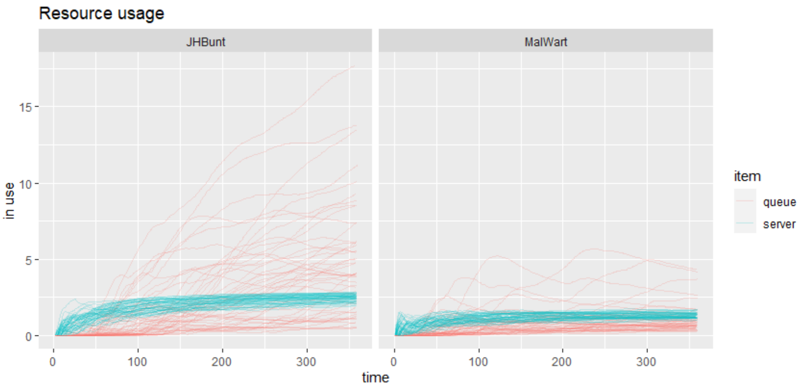
Monitoring and Visualising the Simulations



Outline

Learning Objectives

In this video, we will

- Run the simulations multiple times.
- Monitor and visualise the results of the simulations: arrivals, attributes and resources.

Multiple Replications

- The simulation model simulates the STEM event for 360 minutes.
- We will run the simulation model 50 times.

```
set.seed(2909)
mm2.envs  <- replicate(50,
  simmer("mixer") %>%
  add_resource("MalWart", capacity = 2) %>%
  add_resource("JHBunt", capacity = 3) %>%
  add_generator("Student", student,
    function() rexp(1, 1/2), mon = 2) %>%
    run(until = 360) %>% wrap())
```

```
...
357.646: Student172: Here I am
358.139: Student172: I have affixed my name tag!
358.657: Student167: I shall join the queue and talk to a MalWart recruiter
359.777: Student173: Here I am
```

Target Outcomes

- Recall that we were interested to study three particular performance measures.
  - The utilisation of the recruiters at each company booth.
  - The number of students waiting in line at each booth.
  - The mean wait time of students during the mixer.
- To extract the relevant monitor data that we need, we run the following three lines of code on the 50 simulation environments that we have generated.

```
mon_resources <- get_mon_resources(mm2.envs)
mon_arrivals <- get_mon_arrivals(mm2.envs)
mon_attributes <- get_mon_attributes(mm2.envs)
```

Monitor the Resources Data

- Let's begin by inspecting the rows in the 'mon_resources' table.

```r
knitr::kable(head(mon_resources))
```

```
|resource |      time| server| queue| capacity| system| limit| replication|
|:--------|---------:|------:|-----:|--------:|------:|-----:|-----------:|
|MalWart  |  5.156790|      1|     0|        2|      1|   Inf|           1|
|MalWart  |  5.436570|      0|     0|        2|      0|   Inf|           1|
|JHBunt   |  5.436570|      1|     0|        3|      1|   Inf|           1|
|MalWart  |  9.106437|      1|     0|        2|      1|   Inf|           1|
|MalWart  | 11.495920|      0|     0|        2|      0|   Inf|           1|
|JHBunt   | 11.495920|      2|     0|        3|      2|   Inf|           1|
```

- We can conclude that a student talked to a Malwart recruiter at 5.16 minutes, and he left the counter at 5.44 minutes.

Monitor the Arrivals Data

- Now, let us inspect the rows in the 'mon_arrivals' table.

```r
knitr::kable(head(mon_arrivals))
```

```
|name      | start_time| end_time| activity_time|finished | replication|
|:---------|----------:|--------:|-------------:|:--------|-----------:|
|Student3  |   8.642642| 12.76900|      4.126363|TRUE     |           1|
|Student1  |   4.484858| 25.34722|     20.862361|TRUE     |           1|
|Student8  |  20.554220| 25.36857|      4.814350|TRUE     |           1|
|Student2  |   4.874769| 28.54394|     23.669175|TRUE     |           1|
|Student11 |  27.536154| 34.41472|      6.878562|TRUE     |           1|
|Student0  |   2.255781| 38.55244|     33.108475|TRUE     |           1|
```

- For example, the student 3 arrived at 8.64 minutes, and left at 12.77 minutes. In total, he spent 4.13 minutes in the activities.

