Notat 4 - ST1301

8. februar 2005

1 While- og repeat-løkker

Vi har tidligere sett på bruk av før-løkker. Slike løkker er hensiktsmessig å bruke når vi skal gjenta visse beregninger (løkke-kroppen) et antall ganger og når antallet er kjent på forhånd. I en del tilfeller vil vi imidlertid ønske å stoppe løkken når en visse betingelse er oppfylt. Et eksempel er løsning av ikke-lineære ligninger ved hjelp av Newton's metode. La oss først se på den generelle virkemåten til en while-løkke.

En while-løkke er bygget opp på følgende måte

```
while (logisk uttrykk) {
  sammensatt uttrykk
}
```

Før hver utførelse av løkke-kroppen (et eller flere uttrykk mellom krøll-parantesene) vil det logiske uttrykket i parenteser beregnes. Hvis dette har verdi `FALSE` vil løkken avbrytes, hvis ikke utføres løkke-kroppen en gang til og det logiske uttrykt beregnes pånytt. La oss se på følgende eksempel.

```
i <- 1
while (i<100) {
   i <- i*2
   print(i)
}
```

Her initieres først verdien av variabelen `i` til 1. Ved første gangs beregning av det logiske uttrykket `i<100` har dette dermed verdi `TRUE` og løkke-kroppen utføres slik at `i` får verdien 2 som så skrives til skjerm ved med funksjonen `print`. Så gjentas det hele (det logiske uttrykket beregnes på nytt og løkke-kroppen utføres) helt til `i` blir ikke lenger er mindre 100. Dette inntreffer når `i` har fått verdien 128 etter sjuende gangs utførelse av løkke-kroppen. Da avbrytes løkken:

```
[1] 2
[1] 4
[1] 8
[1] 16
[1] 32
[1] 64
[1] 128
> i
[1] 128
```

Repeat-løkker er mer generelle og har følgende form

```
repeat uttrykk
```

hvor `uttrykk` som gjentas mange ganger inntil det skjer et kall til `break()`, typisk i en gren i en if-setning som vi bruker for å teste om løkken skal avbrytes.

Følgende repeat-løkke gjør det samme som while-løkken ovenfor.

```
i <- 1
repeat {
  if (i>=100)
    break()
  i <- i*2
  print(i)
}
```

Merk at if-setningen vi bruker for å avgjøre om løkka skal avbrytes kan plasseres hvor som helst i løkke-kroppen. Plasseres if-setningen helt til slutt i løkka vil løkke-kroppen dermed alltid utføres minst en gang og dette kan i en del tilfelle være hensiktsmessig.

Merk også at en repeat løkke avbrytes hvis det logiske uttrykket i if-setningen har verdi `TRUE`, mens det tilsvarende uttrykket i en while-løkke må være `FALSE` hvis løkka skal avbrytes.

2 Newton's metode

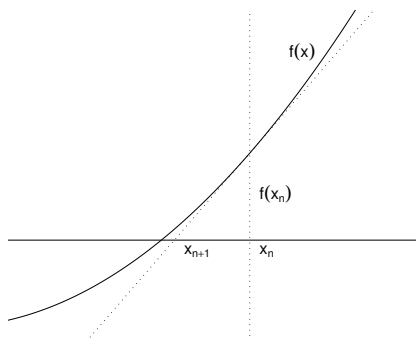
Algoritmen er beskrevet i detalj i Neuhauser (2004) kap. 5.7. Metoden er egnet til å finne røtter til ligninger på formen $f(x) = 0$ forutsatt at vi kjenner funksjonen f og den deriverte av f.

Anta at x_n befinner seg nære løsningen x^* av $f(x) = 0$. Ser vi på tangenten til funksjonen f gjennom punktet $(x_n, f(x_n))$ vil denne skjære første aksen gjennom et nytt punkt x_{n+1} som vil ligge nærmere x^* dersom f er tilnærmet lineær i området fra x^* til x_n (se figur 1). Ut i fra denne betrakningen har vi at forholdet mellom $f(x_n)$ og $x_n - x_{n+1}$ er lik stigningstallet til tangenten, $f'(x_n)$. Løst for x_{n+1} gir dette følgende algoritme:

1. Velg en passende x_0 i nærheten av løsningen.

2. Beregn
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{1}$$
for $n = 1, 2, \ldots$ inntil $|x_n - x_{n-1}| < \epsilon$.

