Labb 3 - Förgreningsprocesser

Sebastijan Babic & Esbjörn Runesson

2025-03-08

Contents

```r
# Denna funktion utgör ett skal för en födelse-dödsprocess med intensiteter
# lambda och mu.
# Funktionen returnerar en lista med tider för tillståndsbyten och tillstånd.
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) {

        # Om tillståndet är 3 så är lambda 0, intensiteten för födelse
        # om redan i 3, kan ej födelse ske
        if (state_now == 3) {
            lambda_now <- 0
        } else {
            lambda_now <- lambda
        }

        # Om tillståndet är 0 så är mu 0, intensiteten för dödsfall
        # om redan i 0, kan ej dödsfall ske
        if (state_now == 0) {
            mu_now <- 0
        } else {
            mu_now <- mu
        }

        time_to_transition <- rexp(1, lambda_now + mu_now) # ty tiden till nästa tillståndsbyte är expo

        # i en födelsedods-process har vi ett tal mellan 0-1 som vi generare slumpmässigt, se runif(1)
        # om det talet är mindre än lambda_now / (lambda_now + mu_now) så sker en födelse, annars en dö
        if (runif(1) < lambda_now / (lambda_now + mu_now)) {
            state_now <- state_now + 1
        } else {
            state_now <- state_now - 1
        }

        time_now <- time_now + time_to_transition # time nu är tiden från starten
        time <- c(time, time_now) # time är en vektor som innehåller alla tider från starten
        state <- c(state, state_now) # state är en vektor som innehåller alla tillstånd från starten
    }
    # Returnera en lista med de två vektorerna tid och state
    list(time = time, state = state)
}
```

1 Förklaring av förgreningsprocessen

Förslag på dem som ska ha tillträde till superdatorn:

1. Endast professorer har tillträde. Då blir $\lambda = 1$ och $\mu = 5$.
2. Professorer och registrerade stundenter har tillträde. Då blir $\lambda = 2$ och $\mu = 5$.
3. Vem som helst har tillträde. Då blir $\lambda = 5$ och $\mu = 5$.

> Förklara vad fördelningen för tiden till nästa tillståndbyte är och vad sannolikheten att processen går upp eller ned ett steg är, beroende på processens tillstånd. Gör det generellt och för förslag 1.

Vi betraktar en Markovkedja med tillstånden $i = 0, 1, 2, 3$, där i representerar antalet jobb i systemet. Övergångarna i systemet sker genom ankomster (födelse) med intensitet λ och serviceavslut (död) med intensitet μ.

Om systemet befinner sig i ett tillstånd där både ankomst och service är möjliga (dvs. $i = 1$ eller 2), är den totala hastigheten för att lämna tillståndet

$$\lambda + \mu.$$

Därför är tiden T till nästa tillståndbyte exponentialfördelad:

$$T \sim \text{Exp}(\lambda + \mu).$$

Väntevärdet av T är därför

$$E(T) = \frac{1}{\lambda + \mu}.$$

I sluttillstånden gäller:

- Vid $i = 0$ sker endast ankomster med intensitet λ:

$$T \sim \text{Exp}(\lambda), \quad E(T) = \frac{1}{\lambda}.$$

- Vid $i = 3$ sker endast serviceavslut med intensitet μ:

$$T \sim \text{Exp}(\mu), \quad E(T) = \frac{1}{\mu}.$$

När båda övergångar är möjliga (t.ex. $i = 1$ eller 2) är sannolikheterna för ett steg upp eller ned:

$$p_{i,i+1} = \frac{\lambda}{\lambda + \mu}, \quad p_{i,i-1} = \frac{\mu}{\lambda + \mu}.$$

I sluttillstånden är:

- Vid $i = 0$:

$$p_{0,1} = 1.$$

- Vid $i = 3$:

$$p_{3,2} = 1.$$

För förslag 1 har vi:

- $\lambda = 1$ jobb per timme,

- $\mu = 5$ jobb per timme.

Därmed får vi:

- För $i = 1$ och 2 är den totala hastigheten

$$\lambda + \mu = 1 + 5 = 6.$$

  Tiden till nästa tillståndbyte är alltså $T \sim \text{Exp}(6)$ med väntevärde

$$E(T) = \frac{1}{6} \text{ timme.}$$

- Övergångssannolikheterna blir:

