R päris algajatele

eriti neile kes pole varem programmeerinud*

Ott Toomet

15. august 2005. a.

1 Mis on R

R on väga kõrgel tasemel programmeerimiskeel. Ta on koostatud eeskätt statistika vajadusi silmas pidades ja seega on tal suur valik mitmesuguseid statistikale (ning ökonomeetriale) sobivaid arvutusvahendeid ning funktsioone. Muu hulgas tähendab too et keel sisaldab maatriksarvutusi, erineva jaotusega juhuslikke arve ning suurt hulka matemaatilisi funktsioone.

R on vaba tarkvara. Keele ja juurde kuuluvad vidinad võib leida ja ka omale installeerida aadressilt http://www.r-project.org.

2 Kuidas R-i käivitada

R käivitamine sõltub nii operatsioonisüsteemist kui kasutusviisist. Kui Sa kasutad windowsi pääl installeeritud R-i on kõige lihtsam klikata ekraani ikoonil.

R on soovitav käivitada õiges kataloogis (folderis). Selleks tee omale kataloog kus Sa hoiad just selle projektiga seotud asju. `Rgui` *file*-menüüs on valik kataloogi vahetamiseks. Koli sinna kataloogi.

Kuidas R-i lõpetada? Selleks on käsk `q()` mis tuleb sisestada *käsureal*, aknas kuhu R trükkis on käivitusinfo. Käsk tuleb lõpetada vajutades *enter*.

```
> q()
Save workspace image? [y/n/c]: y

Process R finished at Wed Feb 16 05:01:19 2005
```

3 Kuidas R-ga arvutada

R kasutamiseks "taskuarvutina" tuleb lihtsal käsureale trükkida arvutustehe:

```
> 1+1
[1] 2
```

R järgib tehete järjekorda:

*See tilluke õpetus sisaldab ainult väga väikest väljavõtet R võimalustest. Vähegi täielikuma info saamiseks pöördu juhendite poole.

```
> 1+2*3
[1] 7
```

Tehete järjekorra muutmiseks kasuta sulge:

```
> (1+2)*3
[1] 9
```

Astendamistehe on `^` ning jagamistehe `/`:

```
> 1+2*3^4
[1] 163
```

Lisaks tavalistele tehetele on olemas veel hulk matemaatilisi funktsioone. Näiteks eksponent ja logaritm:

```
> exp(0.1)
[1] 1.105171
> log(10)
[1] 2.302585
```

Viimasest reast selgub et `log()` tähendab naturaallogaritmi!

Arvutuste tulemused ja muud arvud võib salvestada *muutujatesse*[1] käsuga `<-` ja selle järel opereerida juba muutujatega. Näiteks:

```
> K <- 10
> L <- 20
> alfa <- 0.33
> Y <- K^alfa * L^(1 - alfa)
> Y
[1] 15.91073
```

Viimane käsk, lihtsalt `Y`, trükkis välja muutuja `Y` väärtuse. Muutujate nimed võivad olla üsna suvalised, ainult et nad peavad algama tähega. Lisaks võiks nad olla informatiivsed. Hääd nimed on näiteks `K` (kapital), `k` (kapital intensiivsel kujul), `alpha1` (α_1), `dotK` ($\dot{K}$) jne.

4 Kuidas programme kirjutada

Programm on põhimõtteliselt käskude jada. Tavaliselt kirjutatakse käsud üksteise alla tekstifaili ja siis käivitatakse see fail. Programmi kirjutamiseks võib kasutada erinevaid tekstiredaktoreid, näiteks *notepadi*[2]. Notepadiga salvestamisel peab meeles pidama et ta tahab meeleheitlikult faili nime lõppu panna `.txt`. Seda peab kas vältima või sellega arvestama.

Kirjutame nüüd näiteks programmi ruutvõrrandi lahendamiseks:

```
# ruutvõrrand a*x^2 + b*x + c = 0
a <- 2
b <- 3
```

[1] Siin käsitletakse ainult skalaarseid arvulisi muutujaid. Tegelikult on kõik muutujad R-s vektorid, skalaaride juhul sisaldavad nad ainult ühte komponenti. Märk `[1]` eelnevates näidetes tähendab et tegu on vastuse vektori esimese komponendiga. Mitut arvu üheks vektoriks saab teha näiteks käsuga `c()`.

