Aufgabe zur Warteschlangensimulation

- Diese Aufgabe vereint alle Aspekte, die in dem Abschnitt über Warteschlangen behandelt wurden. Deshalb klingt sie sehr komplex, jedoch lassen sich mit Hilfe der bisherigen Aufgaben viele Teillösungen relativ unkompliziert zur Lösung dieser Aufgabe verbinden.

- Gegenben seien $\lambda = 12, \mu = 13$. Zu simulierende Länge des Stroms $n = 1000$. Dies sei die Anzahl von Anfragen, die pro Stunde eintritt.

- Das System kann höchstens $N = 10$ Anfragen aufnehmen.

- Je nach Schlangenlänge m gelangen neue Anfragen nur gemäß der Scheufunktion

$$G(m) = 1 - (m/N)^2, m \leq N, 0 \text{ sonst}$$

ins System.

- Weiterhin endet jede Anfrage gemäß eines Exponentialprozesses mit einer mittleren Wartezeit von 15 min in der Warteschlange.

- Simulieren Sie diesen Prozess! Gehen Sie dabei schrittweise vor und notieren Sie in natürlicher Sprache, wie die einzelnen Verteilungen ineinandergreifen!

- Welche Intensität (pro Stunde) besitzt der Austrittsprozess?

- Wie lang ist die Schlange im Schnitt?

- Wie groß ist die durchschnittliche Wartezeit im System bis zum Verlassen, entweder nach Bedienung oder nach Abbruch?

Lösung

```
### Initialisierung aller Prozesse. Zeiteinheiten beachten!
n<-250
kapazitaet <- 10
lambda.ankunft <- 12
mu.bearbeitung <- 13
ankunftszeiten <- cumsum(rexp(n, lambda.ankunft))
bearbeitungszeiten <- rexp(n, mu.bearbeitung)
### Mittlere Wartezeit 15 min heißt Intensität 4
zufällige.wartezeiten <- rexp(n,4)
### Scheufunktion
G <- function(m, kapazitaet) if (m <= kapazitaet) 1 - (m/kapazitaet)^2 else 0
```

Simulation eines Schlangendurchlaufs

```
stromsimulation <- function(ankunftszeiten, bearbeitungszeiten,
                                            zufaellige.wartezeiten){
  austrittszeiten <- rep(NA,length(ankunftszeiten))
### Scheufunktion ist 0, da keine Schlange vor dem ersten Kunden
  austrittszeiten[1] <- ankunftszeiten[1] + bearbeitungszeiten[1]
  for (kunde in 2:length(ankunftszeiten)) {
    im.system.bei.ankunft <- 0
    for (i in 1:(kunde-1)){
### Muesste man für große n effizienter gestalten.
### Problem ist, dass die Reihenfolge der Kunden sich durch Abbrecher ändert.
      if( (ankunftszeiten[kunde] < austrittszeiten[i]) &
          (!(ankunftszeiten[i] == austrittszeiten[i]) )){
### Die zweite Bedingung berücksichtigt, dass beim Scheuen keine
### Schlange exisitert.
        im.system.bei.ankunft <-  im.system.bei.ankunft +1
      }
    }
```

```
### Schlangenlänge bei Ankunft  steht fest.
### jetzt Scheufunktion berechnen
    if (runif(1) <= G(im.system.bei.ankunft, kapazitaet)){
### nicht gescheut
      if ((ankunftszeiten[kunde] + zufaellige.wartezeiten[kunde])
                                        < austrittszeiten[kunde-1] ){
### Abbruch wg Ungeduld
        austrittszeiten[kunde] <- ankunftszeiten[kunde] +
                                  zufaellige.wartezeiten[kunde]
      }
      else {
### Normale Bearbeitung wie im Fall der einfachen Schlange
        austrittszeiten[kunde] <- max(ankunftszeiten[kunde],
                                      austrittszeiten[kunde-1]) +
                                      bearbeitungszeiten[kunde]
      }
    }
    else {
```

