Assignment in Discrete-Event Simulation

Philipp Kempter, Michael Scheidegger, Martin Peter Schmitz

03/06/2020

Starting Situation and Work Order

Model and analyze the following situation using a simulation tool (simmer or something else). In a system, customers arrive with a specific distribution:
a) with fixed interarrival times of 90 sec;
b) with interarrival times uniformly distributed between 60 and 120 sec;
c) with exponentially distributed interarrival times with expected value 90 sec.

Each of the customers needs to travel to one service station. The distance is 15m and for the speed 0.6 m/s can be assumed. In total, there are three service stations available for customers.

Consider the following two types of assigning service stations to customers: 1. a round robin method (first customer travels to first service station, second customer to second service station, third customer to third service station, fourth customer to the first service station, etc.) 2. the service stations have a common queue and the next customer is served by the service station which becomes first available (or by an arbitrary service station in case of more than one being available).
Distinguish the two cases that the common queue is before the way to the stations (2a) or after it, i.e. directly in front of the service stations (2b).

Assume that the service stations require service times, which are uniformly distributed between 220 sec and 300 sec. Further specifications of the system are not given and can freely be chosen. Analyze all 9 combinations of the above assumptions (i.e. a1, b1, c1, a2a, b2a, c2a, a2b, b2b, and c2b) by simulation runs of about 4 hours.
During the simulations check for situations of congestion and try to answer the following questions:
-How many customers are served on average during an hour at each of the service stations?
-What is the degree of utilization at each of the service stations?
-What is the average length of the queues?
-What is the average throughput time of the customers?
Please provide a written report of your results. You should describe as well the main modeling concepts (e.g. design of the system and used elements) including code and problems observed during modeling and simulation runs (if there were any).

Preface

To fulfull the assignment the following scripts cover the different assumptions step by step. Therefore, the simulation of the different cases is performed by using simmer. In the end the results of the different simulations are analyzed and compared.

```
set.seed(63)
library(simmer)
library(simmer.plot)
library(dplyr)
library(ggplot2)
library(knitr)
library(forestmangr)
library(data.table)

result_header <- c("customer_per_hour_per_station","degree_of_utilization_1","degree_
of_utilization_2"  ,"degree_of_utilization_3", "average_queue_length_1", "average_que
ue_length_2",
                   "average_queue_length_3","avg_customer_through_put_time")
```

Main Modeling Concepts

The main concepts of the modeling of the simulation are described briefly in this section. To represent the circumstances of the interarrival times of a, b and c the following values were chosen to spawn the patients during simulation:

- For the fixed interarrival times of 90 seconds:
  - `add_generator("Customer", customer, from_to(0, 14400, function() {90}))`
  - => spawning a customer every 90 seconds during the simulation from 0 to 14400 seconds (4hours)
- For the interarrival times uniformly distributed between 60 and 120 seconds:
  - `add_generator("Customer", customer, from_to(0, 14400, function() {runif(1, min = 60, max = 120)}))`
  - => spawning a customer uniformly distributed between 60 and 120 seconds during the simulation from 0 to 14400 seconds (4hours)
- For exponentially distributed interarrival times with an expected value of 90 seconds:
  - `add_generator("Customer", customer, from_to(0, 14400, function() {rexp(1, rate = 1/90)}))`
  - => spawning a customer exponentially distributed with the expected value of 90 during the simulation from 0 to 14400 seconds (4hours)

In case the customers have to get to the service station and overcome the distance of 15 meters with 6 m/s, a timeout of 25 seconds for this action is added at the corresponding position: `timeout(function() {25}) %>%`. The respective positions of the timeout can be found in the representation of the particular case.

Ther time of the service itself is implemented as a uniformly distributed timeout between 220 and 300: `timeout(function() {runif(1, min=220, max=300)}) %>%`.

The difference in the implementation between the cases in 1 and the cases 2A and 2B consists of 3 real resources with an own queue each in 1:

```
add_resource("service_station1", 1) %>%
add_resource("service_station2", 1) %>%
add_resource("service_station3", 1) %>%
```

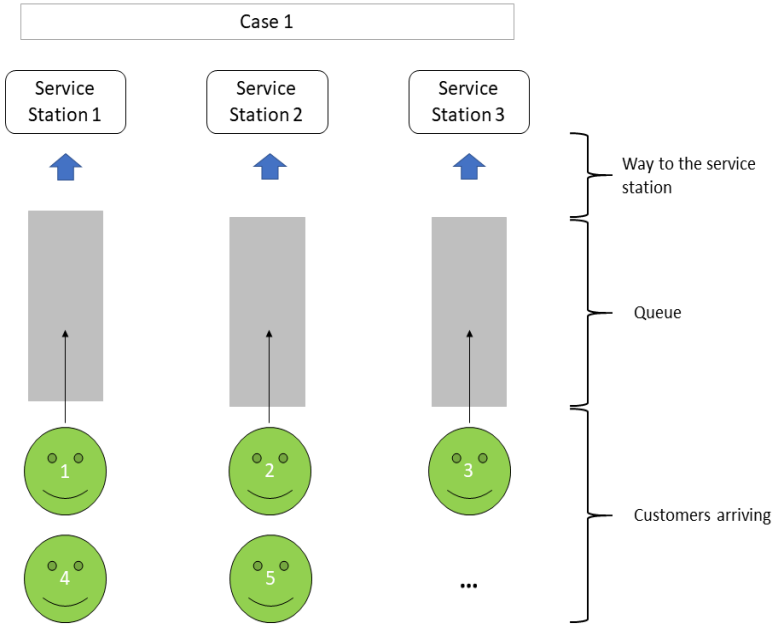
and the change of the capacity for the cases 2A and 2B to meet the requirement of one common queue for all service stations:

```
add_resource("service_station", 3) %>%
```

As consequence of this approach, the results of all cases of 2A and 2B covering the different service stations are divided through 3. This is the case because it is not possible to simulate this situation differently with simmer because adding resources adds a queue automatically which would then violate the requirements of the scenario. Can be seen here. In addition to that the policy of the queues would end up affecting the results in a negative way. So for example all "-available-ending" policies raised errors which can't be handled properly and result in missing customers in the simulation as soon as an error is thrown but not logged. Can be found here.