[2] Notepad on kõige viletsam tekstiredaktor mida ma tean. Parema tekstiredaktori valimiseks vaadake http://www.sciviews.org/_rgui/ ja sääl alajaotust `IDE/Script`.

```
c <- -7
D <- sqrt(b^2 - 4*a*c)
x1 <- (-b + D)/2/a
x2 <- (-b - D)/2/a
cat("x1=", x1, "x2=", x2, "\n")
```

ning salvestame selle faili `ruutvqrrand.R` oma töökataloogis. Programmi saab käivitada käsuga `source(failinimi)`:

```
> source("ruutvqrrand.R")
x1= 1.265564 x2= -2.765564
```

Pane tähele et programmis tulemuste väljatrükkimiseks on kasutatud käsku `cat()`. Käsu kõige viimane argument, `"\n"`, on reavahetussümbol. Vastasel juhul jätkab R trükkimist samalt realt.

Teine oluline tähelepanek näiteprogrammist on esimene rida mis algab sümboliga `#`. See rida on kommentaar. R ignoreerib kõike mis järgneb `#` sümbolile, asja mõte on kirjutada sinna sisse selgitavaid märkusi endale või ka teistele.

5 Programmi kontrollstruktuurid

Automaatse töö juures on sageli vaja sarnaseid ülesandeid korrata ja käigu päält otsuseid vastu võtta. Selleks on olemas kontrollstruktuurid `for()` sama tegevuse kordamiseks, `break` kordamise katkestamiseks ja `if()` programmilõikude selektiivseks käivitamiseks. Kõige parem on seda selgitada näitega. Teeme programmi `kuup.R` mis trükib arvude kuupe:

```
for(x in 1:6) {
  x3 <- x^3
  cat("x=", x, "x^3=", x3, "\n")
}
```

Käivitame selle:

```
> source("kuup.R")
x= 1 x^3= 1
x= 2 x^3= 8
x= 3 x^3= 27
x= 4 x^3= 64
x= 5 x^3= 125
x= 6 x^3= 216
```

Käske mis jäid `for()` taha looksulgude sisse (tsükli sisse) täideti kuus korda, kusjuures muutuja `x` väärtus jooksis ühest kuueni. Näitame ka `if()` ja `break` käskude kasutamist. Muudame oma kuubiarvutusprogrammi niisuguseks:

```
for(x in 1:6) {
  x3 <- x^3
  cat("x=", x, "x^3=", x3, "\n")
  if(x > 3) {
    cat("Nüüd aitab!\n")
    break
  }
}
```

ja käivitame selle:

```
> source("kuup.R")
x= 1 x^3= 1
x= 2 x^3= 8
x= 3 x^3= 27
x= 4 x^3= 64
Nüüd aitab!
```

Iga kord tsüklis kontrolliti ega `x` pole suurem kui kolm, ning kui oli siis katkestati tsükkel (täideti käsud mis jäid `if()` taha looksulgude sisse). Pane tähele et kontroll toimus alles siis kui vastav tulemus oli juba välja trükitud! Teised kasulikud tingimused `if()`-lausele on `>`, `==`[3] (võrdus on *kaks* võrdusmärki!), `>=` ning `<=`.

6 Funktsioonid

Mugav viis interaktiivseks tööks on kasutada funktsioone. Sel juhul on programmi vaja vähem ringi teha ning funktsioone saab oma suva järgi käivitada. Teeme oma ruutvõrrandi programmist funktsiooni:

```
ruutv <- function(a=1, b=0, c=0) {
  ## ruutvõrrand a*x^2 + b*x + c = 0a <- 2
  D <- sqrt(b^2 - 4*a*c)
  x1 <- (-b + D)/2/a
  x2 <- (-b - D)/2/a
  cat("x1=", x1, "x2=", x2, "\n")
}
```

Programmi esimene rida on nüüd funktsiooni definitsioon. Funktsioonil `ruutv()` on kolm *argumenti*, `a`, `b` ning `c`. Argumentide väärtus vaikimisi on 1, 0 ning 0. Funktsiooni põhiosa jääb definitsiooni taha looksulgude sisse. Käivitame programmi:

```
> source("ruutvqrrand.R")
```

Programmi käivitamine käsuga `source` ei teinud nüüd midagi silmaga nähtavat. Funktsiooni kasutamiseks käivitame ta nüüd tema nimega.

```
> ruutv()
x1= 0 x2= 0
> ruutv(1,2,-4)
x1= 1.236068 x2= -3.236068
```

Esimesel korral kasutame vaikimisi väärtusi (s.o `a=1`, `b=0` ning `c=0`) ja saame kaks null-lahendit. Teisel korral anname ette vaikeväärtustest erinevad väärtused. Erinevaid väärtusi võib ette anda ka ühekaupa:

```
> ruutv(b=7)
x1= 0 x2= -7
```

Nüüd tuleb aga eraldi märkida et millise muutuja kohta see väärtus käib.

