Modul Praktikum Pemodelan Stokastik: Rantai Markov Waktu Kontinu

Tim Pengampu Mata Kuliah Pemodelan Stokastik

Pertemuan 7

Contents

1. Pendahuluan

Modul ini dirancang untuk mahasiswa agar memahami konsep dasar proses stokastik yang sering muncul pada pemodelan sistem stokastik, khususnya:

- Rantai Markov waktu kontinu (Continuous-Time Markov Chains, CTMC),
- Proses kelahiran dan kematian (birth and death processes),
- Proses kelahiran murni,
- Proses Yule.

Modul ini berperan sebagai panduan praktikum: setiap subbab memuat definisi singkat, konsep matematis utama, implementasi simulasi dengan menggunakan bahasa pemrograman R, estimasi parameter pada model stokastik, perhitungan ekspektasi dan probabilitas yang berhubungan dengan model stokastik, serta soal latihan.

1.1 Tujuan Pembelajaran

Pada akhir praktikum ini mahasiswa diharapkan mampu:

1. Menjelaskan definisi dan sifat dasar CTMC dan proses kelahiran dan kematian.
2. Melakukan simulasi proses stokastik (proses kelahiran murni, proses Yule) dengan R.
3. Melakukan estimasi parameter dari data simulasi.
4. Menghitung ekspektasi dan probabilitas distribusi terkait.
5. Menginterpretasi hasil simulasi dan estimasi dalam konteks terapan.

2. Rantai Markov Waktu Kontinu

2.1 Definisi

Rantai Markov waktu kontinu (CTMC) adalah proses stokastik $\{X(t) : t \geq 0\}$ dengan ruang keadaan diskrit (misal $\{0, 1, 2, \dots\}$) dan memiliki **sifat Markov**: untuk setiap $s, t \geq 0$ dan bilangan bulat non-negatif $i, j, x(u), 0 \leq u < s$ berlaku

$$P[X(s+t) = j | X(s) = i, X(u) = x(u), 0 \leq u < s] = P[X(t+s) = j | X(s) = i]$$

Dengan kata lain, masa depan sistem hanya bergantung pada keadaan saat ini, bukan pada bagaimana sistem mencapai keadaan tersebut.

Lebih lanjut, jika $P[X(t+s) = j | X(s) = i]$ tidak bergantung pada s, maka CTMC tersebut kita sebut memiliki **peluang transisi stasioner atau homogen**. Hal ini berarti bahwa peluang transisi dari state i ke state j hanya bergantung pada interval waktu yang dibutuhkan untuk bertransisi dan peluang ini sama untuk setiap waktu awal s.

2.2 Laju

Perilaku proses Rantai Markov Waktu Kontinu ditentukan oleh **laju proses**. Kita notasikan laju perpindahan dari state i ke state j dengan $v_{ij} \geq 0$ (atau q_{ij} pada beberapa buku lain). Laju keluar total dari keadaan i adalah jumlah dari semua laju transisi individu keluar dari keadaan i:

$$v_i = \sum_{j \neq i} v_{ij}.$$

Jika saat ini pada state i, **waktu tunggu (holding time)** T_i di state i menyatakan lama waktu proses menghabiskan waktu pada state tersebut sebelum bertransisi ke state lain. Waktu tunggu pada state i berdistribusi eksponensial dengan parameter v_i. Oleh karena itu, lama waktu proses berada dalam state i sebelum berpindah ke state lain memiliki ekspektasi (rata-rata):

$$E[T_i] = \frac{1}{v_i}.$$

Misalkan P_{ij} adalah probabilitas bahwa lompatan berikutnya adalah ke keadaan j, jika proses meninggalkan keadaan i. Peluang ini dihitung sebagai rasio laju transisi dari keadaan i ke keadaan j terhadap laju keluar total:

$$P_{ij} = \frac{v_{ij}}{v_i} = \frac{\text{Laju } i \to j}{\text{Laju Keluar Total dari } i}$$

Sifat-sifat utama dari P_{ij}:

- $P_{ii} = 0$ (Lompatan selalu terjadi ke keadaan yang berbeda).
- $\sum_j P_{ij} = 1$ (Proses harus melompat ke salah satu keadaan lainnya).

