

Package ‘DES’

September 16, 2017

Version 1.0.0

Author Norm Matloff <normmatloff@gmail.com>

Maintainer Norm Matloff <normmatloff@gmail.com>

Date 2017-9-13

Title Discrete Event Simulation

Description Discrete event simulation (DES) involves modeling of systems having discrete, i.e. abrupt, state changes. For instance, when a job arrives to a queue, the queue length abruptly increases by 1. This package is an R implementation of the event-oriented approach to DES; see the tutorial in Matloff (2008)
<<http://heather.cs.ucdavis.edu/~matloff/156/PLN/DESimIntro.pdf>>.

Depends stats,utils

Imports

Suggests

LazyLoad no

License MIT + file LICENSE

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-16 20:11:57 UTC

R topics documented:

newsim,schedevnt,getnextevnt,mainloop,newqueue,appendfcfs,delfcfs,cancelevnt,exparrivals 2

Index

5

newsim,schedevnt,getnextevnt,mainloop,newqueue,appendfcfs,delfcfs,cancelevnt,exparrivals
Discrete-event simulation routines.

Description

Main simulation routines.

Usage

```
newsim(timelim,maxesize,appcols=NULL,aevntset = FALSE,dbg=FALSE)
schedevnt(simlist,evnttime,evnttype,appdata=NULL)
getnextevnt(simlist)
mainloop(simlist)
newqueue(simlist)
appendfcfs(queue,jobtoqueue)
delfcfs(queue)
cancelevnt(rownum,simlist)
exparrivals(simlist,meaninterarr,batchsize = 10000)
```

Arguments

<code>appcols</code>	Names of columns in the event set for application-specific data.
<code>aevntset</code>	If TRUE, <code>exparrivals</code> will be used for arrivals and an arrivals event set will be maintained.
<code>dbg</code>	If TRUE, use debug mode, action pausing for each new event occurrence.
<code>simlist</code>	An R environment containing the simulation, produced by <code>newsim</code> .
<code>evnttime</code>	Occurrence time for an event.
<code>evnttype</code>	Event type.
<code>appdata</code>	Application-specific data.
<code>timelim</code>	Time limit for simulation.
<code>maxesize</code>	Maximum number of rows needed in the event set matrix, excluding separate arrival event rows in the case <code>aevntset = TRUE</code> . (The matrix can be expanded dynamically if needed.)
<code>queue</code>	A queue. Must be in a <code>simlist</code> environment.
<code>jobtoqueue</code>	Job to be placed in a queue.
<code>rownum</code>	Number of the row to be deleted from the event set.
<code>meaninterarr</code>	Mean time between arrivals.
<code>batchsize</code>	Number of arrivals to generate in one call to <code>rexp</code> .

Details

Discrete event simulation, using the event-oriented approach.

Here is an overview of the functions:

- `newsim`: Creates an R environment, containing the event list, current simulated time and so on, including any application-specific data.
- `cancelevnt`: Removes an event from the event set Useful for instance for simulating timeout situations. Removal is done via setting the event time to double `timelim`.
- `schedevnt`: Creates a new event, and then enters it into the event set matrix.
- `getnextevnt`: Removes and returns the earliest event from the event set. Removal is done via setting the event time to double `timelim`.
- `mainloop`: Called by the application to start the simulation and run until the simulated time exceeds the user-specified time limit. At each iteration, calls `getnextevnt` and invokes the application-specific reaction function for the occurred event. If `dbg` is set, then at each iteration the function will enter R browser mode, printing out the current event and simulated time, and giving the user an opportunity to "take a look around."
- `newqueue`: Create a new work queue, an R environment. The main component, `m`, is a matrix representing the queue, with number of columns being application-dependent. The user might add other components, e.g. running totals.
- `appendfcfs`: Appends a job to a First Come, First Served queue. The job is represented by a vector to be added as a row in the queue matrix.
- `delfcfs`: Deletes and returns the head of an FCFS queue.

Reaction Functions

These are user-defined. The DES function `mainloop` will make the call

```
simlist$reactevent(head, simlist)
```

where the user has initially set `simlist$reactevent` to his/her application-specific code. Here `head` is the event just now removed from the head of the event set, and `simlist` is the event set. Let's call this function the "event handler," but note that within it there are if/else cases, one for each event type.

The For example, consider simulation of a single-server queue. When a job arrives, the arrivals section of the event handler will run (coded by the event type, again user-defined). It will record the arrival, update any application-specific totals, and see if service can be started for this job. If so, the code will schedule an event for completion of the service; if not, the code will add the job to the queue.

Outline of Typical Application Code

```
mysim <- newsim()    # create the simlist
set reactevent in mysim
set application-specific variables in mysim, if any
set the first event(s) in mysim$evnts
mainloop(mysim,mysimtimelim)
print results
```

4 *newsim,schedevnt,getnextevnt,mainloop,newqueue,appendfcfs,delfcfs,cancelevnt,exparrivals*

Author(s)

Norm Matloff

Examples

```
# from MachRep.R in examples/  
  
# create a sim list that will run for 100000 simulated time, with 3  
# rows allocated for the event set, and application-specific columns  
# named 'startqtime' and 'startuptime'  
simlist <- newsim(100000,3,appcols=c('startqtime','startuptime'))  
# create a queue  
simlist$queue <- newqueue(simlist)
```

Index

appendfcfs
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

cancelvnt
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

delfcfs
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

exparrivals
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

getnextevnt
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

mainloop
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

newqueue
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

newsim
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)

newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals,
[2](#)

schedevnt
(newsim, schedevnt, getnextevnt, mainloop, newqueue, appendfcfs, delfcfs, cancelvnt, exparrivals),
[2](#)