[3] Kui tegu on ujukomaarvudega (s.t. arvudega mis võivad olla murrulised), siis ei või `==` tüüpi võrdlemist kasutada. Kõik arvutid rehkendavad murdudega ligikaudselt ja sellised võrdused ei kehti peaaegu kunagi. Võib kasutada näiteks konstruktsiooni `if(abs(a - b) < 1e-10)` või midagi sarnast.

7 Vead

Kõik programmeerijad teevad vigu. Oletame et me kirjutasime oma ruutvõrrandi funktsioonis `x1` arvutamise rea valesti:

```
ruutv <- function(a=1, b=0, c=0) {
  ## ruutvõrrand a*x^2 + b*x + c = 0a <- 2
  D <- sqrt(b^2 - 4*a*c)
  x1 -< (-b + D)/2/a    # omistamine valesti!
  x2 <- (-b - D)/2/a
  cat("x1=", x1, "x2=", x2, "\n")
}
```

Käivitame nüüd programmi:

```
> source("ruutvqrrand.R")
Error in parse(file, n, text, prompt) : syntax error on
    line 4
```

Ahhaa! Neljadas reas on midagi mäda!

Vead võivad tekkida ka programmi käigu ajal. Parandame programmi ära ning püüame lahendada niisugust ruutvõrrandit:

```
> ruutv(1,2,3)
x1= NaN x2= NaN
Warning message:
NaNs produced in: sqrt(b^2 - 4 * a * c)
```

Mis juhtus? R teatab et ruutjuure võtmisel tekkis `NaN`. See on lühend väljendist *not a number* ja tähendab et niisugust arvu pole võimalik rehkendada. Ruutjuure puhul on probleemiks ilmselt ruutjuur negatiivsest arvust[4].

Kontrolli ka et sulud oleks õiged: iga algava sulu kohta peab olema vastav lõpusulg.

8 Kust saada lisaabi

Eespool toodud materjalid puudutasid programmerimist üldse, ja R-i eriti, vaid väga pinnapealselt. Samas on ka ainult neid vahendeid kasutades võimalik lahendada palju ettetulevaid ülesandeid. R on vabatarkvara kohta väga hästi dokumenteeritud süsteem. Terve hulk materjale nii HTML kui pdf kujul on olemas R kodulehel http://www.r-project.org. Kõige põhilisemad nendest installeeritakse automaatselt ka teie arvutisse, samasse kataloogi kus R ise. Kasulikumad manuaalid on *An Introduction to R* ning pakettide (eriti `base`-paketi) HTML-kujulised juhendid.

Teiseks on võimalik abi saada ka käsurealt. Näiteks käsk `?sqrt` annab infot ruutjuure funktsiooni kohta. Kui me ei tea meid huvitava funktsiooni nime võime kasutada `help.search()` käsku. Näiteks otsime midagi programmide parandamise/silumise (*debugging*) kohta:

```
> help.search("debug")
Help files with alias or concept or title matching
```

[4] Negatiivse arvu ruutjuure leidmiseks peab R-s arvu deklareerima kompleksarvuna. Proovi `ruutv(1,2,3+0i)`.

```
'debug' using fuzzy matching:
debug(base)        Debug a function
trace(base)        Interactive Tracing and Debugging
of Calls to a Function or Method
...
```

9 Näide dünaamilisest mudelist

Vaatleme hundi-lamba mudelit. Mudelis on hundid H ja lambad L. Lambad söövad rohtu ja nende sündimise määr on α_L. Lambad surevad huntide läbi ja nende surma määr on $\beta_L H$. Seega lammaste arvu dünaamikat kirjeldab seos

$$\dot{L} = \alpha_L L - \beta_L HL. \tag{1}$$

Hundid söövad lambaid, seega sõltub nende iive positiivselt lammaste arvukusest. Olgu nende sündimismäär $\alpha_H L$ ning nad surevad loomulikku surma – suremismäär on β_H. Seega hunide arvukust kirjeldab võrrand

$$\dot{H} = \alpha_H LH - \beta_H H. \tag{2}$$

Koostame programmi millele antakse ette algne huntide ja lammaste arvukus H_0 ning L_0, ja mis tolle järel rehkendab iga ajahetke jaoks arvukuse. Selleks tuleb dünaamilisi kasvu võrrandeid integreerida:

$$L(t + \Delta t) \approx L(t) + \dot{L}(t)\Delta t \tag{3}$$

lammaste tarvis ning analoogiliselt huntide tarvis.

