
{
dxdt<-y[2]
dydt<--y[1]+y[2]*y[3]
dzdt<-p["a"]-y[2]^2
list(c(dxdt,dydt,dzdt))
}
require(deSolve)
out.time<-seq(0,500,0.01)
init.state<-c(0,5,0)
out.state<-
lsoda(init.state,out.time,nosehoover,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",co
l="blue",main="Nosé-Hoover oscillator")
```

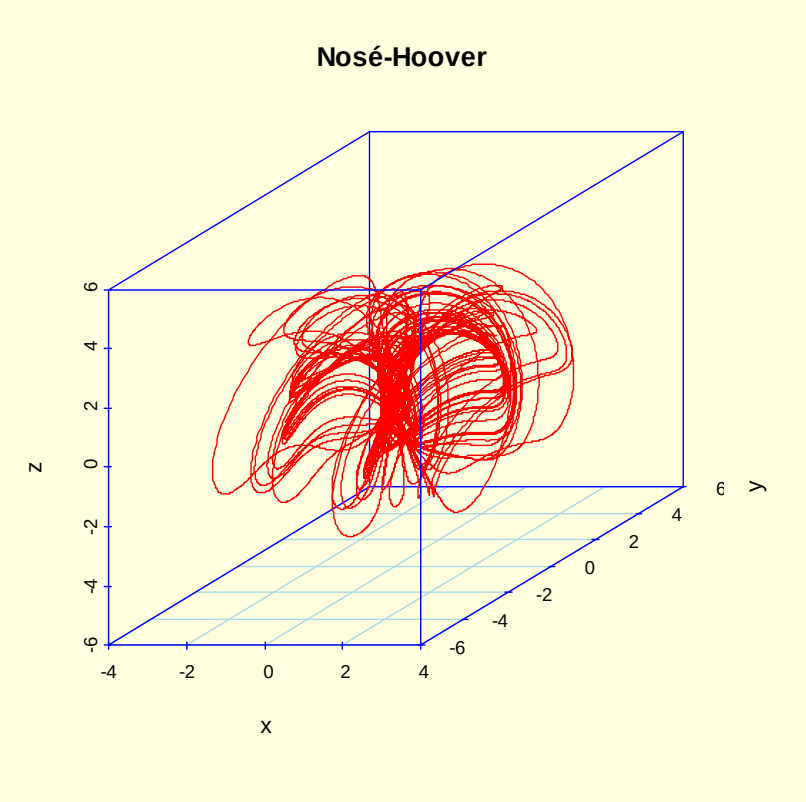


Figur. Fasediagram for Nosé-Hoover oscillator med *a*=1 og initialverdier (0,5,0), t=0-500

3D Nosé-Hoover:

```
#Halvor Aarnes
library(deSolve)
params<-c(a=1)
nosehoover<-function(t,y,p)
{
dxdt<-y[2]
```

```
dydt<--y[1]+y[2]*y[3]
dzdt<-p["a"]-y[2]^2
list(c(dxdt,dydt,dzdt))
}
require(deSolve)
out.time<-seq(0,500,0.01)
init.state<-c(0,5,0)
out.state<-
lsoda(init.state,out.time,nosehoover,params,rtol=1e-4)
par(bg="lightyellow")
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l", xlab="x",ylab="y",zlab="z",main="Nosé-
Hoover",color="red",col.axis="blue",
      col.grid="lightblue",)
```



Figur. 3D-fasediagram for Nosé-Hoover oscillator med *a*=1 og initialverdier (0,5,0), t=0-500

Rucklidge

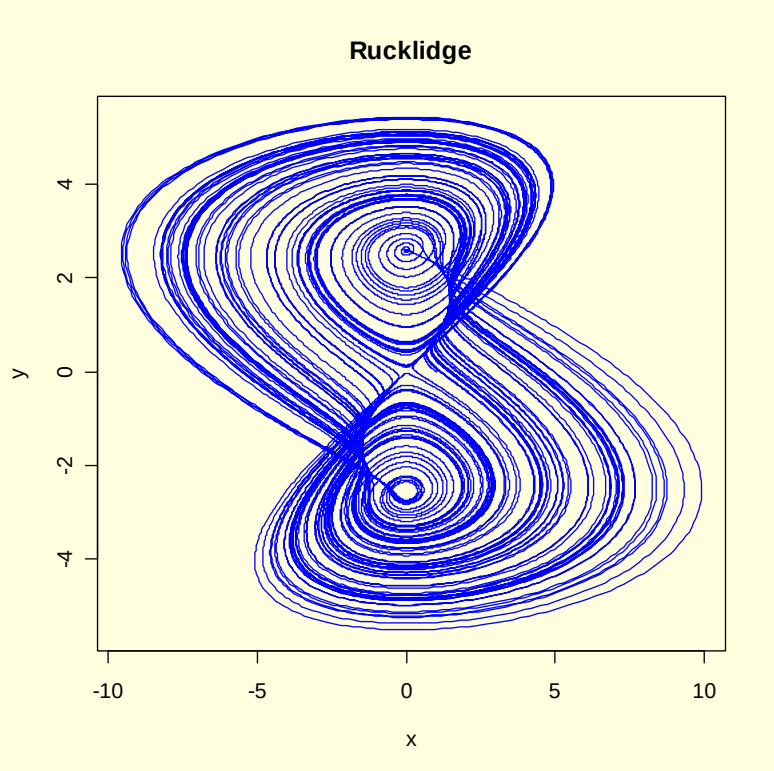
$$\frac{dx}{dt} = -k * x + \lambda * y - y * z$$

$$\frac{dy}{dt} = x$$

$$\frac{dz}{dt} = -z + y^2$$

Rucklidge, A.M. *Chaos in models of double convection*. Journal of Fluid Mechanics 237(1992)209-229.

```
#Halvor Aarnes
library(deSolve)
params<-c(k=2,l=6.7)
rucklidge<-function(t,y,p)
{
dxdt<--p["k"]*y[1]+p["l"]*y[2]-y[2]*y[3]
dydt<-y[1]
dzdt<--y[3]+y[2]^2
list(c(dxdt,dydt,dzdt))
}
require(deSolve)
out.time<-seq(0,500,0.01)
init.state<-c(1,0,4.5)
out.state<-lsoda(init.state,out.time,rucklidge,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",col="blue",main="Rucklidge")
```

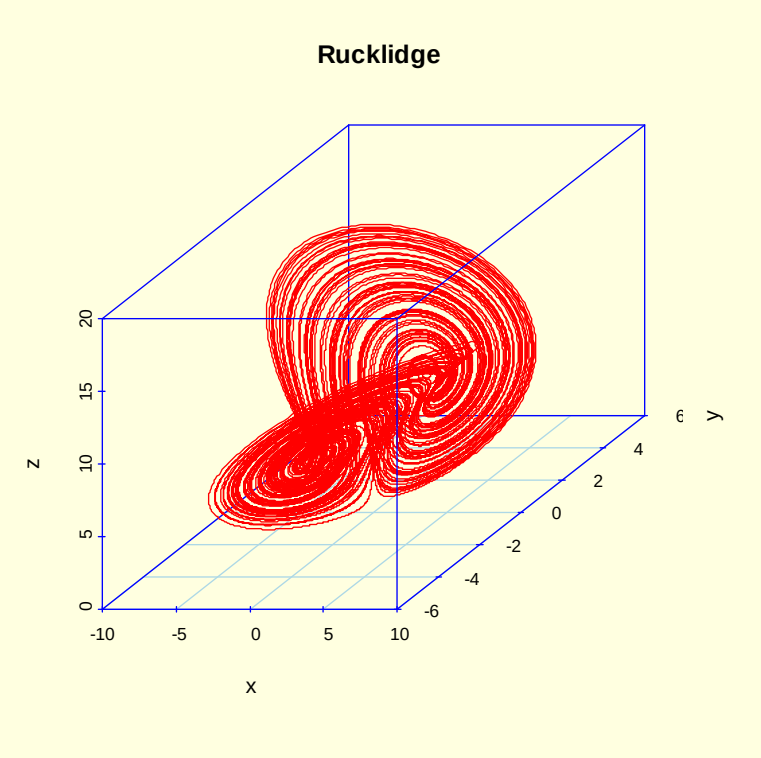


Figur. Fasediagram for Rucklidge med k=2, λ=6.7, initialverdier (1,0,4.5) og t=0-500

3D Rucklidge erstatt plot med:

```
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="l",
```

```
xlab="x",ylab="y",zlab="z",main="Rucklidge",color="red",col.axis="blue",
      col.grid="lightblue",)
```



Figur. Fasediagram for Rucklidge med k=2, λ=6.7, initialverdier (1,0,4.5) og t=0-500

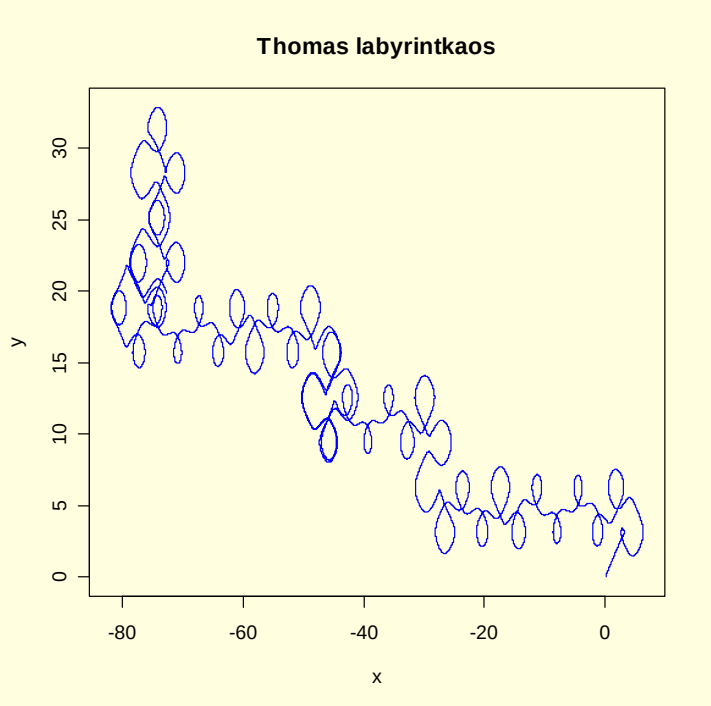
Thomas labyrintkaos

$$\frac{dx}{dt} = \sin y$$
$$\frac{dx}{dt} = \sin z$$
$$\frac{dx}{dt} = \sin x$$

Thomas, R. *Deterministic chaos seen in terms of feedback circuits: analysis, synthesis, "labyrint chaos"*. Journal of Bifurcation and Chaos 9 (1999)1889-1905

```
#Halvor Aarnes
library(deSolve)
params<-c(k=2,l=6.7)
thomas<-function(t,y,p)
{
dxdt<-sin(y[2])
dydt<-sin(y[3])
dzdt<-sin(y[1])
list(c(dxdt,dydt,dzdt))
}
```

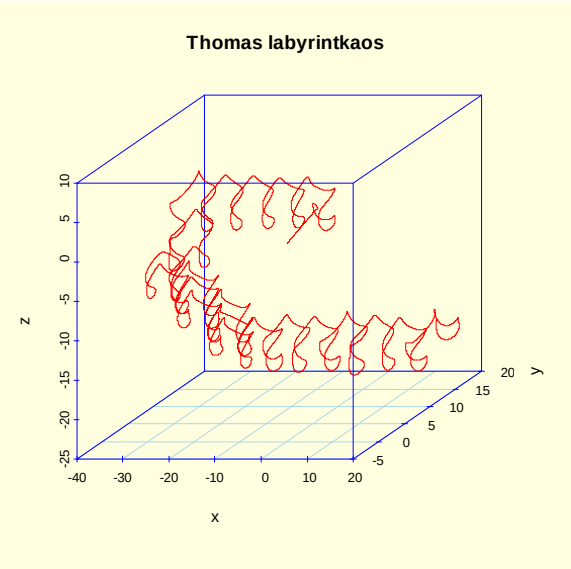
```
require(deSolve)
out.time<-seq(0,500,0.01)
init.state<-c(0.1,0,0)
out.state<-lsoda(init.state,out.time,thomas,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",co
l="blue",main="Thomas labyrintkaos")
```



Figur. Fasediagram for Thomas labyrintkaos med initialverider (0.1,0,0) og t=0-500.

3D Thomas labyrintkaos, hvor plot er erstattet med:

```
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l", xlab="x",ylab="y",zlab="z",main="Thomas
labyrintkaos",color="red",col.axis="blue",
      col.grid="lightblue",)
```



Figur. 3D fasediagram for Thomas labyrintkaos med initialverider (0.1,0,0) og t=0-500.

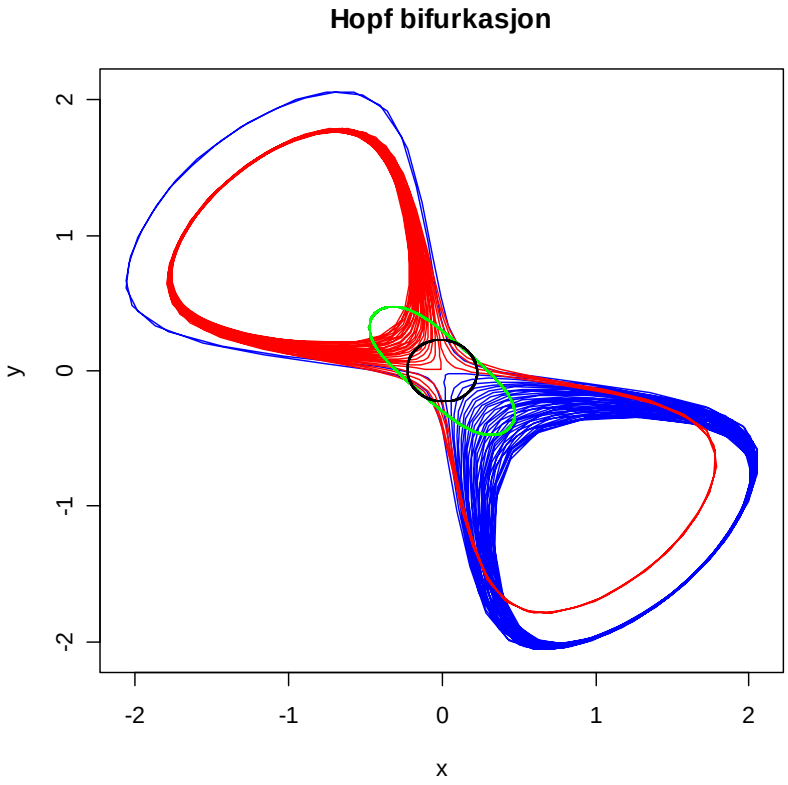
Hopf bifurkasjon

$$\frac{dx}{dt} = -y + x[\mu - (x^2 + y^2)]$$

$$\frac{dy}{dt} = x + y[\mu - (x^2 + y^2)$$

Hopf, E. *A mathematical example displaying the features of turbulence*. Communications of Pure and Applied Mathematics 1 (1948) 303-322.