$$p_{i,i+1} = \frac{1}{6}, \quad p_{i,i-1} = \frac{5}{6}.$$

2 Plot av processen som funktion av tiden

Rita upp $X(t)$ som funktion av t, d.v.s. en kurva som rör sig språngvis mellan de fyra värdena $0, 1, 2$ och 3. För att göra detta ska ni skapa en vektor `time`, 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 `time` har en nolla som första element eftersom vi börjar tidsräkningen därifrån.

```
# Sätt slumptalsfröet för reproducerbarhet
set.seed(040911)



forslag1 <- bd_process(lambda = 1, mu = 5)
time1 <- forslag1$time
state1 <- forslag1$state

forslag2 <- bd_process(lambda = 2, mu = 5)
time2 <- forslag2$time
state2 <- forslag2$state


forslag3 <- bd_process(lambda = 5, mu = 5)
time3 <- forslag3$time
state3 <- forslag3$state


# Kombinera alla tre förslagen i en plot med tre subplots och reserverar en hel sida för dessa plottar
# las = 2 betyder att y-axelns etiketter är horisontella
par(mfrow = c(3, 1))

plot(stepfun(time1[-1], state1), do.points = FALSE,
     xlab = "Tid", ylab = "Tillstånd",
     main = "Förslag 1, lambda = 1, mu = 5",
     yaxt = "n")
axis(2, at = c(0, 1, 2, 3), las = 2)

plot(stepfun(time2[-1], state2), do.points = FALSE,
     xlab = "Tid", ylab = "Tillstånd",
     main = "Förslag 2, lambda = 2, mu = 5",
     yaxt = "n")
axis(2, at = c(0, 1, 2, 3), las = 2)

plot(stepfun(time3[-1], state3), do.points = FALSE,
     xlab = "Tid", ylab = "Tillstånd",
     main = "Förslag 3, lambda = 5, mu = 5",
     yaxt = "n")
axis(2, at = c(0, 1, 2, 3), las = 2)
```