Programm `HundidLambad.R` võiks olla näiteks niisugune (viitamiseks on lisatud reanumbrid):

```
HundidLambad <- function(H=1, L=1, aH=1, bH=1, aL=1, bL
    =1, T=10, dt=1) {
2     ## T   aeg, kui pikka aega arvutada
      ## dt  ajaühik (arvutustäpsus)
4     N <- T/dt
      # N    mitu tsüklit on antud pikkus antud täpsusega
6     H0 <- H
      L0 <- L
8     for(i in 1:N) {
          H <- H0 + dt*(aH*H*L - bH*H)
10        L <- L0 + dt*(aL*L - bL*H*L)
          cat(H, "\t", L, "\n")
12        H0 <- H
          L0 <- L
14    }
}
```

Funktsioonil `HundidLambad()` on kaheksa argumenti: huntide ja lammaste algarvukus (`H` ning `L`), neli mudeli parameetrit, arvutuste kestus `T` ning arvutuste täpsus/ajasamm `dt`.

8.-14. rida on arvutuste põhitsükkel. Selle käigus lisatakse algväärtustele `H0` ja `L0` ajaühiku jooksul toimunud juurdekasv (read 9 ja 10). See vastab integreerimisele seoses (3). Seejärel trükitakse tulemused, ning viimaks tehakse praeguse tsükli uued väärtused `H` ning `L` järgmise ajaperioodi uuteks algväärtusteks `H0` ning `L0` (read 12 ja 13). Kasulik on teada et sümbol `"\t"` trükkimisel on *tabulaator*. Niisugust väljatrükki on lihtsam näiteks tabelarvutusse ümber tõsta ja nii graafikuid joonistada.

Funktsiooni näidisväljund oleks:

```
> HundidLambad()
1       1
1       1
1       1
...
```

Milles asi? Selles et $(H = 1, L = 1)$ on antud parameetrite korral tasakaalupunkt. Proovime mõnda teist algväärtust, näiteks anname huntide arvuks 0,5:

```
> HundidLambad(H=0.5)
0.5     1.5
0.75    1.875
1.40625         1.113281
1.565552        0.4836631
0.7571996       0.6010967
0.4551502       0.9286041
0.4226543       1.464730
0.6190743       2.022683
1.252191        1.512580
1.894040        0.1602736
```

Seekord tõesti toimub midagi. On aimata nii huntide kui lammaste arvukuse tsükliline kõikumine. Et asjast paremat pilti saada on vaja graafilist esitust.

10 Funktsiooni graafik

R on tuntud oma väga võimalusterohke ja kvaliteetse graafika poolest. Funktsiooni graafikuid saab joonistada käsuga `curve()`, argumentideks on funktsioon ning vahemik.

Vaatame näiteks kuidas näeb CRRA kasulikkusfunktsiooni $u(c) = c^{1-\vartheta}/(1-\vartheta)$ välja sõltuvalt parameetrist ϑ. Siin on aga üks aga: antud kujul sõltub mitte ainult funktsiooni kuju vaid ka tema "kõrgus" tugevasti ϑ-st. Et kõrgus paika jääks, lahutame temast konstandi $1/(1-\vartheta)$. Nüüd $u(1) = 0$ sõltumata ϑ-st ning seega kõrguse probleem kaob ära. Uus funktsiooni kuju on seega