2.3 Simulasi umum CTMC

Berikut fungsi R sederhana untuk mensimulasikan CTMC dengan waktu terbatas pada beberapa state numerik:

```
library(tidyverse)
library(broom)
library(gridExtra)
library(ggplot2)
set.seed(123)

# simulasikan CTMC dengan keadaan awal X(0) dan total waktu T
simulate_ctmc <- function(Q, x0 = 1, T = 10){
  # Matriks laju transisi Q (Qij menyatakan laju dari state i ke j)
  states <- seq_len(nrow(Q)) - 1
  t <- 0
  # tibble adalah suatu dataframe
  path <- tibble(time = 0, state = x0)
  current <- x0 + 0  # keadaan saat ini
  while(t < T){
    i <- current + 1
    lambda_i <- -Q[i,i]
    if(lambda_i <= 0) break
    waiting_time <- rexp(1, rate = lambda_i)
    if(t + waiting_time > T) break
    probs <- Q[i,]
    probs[i] <- 0
    probs <- probs / sum(probs)
    j <- sample(states, size = 1, prob = probs)
    t <- t + waiting_time
    current <- j
    path <- bind_rows(path, tibble(time = t, state = j))
  }
  return(path)
}

# Contoh matriks transisi dengan state 0 1 2 3
Q <- matrix(0, nrow = 4, ncol = 4)
colnames(Q) <- rownames(Q) <- 0:3
Q["0","1"] <- 1.5
Q["1","0"] <- 0.5; Q["1","2"] <- 1.0
Q["2","1"] <- 0.4; Q["2","3"] <- 0.8
for(i in 1:4) Q[i,i] <- -sum(Q[i,-i])
print(Q)
```

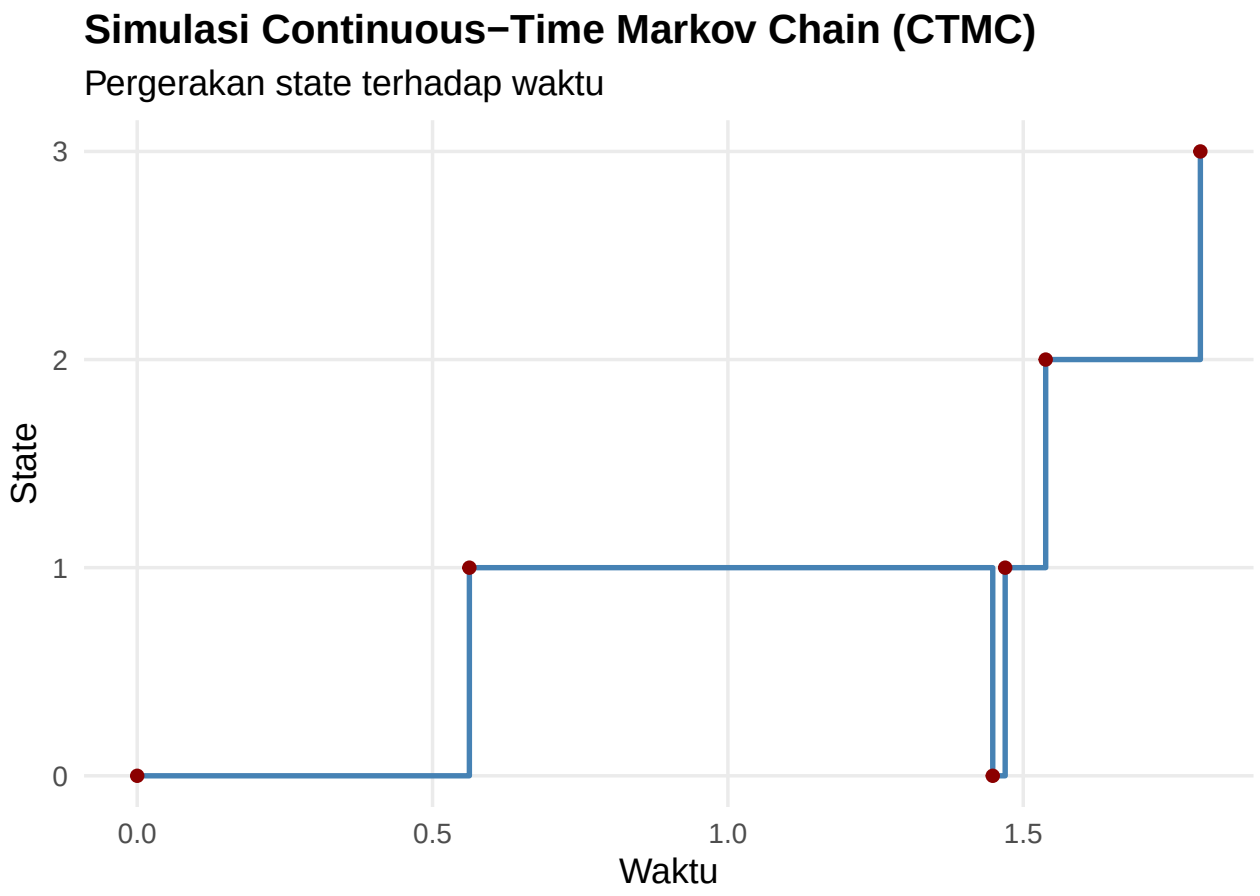
```
##      0    1    2   3
```

```
## 0 -1.5  1.5  0.0 0.0
## 1  0.5 -1.5  1.0 0.0
## 2  0.0  0.4 -1.2 0.8
## 3  0.0  0.0  0.0 0.0
```

```r
# ------------------------------------------------------------
# Hasil simulasi
# ------------------------------------------------------------
sim <- simulate_ctmc(Q, x0 = 0, T = 10)
print(sim)
```

```
## # A tibble: 6 x 2
##    time state
##   <dbl> <dbl>
## 1 0         0
## 2 0.562     1
## 3 1.45      0
## 4 1.47      1
## 5 1.54      2
## 6 1.80      3
```

```r
# ------------------------------------------------------------
# Visualisasi hasil simulasi
# ------------------------------------------------------------
ggplot(sim, aes(x = time, y = state)) +
  geom_step(color = "steelblue", linewidth = 1) +
  geom_point(color = "darkred", size = 2) +
  labs(
    title = "Simulasi Continuous-Time Markov Chain (CTMC)",
    subtitle = "Pergerakan state terhadap waktu",
    x = "Waktu",
    y = "State"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    panel.grid.minor = element_blank()
  )
```


