Package ‘queuecomputer’

April 9, 2021

Title Computationally Efficient Queue Simulation

Version 1.1.0

Description Implementation of a computationally efficient method for
simulating queues with arbitrary arrival and service times.
Please see Ebert, Wu, Mengersen & Ruggeri (2020, <doi:10.18637/jss.v095.i05>)
for further details.

Depends R (>= 3.6)
License GPL-2 | file LICENSE
Encoding UTF-8
Imports stats, Rcpp, tidyr, dplyr, utils
RoxygenNote 7.1.1
Suggests knitr, rmarkdown, testthat, ggplot2
VignetteBuilder knitr
LinkingTo Rcpp, RcppArmadillo (>= 0.7.500.0.0)

URL https://github.com/AnthonyEbert/queuecomputer

Copyright file COPYRIGHTS

R topics documented:

as.server.list .......................................................... 2
as.server.stepfun ...................................................... 3
average_queue .......................................................... 4
as.server.list

Creates a "server.list" object from a list of times and starting availability.

Usage

as.server.list(times, init)

Arguments

times list of numeric vectors giving change times for each server.
init vector of 1s and 0s with equal length to times. It represents whether the server
starts in an available (1) or unavailable (0) state.

Value

an object of class "server.list", which is a list of step functions of range {0, 1}.

See Also

as.server.stepfun, queue_step

Examples

# Create a server.list object with the first server available anytime before time 10,
# and the second server available between time 15 and time 30.
as.server.list(list(10, c(15,30)), c(1,0))
as.server.stepfun

Create a \texttt{server.stepfun} object with a roster of times and number of available servers.

Description

Create a \texttt{server.stepfun} object with a roster of times and number of available servers.

Usage

\texttt{as.server.stepfun(x, y)}

Arguments

\begin{itemize}
\item \texttt{x} numeric vector giving the times of changes in number of servers.
\item \texttt{y} numeric vector one longer than \texttt{x} giving the number of servers available between \texttt{x} values.
\end{itemize}

Details

This function uses the analogy of a step function to specify the number of available servers throughout the day. It is used as input to the \texttt{queue_step} function. Alternatively one may use \texttt{as.server.list} to specify available servers as a list, however \texttt{queue_step} is much faster when \texttt{as.server.stepfun} is used as input rather than \texttt{as.server.list}.

If any of the service times are large compared to any element of \texttt{diff(x)} then the \texttt{as.server.list} function should be used.

Value

A list and \texttt{server.stepfun} object with \texttt{x} and \texttt{y} as elements.

See Also

\texttt{as.server.list, queue_step, stepfun}.

Examples

\begin{verbatim}
servers <- as.server.stepfun(c(15, 30, 50), c(0, 1, 3, 2))
servers
\end{verbatim}
average_queue

*Compute time average queue length*

**Description**

Compute time average queue length

**Usage**

average_queue(times, queuelength)

**Arguments**

times numeric vector of times
queuelength numeric vector of queue lengths

**Examples**

```r
n <- 1e3
arrivals <- cumsum(rexp(n))
service <- rexp(n)
departures <- queue(arrivals, service, 1)

queuedata <- queue_lengths(arrivals, service, departures)
average_queue(queuedata$times, queuedata$queuelength)
```

depart

*get departure times from queue_list object*

**Description**

get departure times from queue_list object

**Usage**

depart(x)

**Arguments**

x an queue_list object

**Value**

departure times
**lag_step**

Examples

```r
arrivals <- cumsum(rexp(10))
service <- rexp(10)
queue_obj <- queue_step(arrivals, service)

depart(queue_obj)
queue_obj$departures_df$departures
```

---

**Description**

Add lag to vector of arrival times.

**Usage**

```r
lag_step(arrivals, service)
```

**Arguments**

- `arrivals`: Either a numeric vector or an object of class `queue_list`. It represents the arrival times.
- `service`: A vector of service times with the same ordering as `arrivals`

**Value**

A vector of response times for the input of arrival times and service times.

**See Also**

`wait_step, queue_step`

**Examples**

```r
# Create arrival times
arrivals <- rlnorm(100, meanlog = 3)

# Create service times
service <- rlnorm(100)
lag_step(arrivals = arrivals, service = service)

# lag_step is equivalent to queue_step with a large number of queues, but it's faster to compute.
cbind(queue(arrivals, service = service, servers = 100),
      lag_step(arrivals = arrivals, service = service))
```
plot.queue_list

ggplot2 method for output from queueing model

Description

ggplot2 method for output from queueing model

Usage

## S3 method for class 'queue_list'
plot(x, which = c(2:6), annotated = TRUE, ...)

Arguments

x
an object of class queue_list

which
Numeric vector of integers from 1 to 6 which represents which plots are to be created. See examples.

annotated
logical, if TRUE annotations will be added to the plot.

