Examples of using QueuesFunction

Sunday, October 30, 2016

This file contains examples showing how we can use our function Queues_Function for different simulation studies

Example 1

Suppose we want to compare two Queing Systems, one of them with uniform distributions and the second one with exponential ones. In this example we want to keep 'nonlinear correlation structure' (use the inverse transorfm method we discussed on Friday). Our Queueing Systems consist of 6 queues each. Serving times in the first system follow $U(0,6)$, $U(1,7)$, $U(2,10)$, $U(0,2)$, $U(0,2)$ and $U(0,4)$. respectively, while in the second system they folow $Exp\left(\frac{1}{3}\right)$, $Exp\left(\frac{1}{4}\right)$, $Exp\left(\frac{1}{6}\right)$,$Exp(1)$, $Exp(1)$ and $Exp\left(\frac{1}{2}\right)$, since we want to have comparable means.

```
source('Queues_Function.R')
# fix T and lambda
T = 100
lambda = 0.5

# fix also parameters min and max (of uniform dist.) for the 3 queues:
# in or case unif(0,6), unif(1,7), unif(2,9)
param_unif_df <- data.frame(min = c(0, 1, 2, 0, 0, 0),
                            max = c(6, 7, 10, 2, 2, 4))

# now fix parameters of corresponding parameters for the exponential distributions
# normally we want to have the same mean values of the corresponding queues
param_exp <- sapply(1:nrow(param_unif_df), function(i){
  2/(param_unif_df$max[i] + param_unif_df$min[i])
})

# create a sample from the Poisson process
set.seed(13)
n <- rpois(1, lambda*T)

# remember to make sure n > 0, otherwise there would be no patients
n
```

```
## [1] 53
```

```
vector_arrival_times <- sort(c(0, runif(n - 1, min = 0, max = T)))
```

Now we create a matrix of $U(0,1)$ random variables and we later transform them to 'our' uniform distributions (on different intervals). We do an analogous thing to obtain corresponding exponentially distributed serving times, but here we use the following transformation

$$F^{-1}(u) = -\frac{\log(1-u)}{\lambda}$$

```r
# create a matrix of unif(0,1) random variables
temp_df <- matrix(runif(n*nrow(param_unif_df)), nrow=n)
serving_times_unif <- sapply(seq_len(ncol(temp_df)), function(i){
  param_unif_df$min[i] + temp_df[,i]*(param_unif_df$max[i] - param_unif_df$min[i])
})

serving_times_exp <- sapply(seq_len(ncol(temp_df)), function(i){
  - log(1-temp_df[,i])/param_exp[i]
  })

# comparison of means
colMeans(serving_times_unif)
```

```
## [1] 2.8460556 3.9643917 6.2487375 0.9303375 0.8901944 2.0330434
```

```r
colMeans(serving_times_exp)
```

```
## [1] 2.9134491 3.8466379 7.0531883 0.8275562 0.9244382 2.3747589
```

```r
# performing QueuesFunction on serving_times_unif and serving_times_exp
queue_unif <- QueuesFunction(serving_times_unif, vector_arrival_times)
queue_exp <- QueuesFunction(serving_times_unif, vector_arrival_times)
```

Example 2

Suppose we cannot use inverse transform method, so we don't care about using the same u's from $U(0,1)$. However, we want to have a 6-element queue consisting of serving times with 6 different distributions.

```r
set.seed(13)
# create a sample from the Poisson process
n <- rpois(1, lambda*T)
n
```

```
## [1] 53
```

```r
vector_arrival_times <- sort(c(0, runif(n - 1, min = 0, max = T)))
serving_times <- as.data.frame(cbind(V1 = abs(rcauchy(n, 0, 1)),
                                     V2 = runif(n, 3, 4),
                                     V3 = abs(rnorm(n, 0, 1)),
                                     V4 = rgamma(n, 5, 6),
                                     V5 = rgamma(n, 7, 100),
                                     V6 = abs(rt(n, 1))))

queue_different <- QueuesFunction(serving_times_unif, vector_arrival_times)
summary(queue_different)
```

```
##      Length Class      Mode
## [1,] 3      data.frame list
## [2,] 3      data.frame list
## [3,] 3      data.frame list
```

```
## [4,] 3      data.frame list
## [5,] 3      data.frame list
## [6,] 3      data.frame list
```