Simulasi Continuous−Time Markov Chain (CTMC)
Pergerakan state terhadap waktu
State
Waktu
0
1
2
3
0.0
0.5
1.0
1.5

3. Proses Kelahiran dan Kematian

3.1 Definisi

Proses kelahiran dan kematian adalah Rantai Markov Waktu Kontinu dengan ruang keadaan bilangan bulat non-negatif $\{0, 1, 2, \dots\}$ dengan transisi terjadi hanya dari state i ke $i+1$ (kelahiran) atau $i-1$ (kematian). Umumnya proses ini merepresentasikan kuantitas seperti ukuran populasi atau jumlah pelanggan. Laju transisi pada proses ini diberikan oleh:

- dari i ke $i+1$ dengan laju λ_i,
- dari i ke $i-1$ dengan laju μ_i.

3.2 Laju

Proses Kelahiran dan Kematian ditandai oleh dua laju yang bergantung pada keadaan: (1) Laju kelahiran λ_i pada keadaan i yang merupakan laju di mana proses bertransisi dari state i ke state $i+1$. Laju ini mewakili kedatangan customer, penambahan, atau peristiwa pertumbuhan. (2) Laju kematian μ_i pada keadaan i yang merupakan laju di mana proses bertransisi dari state i ke state $i-1$. Mewakili kepulangan customer, penghapusan, atau peristiwa penurunan. Kita asumsikan $\mu_0 = 0$ karena proses tidak dapat "mati" ketika populasi sudah nol.

Berikut ini sifat dari laju dan probabilitas transisi pada Proses Kelahiran dan Kematian:

1. Ketika sistem berada di state 0 (misalnya, populasi nol), hanya kelahiran yang dapat terjadi.
$$\mathbf{v_0} = \lambda_\mathbf{0}$$
2. Untuk state apa pun di atas nol, proses dapat mengalami kelahiran (ke $i+1$) atau kematian (ke $i-1$).
$$v_i = \lambda_i + \mu_i, i > 0$$
3. Jika sistem meninggalkan state 0, ia harus bergerak ke state 1 dengan peluang 1.
$$P_{01} = 1$$
4. Peluang terjadinya kelahiran saat sistem berada di state i:
$$P_{i,i+1} = \frac{\lambda_i}{\lambda_i + \mu_i}, i > 0$$
5. Peluang terjadinya kematian saat sistem berada di state i:
$$P_{i,i-1} = \frac{\mu_i}{\lambda_i + \mu_i}, i > 0$$