All further modeling aspects are briefly described and illustrated in the simulation of the respective case. In addition to that comments in the code describe the reason of the actions where they are performed.

Scenario 1



Simulation Model for Case 1

CASE A1

Short description:

- Customers arrive with fixed interarrival times of 90 seconds
- The queues are chosen by the round robin mehtod
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  ## setting the start time for logging purposes
  set_attribute("start_time", function() {now(service_system)}) %>%
  ## Selection of the queue based on the round-robin method
  simmer::select(c("service_station1", "service_station2","service_station3"), policy
= "round-robin") %>%
  ## Block the selected service station
  seize_selected() %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  ## Start the walk from the queue to the service station
  timeout(function() {25}) %>%
  log_("Customer arrives at the service station") %>%
  ## Timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
```

```
  ## Free the service station for the next customer
  release_selected() %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with 3 serice stations --> round robin
  add_resource("service_station1", 1) %>%
  add_resource("service_station2", 1) %>%
  add_resource("service_station3", 1) %>%
  ## customer arrives every 90 seconds until 4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {90}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# merged_results <- merge(result,get_mon_resources(service_system), by = "time", "sta
rt_time")
# filtered_1 <- filter(merged_results,resource == "service_station1")
# filtered_2 <- filter(merged_results,resource == "service_station2")
# filtered_3 <- filter(merged_results,resource == "service_station3")
```

Calculation of the averaged customers served per hour:

```
results_a1 <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations:

```
results_a1 <- c(results_a1,sum(filtered_1$actual_service_time)/(14400),
                  sum(filtered_2$actual_service_time)/(14400),
                  sum(filtered_3$actual_service_time)/(14400))
```

Calculation of the queue means:

```
results_a1 <- c(results_a1, mean(filtered_1$queue),
                  mean(filtered_2$queue),
                  mean(filtered_3$queue))
```

Calculation of the throughput time

```
results_a1 <- c(results_a1,mean(result$activity_time+result$waiting_time))
```

CASE B1

Short description:

- Customers arrive with interarrival times which are uniformly distributed between 60 and 120 seconds
- The queues are chosen by the round robin mehtod
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  ## setting the start time for logging purposes
  set_attribute("start_time", function() {now(service_system)}) %>%
  ## Selection of the queue based on the round-robin method
  simmer::select(c("service_station1", "service_station2","service_station3"), policy
= "round-robin") %>%
  ## Block the selected service station
  seize_selected() %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  ## Start the walk from the queue to the service station
  timeout(function() {25}) %>%
  log_("Customer arrives at the service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  ## Free the service station for the next customer
  release_selected() %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with 3 serice stations --> round robin
  add_resource("service_station1", 1) %>%
  add_resource("service_station2", 1) %>%
  add_resource("service_station3", 1) %>%
  ## customer arrives in times uniformly distributed between 60 and 120 seconds until
4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {runif(1, min = 60
, max = 120)}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
```

```
# merged_results <- merge(result,get_mon_resources(service_system), by = "time", "sta
rt_time")
# filtered_1 <- filter(merged_results,resource == "service_station1")
# filtered_2 <- filter(merged_results,resource == "service_station2")
# filtered_3 <- filter(merged_results,resource == "service_station3")
```

Calculation of the averaged customers served per hour:

```
results_b1 <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations:

```
results_b1 <- c(results_b1,sum(filtered_1$actual_service_time)/(14400),
                  sum(filtered_2$actual_service_time)/(14400),
                  sum(filtered_3$actual_service_time)/(14400))
```

Calculation of the queue means:

```
results_b1 <- c(results_b1, mean(filtered_1$queue),
                  mean(filtered_2$queue),
                  mean(filtered_3$queue))
```

Calculation of the throughput time

```
results_b1 <- c(results_b1,mean(result$activity_time+result$waiting_time))
```

CASE C1

Short description:

- Customers arrive with exponentially distributed interarrival times with expected value 90 sec
- The queues are chosen by the round robin mehtod
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  ## setting the start time for logging purposes
  set_attribute("start_time", function() {now(service_system)}) %>%
  ## Selection of the queue based on the round-robin method
  simmer::select(c("service_station1", "service_station2","service_station3"), policy
= "round-robin") %>%
  ## Block the selected service station
  seize_selected() %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  ## Start the walk from the queue to the service station
  timeout(function() {25}) %>%
  log_("Customer arrives at the service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  ## Free the service station for the next customer
  release_selected() %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with 3 serice stations and a single queue --> round robin
  add_resource("service_station1", 1) %>%
  add_resource("service_station2", 1) %>%
  add_resource("service_station3", 1) %>%
  ## customer arrives exponentially distributed with the expection of 90 seconds, unt
il 4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {rexp(1, rate = 1/
90)}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
```