```
#Halvor Aarnes
library(deSolve)
#Parameterverdier (params)
params<-c(m=5)
hopf<-function(t,x,p)
{
dxdt<--x[2]-x[1]*(p["m"]-(x[1]^2+x[2]^2))
dydt<-x[1]+x[2]*(p["m"]-(x[1]^2+x[2]^2))
list(c(dxdt,dydt))
}
require(deSolve)
out.time<-seq(0,100,0.1)
init.state<-c(0.1,0.2)
out.state<-lsoda(init.state,out.time,hopf,params,rtol=1e-4)
plot(out.state[,2],out.state[,3],type="n",ylab="y",xlab="x",ma
in="Hopf bifurkasjon")
lines(out.state[,2],out.state[,3],col="blue")
```



Figur. Fasediagram Hopf bifurkasjon med initialverdier (0.1,0.2) og μ=5 (blå),μ=4 (rød) og μ=1 (grønn), μ=0.1 (svart), t=0-100

van der Pol

$$\frac{dx}{dt} = y$$

$$\frac{dy}{dt} = b(1 - x^2)y - x$$

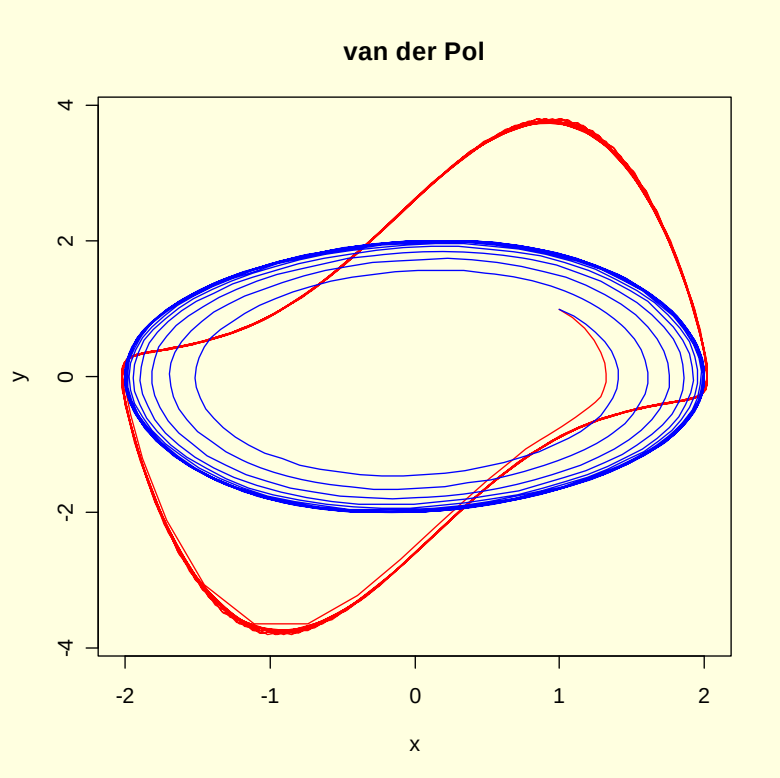
van der Pol ligningen har en begrenset syklus for hver verdi av b >0. Har vært brukt i modellering av hjerteslag, solflekksyklus og pulserende stjerner (Cepheidene). Ligningene er det samme som:

$$\frac{d^2x}{dt^2} + b(x^2 - 1)\frac{dx}{dt} + x = 0$$

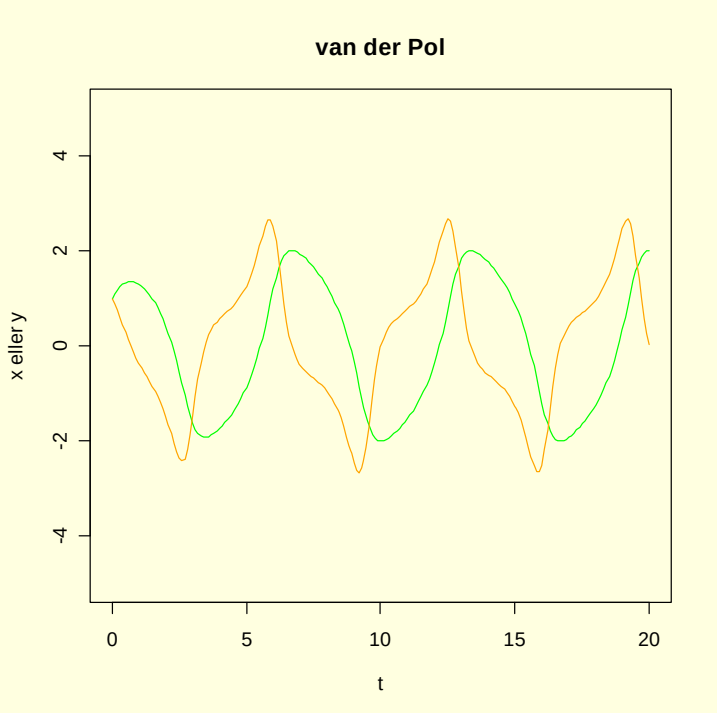
Van der Pol, B. *On relaxation oscillations*. Philosophical Magazine 2 (1926) 978-992

```
#Halvor Aarnes
library(deSolve)
params<-c(b=0.1)
```

```
vanderpol<-function(t,x,p)
{
dxdt<-x[2]
dydt<-p["b"]*x[2] - p["b"]*x[2]*x[1]^2-x[1]
list(c(dxdt,dydt))
}
require(deSolve)
out.time<-seq(0,100,0.1)
init.state<-c(1,1)
out.state<-lsoda(init.state,out.time,vanderpol,params,rtol=1e-
4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],type="n",ylab="y",xlab="x",ma
in="van der Pol")
lines(out.state[,2],out.state[,3],col="blue")
```



Figur. van der Pol ligningen er en begrenset syklus som avhenger av verdien for b: b=2 (rød), b=0.1 (blå), initialverdier (1,1).Sirkulært fasetrajektorium for b<1. Ved større b er trajektoriene to ganger utenfor sirkelen per syklus.



Figur van der Pol tidsseriediagram b=1, initialbetingelser (1,1) x(grønn), y (oransje)

Hénon-Heiles

$$\frac{dx}{dt} = v$$

$$\frac{dy}{dt} = w$$

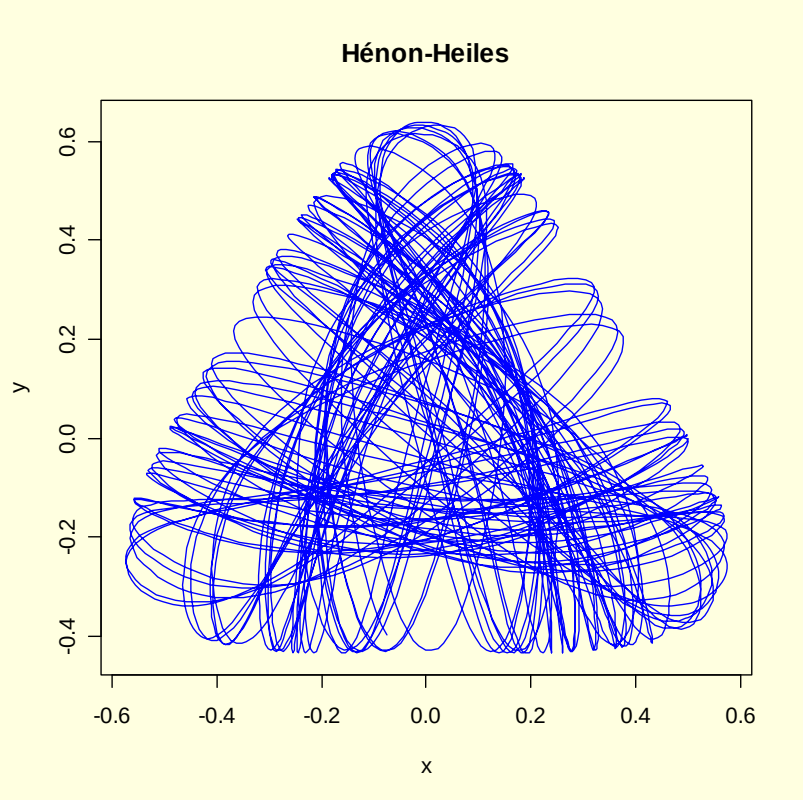
$$\frac{dv}{dt} = -x - 2xy$$

$$\frac{dw}{dt} = -y - x^2 + y^2$$

Hénon, M. & Heiles, C. *The applicability of the third integral of motion: some numerical experiments*. Astrophysical Journal 69 (1964)73-79