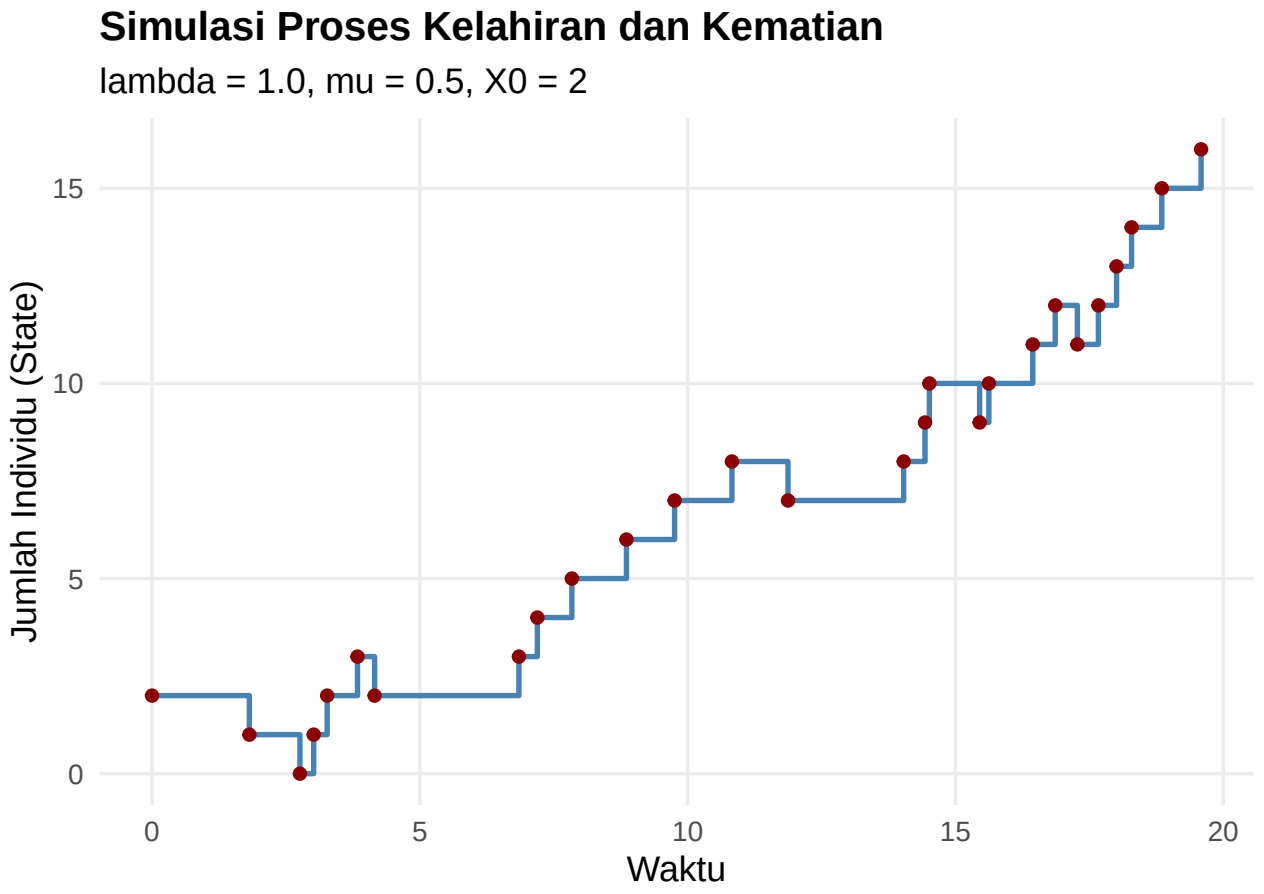
3.3 Simulasi Proses Kelahiran dan Kematian

```r
# Simulasi BnD process dengan laju lahir lambda, laju mati mu
simulate_birth_death <- function(lambda, mu, x0 = 0, T = 10){
  # lambda, mu dapat berupa fungsi atau vektor berdasarkan state
  t <- 0
  state <- x0
  records <- tibble(time = 0, state = state)
  while(t < T){
    lam_i <- if(is.function(lambda)) lambda(state) else
      lambda[min(state+1, length(lambda))]
    mu_i <- if(is.function(mu)) mu(state) else mu[min(state+1, length(mu))]
    total <- lam_i + mu_i
    if(total == 0) break
    waiting_time <- rexp(1, rate = total)
    if(t + waiting_time > T) break
    u <- runif(1)
    if(u < lam_i/total) state <- state + 1 else state <- max(0, state - 1)
    t <- t + waiting_time
    records <- bind_rows(records, tibble(time = t, state = state))
  }
  records
}

# ------------------------------------------------------------
# Contoh simulasi
# ------------------------------------------------------------
sim_bd <- simulate_birth_death(lambda = 1, mu = 0.5, x0 = 2, T = 20)
head(sim_bd, 10)
```

```
## # A tibble: 10 x 2
##     time state
##    <dbl> <dbl>
##  1  0        2
##  2  1.82     1
##  3  2.76     0
##  4  3.02     1
##  5  3.27     2
##  6  3.84     3
##  7  4.16     2
##  8  6.85     3
##  9  7.19     4
## 10  7.84     5
```

```r
# ------------------------------------------------------------
# Visualisasi hasil simulasi
```

```r
# ------------------------------------------------------------
ggplot(sim_bd, aes(x = time, y = state)) +
  geom_step(color = "steelblue", linewidth = 1) +
  geom_point(color = "darkred", size = 2) +
  labs(
    title = "Simulasi Proses Kelahiran dan Kematian",
    subtitle = "lambda = 1.0, mu = 0.5, X0 = 2",
    x = "Waktu",
    y = "Jumlah Individu (State)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    panel.grid.minor = element_blank()
  )
```