```
# merged_results <- merge(result,get_mon_resources(service_system), by = "time", "sta
rt_time")
# filtered_1 <- filter(merged_results,resource == "service_station1")
# filtered_2 <- filter(merged_results,resource == "service_station2")
# filtered_3 <- filter(merged_results,resource == "service_station3")
```

Calculation of the averaged customers served per hour:

```
results_c1 <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations:

```
results_c1 <- c(results_c1,sum(filtered_1$actual_service_time)/(14400),
                  sum(filtered_2$actual_service_time)/(14400),
                  sum(filtered_3$actual_service_time)/(14400))
```

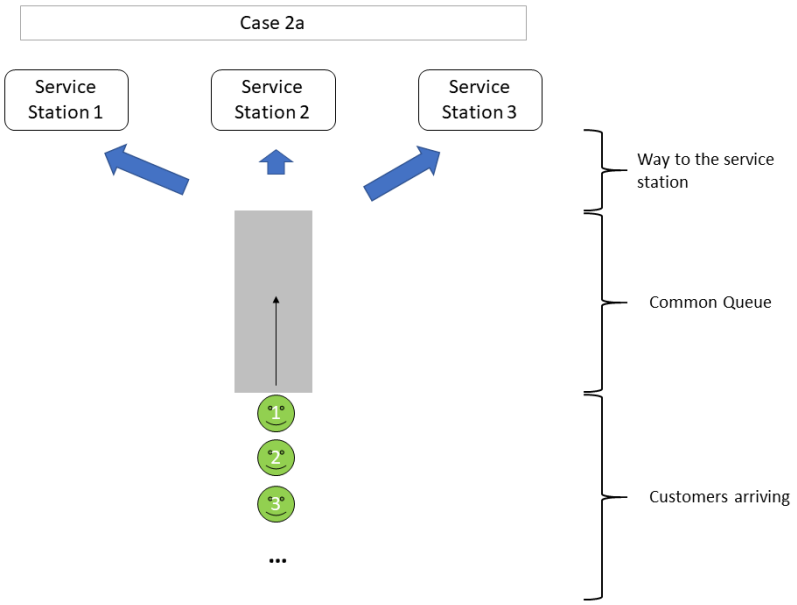
Calculation of the queue means:

```
results_c1 <- c(results_c1, mean(filtered_1$queue),
                  mean(filtered_2$queue),
                  mean(filtered_3$queue))
```

Calculation of the throughput time:

```
results_c1 <- c(results_c1,mean(result$activity_time+result$waiting_time))
```

Scenario 2A



Simulation Model for Case 2a

CASE A2A

Short description:

- Customers arrive with fixed interarrival times of 90 seconds
- Common queue before the way to the service station
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <-
  trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  ## setting the start time for logging purposes
  set_attribute("start_time", function() {now(service_system)}) %>%
  seize("service_station") %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  log_("Customer moves to the service station") %>%
  timeout(25) %>%
  log_("Customer arrives service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  release("service_station") %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})
```

```
service_system <-
  simmer("service_system") %>%
  ## service system with a capacity of 3 and a single queue
  add_resource("service_station", 3) %>%
  ## customer arrives every 90 seconds until 4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {90}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# result <- merge(result, get_mon_resources(service_system), by = "time", "start_time
")
```

Calculation of the averaged customers served per hour:

```
results_a2a <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations: (3 times the same value to meet the data frame length for later joins)

```
results_a2a <- c(results_a2a,sum(result$actual_service_time)/(14400*3),
                 sum(result$actual_service_time)/(14400*3),
                 sum(result$actual_service_time)/(14400*3))
```

Calculation of the queue means: (3 times the same value to meet the data frame length for later joins)

```
results_a2a <- c(results_a2a, mean(result$queue),
                 mean(result$queue),
                 mean(result$queue))
```

Calculation of the throughput time:(3 times the same value to meet the data frame length for later joins)

```
results_a2a <- c(results_a2a,mean(result$activity_time+result$waiting_time))
```

CASE B2A

Short description:

- Customers arrive with interarrival times which are uniformly distributed between 60 and 120 seconds
- Common queue before the way to the service station
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <-
  trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  ## setting the start time for logging purposes
  set_attribute("start_time", function() {now(service_system)}) %>%
  seize("service_station") %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  log_("Customer moves to service station") %>%
  timeout(25) %>%
  log_("Customer arrives at the service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  release("service_station") %>%
  log_("Customer leaves")
#  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with a capacity of 3 and a single queue
  add_resource("service_station", 3) %>%
  ## customer arrives in times uniformly distributed between 60 and 120 seconds until
4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {runif(1, min = 60
, max = 120)}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# result <- merge(result, get_mon_resources(service_system), by = "time", "start_time
")
```

Calculation of the averaged customers served per hour:

```
results_b2a <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations: (3 times the same value to meet the data frame length for later joins)

```
results_b2a <- c(results_b2a,sum(result$actual_service_time)/(14400*3),
                   sum(result$actual_service_time)/(14400*3),
                   sum(result$actual_service_time)/(14400*3))
```

Calculation of the queue means:(3 times the same value to meet the data frame length for later joins)

```
results_b2a <- c(results_b2a, mean(result$queue),
                   mean(result$queue),
                   mean(result$queue))
```

This queue would would rise to infinity.