```
#Halvor Aarnes
library(deSolve)
params<-
henonheiles<-function(t,x,p)
{
dxdt<-x[3]
dydt<-x[4]
dvdt<--x[1]-2*x[1]*x[2]
dwdt<--x[2]-x[1]^2+x[2]^2
list(c(dxdt,dydt,dvdt,dwdt))
}
```

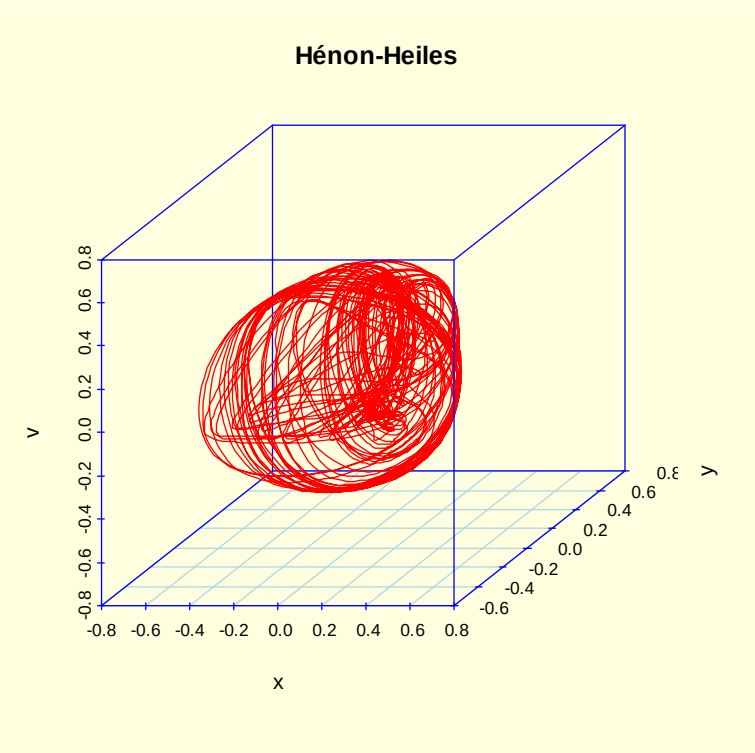
```
require(deSolve)
out.time<-seq(0,500,0.1)
init.state<-c(0.499,0,0,-0.045)
out.state<-
lsoda(init.state,out.time,henonheiles,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],type="n",ylab="y",xlab="x",ma
in="Hénon-Heiles")
lines(out.state[,2],out.state[,3],col="blue")
```



Figur. Fasediagram Hénon-Heiles med initialverdier c(0.499,0,0,-0.045)

3D: Istedet for plot bruk:

```
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l", xlab="x",ylab="y",zlab="v",main="Hénon-
Heiles",color="red",col.axis="blue",
      col.grid="lightblue",)
```



Figur. 3D fasediagram Hénon-Heiles for x, y og v med initialverdier c(0.499,0,0,-0.045)

Chen og Ueta

$$\frac{dx}{dt} = a \cdot (y - x)$$

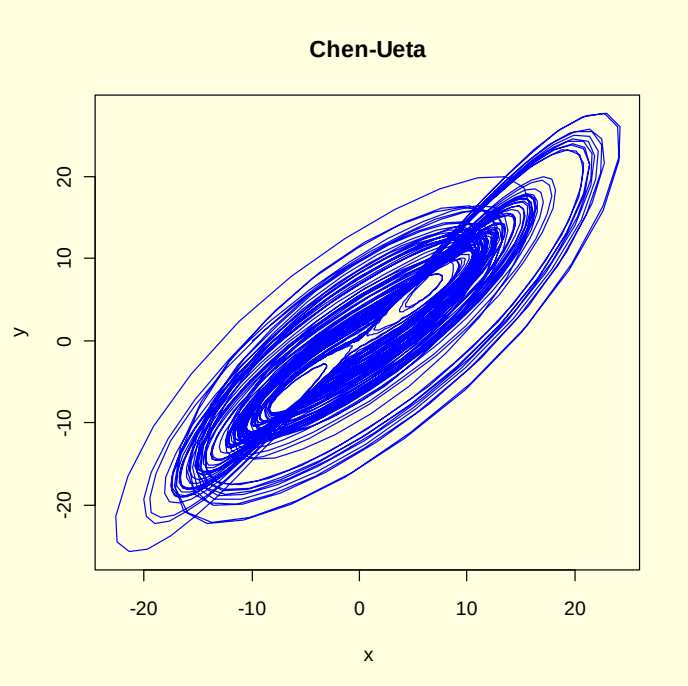
$$\frac{dy}{dt} = (c - a) \cdot x - x \cdot z + c \cdot y$$

$$\frac{dz}{dt} = x \cdot y - b \cdot z$$

Chen, G. & Ueta,T. *Yet another chaotic attractor.* International Journal of Bifurcation and Chaos 9 (1999)1465-1466

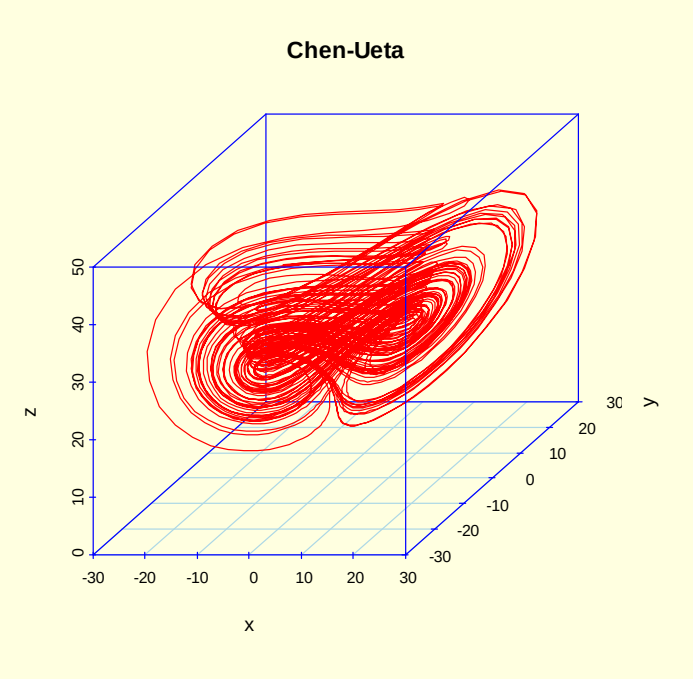
```
#Halvor Aarnes
library(deSolve)
params<-c(a=35,b=3,c=28)
chen<-function(t,y,p)
{
dxdt<- p["a"]*(y[2]-y[1])
dydt<-(p["c"]-p["a"])*y[1]-y[1]*y[3]+p["c"]*y[2]
dzdt<-y[1]*y[2]-p["b"]*y[3]
list(c(dxdt,dydt,dzdt))
}
require(deSolve)
out.time<-seq(0,50,0.01)
init.state<-c(-10,0,37)
```

```
out.state<-lsoda(init.state,out.time,chen,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",co
l="blue",main="Chen-Ueta")
```



Figur. Fasediagram Chen-Ueta med parametere a=35, b=3, c=28, intialverdier (-10,0,37) og t=0-50 i intervall 0.01