Størrelsen ϵ bestemmer den numeriske nøyaktigheten på løsningen og velges lik f.eks. 10^8 hvis vi ønsker en løsning med 8 desimalers nøyaktighet. Andre stoppkriterier kan brukes avhengig av hva slags problem vi studerer.



Figur 1: Netwon's metode

2.1 Eksempel

Anta at vi ønsker å finne roten av et positivt tall a, altså løsningen av ligningen

$$\begin{aligned} x^2 &= a \\ x^2 - a &= 0 \end{aligned} \tag{2}$$

som er på formen $f(x) = 0$ om vi lar $f(x) = x^2 - a$. Deriverer vi f får vi

$$f'(x) = 2x, \tag{3}$$

og iterasjonsligningen blir dermed

$$\begin{aligned} x_{n+1} &= x_n - \frac{x_n^2 - a}{2x_n} \\ x_{n+1} &= x_n - \frac{x_n}{2} + \frac{a}{2x_n}. \end{aligned} \tag{4}$$

En funksjon som beregner roten av a ved bruk av metoden over kan se slik ut:

```
minrot <- function(a,x0=a/2,tol=1e-8) {
   x <- x0
   repeat {
      forrige.x <- x
      x <- x - x/2 + a/(2*x)
      if (abs(x-forrige.x)<tol)
         break()
   }
   x
}
```

Vi trenger i praksis ikke å ta vare på alle x_n'ene; det er nok å bruke to lokale variabler `x` og `forrige.x`. Variabelen `forrige.x` brukes til å ta vare på forrige verdi av x før vi beregner neste verdi av x, slik at vi kan sammenligne x_{n+1} og x_n i det logiske uttrykket i første linje av while-løkken. I dette tilfelle har ligningen $f(x) = 0$ flere røtter, disse kan finnes ved å gi x_0 passende start verdier i nærheten av den roten vi søker:

```
> minrot(2)
[1] 1.414214
> minrot(2,x0=-2)
[1] -1.414214
```

Newton's metode vil ikke alltid konverger mot noen løsning, se Neuhauser (2004) for eksempler. En rekke varianter av Newtons metode og andre algoritmer eksisterer for å løse tilsvarende problem. R's innebygde funksjon `uniroot` bruker en annen og langsommere algoritme og søker seg fram til røtter til en funksjon f forutsatt at $f(x)$ har forskjellig fortegn på endepunktene av intervallet $x = a$ og $x = b$:

```
> f <- function(x,a) x^2-a
> uniroot(f,lower=0,upper=4,a=2)
$root
[1] 1.414208

$f.root
[1] -1.503627e-05

$iter
[1] 8

$estim.prec
[1] 6.103516e-05

> uniroot(f,lower=-4,upper=0,a=2)
$root
[1] -1.414208

$f.root
[1] -1.503627e-05

$iter
[1] 8

$estim.prec
[1] 6.103516e-05
```

2.2 Eksempel: Euler-Lotka ligningen

La oss tenke oss en populasjon bestående av individer av ulik alder. La n være maksimal alder. La m_i være antall avkom en hunn produserer på alderstrinn i (fekunditeten) og la l_i være overlevelse opp til alder i. Anta at disse parameterne ikke endrer seg over tid. Dette vil kunne være tilfelle så lenge den totale populasjonsstørrelsen er liten slik at det ikke er konkurranse om plass og andre begrensede ressurser. Selv om en slik antakelse kanskje ikke er realistisk vil det kunne være interessant å undersøke implikasjonen av en disse antakelsene. Et spørsmål er hvor raskt en slik populasjon vil vokse.

