

# Package ‘queuecomputer’

May 9, 2026

**Title** Computationally Efficient Queue Simulation

**Version** 1.2.0

**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](https://doi.org/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** 7.1.2

**Suggests** knitr, rmarkdown, testthat, ggplot2

**VignetteBuilder** knitr

**LinkingTo** Rcpp, RcppArmadillo (>= 0.7.500.0.0)

**URL** <https://github.com/AnthonyEbert/queuecomputer>

**Copyright** file COPYRIGHTS

**NeedsCompilation** yes

**Author** Anthony Ebert [aut, cre] (ORCID: <<https://orcid.org/0000-0003-3002-6300>>),  
Kerrie Mengersen [ths],  
Paul Wu [ths],  
Fabrizio Ruggeri [ths]

**Maintainer** Anthony Ebert <[anthonyebert+CRAN@gmail.com](mailto:anthonyebert+CRAN@gmail.com)>

**Repository** CRAN

**Date/Publication** 2022-11-15 21:20:04 UTC

## Contents

as.server.list . . . . .	2
as.server.stepfun . . . . .	3

average_queue . . . . .	4
depart . . . . .	4
lag_step . . . . .	5
plot.queue_list . . . . .	6
print.summary_queue_list . . . . .	7
ql_summary . . . . .	8
queue . . . . .	8
queue_lengths . . . . .	9
queue_step . . . . .	11
summary.queue_list . . . . .	12
wait_step . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

---

as.server.list	<i>Creates a "server.list" object from a list of times and starting availability.</i>
----------------	---

---

## Description

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	<i>Create a server.stepfun object with a roster of times and number of available servers.</i>
-------------------	---

---

### Description

Create a server.stepfun object with a roster of times and number of available servers.

### Usage

```
as.server.stepfun(x, y)
```

### Arguments

x	numeric vector giving the times of changes in number of servers.
y	numeric vector one longer than x giving the number of servers available between x values.

### 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 [queue\\_step](#) function. Alternatively one may use `as.server.list` to specify available servers as a list, however `queue_step` is much faster when `as.server.stepfun` is used as input rather than `as.server.list`.

If any of the service times are large compared to any element of `diff(x)` then the [as.server.list](#) function should be used.

### Value

A list and server.stepfun object with x and y as elements.

### See Also

[as.server.list](#), [queue\\_step](#), [stepfun](#).

### Examples

```
servers <- as.server.stepfun(c(15,30,50), c(0, 1, 3, 2))
servers
```

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**

```
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

**Examples**

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

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

---

lag_step	<i>Add lag to vector of arrival times.</i>
----------	--

---

**Description**

Add lag to vector of arrival times.

**Usage**

```
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**

```
# 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

```
## Not run:  
  
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)  
  
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  
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**

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

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

---

ql_summary	<i>Summarise queue lengths</i>
------------	--------------------------------

---

**Description**

Summarise queue lengths

**Usage**

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

**Arguments**

times	numeric vector of times
queuelength	numeric vector of queue lengths

**Examples**

```
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	<i>Compute the departure times for a set of customers in a queue from their arrival and service times.</i>
-------	--

---

**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**

```
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 <code>server.stepfun</code> or an object of class <code>server.list</code> .
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**

```
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) %>%

```

```

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	<i>Compute the departure times and queue lengths for a queueing system from arrival and service times.</i>
------------	--

---

### Description

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

### Usage

```
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 <code>server.stepfun</code> or an object of class <code>server.list</code> .
labels	character vector of customer labels (deprecated).

### Details

If only departure times are needed, the [queue](#) function is faster.

### Value

An list object of class `queue_list` with the following components:

- `departures` - A vector of response times for the input of arrival times and service times.
- `server` - A vector of server assignments for the input of arrival times and service times.
- `departures_df` - A data frame with arrivals, service, departures, waiting, system time, and server assignments for each customer.

- queuelength\_df - A data frame describing the evolution of queue length over time
- systemlength\_df - A data frame describing the evolution of system length over time
- servers\_input - A copy of the server argument
- state - A vector of availability times for the servers

### 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)
}
```

**Description**

Summary method for queue\_list object

**Usage**

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

**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.

---

wait_step	<i>Compute maximum time for each row from two vectors of arrival times.</i>
-----------	---

---

**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**

```

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.
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.

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)
}

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

# Find the time when customers can leave with their bags.
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](#), [12](#)  
`print.summary_queue_list`, [7](#)  
  
`ql_summary`, [8](#)  
`queue`, [8](#), [11](#), [12](#)  
`queue_lengths`, [9](#)  
`queue_step`, [2](#), [3](#), [5](#), [9](#), [11](#), [13](#)  
  
`stepfun`, [3](#)  
`summary.queue_list`, [12](#), [12](#)  
  
`wait_step`, [5](#), [13](#)