Laboration 3: Födelse-dödsprocesser

Benjamin Kjellson (tidigare version skriven av Anders Björkström)

2016-05-26

Viktigt: Innan ni läser vidare

Gör först följande:

1. I den övre menyn i RStudio, tryck på `Tools --> Global Options...`. I rutan som dyker upp, där det står "Default text encoding", tryck "Change" och välj "UTF-8".
2. Gå in på kurshemsidan och ladda ner mallen för denna laboration (döp den till något vettigt när ni sparar den på er dator). Öppna den i RStudio. Skriv er rapport i denna fil.

För dig som använder en av skolans datorer

Innan du stänger av datorn, se till att maila dina labbfiler till dig själv eller lägga över dem på ett USB-minne.

Krav för laboration 3

Följande **krav** ställs på er rapport:

- Rapporten måste kunna läsas av någon som inte har läst labbinstruktionerna. Så ni måste skriva vad det är ni ska göra innan ni gör det, och berätta vad syftet är.
- Rapporten måste vara skriven i **R Markdown** eller knitr.
- All kod som används måste synas i labbrapporten, men ska inte beskrivas i detalj i rapporten.
- Alla **tabeller och diagram** måste förses med **numrering och beskrivande text**, och refereras till i rapportens vanliga text på rätt sätt. Diagram måste ha lämpliga rubriker på axlar och tabeller lämpliga rubriker på kolumner.

Bakgrund

Vid institutionen för experimentell ekonomi finns en dator, dit forskare och studenter sänder beräkningstunga jobb för exekvering. Jobb anländer till datorn enligt en Poissonprocess med intensiteten λ jobb per timme. Tiderna det tar att exekvera jobben kan betraktas som oberoende och exponentialfördelade med parameter μ. Det tar i genomsnitt alltså $1/\mu$ timmar för ett jobb att bli klart.

Jobb som kommer när datorn är ledig börjar genast exekveras; övriga jobb lagras i en buffert och behandlas i turordning. Bufferten har plats för två väntande jobb (det jobb som exekveras tar ingen plats i bufferten). Om ett jobb kommer när bufferten är fylld så får användaren ett felmeddelande, med uppmaning att försöka igen om en liten stund. Institutionsstyrelsen har beslutat att högst 5% av alla jobb som skickas till datorn ska behöva råka ut för detta.

För att bestämma vilka som ska ha tillträde till datorn finns tre förslag:

1. Endast professorer har tillträde. Då blir $\lambda = 2$ och $\mu = 10$.
2. Professorer och registrerade studenter har tillträde. Då blir $\lambda = 6$ och $\mu = 10$.
3. Vem som helst har tillträde. Då blir $\lambda = 10$ och $\mu = 10$.

Vi ska undersöka vilka av förslagen 1, 2 och 3 som uppfyller institutionsstyrelsens krav. Eftersom jobben anländer enligt en Poissonprocess så är kravet likvärdigt med att bufferten får vara full högst 5% av tiden. Detta undersöker vi dels genom att simulera hur antalet jobb i systemet utvecklar sig i tiden, under en tillräckligt lång tidsperiod, dels genom att räkna ut den stationära fördelningen för systemet.

Låt $X(t)$ vara antalet jobb som antingen väntar i bufferten eller åtgärdas av datorn vid tiden t. Processen $\{X(t), t \geq 0\}$ är en Markovkedja i kontinuerlig tid, närmare bestämt en födelse-dödsprocess med tillståndrummet $S = \{0, 1, 2, 3\}$. I simuleringen antas att $X(0) = 0$.

Om vi ritar upp $X(t)$ som funktion av t kommer vi få en kurva som rör sig språngvis mellan de fyra värdena $0, 1, 2$ och 3. För att göra detta kommer vi skapa en vektor `tid`, som innehåller tidpunkterna för sprången, och en vektor `state`, där motsvarande element anger vilket tillstånd språnget leder till. Vi låter vektorn `state` börja med en nolla, eftersom det är initialtillståndet, och vektorn `tid` har en nolla som första element eftersom vi börjar tidsräkningen därifrån.

Tanken är nu att ni ska kopiera "funktionsskelettet" på följande sida till er rapport, och fylla i det som saknas där det står `....` Glöm inte att få med indenteringen (mellanslagen framför koden) så att koden blir lätt att läsa. Där det står `# ?` eller `# [fråga]?` ska ni **byta ut det efter # mot en mening som förklarar vad koden nedanför motsvaras av för händelse i födelse-dödsprocessen**, i vanliga ord.

Ledning: Använd formlerna i början av avsnitt 6.3 i Ross (avsnittet börjar på sida 359 i upplaga 11, sida 374 i upplaga 10, och sida 368 i upplaga 9).

```r
bd_process <- function(lambda, mu, initial_state = 0, steps = 100) {

  time_now <- 0
  state_now <- initial_state

  # Dessa vektorer ska byggas på i loopen nedan
  time <- 0
  state <- initial_state

  for (i in 1:steps) {

    # ?
    if (state_nu == 3) {
      lambda_now <- 0
    } else {
      lambda_now <- lambda
    }

    # ?
    if (state_now == 0) {
      mu_now <- 0
    } else {
      mu_now <- mu
    }

    # ?
    time_to_transition <- ...

    # ?
    if (...) {
      state_now <- state_now - 1
    } else {
      state_now <- state_now + 1
    }

    time_now <- time_now + time_to_transition # vad är time now?
    time <- c(time, time_now) # vad innehåller vektorn time?
    state <- c(state, state_now) # vad innehåller vektorn state?
  }

  # Returnera en lista med de två vektorerna tid och state
  list(tid = tid, state = state)
}
```