```
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l", xlab="x",ylab="y",zlab="z",main="Chen-
Ueta",color="red",col.axis="blue",
      col.grid="lightblue",)
```



Figur. 3D fasediagram Chen-Ueta med parametere a=35, b=3, c=28, intialverdier (-10,0,37) og t=0-50 i intervall 0.01

Thomas

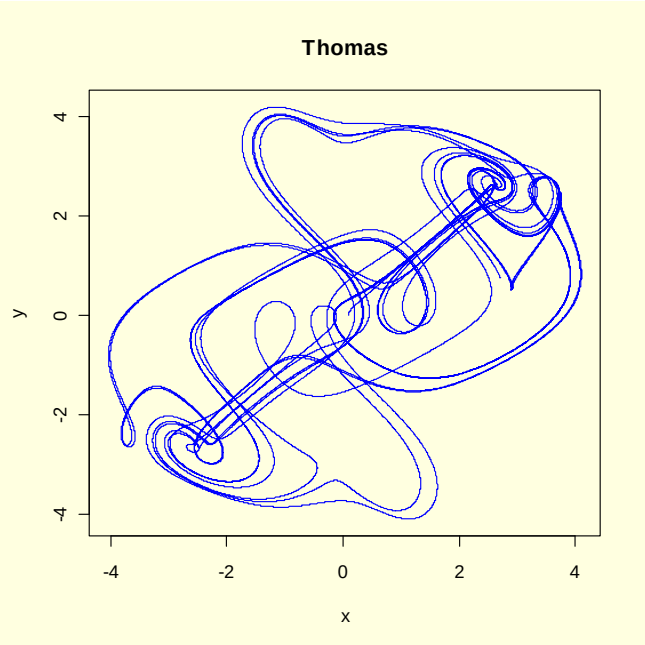
$$\frac{dx}{dt} = -b \cdot x + \sin y$$

$$\frac{dy}{dt} = -b \cdot y + \sin z$$

$$\frac{dz}{dt} = -b \cdot z + \sin x$$

Thomas, R. *Deterministic chaos seen in terms of feedback circuits: analysis, synthesis, "labyrinth chaos"*. International Journal of Bifurcation and Chaos 9 (1999)1889-1905

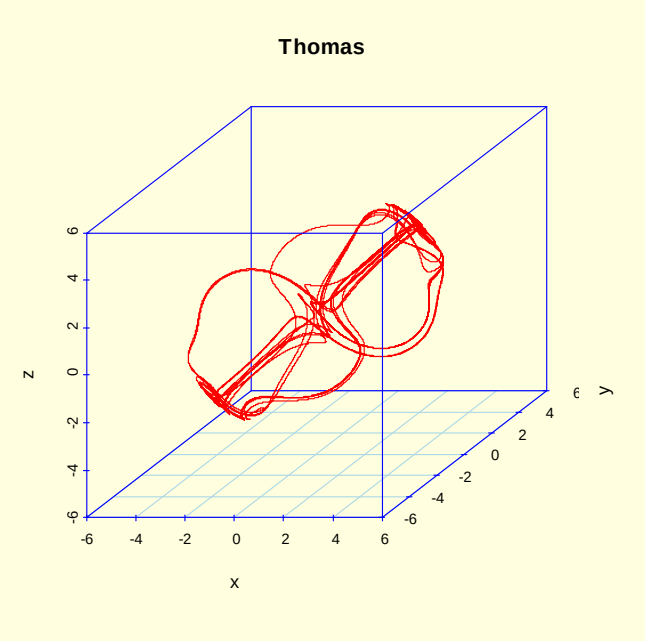
```
#Halvor Aarnes
library(deSolve)
params<-c(b=0.18)
thomas<-function(t,y,p)
{
dxdt<- -p["b"]*y[1] + sin(y[2])
dydt<- -p["b"]*y[2]+ sin(y[3])
dzdt<- -p["b"]*y[3] + sin(y[1])
list(c(dxdt,dydt,dzdt))
}
require(deSolve)
out.time<-seq(0,500,0.01)
init.state<-c(0.1,0,0)
out.state<-lsoda(init.state,out.time,thomas,params,rtol=1e-4)
par(bg="lightyellow")
plot(out.state[,2],out.state[,3],xlab="x",ylab="y",type="l",co
l="blue",main="Thomas")
```



Figur. Fasediagram Thomas syklisk symmetrisk tiltrekker med parameter b=0.18, initialverdier (0.1,0,0) og tid 0-500 i step 0.01.

3D-plot:

```
require(scatterplot3d)
scatterplot3d(out.state[,2],out.state[,3],out.state[,4],type="
l",
xlab="x",ylab="y",zlab="z",main="Thomas",color="red",col.axis=
"blue", col.grid="lightblue",)
```



Figur. 3D fasediagram Thomas syklisk symmetrisk tiltrekker med parameter b=0.18, initialverdier (0.1,0,0) og tid 0-500 i step 0.01.

Stabilitet og likevektspunkter

Hvis vi har følgende generelle system av todimensjonale lineære differensialligninger med parameterverdier *a*, *b*, *c* og *d*.

$$\frac{dx}{dt} = ax + by$$
$$\frac{dy}{dt} = cx + dy$$

Uttrykt i matriseform blir disse:

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Koffisientmatrisen *A* (*2x2* matrise) blir:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Hvis vi har en *n x n* matrise *A* så vil sammenhengen mellom **egenverdien λ** og **egenvektoren ν** være følgende:

$$Av = \lambda v$$

Som kan omskrives til:

$$(A - \lambda I)v = 0$$

hvor *I* er identitetsmatrisen.
Som gir:

$$A - \lambda I = \begin{pmatrix} a - \lambda & b \\ c & d - \lambda \end{pmatrix}$$

Sammenhengen mellom matrisen A, den **inverse matrisen** A^{-1} og identitetsmatrisen *I* er:

$$A \cdot A^{-1} = I$$

Egenverdiene til en matrise A er gitt ved **den karakteristiske ligningen**:

$$det(A - \lambda I)v = 0$$

Vi har videre **determinanten** til *A*:

$$detA = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Og derfor blir:

$$\det(A - \lambda I) = (a - \lambda)(d - \lambda) - bc = \lambda^2 - (a + d)\lambda + (ad - bc)$$

Determinanten til en enhetsmatrise er lik 1.
Hvis *t* er et tall så er:

$$det \begin{vmatrix} a & tb \\ c & td \end{vmatrix} = t \cdot det \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

Hvis to kolonner bytter plass så skifter fortegnet på determinanten:

$$det\begin{vmatrix} a & b \\ c & d \end{vmatrix} = -det\begin{vmatrix} b & a \\ d & c \end{vmatrix}$$

To vektorer:

$$\binom{a}{c} og \binom{b}{d}$$

er lineært avhengige hvis og bare hvis ad-bc=0

Trace til A (tr(A)) er summen av alle diagonalelementene i matrisen:

$$tr(A) = a + d$$

Vi har det karakteristiske polynomet til matrisen *A* ($P_A(\lambda)$):

$$P_A(\lambda) = det(A - \lambda I)$$

og generelt vil matrisen til polynomet bli:

$$P_A(\lambda) = \lambda^2 - tr(A)\lambda + det(A)$$

Egenverdien λ til matrisen *A* er roten til et polynom til matrisen *A*. Vi kaller røttene egenverdiene λ_1 og λ_2:

$$P_A(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) = \lambda^2 - (\lambda_1 + \lambda_2)\lambda + \lambda_1 \cdot \lambda_2$$

Derved kan egenverdiene λ_1 og λ_2 til matrisen *A* uttrykkes i form av determinanten og trace til *A*:

$$det(A) = \lambda_1 \cdot \lambda_2$$
$$tr(A) = \lambda_1 + \lambda_2$$

Vi kan beregne **diskriminanten** *D* til en matrise *A*:

$$D = [tr(A)]^2 - 4 \cdot \det(A)$$

Hvis determinanten *D*=0 så er egenverdiene til *A* relle og like. Er *D*>0 er egenverdiene relle og forskjellige. Er *D*<0 er egenverdiene komplekse tall i konjugate par.
Vi kan finne røttene (egenverdiene λ_1 og λ_2) ved å bruke:

$$P_A(\lambda) = \lambda^2 - tr(A)\lambda + det(A)$$

Som har løsningene for λ_1 og λ_2:

$$\frac{1}{2}\left(tr(A) \pm \sqrt{[tr(A)]^2 - 4 \cdot det(A)}\right) = \frac{tr(A) \pm \sqrt{D}}{2}$$

Hvis vi har funnet egenverdiene λ_1 og λ_2 kan vi finne egenvektorene ν_1 og ν_2 som hører sammen med disse egenverdiene ved hjelp av:

$$(A - \lambda I)\nu = 0$$

Ut fra om egenverdiene er reelle eller komplekse tall og fortegnene på egenverdiene så kan man si noe om stabiliteten ved likevektspunktene. Hyperbolsk eller ikke-hyperbolsk stabilitet med spiraler (fokus), radialpunkter (node) eller sadelpunkter. Vi har maksimum eller minimum når den deriverte dx/dt=0. Når den deriverte er >0 er det en stigende kurve og når den deriverte er <0 er det en synkende kurve. Ved vendepunkter med vendetangenter er den andrederiverte lik 0. Når den andrederiverte er >0 er det et minimum og når den andrederiverte er < 0 er det et maksimum.
For n=2 har vi *e.g.*:
To negative egenverdier gir stabil node.

En positiv og en negativ egenverdi gir et sadelpunkt
To positive egenverdier gir en ustabil node.
Hvis egenverdiene er komplekse tall (imaginære tall) vil det bli et fiksert tall som senter eller spiral. Komplekse tall kan vises grafisk i et Argand diagram (Wessel-diagram) hvor den horisontale x-aksen er den reelle delen av λ og y-aksen er den imaginære delen.

I et system med to variable *e.g.* Lotka-Volterra kan vi finne type likevekt ved å se på fortegnet og verdi til trace og determinant til Jacobimatrisen.

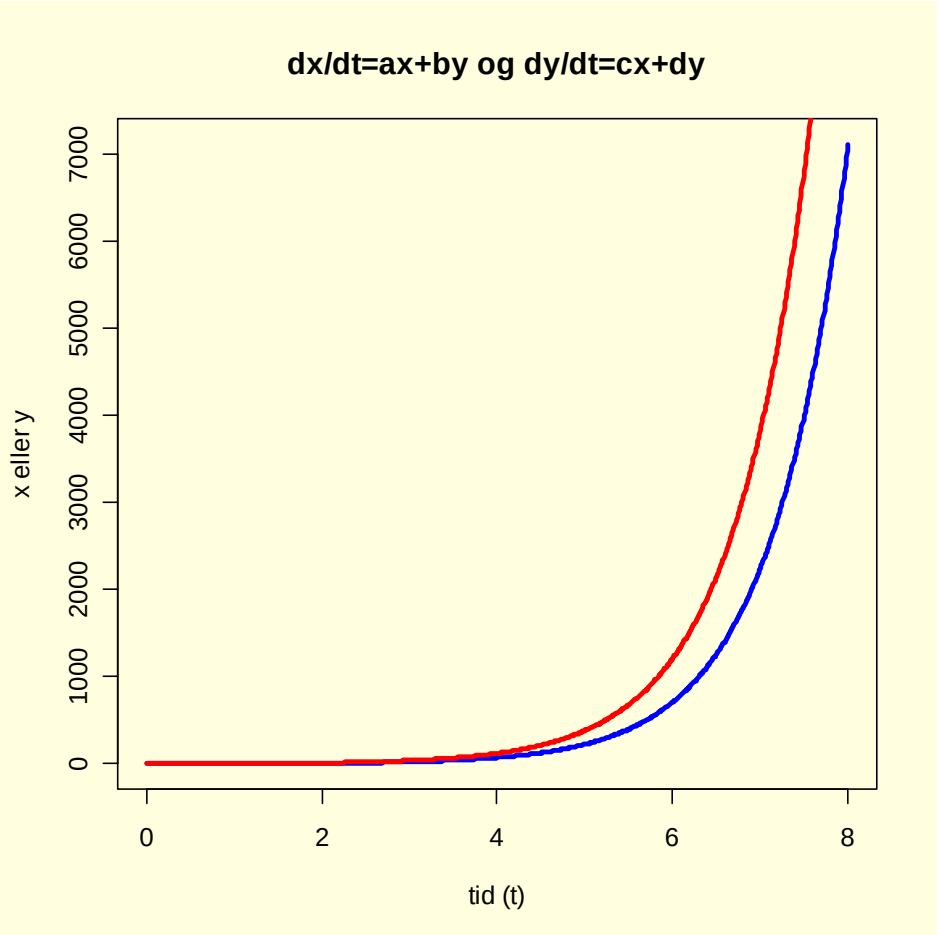
Vi har to differensialligninger:

$$\frac{dx}{dt} = ax + by$$

$$\frac{dy}{dt} = cx + dy$$

En numerisk løsning av disse indikerer at løsningen er to eksponentialfunksjoner:

```
#Eksempel1
params<-c(a=0.3,b=0.5,c=0.6,d=0.8)
eks1<-function(t,x,p)
{
dxdt<-p["a"]*x[1]+p["b"]*x[2]
dydt<-p["c"]*x[1]+p["d"]*x[2]
list(c(dxdt,dydt))
}
require(deSolve)
out.time<-seq(0,8,0.01)
init.state<-c(1,1)
out.state<-
as.data.frame(lsoda(init.state,out.time,eks1,params,rtol=1e-
4))
par(bg="lightyellow")
plot(out.state[,1],out.state[,2],type="n",xlab="tid
(t)",ylab="x eller y",
main="dx/dt=ax+by og dy/dt=cx+dy")
lines(out.state[,1],out.state[,2],col="blue",lwd=3)
lines(out.state[,1],out.state[,3],col="red",lwd=3)
```



Parameter og initialverdier (params<-c(a=0.3,b=0.5,c=0.6,d=0.8), init.state<-c(1,1)), x er blå og y er rød.

Vi kan finne egenverdiene og egenvektorene til matrise *A* med kommandoen **eigen:**

```
 A<-matrix(c(0.3,0.6,0.5,0.8),nrow=2);A
     [,1] [,2]
[1,]  0.3  0.5
[2,]  0.6  0.8
 eigen(A)
$values
[1]  1.15207973 -0.05207973

$vectors
           [,1]       [,2]
[1,] -0.5060999 -0.8176310
[2,] -0.8624749  0.5757426
```

Vi kan finne **trace** til en matrise, hvor **diag** bestemmer diagonalelementene i matrisen:

```
trA<-sum(diag(A));trA
[1] 1.1
```

Determinanten til A bestemmes med kommandoen **det(A):**

```
det(A)
[1] -0.06
```

En matrise kan **inverteres** med kommandoen solve:

```
B<-solve(A);B
          [,1]      [,2]
[1,] -13.33333  8.333333
[2,]  10.00000 -5.000000
```

Vi kan også bruke pakken **rootSolve** for å finne egenverdier:

```
require(rootSolve)
C<-jacobian.full(y=c(x=0,y=0),func=eks1,parms=c(a=0.3,b=0.5,c=0.6,d=0.8));C
     x   y
[1,] 0.3 0.5
[2,] 0.6 0.8
D<-
eigen(jacobian.full(y=c(x=0,y=0),func=eks1,parms=c(a=0.3,b=0.5,c=0.6,d=0.8)
));D
$values
[1]  1.15207973 -0.05207973
$vectors
           [,1]       [,2]
[1,] -0.5060999 -0.8176310
[2,] -0.8624749  0.5757426
```

Derivasjon og integrasjon

Se R-manualen

```
?deriv
?integrate
```

Derivasjon

Vi kan bruke R til å derivere funksjoner.

Den deriverte av x^3 som blir lik $3x^2$:

```
 D(expression(x^3),"x")
3 * x^2
```

Den deriverte av sin(x)som blir lik cos(x):

```
 D(expression(sin(x)),"x")
cos(x)
```

Den deriverte av cos(x)som blir lik –sin(x):