Calculation of the throughput time:

```
results_b2a <- c(results_b2a,mean(result$activity_time+result$waiting_time))
```

CASE C2A

Short description:

- Customers arrive with exponentially distributed interarrival times with expected value 90 sec
- Common queue before the way to the service station
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  ## setting the start time for logging purposes
  set_attribute("start_time", function() {now(service_system)}) %>%
  seize("service_station") %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  log_("Customer arrives service station") %>%
  timeout(25) %>%
   ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  release("service_station") %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with a capacity of 3 and a single queue
  add_resource("service_station", 3) %>%
  ## customer arrives exponentially distributed with the expection of 90 seconds, unt
il 4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {rexp(1, rate = 1/
90)}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# result <- merge(result, get_mon_resources(service_system), by = "time", "start_time
")
```

Calculation of the averaged customers served per hour:

```
results_c2a <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations: (3 times the same value to meet the data frame length for later joins)

```
results_c2a <- c(results_c2a,sum(result$actual_service_time)/(14400*3),
                   sum(result$actual_service_time)/(14400*3),
                   sum(result$actual_service_time)/(14400*3))
```

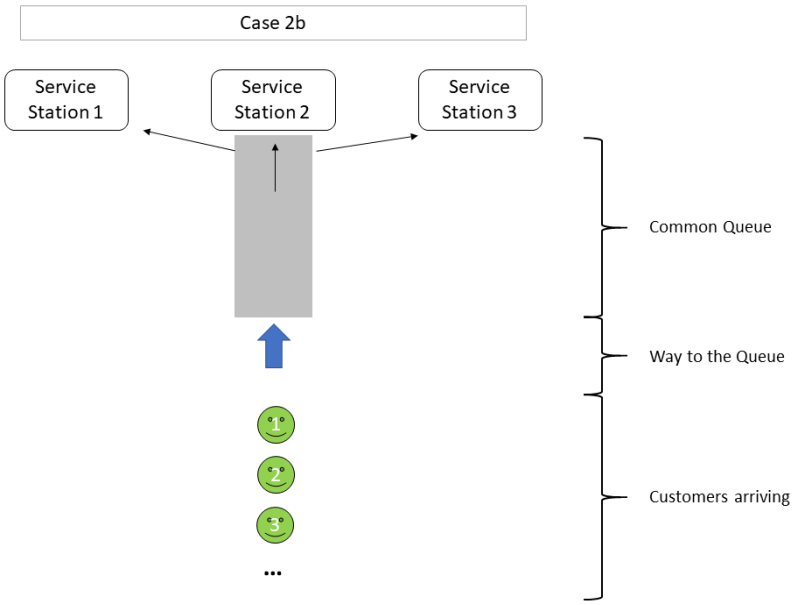
Calculation of the queue means:(3 times the same value to meet the data frame length for later joins)

```
results_c2a <- c(results_c2a, mean(result$queue),
                   mean(result$queue),
                   mean(result$queue))
```

Calculation of the throughput time:

```
results_c2a <- c(results_c2a,mean(result$activity_time+result$waiting_time))
```

Scenario 2B



Simulation Model for Case 2b

CASE A2B

Short description:

- Customers arrive with fixed interarrival times of 90 seconds
- The Common queue is located directly in front of the service station. In this case, the assumption is made as shown above that the way now is infront of the queue instead of the service station. The start time is set as soon as the customer arrives the queue.
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  # walk to the queue takes 25 seconds before entering it
  log_("Customer walks to the queue") %>%
  timeout(function() {25}) %>%
  log_("Customer arrives at the queue") %>%
  ## setting the start time
  set_attribute("start_time", function() {now(service_system)}) %>%
  seize("service_station") %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  log_("Customer arrives service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  release("service_station") %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})
```

```
service_system <-
  simmer("service_system") %>%
  ## service system with a capacity of 3 and a single queue
  add_resource("service_station", 3) %>%
  ## customer arrives every 90 seconds until 4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {90}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# result <- merge(result, get_mon_resources(service_system), by = "time", "start_time
")
```

Calculation of the averaged customers served per hour:

```
results_a2b <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations: (3 times the same value to meet the data frame length for later joins)

```
results_a2b <- c(results_a2b,sum(result$actual_service_time)/(14400*3),
                 sum(result$actual_service_time)/(14400*3),
                 sum(result$actual_service_time)/(14400*3))
```

Calculation of the queue means:(3 times the same value to meet the data frame length for later joins)

```
results_a2b <- c(results_a2b, mean(result$queue),
                 mean(result$queue),
                 mean(result$queue))
```

Calculation of the throughput time:

```
results_a2b <- c(results_a2b,mean(result$activity_time+result$waiting_time))
```

CASE B2B

Short description:

- Customers arrive with interarrival times which are uniformly distributed between 60 and 120 seconds
- The Common queue is located directly in front of the service station. In this case, the assumption is made as shown above that the way now is infront of the queue instead of the service station. The start time is set as soon as the customer arrives the queue.
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  # walk to the queue takes 25 seconds before entering it
  log_("Customer walks to the queue") %>%
  timeout(function() {25}) %>%
  log_("Customer arrives at the queue") %>%
  ## setting the start time
  set_attribute("start_time", function() {now(service_system)}) %>%
  seize("service_station") %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  log_("Customer arrives service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  release("service_station") %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with a capacity of 3 and a single queue
  add_resource("service_station", 3) %>%
  ## customer arrives in times uniformly distributed between 60 and 120 seconds until
4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {runif(1, min = 60
, max = 120)}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# result <- merge(result, get_mon_resources(service_system), by = "time", "start_time
")
```

Calculation of the averaged customers served per hour:

```
results_b2b <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations: (3 times the same value to meet the data frame length for later joins)

```
results_b2b <- c(results_b2b,sum(result$actual_service_time)/(14400*3),
                 sum(result$actual_service_time)/(14400*3),
                 sum(result$actual_service_time)/(14400*3))
```

Calculation of the queue means:(3 times the same value to meet the data frame length for later joins)

```
results_b2b <- c(results_b2b, mean(result$queue),
                 mean(result$queue),
                 mean(result$queue))
```

Calculation of the throughput time:

```
results_b2b <- c(results_b2b,mean(result$activity_time+result$waiting_time))
```

CASE C2B

Short description:

- Customers arrive with exponentially distributed interarrival times with expected value 90 sec
- Common queue before the way to the service station
- The way to the service station is between the queue and the actual service station. So, the block of the service station is already done as soon as the customer leaves the queue and gets on the way to the service station
- The service times are uniformly distributed between 220sec and 300sec

```
service_system <- simmer()

customer <- trajectory("serve Customer") %>%
  log_("Customer arrives") %>%
  # walk to the queue takes 25 seconds before entering it
  log_("Customer walks to the queue") %>%
  timeout(function() {25}) %>%
  log_("Customer arrives at the queue") %>%
  ## setting the start time
  set_attribute("start_time", function() {now(service_system)}) %>%
  seize("service_station") %>%
  log_(function() {paste("Waited: ", now(service_system) - get_attribute(service_syst
em, "start_time"))}) %>%
  log_("Customer arrives service station") %>%
  ## timeout is the service time which is uniformly distributed between 220 and 300
  timeout(function() {runif(1, min=220, max=300)}) %>%
  release("service_station") %>%
  log_("Customer leaves")%>%
  log_(function() {paste("Finished: ", now(service_system))})

service_system <-
  simmer("service_system") %>%
  ## service system with a capacity of 3 and a single queue
  add_resource("service_station", 3) %>%
  ## customer arrives exponentially distributed with the expection of 90 seconds, unt
il 4h -> 14400sec
  add_generator("Customer", customer, from_to(0, 14400, function() {rexp(1, rate = 1/
90)}))

  # Code for analysis (suppressed)

# service_system %>% run(until = 14400)
# result <- data.frame(service_system %>%
#   get_mon_arrivals() %>%
#   transform(waiting_time = end_time - start_time - activity_time) %>%
#   transform(actual_service_time = activity_time-25)
#   )
# result <- merge(result, get_mon_resources(service_system), by = "time", "start_time
")
```

Calculation of the averaged customers served per hour:

```
results_c2b <- c(length(result$finished==TRUE)/4/3)
customers_served <- length(result$finished==TRUE)/4/3
#customers_served
```

Calculation of the occupation of the service stations: (3 times the same value to meet the data frame length for later joins)

```
results_c2b <- c(results_c2b,sum(result$actual_service_time)/(14400*3),
                   sum(result$actual_service_time)/(14400*3),
                   sum(result$actual_service_time)/(14400*3))
```

Calculation of the queue means:(3 times the same value to meet the data frame length for later joins)

```
results_c2b <- c(results_c2b, mean(result$queue),
                   mean(result$queue),
                   mean(result$queue))
```

Calculation of the throughput time:

```
results_c2b <- c(results_c2b,mean(result$activity_time+result$waiting_time))
```

Result Interpretation and Comparison

```
result_matrix <- round_df(data.frame("Values"=result_header, "CASE A1"= results_a1, "
CASE B1" = results_b1, "CASE C1" = results_c1, "CASE A2A" = results_a2a, "CASE B2A" =
results_b2a, "CASE C2A" = results_c2a, "CASE A2B" = results_a2b, "CASE B2B" = results
_b2b, "CASE C2B" = results_c2b),digits = 2)

t_result_matrix <- transpose(result_matrix)
colnames(t_result_matrix) <- rownames(result_matrix)
rownames(t_result_matrix) <- colnames(result_matrix)
```

Overview of all results

The following table shows all results of the different simulations performed with simmer.

```
kable(result_matrix)
```

Values	CASE.A1	CASE.B1	CASE.C1	CASE.A2A	CASE.B2A	CASE.C2A	CASE.A2B	CASE.B2B	CASE.C2B
customer_per_hour_per_station	12.50	12.50	12.08	12.50	12.33	11.83	13.08	13.17	11.92
degree_of_utilization_1	0.91	0.91	0.86	0.90	0.90	0.86	0.95	0.95	0.87
degree_of_utilization_2	0.90	0.90	0.87	0.90	0.90	0.86	0.95	0.95	0.87
degree_of_utilization_3	0.89	0.90	0.86	0.90	0.90	0.86	0.95	0.95	0.87
average_queue_length_1	2.02	2.22	1.14	4.05	3.80	8.70	0.59	0.91	4.34
average_queue_length_2	2.12	1.60	1.56	4.05	3.80	8.70	0.59	0.91	4.34
average_queue_length_3	1.60	1.56	1.19	4.05	3.80	8.70	0.59	0.91	4.34
avg_customer_through_put_time	676.61	649.61	518.80	625.62	600.52	1074.62	299.62	331.57	627.02

A detailed analysis can be found in the answers of the following questions.