La $N_{t,0}$ betegne antall nyfødte i år t. Disse nyfødte er avkom av hunner født i tidligere år — av antall hunner født i år tilbake, $N_{t-i,0}$, vil en andel l_i være i live ved tidspunkt t. Hver av disse vil bidra med m_i avkom slik at det

totale bidraget fra årsklassen født i år tidligere blir $N_{t-i,0}l_im_i$ avkom. Totalt antall avkom født i år t kan dermed skrives som

$$N_{t,0} = \sum_{i=1}^{n} N_{t-i,0}l_im_i. \tag{5}$$

Hvordan vil $N_{t,0}$ endre seg over tid? Det virker rimelig å anta at $N_{t,0}$ etter hvert vil vokse eller avta eksponentielt med t, altså at en mulig løsning vil kunne være på formen

$$N_{t,0} = Ke^{rt}. \tag{6}$$

Setter vi (6) inn i (5) får vi at

$$Ke^{rt} = \sum_{i=1}^{n} Ke^{r(t-i)}l_im_i. \tag{7}$$

Deler vi begge sider med Ke^{rt} får vi

$$1 = \sum_{i=1}^{n} e^{-ri}l_im_i. \tag{8}$$

som er den såkalte Euler-Lotka ligningen. Hvis det eksisterer en r som er løsning av (8) betyr det at en løsning på formen (6) vil passe i (5), altså at populasjonen vil kunne vokse eksponentielt med vekstrate r.

Bare når $n \leq 2$ kan (8) løses ved vanlige metoder — for $n > 2$ ser vi at (8) er en n'te grads ligning i e^r. Denne må generelt løses numerisk, f.eks. ved bruk av Newton's metode. Vi må da skrive om (8) slik at den er på formen

$$f(r) = 0, \tag{9}$$

finne $f'(r)$, og så bruke iterasjonsligningen

$$r' = r - \frac{f(r)}{f'(r)}, \tag{10}$$

for å regne oss fram til løsningen.

Euler-Lotka ligningen og alderstrukturerte modeller generelt er nyttige i mange sammenhenger, f.eks. kan slike modeller brukes i befolkningsframskrivninger for å predikere fremtidige befolkningsstørrelser under ulike scenarier. Innen evolusjonsbiologi er slike modeller viktige for å forstå evolusjon av ulike livshistoriestrategier — slike modeller gir oss da innsikt i hva som er årsaken til fenomen som aldring, hva som påvirker alder for kjønnsmodning o.l.

Alderstrukturerte modeller kan håndteres mer generelt ved bruk av metoder fra lineær algebra. Dette vil vi se nærmere på senere i kurset.

2.3 Eksempel: Sannsynlighetsmaksimering

La oss tenke oss en levetidsmodell med dødsrate (intensitet)

$$\lambda(t) = \frac{1}{t+a}, \tag{11}$$

hvor parameter a er en positiv konstant. Dødsraten, altså sannsynligheten for at et individ dør i et lite tidsintervall $t, t + \Delta t$, gitt at det er i live ved tidspunkt t avtar altså med tiden t og går mot null når t går mot uendelig.

Generelt (se notat om levetidsfordelinger fra brukerkurset i sannsynlighetsregning) er kumulativ fordeling til T uttrykt ved dødsraten gitt ved

$$F_T(t) = P(T \leq t) = 1 - e^{-\int_0^t \lambda(u)du}. \tag{12}$$

For modell (11) får vi etter litt regning at dette blir

$$F_T(t) = 1 - \frac{a}{t+a}. \tag{13}$$

Deriverer vi får vi at tettheten til T blir

$$f_T(t) = \frac{a}{(t+a)^2}. \tag{14}$$

La oss tenke oss at $t_1, t_2, \ldots, t_n$ er observerte levetider fra modellen over og at vi ønsker å estimere parameteren a ved bruk av sannsynlighetsmaksimering. Vi trenger da likelihoodfunksjonen, altså sannsynligheten for dataene gitt a, som blir

$$L(a) = \prod_{i=1}^{n} \frac{a}{(t_i + a)^2}, \tag{15}$$

og log-likelihoodfunksjonen

$$\ln L(a) = n \ln a - 2 \sum_{i=1}^{n} \ln(t_i + a)^2. \tag{16}$$

I likelihoodfunksjonens maksimum er

$$\frac{\partial}{\partial a} \ln L(a) = 0, \tag{17}$$

slik at

$$\frac{n}{a} - 2 \sum_{i=1}^{n} \frac{1}{t_i + a} = 0. \tag{18}$$

Løsningen av denne ligningen er sannsynlighetsmaksimeringsestimatet av a. Vi ser at (18) er en $(n+1)$'te gradsligning i a som derfor ikke kan løses ved vanlige metoder med mindre $n = 1$. Derfor må denne estimeringsligningen løses numerisk. Vi ser at vi har en ligning på formen $f(a) = 0$ slik at vi kan finne løsningen ved å bruke Newton's metode. Vi trenger da $f'(a)$ som blir