```
 D(expression(cos(x)),"x")
-sin(x)
```

Den deriverte av ln(x) som blir lik 1/x (husk at i R betyr **log** ln og **log10** er log med grunntall 10):

```
 D(expression(log(x)),"x")
1/x
```

Den deriverte av e^x som blir lik seg selv:

```
 D(expression(exp(x)),"x")
exp(x)
```

Den deriverte av tan(x):

```
 D(expression(tan(x)),"x")
1/cos(x)^2
```

Den deriverte av a^x som blir lik $a^x \cdot \ln(a)$:

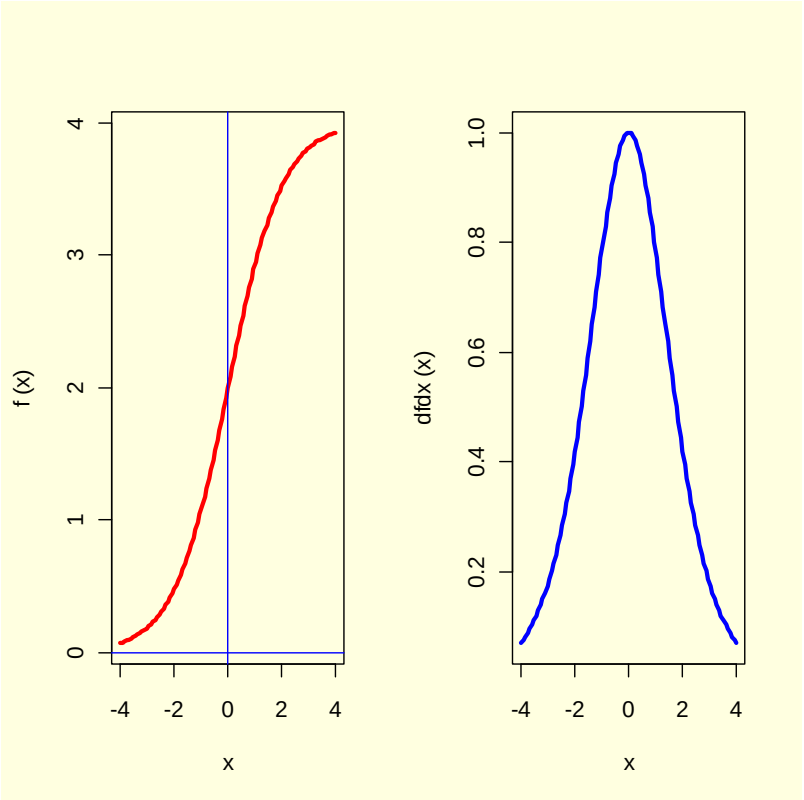
```
 D(expression(a^x),"x")
a^x * log(a)
```

Vi har for eksempel en funksjon:

$$f(x) = \frac{4 \cdot e^x}{e^x + 1}$$

Vi bruker programmeringsspråket R til å lage en funksjon, plotter funksjonen og skriver inn x- og y-akse:

```
f<-function (x){4*exp(x)/(exp(x)+1)}
par(mfrow=c(1,2),bg="lightyellow")
plot(f,-4,4,col="red",lwd=3)
abline(h=0,v=0,col="blue")
D(expression((4*exp(x))/(exp(x)+1)),"x")
4 * exp(x)/(exp(x) + 1) - (4 * exp(x)) * exp(x)/(exp(x) + 1)^2
dfdx<- function (x)4 * exp(x)/(exp(x) + 1) - (4 * exp(x) * exp(x))/(exp(x)
+ 1)^2
x<-seq(-4,4,0.1)
plot(dfdx,x,col="blue",lwd=3,xlim=c(-4,4))
```



Figur. Vi ser at den deriverte skifter fortegn ved x=0 og vi har et vendepunkt

Den dobbeltderiverte eller høyere ordens deriverte via kommandoen **DD** (se R-manualen)

```
DD <- function(expr,name, order = 1) {
   if(order < 1) stop("'order' must be >= 1")
   if(order == 1) D(expr,name)
   else DD(D(expr, name), name, order - 1)
```

Den andre deriverte:

```
DD(expression((4*exp(x))/(exp(x)+1)), "x", 2)
4 * exp(x)/(exp(x) + 1) - 4 * exp(x) * exp(x)/(exp(x) + 1)^2 -
    ((4 * exp(x) * exp(x) + (4 * exp(x)) * exp(x))/(exp(x) +
        1)^2 - (4 * exp(x)) * exp(x) * (2 * (exp(x) * (exp(x) +
        1)))/((exp(x) + 1)^2)^2)
```

Vi kan også gjøre følgende:

```
dfdx<- deriv(~4*exp(x)/(exp(x)+1),"x");dfdx
expression({
    .expr1 <- exp(x)
    .expr2 <- 4 * .expr1
    .expr3 <- .expr1 + 1
    .expr4 <- .expr2/.expr3
    .value <- .expr4
    .grad <- array(0, c(length(.value), 1L), list(NULL, c("x")))
    .grad[, "x"] <- .expr4 - .expr2 * .expr1/.expr3^2
    attr(.value, "gradient") <- .grad
    .value
})
```

Integrasjon

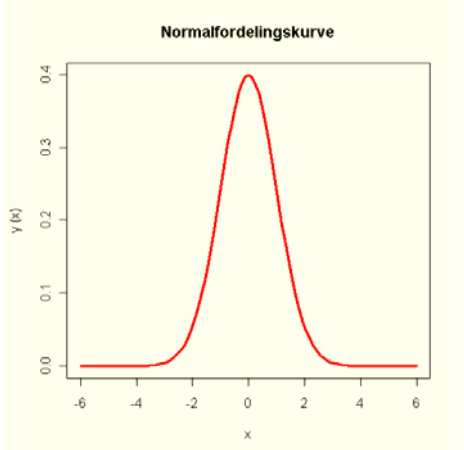
Vi kan integrere normalfordelingskurven og finne arealet under kurven fra -1.96 til 1.96 som tilsvarer 95% av arealet og middeltallet pluss minus ett standardavvik som tilsvarer ca. 68.3% av arealet under normalfordelingskurven:

```
 integrate(dnorm,-1.96,1.96)
0.9500042 with absolute error < 1.0e-11
 integrate(dnorm,-1,1)
0.6826895 with absolute error < 7.6e-15
```

Integrasjon av tetthetsfunksjoner

Vi skriver funksjonen for **normalfordelingskurven**, lager et plot og deretter integrerer. Vi ser at arealet under kurven er lik 1, en karakteristisk egenskap ved sannsynlighetstetthetskurver

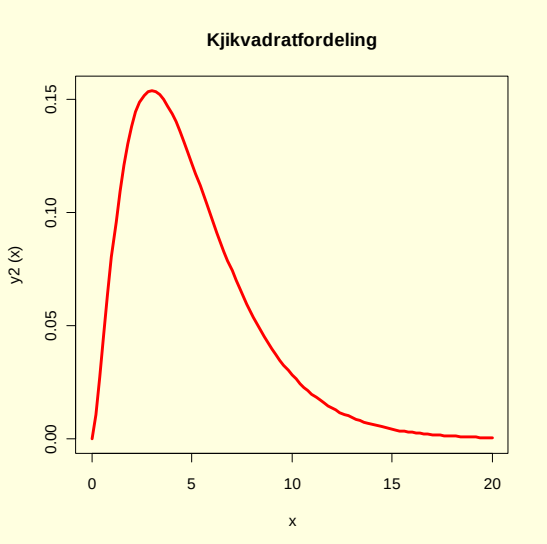
```
sigma<-1
my<-0
y<-function (x) {return (1/(sigma*sqrt(2*pi))*exp(-(x-
my)^2/(2*sigma^2)))}
par(bg="lightyellow")
plot(y,-6,6,col="red",lwd=3,main="Normalfordelingskurve")
integrate(y,-10,10)
1 with absolute error < 7.4e-05
```



Figur. Normalfordelingskurve med μ=0 og σ=1. Integralet er lik 1.

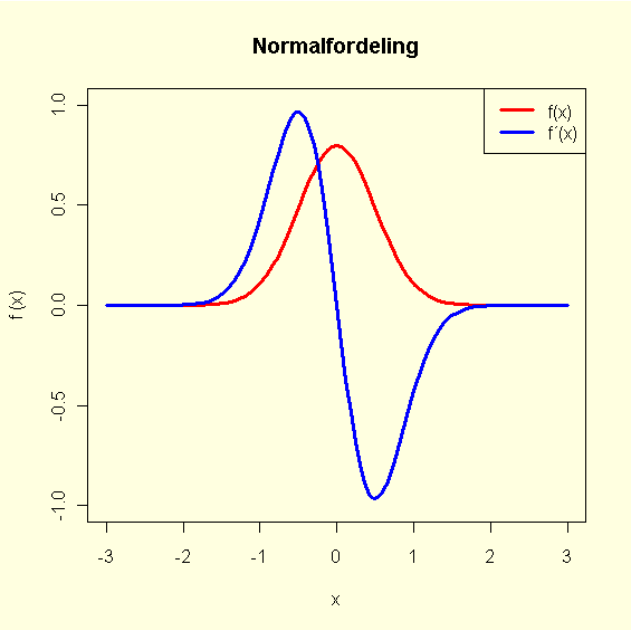
Vi gjør det samme for kjikvadratfordelingen hvor gammafunksjonen (Γ)inngår. Vi ser at også her blir integralet lik 1.