$$u(c) = \frac{c^{1-\vartheta} - 1}{1 - \vartheta}. \tag{4}$$

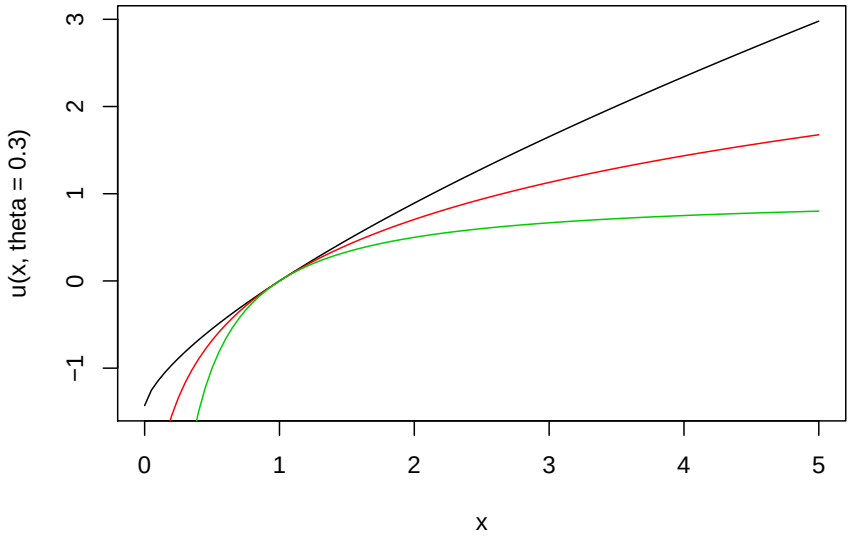
Funktsioon `u()` võiks nüüd välja näha niisugune:

```
u <- function(c, theta=0.9) {
    (c^(1 - theta) - 1)/(1 - theta)
}
```

Selle funktsiooni graafiku võime nüüd joonistada käsuga `curve`. Näiteks nii:

```
> curve(u(x, theta=0.3), 0, 5)
> curve(u(x, theta=0.95), 0, 5, add=TRUE, col=2)
> curve(u(x, theta=2), 0, 5, add=TRUE, col=3)
```

Esimene käsk `curve(u(x, theta=0.3), 0, 5)` joonistab funktsiooni `u()` graafiku parameetri väärtusel $\vartheta = 0{,}3$ vahemikus $c = 0$ kuni 5. Pane tähele et meid huvitav sõltumatu muutuja, antud juhul tarbimini c, tuleb tähistada `x`-ga. Järgmine rida teeb sama parameetriga $\vartheta = 0{,}95$. Lisaks ütleme et uus graafik tuleb lisada vanale ja mitte vana ära kustutada (`add=TRUE`) ning et joonistada punase värviga (`col=2`). Kolmas rida on analoogiline (`col=3` on roheline). Tulemus näeb välja nii nagu joonisel 1.



Joonis 1: CRRA kasulikkusfunktsioon eri ϑ väärtustel

Näeme et väikese ϑ korral on kasulikkus peaaegu lineaarne. $\vartheta = 0{,}95$ on sarnane logaritmilise kasulikkusega ($\vartheta = 1$ juhtu ei saa niimoodi plottida sest siis tekiks jagamine nulliga) ning kui $\vartheta = 2$ siis langeb piirkasulikkus tarbimise suurenedes väga järsult.

11 Hundid-lambad: graafiline esitus

Huntide-lammaste arvukuse võib küll ekraanile trükkida, tabelarvutusse ümber tõsta, ning siis tollega plottida, kuid niisugune töö on kohmakas. Teeme seda R-ga automaatselt. Kõigepäält maname graafikaakna esile käsuga `plot()` ning lisame sinna informatsiooni käsuga `abline()` ning `arrows()`.