$$-\frac{n}{a^2} + 2\sum_{i=1}^{2} \frac{1}{(t_i + a)^2}. \tag{19}$$

En funksjon som tar en vektor av observasjoner som innargument og beregner SME av a, $\hat{a}$, ved hjelp av Newton's metode blir dermed:

```
ahat <- function(t,a=1,tol=1e-6) {
   n <- length(t)
   repeat {
      forrigea <- a
      a <- a - (n/a-2*sum(1/(t+a)))/(-n/a^2+2*sum(1/(t+a)^2))
      if (abs(a-forrigea)<tol)
         break()
      print(a)
   }
   a
}
```

3 Newtons metode i flere dimensjoner

Newton's metode kan generaliseres til å løse sett av n ligninger med n ukjente. Skal vi f.eks. finne sannsynlighetsmaksimeringsestimater i en modell med n ukjente parametere vil dette gi opphav til et slikt sett av ligninger når vi setter de deriverte med hensyn på de ulike parameterne lik 0.

La oss se på det tilfelle at vi har et sett av to ikke-lineære ligninger med to ukjente. Vi har altså et problem på formen

$$\begin{aligned} f(x, y) &= 0, \\ g(x, y) &= 0. \end{aligned} \tag{20}$$

og ønsker å finne en løsning $\mathbf{z} = (x, y)$ slik at (20) er tilfredsstilt.

Vi gjetter først på en løsning $\mathbf{z}_0 = (x_0, y_0)$. Ideen som ligger bak løsningsalgoritmen er den samme som når vi bruker Newton's metode for å løse én

ligning med én ukjent. Vi lager oss tilnærminger til funksjonene f og g rundt punktet $\mathbf{z}_0 = (x_0, y_0)$ — lineariseringene

$$\begin{aligned} f(x,y) &\approx f(x_0,y_0) + \frac{\partial f}{\partial x}(x-x_0) + \frac{\partial f}{\partial y}(y-y_0) \\ g(x,y) &\approx g(x_0,y_0) + \frac{\partial g}{\partial x}(x-x_0) + \frac{\partial g}{\partial y}(y-y_0) \end{aligned} \tag{21}$$

De partiellderiverte beregnes i punktet $\mathbf{z}_0$. Vi ser altså på tangentplanene til funksjonene f og g i punktet $\mathbf{z}_0$.

I stedet for å forsøke å løse det virkelige ikke-lineære ligningssystemet er ideen så at vi lager vi oss en forbedret tilnærmet løsning ved å sette tilnærmingene av funksjonene lik null. Vi får da det lineære ligningssystemet

$$\begin{aligned} f(x_0,y_0) + \frac{\partial f}{\partial x}(x-x_0) + \frac{\partial f}{\partial y}(y-y_0) &= 0, \\ g(x_0,y_0) + \frac{\partial g}{\partial x}(x-x_0) + \frac{\partial g}{\partial y}(y-y_0) &= 0, \end{aligned} \tag{22}$$

som på matriseform kan skrives som

$$\mathbf{F}(\mathbf{z}_0) + \mathbf{J}(\mathbf{z} - \mathbf{z}_0) = 0, \tag{23}$$

hvor

$$\mathbf{F}(\mathbf{z}) = \begin{bmatrix} f(x,y) \\ g(x,y) \end{bmatrix}, \tag{24}$$

altså en funksjon som har en vektor som funksjonsverdi, og hvor

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}. \tag{25}$$

Vi lar løsningen på det lineariserte ligningssystemet (23) være neste tilnærmede løsning av (20) slik at vi generelt får iterasjonsformelen

$$\mathbf{z}_{n+1} = \mathbf{z}_n - \mathbf{J}^{-1}\mathbf{F}(\mathbf{z}_n). \tag{26}$$

Merk likheten til (1).