```
n<-5
y2<-function (x){return(((1/2)^(n/2))/(gamma(n/2))*x^(n/2-
1)*exp(-x/2))}
par(bg="lightyellow")
plot(y2,0,20,col="red",lwd=3,main="Kjikvadratfordeling")
integrate(y2,0,100)
1 with absolute error < 1.5e-05
```

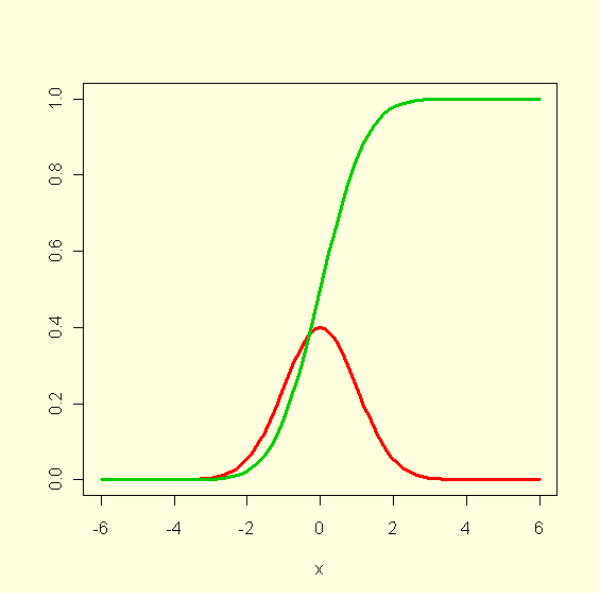


Figur. Kjikvadratfordeling med n=5 og areal=1

```
#Normalfordelingsfunksjon og den deriverte
sigma<-0.5
my<-0
f<-function (x) {(1/(sigma*sqrt(2*pi))*(exp(-1/2*((x-
my)/sigma)^2)))}
par(bg="lightyellow")
plot(f,-3,3,lwd=3,col=2,
ylim=c(-1,1),main="Normalfordeling")
#den deriverte
derivert<-D(expression((1/(sigma*sqrt(2*pi))*(exp(-1/2*((x-
my)/sigma)^2)))),"x")
#Lager en funksjon av den deriverte
fderivert<-function(x) {eval(derivert)}
#plotter funksjonen for den førstederiverte
curve(fderivert,-3,3,col=4,lwd=3,add=T)
#Setter inn hjelpetekst
legend("topright",c("f(x)","f´(x)"),lty=c(1,1),lwd=c(3,3),col=
c(2,4))
```




```
h<-function (x) {dnorm(x)}
k<-function (x) {pnorm(x)}
par(bg="lightyellow")
curve(h,-6,6,col=2,lwd=3,ylim=c(0,1),ylab="")
curve(k,-6,6,col=3,lwd=3,add=T)
```



Sannsynlighetstetthetsfunksjonen dnorm viser ikke sannsynligheten direkte, men det gjør det akkumulerte arealet via pnorm.

Topologi

Dei gamle fjell i syningom er alltid eins å sjå
Ivar Aasen

Topologi er studiet av posisjon. En oversikt over T-banenettet i Oslo er ikke geografisk riktig, men nettverket og mønsteret er riktig. Hvis vi tegner et nettverk av punkter (P) og linjer

(L) som danner flater (F)med trekanter så viste Euler at det var følgende sammenheng mellom antallet (Eulerkarakteristikk):

$$P - K + F = 1$$

I et tredimensjonalt nettverk kan nettverkstrådene gå over eller under hverandre, men det er ikke mulig i et todimensjonalt hvor de må krysse hverandre.

Hvis vi lager nettverket på en kule, for eksempel på en appelsin blir sammenhengen:

$$P - K + F = 2$$

For en torus er eulerkarakteristikken:

$$P - K + F = 0$$

En kule og en torus har ingen kanter og er en lukket overflate. En sylinder har to kanter. Et Möbiusbånd har bare en kant og er en flate som ser ut som bare en side. (Augustus.F. Möbius). To Möbiusbånd limt sammen kant mot kant gir en Kleinflaske. Knuteteori og manifoldteori er en del av topologien. Objekter kan bli transformert ved bøyning, vridning, strekning og sammenpressing.

Eulers polyederformel:

$$f + h = k + 2$$

hvor f=flater, h=hjørner, k=kanter.
En kube har 6 flater, 12 kanter og 8 hjørner.

Polygoner kan klassifiseres etter hvor mange kanter og flater, lengden av kanter og vinkelen mellom dem.

En fotball består av 12 femkanter og 20 sekskanter.
Knutetopologi. En knute er en lukket løkke.

Møbiusbånd. M.C. Escher

Escher, M.C. & Locher, J.L.: *The infinite world of M.C. Escher*. Abradale Press 1984.

Litteratur:

Crawley, M.J.: *Statistics. An introduction using R*. John Wiley & Sons Ltd. 2005.

Crawley, M.J.: *Statistical computing. An introduction to data analysis using S-Plus*. John Wiley & Sons Ltd. 2003

Crawley, M.J: *The R book*. John Wiley & Sons,Ltd. 2007.

Dalgaard, P.: *Introductory Statistics with R*. Springer 2002.

Krause, A.& Olson, M.: *The Basics of S-PLUS*. 3/E. Springer-Verlag, New York,Inc. 2002

Soetaert, K. & Herman, P.M. *A practical guide to ecological modelling. Using R as a simulation platform*. Springer 2009.

Sprott, J.C.: *Chaos and time-series analysis*. Oxford university press 2003.

Strogatz, S.H.: *Nonlinear dynamics and chaos*. Perseus Books publ. 1994.

Venables, W.N. & Ripley, B.D.: *Modern applied statistics with S*. Springer Verlag 2002

Verzani, J.: *Using R for introductory statistics*. Chapman&Hall/CRC 2005

WEB-adresser:

http://www.r-project.org/ (The R Project for Statistical Computing)

http://stat.ethz.ch/R-manual/R-patched/library/MASS/html/00Index.html (Main Package of Venables and Ripley's MASS)

Tom Lehrer: Mathematics
http://www.youtube.com/watch?v=v0XUqliEnPQ
Andrew Wiles løste Fermats siste sats
QED : Quod erat demonstrandum – herved skulle bevises

Fil:haa/rmanual5.docx