4. Proses Kelahiran Murni dengan Laju Konstan

4.1 Definisi

Proses Kelahiran Murni (pure birth process) adalah Proses Kelahiran dan Kematian dengan $\mu_i = 0$ untuk semua i. Artinya hanya ada transisi naik (kelahiran) $i \to i+1$ dan tidak pernah terjadi transisi turun (kematian) $i \to i-1$.

Tinjau Proses Kelahiran Murni dengan laju kelahiran konstan $\lambda_i = \lambda$ untuk semua i. Jika diketahui keadaan awal $X(0) = 0$, maka $X(t) \sim \text{Poisson}(\lambda t)$ — ini adalah proses Poisson standar. Jika diketahui keadaan awal $X(0) = k > 0$ distribusi $X(t)$ mengikuti *Shifted Poisson process* dengan $X(t) = k + N(t)$ di mana $N(t) \sim \text{Poisson}(\lambda t)$. Pada kasus ini, ekspektasi dan varians populasi diberikan oleh

$$E[X(t)|X(0) = k] = k + \lambda t$$

dan

$$\text{Var}(X(t)|X(0) = k) = \lambda t.$$

4.2 Simulasi dengan R

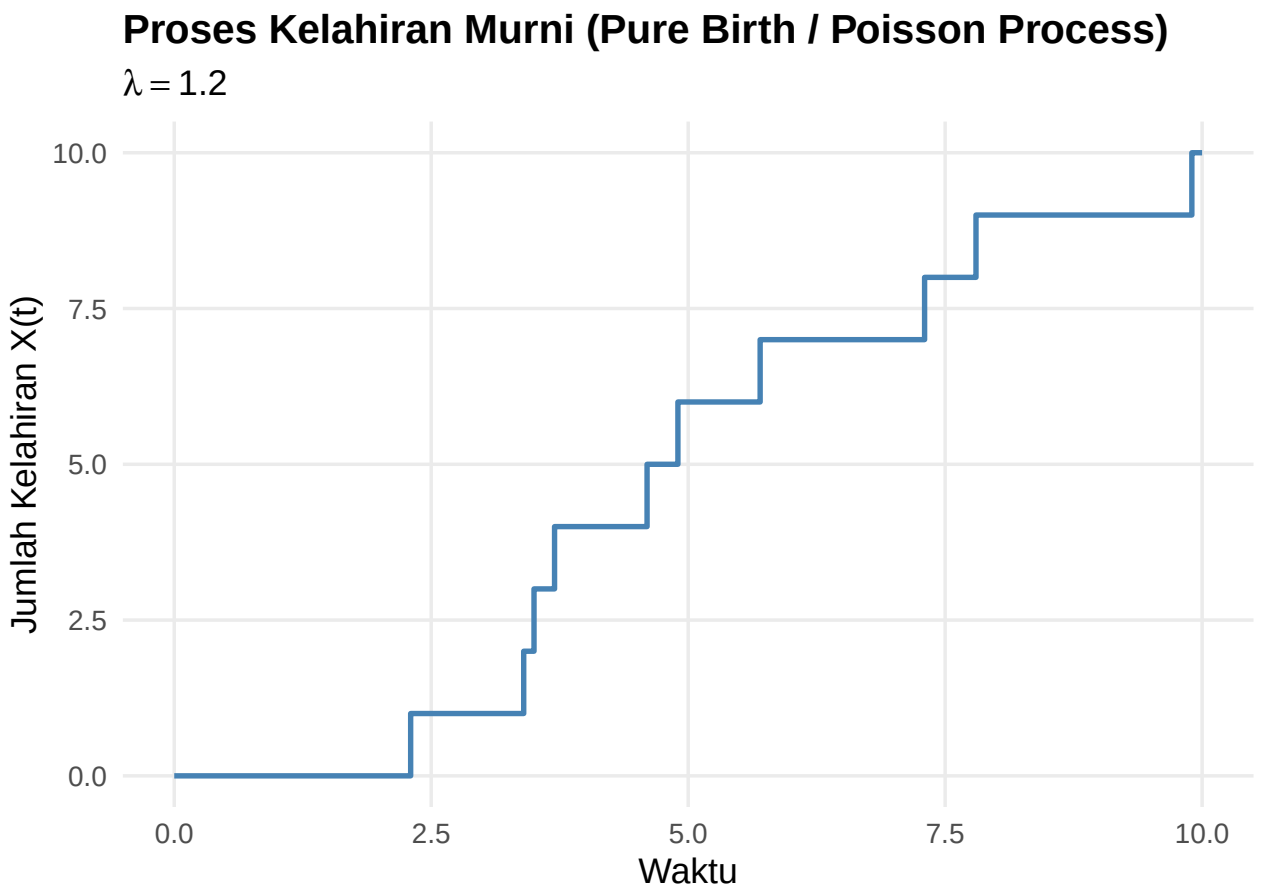
Simulasi dapat dilakukan dengan cara mensimulasikan jumlah kejadian Poisson secara langsung.

```r
# ------------------------------------------------------------
# Simulasi Proses Kelahiran Murni (Pure Birth Process / Poisson Process)
# ------------------------------------------------------------
simulate_pure_birth_poisson <- function(lambda, n0 = 0, T = 10, dt = 0.1) {
  times <- seq(0, T, by = dt)
  # Simulasi waktu antar-kelahiran (exponential waiting times)
  t <- 0
  births <- 0
  event_times <- numeric()
  while (t < T) {
    t <- t + rexp(1, rate = lambda)
    if (t <= T) event_times <- c(event_times, t)
  }

  # Hitung jumlah kelahiran sampai setiap waktu dalam grid
  counts <- numeric(length(times))
  for (k in seq_along(times)) {
    counts[k] <- n0 + sum(event_times <= times[k])
  }

  tibble(time = times, X = counts)
}
```

```r
# ------------------------------------------------------------
# Simulasi
# ------------------------------------------------------------
sim_pb <- simulate_pure_birth_poisson(lambda = 1.2, n0 = 0, T = 10, dt = 0.1)
head(sim_pb)
```

```
## # A tibble: 6 x 2
##    time     X
##   <dbl> <dbl>
## 1   0       0
## 2   0.1     0
## 3   0.2     0
## 4   0.3     0
## 5   0.4     0
## 6   0.5     0
```

```r
# ------------------------------------------------------------
# Plot hasil simulasi
# ------------------------------------------------------------
ggplot(sim_pb, aes(x = time, y = X)) +
  geom_step(color = "steelblue", linewidth = 1) +
  #geom_point(color = "darkred", size = 2) +
  labs(
    title = "Proses Kelahiran Murni (Pure Birth / Poisson Process)",
    subtitle = expression(lambda == 1.2),
    x = "Waktu",
    y = "Jumlah Kelahiran X(t)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 16),
    panel.grid.minor = element_blank()
  )
```