-How many customers are served on average during an hour at each of the service stations?
The results of the service systems can be found in the table below:

```
kable(data.frame("Context" = rownames(t_result_matrix), "Value" = t_result_matrix[,1]
))
```

Context	Value
Values	customer_per_hour_per_station
CASE.A1	12.5
CASE.B1	12.5
CASE.C1	12.08
CASE.A2A	12.5
CASE.B2A	12.33
CASE.C2A	11.83
CASE.A2B	13.08
CASE.B2B	13.17
CASE.C2B	11.92

We calculated our results under the condition set.seed(63). By setting the seed to 63 we ensured that we use a random number generator, which is useful for creating simulations that can be reproduced. Every simulation run has therefore the same result for the 4 hours simulation.

With the settings used in the simulation the most efficient service station would be B2B by serving 13.17 customers on average during an hour service. Nevertheless, this could also just be the case because of the random distribution on set.seed(63). Under other conditions where the disitribution of the arrival time would lean more to 120 seconds instead of 60 seconds the result could differ. Since A2B with fixed interarrival times of 90 seconds also is showing a good result we can asume that the service system is working better than the others. If we calculate the average arriving rate with 3600/90/3 = **13.33** it would indicate that all systems are not stable since the arrival rate is larger than the service rate. Nevertheless, A2B and B2B would be really close to this number.

-What is the degree of utilization at each of the service stations?

The results of the service systems can be found in the table below. Please consider that in simmer for a service system with one queue for multiple service station (counters) is simulated as a capacity of three for one service station. Therefore, the individual service station results of 2a and 2b cannot be displayed separately which leads to three identical values in these cases.

```
kable(data.frame("Value" = t_result_matrix[,2:4]))
```

	Value.2	Value.3	Value.4
Values	degree_of_utilization_1	degree_of_utilization_2	degree_of_utilization_3
CASE.A1	0.91	0.9	0.89
CASE.B1	0.91	0.9	0.9
CASE.C1	0.86	0.87	0.86
CASE.A2A	0.9	0.9	0.9
CASE.B2A	0.9	0.9	0.9
CASE.C2A	0.86	0.86	0.86
CASE.A2B	0.95	0.95	0.95
CASE.B2B	0.95	0.95	0.95
CASE.C2B	0.87	0.87	0.87

The degree of utilization at each of the service stations is bellow 100% as in the beginning the first 2 service stations are idle. The third service station gets the first customer on average only after 180s of service. An additional factor for the service stations are not beeing fully utilized is the time where the customer walks to the service station (25 seconds). During this time the service stations are idle as well. So even if the customer arrival rate is higher than the service rate the service station with the highest utilization is only at 0.95 % (A2B and B2B). It is therefore no considered that these two stations had the highest customer service rate since all stations have equal service rates but these 2 stations are just more in use.

-What is the average length of the queues?

The results of the service systems can be found in the table below. Please consider that in simmer for a service system with one queue for multiple service station (counters) is simulated as a capacity of three for one service station. Therefore, the individual service station results of 2a and 2b cannot be displayed separately which leads to three identical values in these cases.

```
kable(data.frame("Value" = t_result_matrix[,5:7]))
```

	Value.5	Value.6	Value.7
Values	average_queue_length_1	average_queue_length_2	average_queue_length_3
CASE.A1	2.02	2.12	1.6
CASE.B1	2.22	1.6	1.56
CASE.C1	1.14	1.56	1.19
CASE.A2A	4.05	4.05	4.05
CASE.B2A	3.8	3.8	3.8
CASE.C2A	8.7	8.7	8.7
CASE.A2B	0.59	0.59	0.59
CASE.B2B	0.91	0.91	0.91
CASE.C2B	4.34	4.34	4.34

The average length of the queues is between 0.59(A2B) & 8.7 (C2A) and differs therefore heavily. The aspect that the simulation is only done for 4 hours has to be considered when looking at these numbers. In the beginning there was at in none of the scenarios a queue since the customers just arrived one after another. As the plots in the following answers show are most queues having a positive slope for the customer throughput. The rising throughput time obviously impacts the queue length. The length of the queue for the simulations would thus rise to infinity if the simulation time of A1/B1/A2A/B2A/C2A be prolonged. C1 and C2B look suspicious but no definite Conclusion can be drawn. Only A2A and B2B seems to look stable.

-What is the average throughput time of the customers?

The results of the service systems can be found in the table below:

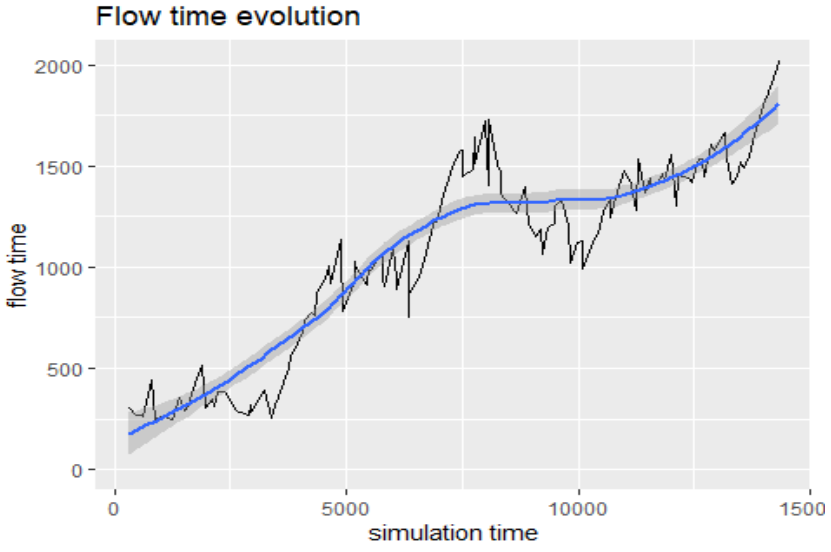
```
kable(data.frame("Context" = rownames(t_result_matrix), "Value" = t_result_matrix[,8]
))
```