När ni sedan har fyllt i funktionen korrekt kan ni enkelt kalla den med värden på `lambda` (dvs λ) och `mu` (dvs μ) enligt de tre förslagen som gavs ovan. Exempel:

```r
set.seed(19880210) # fyll i erat eget födelsedatum här istället


forslag1 <- bd_process(lambda = 2, mu = 10)
# Här låter vi argumenten initial_state och steps vara på sina defaultvärden,
# så de behöver inte anges.
```

```r
# Hämta ut vektorerna tid och state
time1 <- forslag1$time
state1 <- forslag1$state
```

För att sedan rita upp $X(t)$ som funktion av t, i ett så kallat **trappstegsdiagram**, så kan ni göra det enligt följande exempel (OBS! påhittade värden för `time1` och `state1`):

```r
plot(stepfun(time1[-1], state1),
     do.points = FALSE,
     xlab = "Tid",
     ylab = "Tillstånd",
     main = "",
     yaxt = "n")
axis(2, at = c(0, 1, 2, 3), las = 2)
```

Viktig lärdom: istället för att upprepa ovanstående kodstycke tre gånger för de olika kombinationerna av λ och μ ni ska testa så har ni nu skrivit en funktion som ni enkelt kan återanvända för olika värden på λ och μ. Genom att skriva en generell funktion istället för att kopiera kodstycken så spar ni tid och minskar risken för fel. Felet finns nu på en plats istället för tre, eller hur många gånger ni nu skulle ha kopierat ett och samma kodstycke.

Uppgift 1

Beskriv med ord och symboler systemet som definierar födelse-dödsprocessen. Skriv ned vad fördelningen för tiden till nästa tillståndsbyte är och vad sannolikheten att processen går upp eller ned ett steg är, beroende på processens rådande tillstånd. När ni gjort detta för generella λ och μ, gör det specifikt för förslag 1 också.

Uppgift 2

Visa den ifyllda koden, inklusive kommentarer. Koden ska vara indenterad, dvs ha olika många mellanslag beroende på "nivån" i koden, precis som den är skriven i labbinstruktionen.

Uppgift 3

Rita upp ett simulerat trappstegsdiagram för vart och ett av förslagen 1–3 ovan (tre olika λ-värden). Kommentera diagrammen: varför ser de ut som de gör, och varför ser de olika ut? Till funktionen `bd_process`, använd argumenten `steps = 100` och `initial_state = 0` (dvs default, så du behöver inte fylla i något).

Uppgift 4

Visa i en tabell hur lång tid det tog innan systemet hade ändrat tillstånd 500 gånger, för alla tre förslag. För att göra detta behöver du kalla funktionen på nytt för de tre förslagen, med ett nytt värde på `steps` (vilket?). Visa **inte** något diagram, men jämför tiderna mellan de tre förslagen och förklara varför de skiljer sig åt (eller inte). Avrunda tiderna till ett lämpligt antal decimaler, förslagsvis 0.

Uppgift 5

Del 1

Skriv en funktion `proportion_in_state` som tar ett tillstånd `s` (0, 1, 2 eller 3) och en simulerad födelse-dödsprocess `bdp` (som t.ex. variabeln `forslag1` ovan) och returnerar andelen tid som processen spenderade i det givna tillståndet. Alltså, det är fördelningen av tiden (t.ex. i procent) som efterfrågas, inte hur många tidsenheter. Här är en mall:

```
proportion_in_state <- function(s, bdp) {
  ... # fyll i kod här
}

# Exempel på användning:
proportion_in_state(2, forslag1) # räkna ut andelen tid i tillstånd 2
```

Tips:

- Fundera på vad tiderna i tid-vektorn `time` som fås från `bd_process` är och hur de kan omvandlas till tiderna processen spenderade i varje tillstånd den besökte, och vilka element i tillståndsvektorn `state` som ska användas.
- Bortse från det sista elementet i tillståndsvektorn, eftersom ingen tid spenderats i det sista tillståndet processen hoppade till.

Del 2

Kalla nu på funktionen `bd_process` på nytt med `steps = 1000` för vart och ett av förslagen, och räkna för vart och ett av dem ut hur stor del av tiden som har tillbringats i tillstånd $0, 1, 2$ respektive 3. Visa resultaten i en tabell med tillstånden längs kolumnerna och förslagen längs raderna.

Avgör vilket eller vilka av förslagen 1–3 som är acceptabla för institutionsstyrelsen.

Uppgift 6

Ange en formel för den stationära fördelningen som funktion av kvoten $\rho = \lambda/\mu$. Bestäm den stationära fördelningen för vart och ett av de tre förslagen (vi vill se siffror, inte symboler). Visa dessa som i en tabell, precis som i föregående uppgift, och gör en jämförelse. För vilka förslag "borde" systemet få godkänt?

Redovisning

Laborationen skall utföras i grupper om högst två personer. Redovisning sker genom skriftliga svar på uppgifterna 1–6, som ska vara inlämnade senast det datum som anges på schemat. De som har skrivit sitt namn på redogörelsen ska vara beredda på att besvara muntliga följdfrågor som examinatorn kan vilja ställa. Var noga med att följa kraven som ställdes i början av denna instruktion!