... other parameters to be passed through to plotting functions.

Examples

if(require(ggplot2, quietly = TRUE)){

n_customers <- 50
arrival_rate <- 1.8
service_rate <- 1
arrivals <- cumsum(rexp(n_customers, arrival_rate))
service <- rexp(n_customers, service_rate)
queue_obj <- queue_step(arrivals, service, servers = 2)
plot(queue_obj)
}

## Not run:

library(ggplot2)

## density plots of arrival and departure times
plot(queue_obj, which = 1)

## histograms of arrival and departure times
plot(queue_obj, which = 2)

## density plots of waiting and system times
plot(queue_obj, which = 3)

## step function of queue length
print.summary_queue_list

plot(queue_obj, which = 4)
## line range plot of customer and server status
plot(queue_obj, which = 5)
## empirical distribution plot of arrival and departure times
plot(queue_obj, which = 6)

## End(Not run)

---

print.summary_queue_list

Print method for output of summary.queue_list.

Description

Print method for output of summary.queue_list.

Usage

## S3 method for class 'summary_queue_list'
print(x, ...)

Arguments

x

an object of class summary_queue_list, the result of a call to summary.queue_list().

...

further arguments to be passed to or from other methods.

Value

A list of performance statistics for the queue:

"Total customers": Total customers in simulation,
"Missed customers": Customers who never saw a server,
"Mean waiting time": The mean time each customer had to wait in queue for service,
"Mean response time": The mean time that each customer spends in the system (departure time - arrival time),
"Utilization factor": The ratio of available time for all servers and time all servers were used. It can be greater than one if a customer arrives near the end of a shift and keeps a server busy,
"Mean queue length": Average queue length, and
"Mean number of customers in system": Average number of customers in queue or currently being served.
Examples

```r
n <- 1e3
arrivals <- cumsum(rexp(n, 1.8))
service <- rexp(n)

queue_obj <- queue_step(arrivals, service, servers = 2)
summary(queue_obj)
```

### ql_summary

**Summarise queue lengths**

**Description**

Summarise queue lengths

**Usage**

```r
ql_summary(times, queuelength)
```

**Arguments**

- `times`: numeric vector of times
- `queuelength`: numeric vector of queue lengths

**Examples**

```r
n <- 1e3
arrivals <- cumsum(rexp(n))
service <- rexp(n)
departures <- queue(arrivals, service, 1)

queuedata <- queue_lengths(arrivals, service, departures)
ql_summary(queuedata$times, queuedata$queuelength)
```

### queue

*Compute the departure times for a set of customers in a queue from their arrival and service times.*

**Description**

`queue` is a faster version of `queue_step` but the input returned is much simpler. It is not compatible with the `summary.queue_list` method or the `plot.queue_list` method.

**Usage**

```r
queue(arrivals, service, servers = 1, serveroutput = FALSE)
```
Arguments

arrivals  numeric vector of non-negative arrival times
service  numeric vector of non-negative service times
servers  a non-zero natural number, an object of class `server.stepfun` or an object of class `server.list`.
serveroutput  boolean whether the server used by each customer should be returned.

Details

If the arrival vector is out of order the function will reorder it. The same reordering will be applied to the service vector, this is so each customer keeps their service time. Once the queue is computed the original order is put back.

See Also

`queue_step`

Examples

```r
n <- 1e2
arrivals <- cumsum(rexp(n, 1.8))
service <- rexp(n)

departures <- queue(arrivals, service, servers = 2)

head(departures)
curve(ecdf(departures)(x) * n,
   from = 0, to = max(departures),
   xlab = "Time", ylab = "Number of customers")
curve(ecdf(arrivals)(x) * n,
   from = 0, to = max(departures),
   col = "red", add = TRUE)
```

---

`queue_lengths`  
*Compute queue lengths from arrival, service and departure data*

Description

Compute queue lengths from arrival, service and departure data

Usage

`queue_lengths(arrivals, service = 0, departures, epsilon = 1e-10, ...)`
Arguments

arrivals vector of arrival times
service vector of service times. Leave as zero if you want to compute the number of customers in the system rather than queue length.
departures vector of departure times
epsilon numeric small number added to departures to prevent negative queue lengths
... additional arguments - does nothing, for compatibility

Examples

library(dplyr)
library(queuecomputer)

set.seed(1L)
n_customers <- 100

queueoutput_df <- data.frame(
  arrivals = runif(n_customers, 0, 300),
  service = rexp(n_customers)
)

queueoutput_df <- queueoutput_df %>% mutate(
  departures = queue(arrivals, service, servers = 2)
)

queue_lengths(
  queueoutput_df$arrivals,
  queueoutput_df$service,
  queueoutput_df$departures
)