4.3 Estimasi parameter

Kita dapat mengestimasi parameter pada Proses Kelahiran Murni dengan laju konstan dengan menggunakan Maximum Likelihood Estimation (MLE). Jika kita mengamati proses Poisson (pure birth with constant rate) pada interval $[0, T]$ dan mencatat jumlah kejadian $N(T)$, likelihood untuk λ adalah:

$$L(\lambda) = P(N(T) = n) = e^{-\lambda T} \frac{(\lambda T)^n}{n!}.$$

Maximum Likelihood Estimator untuk λ adalah:

$$\hat{\lambda} = \frac{N(T)}{T}.$$

Contoh estimasi dari simulasi yang sudah kita lakukan sebelumnya:

```
# ------------------------------------------------------------
# Estimasi parameter lambda dengan MLE
# ------------------------------------------------------------
T = 10            # interval waktu
lambda_true = 1.2 # laju kelahiran sesungguhnya
```

```r
n0 = 0              # keadaan awal

# Estimator: lambda_hat = N(T) / T
pembilang <- tail(sim_pb$X, 1) - n0 # selisih nilai X(T) awal dan terakhir
lambda_hat <- pembilang / T

cat("Nilai sebenarnya (true lambda):", lambda_true, "\n")
```

```
## Nilai sebenarnya (true lambda): 1.2
```

```r
cat("Estimasi MLE (lambda hat):", lambda_hat, "\n")
```

```
## Estimasi MLE (lambda hat): 1
```

Dapat kita lihat bahwa hasil estimasi parameter lambda yang kita peroleh memiliki nilai yang cukup dekat dengan nilai lambda sesungguhnya.

5. Proses Yule

5.1 Definisi

Proses Yule adalah model pertumbuhan sederhana (sel atau partikel membelah) di mana setiap individu membelah secara independen dengan laju konstan λ. Proses Yule juga dikenal sebagai proses kelahiran murni dengan laju proporsional terhadap populasi, yaitu laju proses keseluruhan diberikan oleh $\lambda_i = i\lambda$.