Context	Value
Values	avg_customer_through_put_time
CASE.A1	676.61
CASE.B1	649.61
CASE.C1	518.8
CASE.A2A	625.62
CASE.B2A	600.52
CASE.C2A	1074.62
CASE.A2B	299.62
CASE.B2B	331.57
CASE.C2B	627.02

The average throughput time of the customers is between 299.62 seconds (A2B) & 1074.62 seconds (C2A) and differs therefore also heavily. The aspect that the simulation is only done for 4 hours has to be considered when looking at these numbers. As the plots in the paper show most models have as well a positive slope for the throughput time which would therefore also rise to infinity if the simulation time would be prolonged (see example of C2A). In contrast the slope of A2B looks stable.

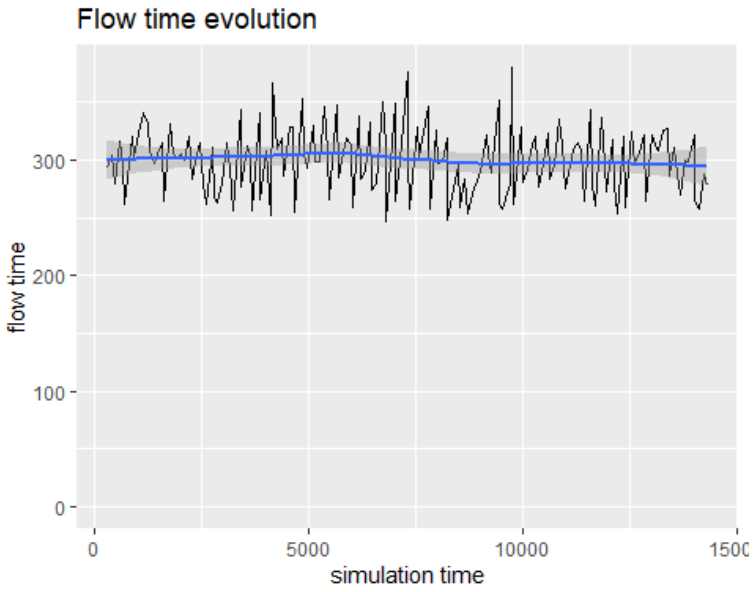
See example C2A with a positive slope for the flow time

```
plot(arrivals_c2a, metric = "flow_time")
```



See example A2B with a stable flow time

```
plot(arrivals_a2b, metric = "flow_time")
```



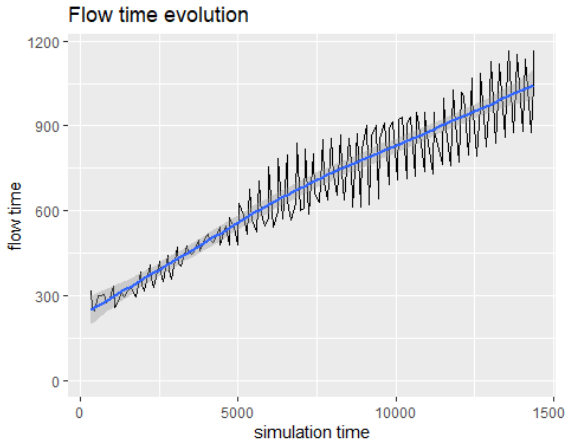
Conclusion

If we just consider the average service rate per service station and the average arrival rate of the customers (~ 90 seconds) all systems seem to not be stable. When λ is greater or equal to μ, the queue becomes infinite long. The workload ρ should always be smaller than 1 which is in none of the scenatios the case. Nevertheless the two best functioning simulations were A2A and B2B which seem to function properly in the simulations done.

Plots of the flow time from all models:

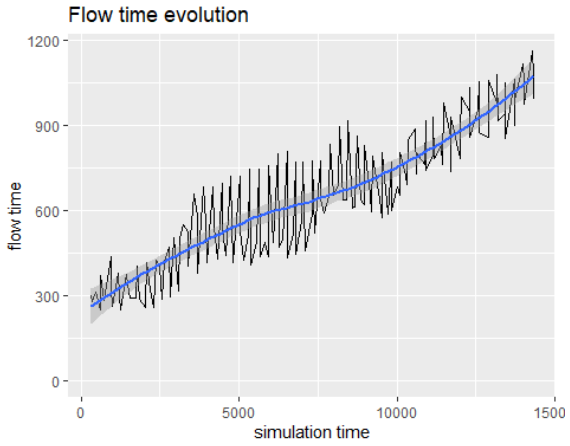
CASE A1

```
plot(arrivals_a1, metric = "flow_time")
```