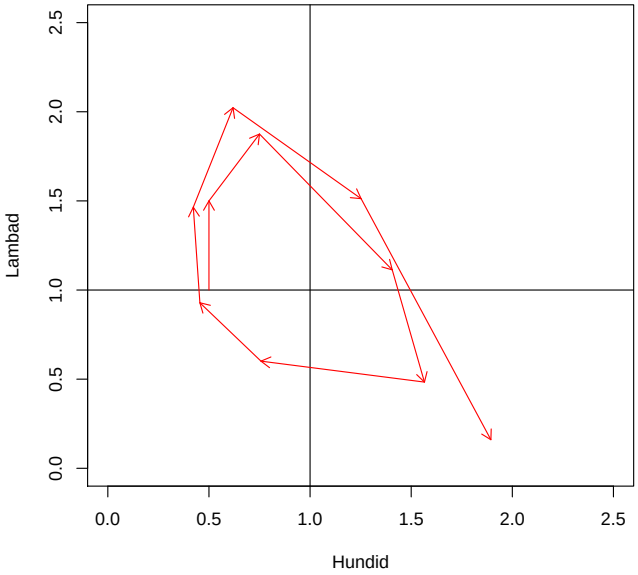
Graafilise esitusega võiks hundi-lamba programm välja näha selline:

```
HundidLambad <- function(H=1, L=1, aH=1, bH=1, aL=1, bL
    =1, T=10, dt=1) {
  ## T   aeg, kui pikka aega arvutada
  ## dt  ajaühik (arvutustäpsus)
  N <- T/dt
  ## N    mitu tsüklit on antud pikkus antud täpsusega
  plot(1:2, xlim=c(0,2.5), ylim=c(0,2.5), type="n",
       xlab="Hundid", ylab="Lambad")
  abline(v=aL/bL)
  abline(h=bH/aH)
  H0 <- H
  L0 <- L
  for(i in 1:N) {
    H <- H0 + dt*(aH*H*L - bH*H)
    L <- L0 + dt*(aL*L - bL*H*L)
    arrows(H0, L0, H, L, length=0.1, col=2)
    H0 <- H
    L0 <- L
  }
}
```

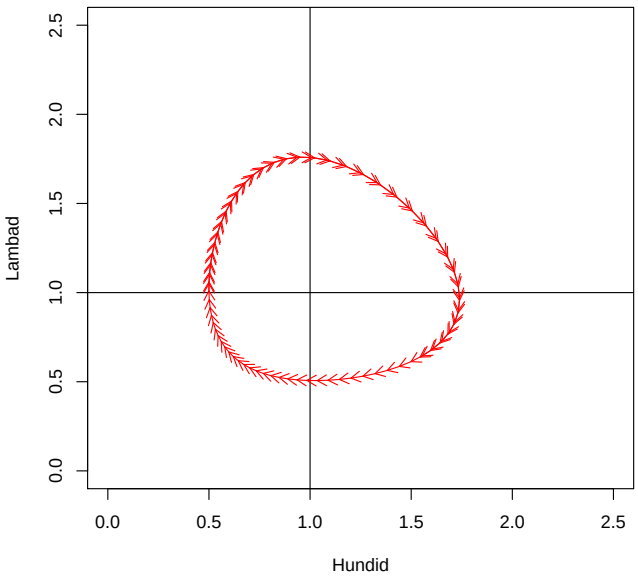
Muutused võrreldes eelmise variandiga on minimaalsed. Rida 6 ja 7 sisaldavad `plot()`-käsku mis joonistab teljed vahemikus 0 kuni 2,5 ning annab telgedele nimed. Esimene argument `1:2` ütleb et mida plottida ja `type="n"` ütleb et see mida plottida jätta plottimata. Ehk siis ainult joonistada teljed. Järgnevad kaks rida joonistavad vertikaalse ja horisontaalse joone mis vastavad tingimustele $\dot{H} = 0$ ning $\dot{L} = 0$ (antud mudelis on need sirgjooned). Trükkimise asemel on rida 15 nüüd noole joonistamine: joon tõmmatakse punktist (`H0,L0`) punkti (`H,L`) värviga 2 (mis on punane).

Käsu `HundidLambad(H=0.5)` tulemus on nüüd joonisel 2. Näeme et süsteem joonistab hundid-lambad teljestikus tsõõre. Pilt on aga väga sakiline. Vähendame ajaühikut et arvutustäpsus paraneks: `HundidLambad(H=0.5, dt=0.1)`. Tulemus on näha joonisel 3.

Uus tulemus on palju parem. Näeme et huntide ja lammaste arvukus kõigub tsükliliselt kusjuures tsüklite maksimumid ei lange kokku. Intuitsioon on lihtne: algne madal huntide arvukus viib lammaste arvuku kiirele kasvule, kõrge lammaste arvukus aga tähendab et hundid paljunevad kiiresti. Lambad süüakse ära ja toidu puudusel hunte enam juurde ei sünni ning nende arv väheneb. Tsükkel algab otsast pääle. Selline kiskjate saakloomade arvukuse kõikumine on vaadeldav mõnes lihtsas koosluses nagu saartel. Tüüpiline tsükli pikkus on seitse aastat.



Joonis 2: Joonis käsuga `HundidLambad(H=0.5)`



Joonis 3: Joonis käsuga `HundidLambad(H=0.5, dt=0.1)`