Jika keadaan awal $X(0) = k$, maka distribusi $X(t)$ adalah **Negative Binomial**. Dengan ekspektasi dan varians diberikan oleh

$$E[X(t)] = k \cdot e^{\lambda t}$$

dan

$$\text{Var}(X(t)|X(0) = k) = k \cdot e^{\lambda t}(e^{\lambda t} - 1).$$

5.2 Simulasi dengan R

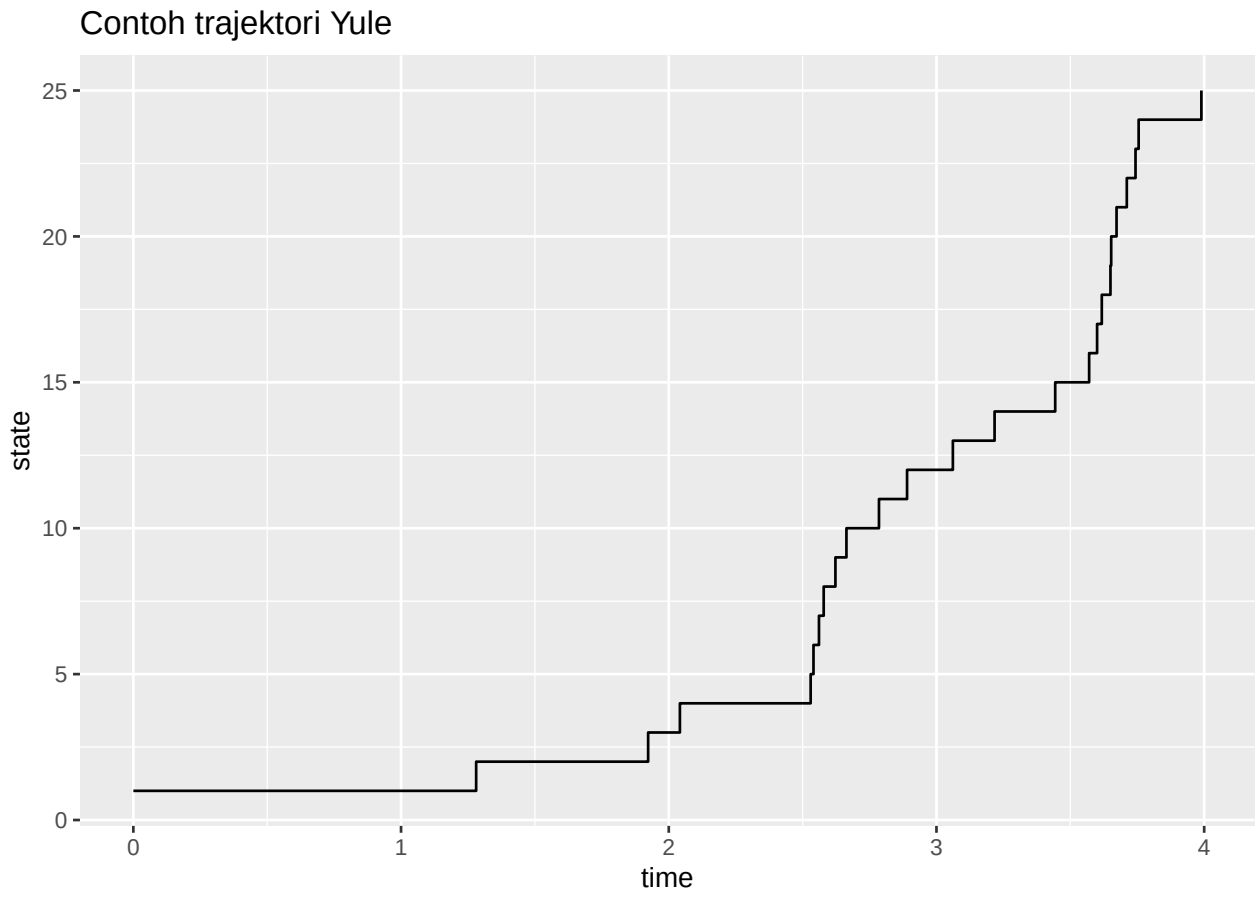
Kita dapat mensimulasikan proses Yule event-by-event: tiap individu membelah, sehingga laju total = current_state * lambda.

```
simulate_yule <- function(lambda, x0 = 1, T = 5){
  t <- 0; state <- x0
  rec <- tibble(time = 0, state = state)
  while(t < T){
    rate <- state * lambda
    wait <- rexp(1, rate = rate)
    t <- t + wait
    if(t > T) break
    state <- state + 1
    rec <- bind_rows(rec, tibble(time = t, state = state))
  }
  rec
}

sim_yule <- simulate_yule(lambda = 0.8, x0 = 1, T = 4)
head(sim_yule, 10)
```

```
## # A tibble: 10 x 2
##     time state
##    <dbl> <dbl>
##  1  0        1
##  2  1.28     2
##  3  1.92     3
##  4  2.04     4
##  5  2.53     5
```

```
##  6  2.54     6
##  7  2.56     7
##  8  2.58     8
##  9  2.62     9
## 10  2.66    10
```

```r
# Sample trajectory plot:
ggplot(sim_yule, aes(x=time, y=state)) +
  geom_step() + labs(title = "Contoh trajektori Yule")
```