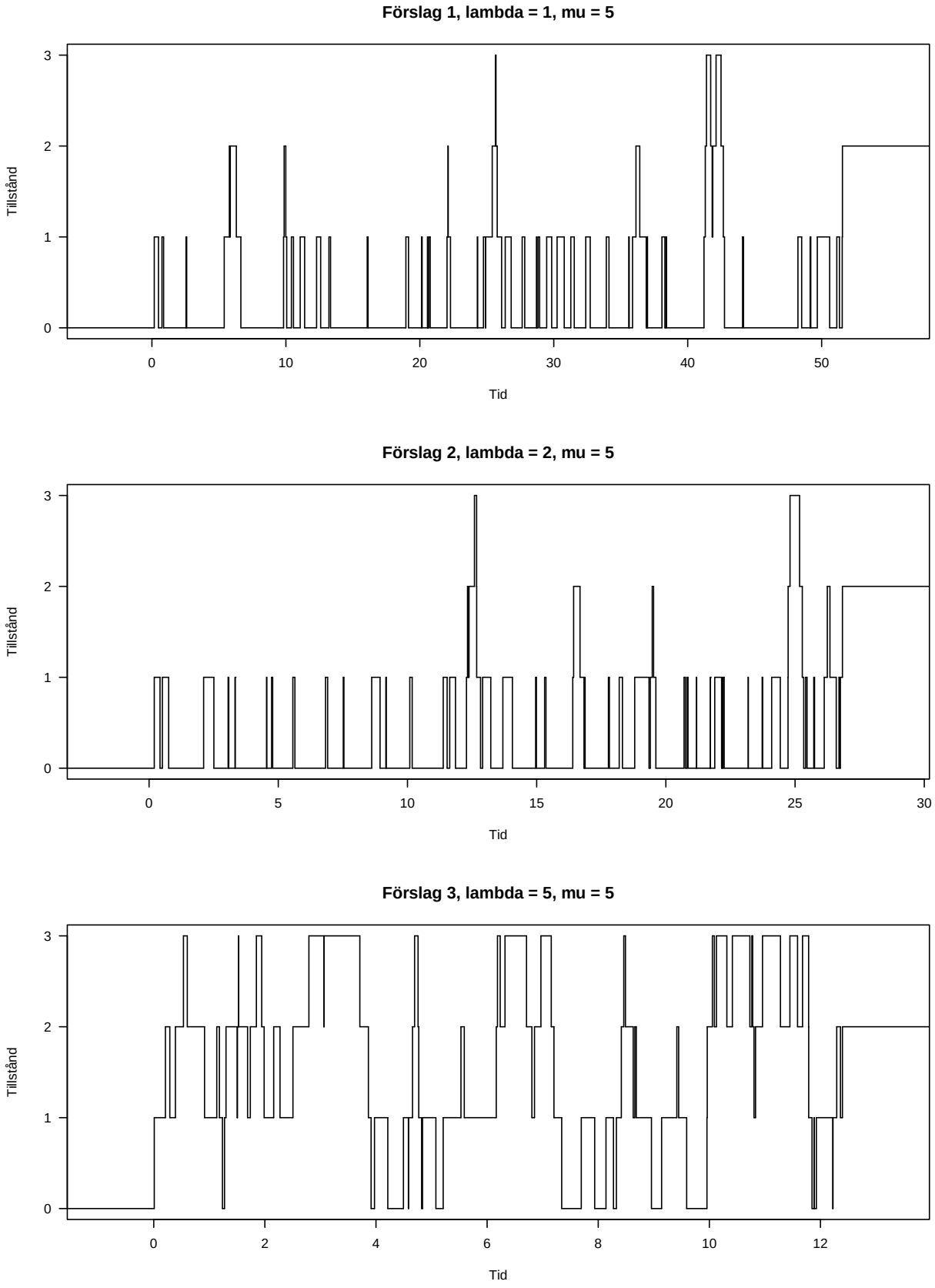

Figure 1: Simulerad födelse-dödsprocess för dem tre olika förslag

3 Tid för att systemet hade ändrat tillstånd 500 gånger

Räkna ut 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. Jämför tiderna mellan de tre förslagen och reflektera över varför de skiljer sig åt (eller inte).

```
# steps representerar antalet tillståndsbyten
forslag1_500 <- bd_process(lambda = 1, mu = 5, steps = 500)
forslag2_500 <- bd_process(lambda = 2, mu = 5, steps = 500)
forslag3_500 <- bd_process(lambda = 5, mu = 5, steps = 500)

# hämtar det sista elementet i tid-vektorn för intresserade av det totala tiden
# tail(time,1) där 1 är antalet element från slutet som ska hämtas
tid1 <- round(tail(forslag1_500$time, 1), 0)
tid2 <- round(tail(forslag2_500$time, 1), 0)
tid3 <- round(tail(forslag3_500$time, 1), 0)

# skapar en data.frame med de olika tiderna
bd_process_500 <- data.frame(
  Förslag = c("1", "2", "3"),
  Tid = c(tid1, tid2, tid3)
)

knitr::kable(bd_process_500, caption = "Tid för att systemet hade ändrat tillstånd 500 gånger")
```

Table 1: Tid för att systemet hade ändrat tillstånd 500 gånger

Förslag	Tid
1	264
2	126
3	65

Ser att förslag 3 är det snabbast att få systemet att ändra tillstånd 500 gånger. Det är rimligt ty högre födelseintensitet ger fler födelser och därmed fler tillståndsbyten. Motsvarande för förslag 1 som visade sig vara det långsammaste.

4 Andel tid spenderad i varje tillstånd

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. Bortse från det sista elementet i tillståndsvektorn, eftersom ingen tid spenderats i det sista tillståndet processen hoppade till.

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.

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

Börjar med att skriva en funktion som beräknar andelen tid som processen spenderade i varje tillstånd.

```
proportion_in_state <- function(state, bdp) {

  # Beräkna tidsintervallerna mellan tillståndsbyten
  time_intervals <- diff(bdp$time)

  # Ignorera det sista elementet i state-vektorn för att inte räkna med tiden som spenderats i det sist
  visited_states <- bdp$state[-length(bdp$state)]

  # Beräkna andelen tid i det givna tillståndet
  proportion <- sum(time_intervals[visited_states == state]) / sum(time_intervals)
  return(proportion)
}
```

För att förstå 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 kan vi tänka oss att vi har en process som börjar i tillstånd 0 och sedan hoppar till tillstånd 1, 2 eller 3. De omvandlas till tiderna processen spenderade i varje tillstånd genom att vi tar tiden mellan tillståndsbyten. Tillståndsvektorn state innehåller de tillstånd som processen besökt.

Nu kallar vi på funktionen `bd_process` med 1000 steg och räknar ut andelen tid som processen spenderade i varje tillstånd.

```
# Generera simulerade data för 1000 tillståndsbyten för varje förslag
forslag1_1000 <- bd_process(lambda = 1, mu = 5, steps = 1000)
forslag2_1000 <- bd_process(lambda = 2, mu = 5, steps = 1000)
forslag3_1000 <- bd_process(lambda = 5, mu = 5, steps = 1000)

# Beräkna andelen tid i varje tillstånd för varje förslag
forslag1 <- c(proportion_in_state(0, forslag1_1000),
              proportion_in_state(1, forslag1_1000),
              proportion_in_state(2, forslag1_1000),
              proportion_in_state(3, forslag1_1000))
forslag2 <- c(proportion_in_state(0, forslag2_1000),
              proportion_in_state(1, forslag2_1000),
              proportion_in_state(2, forslag2_1000),
```

```
              proportion_in_state(3, forslag2_1000))
forslag3 <- c(proportion_in_state(0, forslag3_1000),
              proportion_in_state(1, forslag3_1000),
              proportion_in_state(2, forslag3_1000),
              proportion_in_state(3, forslag3_1000))

# Skapa en snygg data.frame med tydliga kolumnnamn och rundade värden
bd_process_1000 <- data.frame(
  State = c(0, 1, 2, 3),
  Förslag1 = round(forslag1, 3),
  Förslag2 = round(forslag2, 3),
  Förslag3 = round(forslag3, 3)
)

# Visa dataframen med knitr::kable
knitr::kable(bd_process_1000, caption = "Andel tid spenderad i varje tillstånd")
```

Table 2: Andel tid spenderad i varje tillstånd

State	Förslag1	Förslag2	Förslag3
0	0.816	0.599	0.261
1	0.141	0.266	0.244
2	0.032	0.092	0.249
3	0.010	0.043	0.246

Ser att förslag 1 och 2 uppfyller institinutionens krav på att vi ska befinna oss i tillstånd 3 mindre än 5% av tiden.