Monitor the Arrivals Data

cont'd

- To know the breakdown activity time per resource, we can use the following code.

```
mon_arrivals_sub <- get_mon_arrivals(mm2.envs, per_resource = TRUE)
knitr::kable(head(mon_arrivals_sub))
```

```
|name      | start_time| end_time| activity_time|resource | replication|
|:---------|----------:|--------:|-------------:|:--------|-----------:|
|Student1  |   5.156790|  5.43657|     0.2797801|MalWart  |           1|
|Student3  |   9.106437| 11.49592|     2.3894828|MalWart  |           1|
|Student3  |  11.495920| 12.76900|     1.2730851|JHBunt   |           1|
|Student8  |  20.950618| 23.24390|     2.2932834|MalWart  |           1|
|Student2  |  23.704085| 24.68430|     0.9802124|MalWart  |           1|
|Student1  |   5.436570| 25.34722|    19.9106486|JHBunt   |           1|
```

```
queueing_time = (end_time - start_time) - activity_time
```

Monitor the Attributes Data

- Next, let us inspect the rows in the 'mon_attributes' table.

```
knitr::kable(tail(mon_attributes))
```

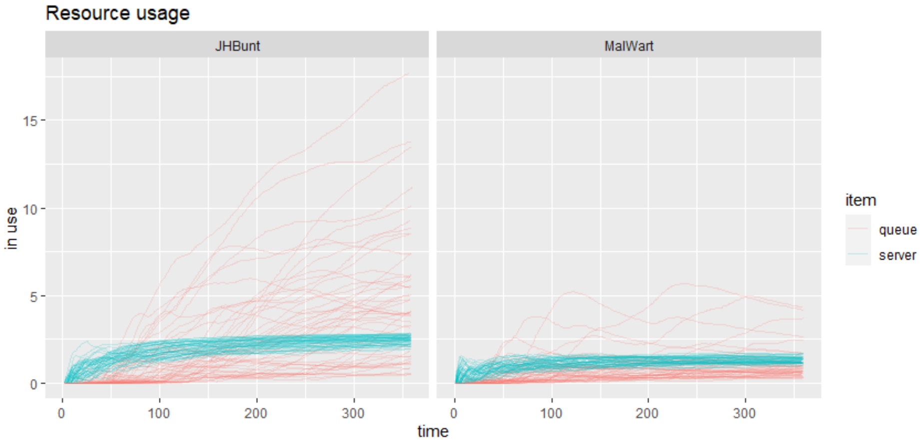
```
|      |     time|name        |key  | value| replication|
|:-----|--------:|:-----------|:----|-----:|-----------:|
|13367 | 352.5556|Student169 |type |     1|          50|
|13368 | 352.6140|Student170 |type |     1|          50|
|13369 | 355.8731|Student161 |type |     2|          50|
|13370 | 357.0327|Student171 |type |     1|          50|
|13371 | 358.1386|Student172 |type |     1|          50|
|13372 | 358.6568|Student167 |type |     2|          50|
```

- In total, there are 3 types of students.

Visualise the Resources Data

- To address the question of queue lengths at the two booths, we can begin with a plot.

```
plot(mon_resources, metric="usage", items=c("queue", "server"),
     limits=FALSE)
```

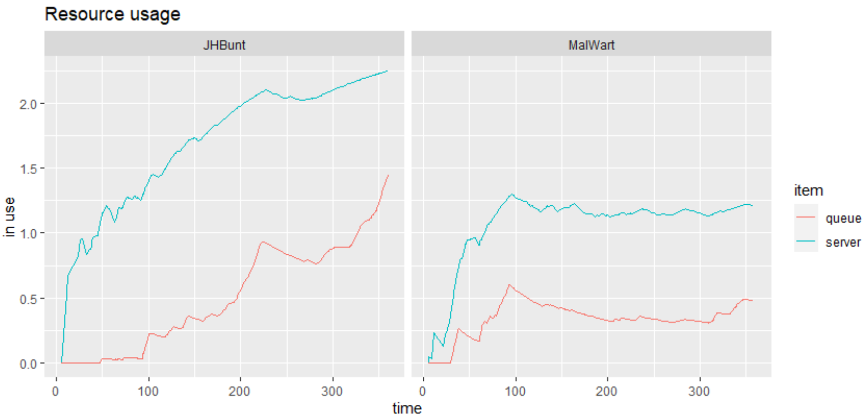


Visualise the Resources Data

cont'd

- To zoom into one specific simulation, say, the first replication, we can use the following code.

```
plot(mon_resources[mon_resources$replication == 1, ],
     items=c("queue", "server"), limits=FALSE)
```

