Assignment 8

Angewandte Modellierung 25

Carl Colmant

June 27, 2025

Exercise 1.

Um eine Verteilung für den Zug und dne Bus zu berechnen müssen wir zuerst die Ankunftszeiten in Integer Werte umwandeln, dazu habe ich die Zeiten in Minuten umgerechnet. Dann dann habe ich die in der Aufgabe vorgegebenen Werte als Variablen definiert. Der Zug ist nun zu spät wenn der Wert der Zufallsvariable größer ist als 525 (8:45) ist. Außerdem habe ich die Wahrscheinlichkeit für den Bus berechnet, dass dieser for dem Zug abfährt.

```python
from scipy.stats import norm

# Convert times to “minutes after midnight”
mu_train    = 8*60 + 44   # 08:44 → 524 min
sigma_train = 3           # minutes
due_train   = 8*60 + 45   # 08:45 → 525 min

mu_bus      = 8*60 + 50   # 08:50 → 530 min
sigma_bus   = 1           # minutes

# 1) P(train is late) = P(X_train > 525)
p_train_late = 1 - norm.cdf(due_train, loc=mu_train, scale=sigma_train)

# 2) P(bus departs before train arrives) = P(X_train - Y_bus > 0)
mu_diff     = mu_train - mu_bus
sigma_diff  = (sigma_train**2 + sigma_bus**2)**0.5
p_bus_before_train = 1 - norm.cdf(0, loc=mu_diff, scale=sigma_diff)
```

```r
# Convert times to “minutes after midnight”
mu_train    <- 8*60 + 44  # 524
sigma_train <- 3
due_train   <- 8*60 + 45  # 525

mu_bus      <- 8*60 + 50  # 530
sigma_bus   <- 1

# 1) P(train is late)
p_train_late <- 1 - pnorm(due_train, mean = mu_train, sd = sigma_train)

# 2) P(bus leaves before train arrives)
mu_diff      <- mu_train - mu_bus
sigma_diff   <- sqrt(sigma_train^2 + sigma_bus^2)
p_bus_before_train <- 1 - pnorm(0, mean = mu_diff, sd = sigma_diff)

cat("P(train arrives after 08:45) =", round(p_train_late, 4), "\n")
cat("P(bus leaves before train arrives) =", round(p_bus_before_train, 4), "\n")
```

Ausgabe: P(train arrives after 08:45) = 0.3694
P(bus leaves before train arrives) = 0.0289

Exercise 4.

Die Monte Carlo Methode zur Berechnung von π ist relativ simple, man generiert random punkte in einem viertel-kreis und bildet den Anteil der Punkte die im Kreis liegen zu der Gesamtzahl der Punkte. Der Anteil der Punkte im Kreis sollte dann $\frac{\pi}{4}$ sein.

```python
def estimate_pi(n):
    count = 0
    for _ in range(n):
        x = random.random()
        y = random.random()
        if (x - 0.5)**2 + (y - 0.5)**2 <= 0.5**2:
            count += 1
    return 4 * count / n
```

```r
# Übung 4: Monte-Carlo-Schätzung von π
estimate_pi <- function(n) {
  x <- runif(n)
  y <- runif(n)
  inside <- (x - 0.5)^2 + (y - 0.5)^2 <= 0.5^2
  return(mean(inside) * 4)
}
```

Exercise 5.

In dieser Aufgabe soll das Beta-integral approximiert werden. Die Funktion ist gegeben mit

$$B(0.5, 2) = \int_0^1 x^{-0.5}(1-x)^{2-1}dx$$

Als Riemann Summe geschrieben erhalten wir dann:

$$B(0.5, 2) = \sum_{i=0}^{1} x^{-0.5}(1-x)^{2-1} = \frac{1}{N}\sum_{i=0}^{N} x^{-0.5}(1-x)^{2-1}$$

Das bewerkstelligt folgender Python Code:

```python
def estimate_beta(z, w, n):
    total = 0.0
    for _ in range(n):
        x = random.random()
        total += x**(z - 1) * (1 - x)**(w - 1)
    return total / n
```

```r
# Übung 5: Monte-Carlo-Integral für die Beta-Funktion
estimate_beta <- function(z, w, n) {
  x <- runif(n)
  return(mean(x^(z - 1) * (1 - x)^(w - 1)))
}
```

Github

Wie immer sind alle meine benutzten Dateien auf meinem Github zu finden.