5.3 Estimasi parameter

Untuk data yang mengamati jumlah individu pada waktu t, untuk kasus $X(0) = 1$ dan dengan observasi akhir $X(T)$:

$$P(X(T) = k) = e^{-\lambda T}(1 - e^{-\lambda T})^{k-1}.$$

Log-likelihood untuk sampel k adalah

$$\ell(\lambda) = -\lambda T + (k-1)\log(1 - e^{-\lambda T}).$$

Kita dapat maksimumkan log-likelihood ini sebagai berikut.

```
# MLE for lambda given observed k at time T (single trajectory start at 1)
loglik_lambda <- function(lambda, k, T){
  -lambda * T + (k-1) * log(1 - exp(-lambda * T))
}

estimate_lambda_mle <- function(k, T){
  opt <- optimize(function(b) -loglik_lambda(b, k, T), interval = c(1e-6, 5),
                  maximum = FALSE)
  # We'll use optimize to maximize by negating appropriately:
  opt2 <- optimize(function(b) -loglik_lambda(b, k, T), interval = c(1e-6, 5))
  lambda_hat <- opt2$minimum
  lambda_hat
}

# ------------------------------------------------------------
# Estimasi parameter lambda dari hasil simulasi
# ------------------------------------------------------------
Tobs = 5
lambda_true = 0.8

k_obs <- tail(sim_yule$state, 1)  # jumlah individu di akhir waktu T
lambda_hat <- estimate_lambda_mle(k = k_obs, T = Tobs)

cat("Jumlah individu pada waktu T =", k_obs, "\n")
```

```
## Jumlah individu pada waktu T = 25
```

```
cat("lambda sebenarnya:", lambda_true, "\n")
```

```
## lambda sebenarnya: 0.8
```

```
cat("lambda hasil estimasi (MLE):", lambda_hat, "\n")
```

```
## lambda hasil estimasi (MLE): 0.6437706
```

Dapat kita lihat bahwa hasil estimasi parameter lambda yang kita peroleh memiliki nilai yang cukup dekat dengan nilai lambda sesungguhnya.

6. Latihan Praktikum

Berikut beberapa soal praktikum yang harus dikerjakan menggunakan R. Setiap soal disertai petunjuk pemecahan dan solusi R.

Soal 1 — Simulasi CTMC sederhana

Soal: Buat simulasi CTMC untuk 4 state (0,1,2,3) dengan generator matrix (matriks transisi):

$$Q = \begin{pmatrix} -1.5 & 1.5 & 0 & 0 \\ 0.5 & -1.5 & 1.0 & 0 \\ 0 & 0.4 & -1.2 & 0.8 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Mulai dari state 0, simulasikan hingga waktu $T = 20$. Plot trajectory.

Soal 2 — Estimasi lambda pada proses Poisson (pure birth)

Soal: Simulasikan proses Poisson pada interval $[0, 10]$ dengan laju $\lambda = 1.5$. Dari data observasi jumlah kejadian $N(10)$, hitung estimator MLE $\hat{\lambda}$ dan bandingkan dengan nilai sebenarnya.

Soal 3 — Simulasi dan estimasi Proses Yule

Soal: Untuk $\lambda = 0.6$ dan $T = 3$, simulasi trajektori Yule dimulai dari keadaan awal 1. Plot trajektori ambil $X(T)$ dan lakukan estimasi $\hat{\lambda}$ menggunakan metode MLE/pendekatan numerik.

---

7. Kesimpulan

Modul ini memperkenalkan praktek komputasi untuk beberapa proses stokastik dasar: CTMC, birth-death, pure birth (Poisson) dan proses Yule. Anda diminta mengerjakan latihan di atas, memastikan kode berjalan di RStudio, dan mempelajarinya kembali di luar kelas.