5 Beräkning av stationär fördelning för varje förslag

> 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. För vilka förslag "borde" systemet få godkänt?

Vi betraktar en födelse-dödsprocess med ett begränsat tillståndsrum $\{0, 1, 2, 3\}$ där:

- Vid en födelse (övergång uppåt) sker övergången med intensiteten λ (för de tillstånd där födelse är möjlig) och
- Vid ett dödsfall (övergång nedåt) sker övergången med intensiteten μ (för de tillstånd där dödsfall är möjliga).

Antag att systemet har en stationär fördelning $\pi = (\pi_0, \pi_1, \pi_2, \pi_3)$. För en klassisk födelse-dödsprocess med konstant födelse- och dödsintensitet (och med reflekterande gränser så att födelse inte sker i tillstånd 3 och dödsfall inte sker i tillstånd 0) är den stationära fördelningen given av

$$\pi_i = \frac{\rho^i}{\sum_{j=0}^{3} \rho^j}, \quad i = 0, 1, 2, 3,$$

där

$$\rho = \frac{\lambda}{\mu}.$$

Vi kan då alltså beräkna den stationära fördelningen för varje förslag.

Förslag 1: $\lambda = 1$ och $\mu = 5$ ger

$$\rho = \frac{1}{5} = 0.2.$$

Då blir:

$$\begin{aligned} \pi_0 = \frac{1}{1 + 0.2 + 0.2^2 + 0.2^3} = \frac{1}{1 + 0.2 + 0.04 + 0.008} = \frac{1}{1.248} &\approx 0.8013, \\ \pi_1 = \frac{0.2}{1.248} &\approx 0.1603, \\ \pi_2 = \frac{0.04}{1.248} &\approx 0.0321, \\ \pi_3 = \frac{0.008}{1.248} &\approx 0.00641. \end{aligned}$$

Förslag 2: $\lambda = 2$ och $\mu = 5$ ger

$$\rho = \frac{2}{5} = 0.4.$$

Då får vi:

$$\pi_0 = \frac{1}{1+0.4+0.4^2+0.4^3} = \frac{1}{1+0.4+0.16+0.064} = \frac{1}{1.624} \approx 0.615,$$
$$\pi_1 = \frac{0.4}{1.624} \approx 0.246,$$
$$\pi_2 = \frac{0.16}{1.624} \approx 0.0985,$$
$$\pi_3 = \frac{0.064}{1.624} \approx 0.0394.$$

Förslag 3: $\lambda = 5$ och $\mu = 5$ ger

$$\rho = \frac{5}{5} = 1.$$

Då blir:

$$\pi_i = \frac{1^i}{1+1+1+1} = \frac{1}{4} = 0.25, \quad i = 0, 1, 2, 3.$$

Eftersom alla ρ är 1 så är alla $\pi_i = 0.25$.

Tolkning:

- **Förslag 1** ger att systemet är i tillstånd 0 (tomt) ca 80% av tiden. Det skulle kunna innebära att systemet är undernyttjad och därmed inte så effektiv.
- **Förslag 2** ger en stationär fördelning där:
  - $\pi_0 \approx 61.5\%$ (systemet är tomt en stor del av tiden),
  - $\pi_1 \approx 24.6\%$,
  - $\pi_2 \approx 9.85\%$,
  - $\pi_3 \approx 3.94\%$ (systemet är sällan fullt).
- **Förslag 3** har en jämn fördelning med 25% i varje tillstånd. Dock spenderas 25% av tiden i tillstånd 3, vilket kan innebära att systemet ofta är fullt eller överbelastat.

Ur ett perspektiv där vi vill undvika att systemet är för ofta tomt (ineffektivt) eller överbelastat (risk för att inte kunna hantera ytterligare jobb) är förslag 2 det mest effektiva förslaget. Det resulterar i en låg sannolikhet för att systemet når det högsta tillståndet (3), samtidigt som det inte är övervägande tomt.

Slutsats:
Systemet borde få godkänt för förslag 2 eftersom den stationära fördelningen där $\rho = 0.4$ visar att systemet är relativt välbalanserat, med en låg sannolikhet för överbelastning (endast ca 3.94% i tillstånd 3) och ändå en rimlig användning av resurser.