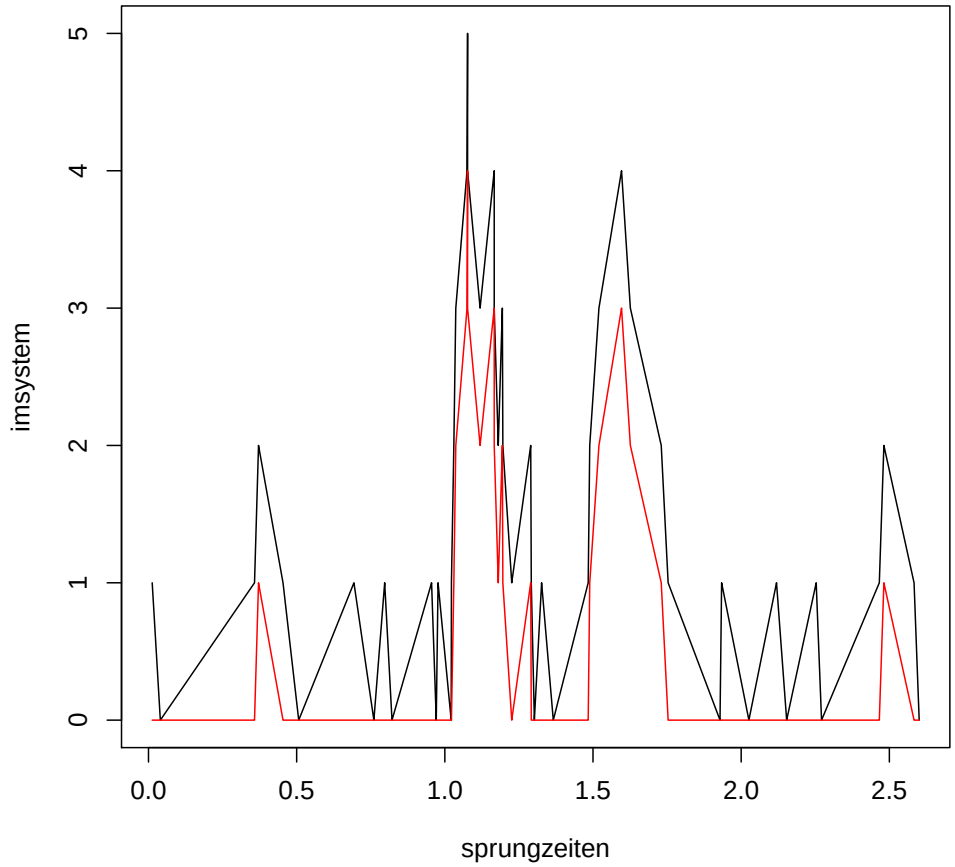
```
### Endzeit ist Ankunftszeit ist auch eindeutig scheuen!
      austrittszeiten[kunde] <- ankunftszeiten[kunde]
    }
  }
  austrittszeiten
}

### Wenn diese Funktion definiert ist, weiter wie im einfachen Fall
exits <- stromsimulation(ankunftszeiten, bearbeitungszeiten,
                         zufaellige.wartezeiten)

rein <- rep(1, n); raus <- rep(-1, n)
allezeiten <- c(ankunftszeiten, exits)
alleaenderungen <- c(rein, raus)
sortierterindex <- sort.int(allezeiten, index.return=TRUE)$ix
sprungzeiten <- allezeiten[sortierterindex]
alleaenderungen <- alleaenderungen[sortierterindex]
imsystem <- cumsum(alleaenderungen)
```

```
inschlange <- imsystem-1
inschlange[inschlange <0] <- 0

plot(sprungzeiten,imsystem, t="l")
lines(sprungzeiten,inschlange, t="l", col="red")
### Und mit den vorliegenden Größen alle interessanten Kennzahlen berechnen
```


imsystem
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
sprungzeiten