# The dplyr way
queueoutput_df %>% do(
  queue_lengths(.arrivals, .service, .departures))

n_customers <- 1000

queueoutput_df <- data.frame(
  arrivals = runif(n_customers, 0, 300),
  service = rexp(n_customers),
  route = sample(c("a", "b"), n_customers, TRUE)
)

server_df <- data.frame(
  route = c("a", "b"),
  servers = c(2, 3)
)

output <- queueoutput_df %>%
  left_join(server_df) %>%
  ...
queue_step

```r

group_by(route) %>%
mutate(
  departures = queue(arrivals, service, servers = servers[1])
) %>%
do(queue_lengths(.$arrivals, .$service, .$departures))

if(require(ggplot2, quietly = TRUE)){
  ggplot(output) +
  aes(x = times, y = queuelength) + geom_step() +
  facet_grid(~route)
}
```

---

**queue_step**

*Compute the departure times and queue lengths for a queueing system from arrival and service times.*

**Description**

Compute the departure times and queue lengths for a queueing system from arrival and service times.

**Usage**

```r
queue_step(arrivals, service, servers = 1, labels = NULL)
```

**Arguments**

- `arrivals`: numeric vector of non-negative arrival times
- `service`: numeric vector of service times with the same ordering as arrival_df.
- `servers`: a non-zero natural number, an object of class server.stepfun or an object of class server.list.
- `labels`: character vector of customer labels.

**Value**

A vector of response times for the input of arrival times and service times.

**See Also**

`queue`, `summary.queue_list`, `plot.queue_list`
Examples

# With two servers
set.seed(1)
n <- 100

arrivals <- cumsum(rexp(n, 3))
service <- rexp(n)

queue_obj <- queue_step(arrivals,
    service = service, servers = 2)

summary(queue_obj)
plot(queue_obj, which = 5)

# It seems like the customers have a long wait.
# Let's put two more servers on after time 20

server_list <- as.server.stepfun(c(20), c(2,4))

queue_obj2 <- queue_step(arrivals,
    service = service,
    servers = server_list)

summary(queue_obj2)
if(require(ggplot2, quietly = TRUE)){
    plot(queue_obj2, which = 5)
}

summary.queue_list  

Summary method for queue_list object

Description

Summary method for queue_list object

Usage

## S3 method for class 'queue_list'
summary(object, ...)

wait_step

Arguments

object: an object of class queue_list, the result of a call to queue_step.

... further arguments to be passed to or from other methods.

Description

Compute maximum time for each row from two vectors of arrival times.

Usage

wait_step(arrivals, service)

Arguments

arrivals: Either a numeric vector or an object of class queue_list. It represents the arrival times.

service: A vector of times which represent the arrival times of the second type of customers. The ordering of this vector should have the same ordering as arrivals.

Details

A good real-world example of this is finding the departure times for passengers after they pick up their bags from the baggage carousel. The time at which they leave is the maximum of the passenger and bag arrival times.

Value

The maximum time from two vectors of arrival times.

See Also

lag_step, queue_step.

Examples

```r
set.seed(500)
arrivals <- rlnorm(100, meanlog = 4)
service <- rlnorm(100)

#Airport example ------------------------

# Create a number of bags for each of 100 customers
bags <- rpois(100,1)
```
# Create a bags dataframe, with each bag associated with one customer.

```r
bags.df <- data.frame(BagID = 1:sum(bags),
                       ID = rep(1:100, bags), times = rlnorm(sum(bags), meanlog = 2))
```

# Create a function which will return the maximum time from each customer’s set of bags.

```r
reduce_bags <- function(bagdataset, number_of_passengers){
  ID = NULL
  times = NULL

  zerobags <- data.frame(BagID = NA, ID = c(1:number_of_passengers), times = 0)
  reduced_df <- as.data.frame(dplyr::summarise(dplyr::group_by(
    rbind(bagdataset, zerobags), ID), n = max(times, 0)))
  ord <- order(reduced_df$ID)
  reduced_df <- reduced_df[order(ord),]
  names(reduced_df) <- c("ID", "times")
  return(reduced_df)
}
```

```r
arrivals2 <- reduce_bags(bags.df, 100)$times
```

# Find the time when customers can leave with their bags.

```r
wait_step(arrivals = arrivals, service = arrivals2)
```
Index

as.server.list, 2, 3
as.server.stepfun, 2, 3
average_queue, 4
depart, 4
lag_step, 5, 13
plot.queue_list, 6, 11
print.summary_queue_list, 7
ql_summary, 8
queue, 8, 11
queue_lengths, 9
queue_step, 2, 3, 5, 9, 11, 13
stepfun, 3
summary.queue_list, 11, 12
wait_step, 5, 13