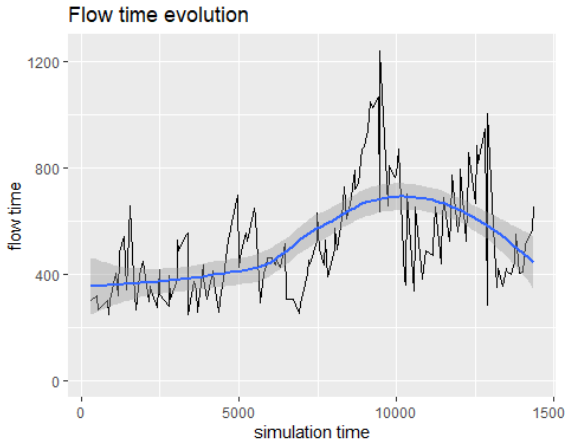
CASE B1

```
plot(arrivals_b1, metric = "flow_time")
```



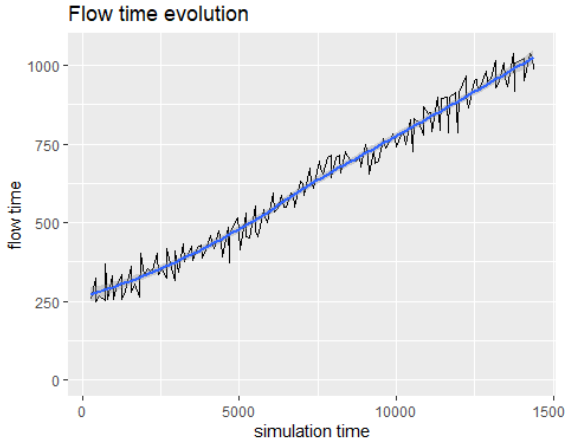
CASE C1

```
plot(arrivals_c1, metric = "flow_time")
```



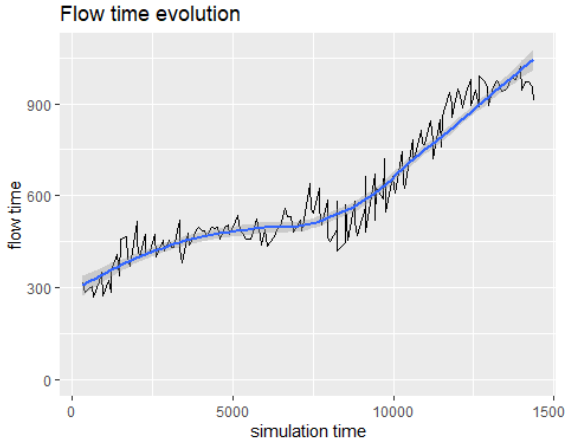
CASE A2A

```
plot(arrivals_a2a, metric = "flow_time")
```



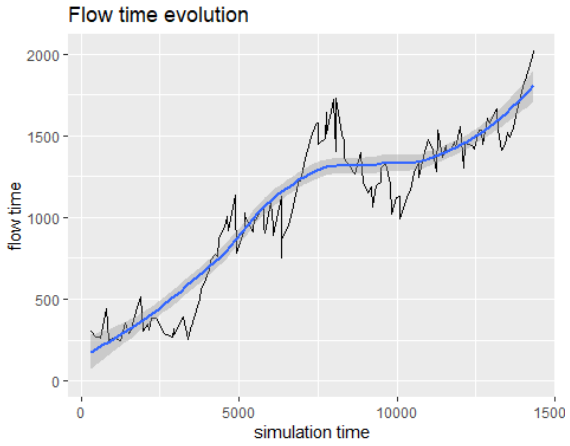
CASE B2A

```
plot(arrivals_b2a, metric = "flow_time")
```



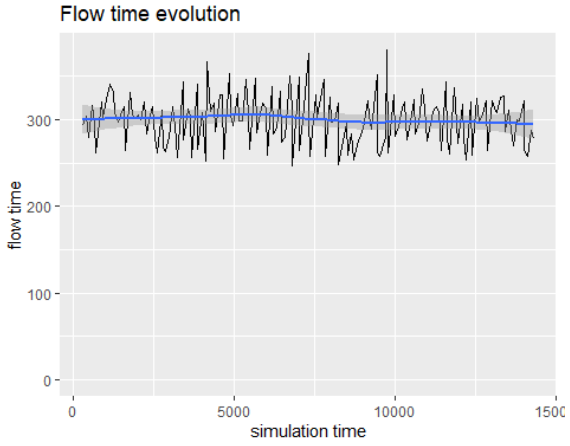
CASE C2A

```
plot(arrivals_c2a, metric = "flow_time")
```



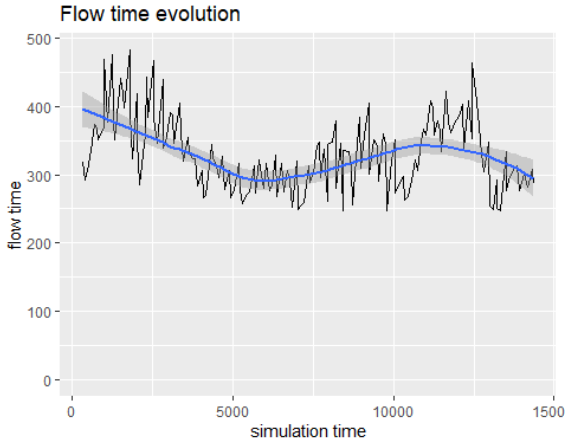
CASE A2B

```
plot(arrivals_a2b, metric = "flow_time")
```



CASE B2B

```
plot(arrivals_b2b, metric = "flow_time")
```



CASE C2B

```
plot(arrivals_c2b, metric = "flow_time")
```