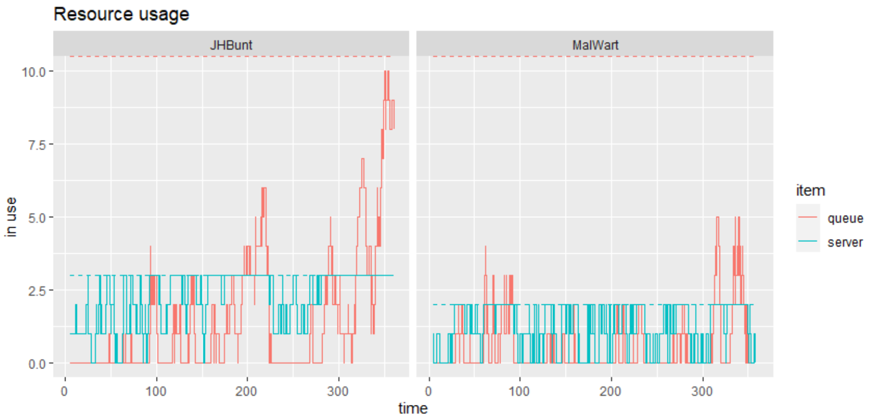


Visualise the Resources Data

cont'd

- If our interest is the instantaneous number of students in the queue, we can use the following code.

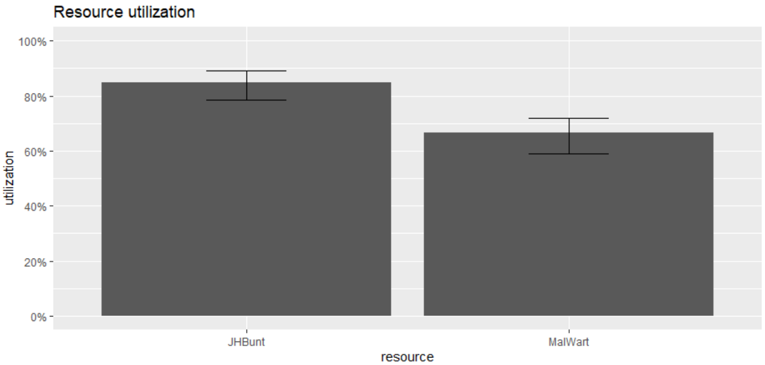
```
plot(mon_resources[mon_resources$replication == 1, ],
     items=c("queue", "server"), limits=FALSE, step = TRUE)
```



Visualise the Resources Data

cont'd

```
plot(mon_resources, metric="utilization")
```



- In conclusion, the utilisation of 3 recruiters at the JHBunt booth is close to 85%, and the utilisation of 2 recruiters at the Malwart booth is slightly above 65%.

Utilise the Arrivals Data

to compute the waiting time at booths

- The arrivals data contain all the information we need to compute the waiting time for each student.
- To answer the last question on the mean waiting time at booths, our approach is going to be as follows:
  - We first write a simple function to extract the summary measure that we need for a single replication.
  - We next split the data frame by the replication number and use ***sapply*** to call this function on each sub data frame.

Waiting Time at Booths

define a function, split the data frame, and calculate the mean for each replication

- Our function is going to take in a data frame, and compute the mean waiting time for all students in that particular replication.

```
ave_wait_time <- function(df) {
  mean(df$end_time - df$start_time - df$activity_time)
}
```

- To split the arrivals data frame, we can use the ***split*** function from R. It returns a list of data frames, which we can then feed into ***sapply***.

```
s_out <- split(mon_arrivals, ~replication)
w_vec <- sapply(s_out, ave_wait_time)
```

- ***w_vec*** records a vector of 50 values, which are some mean waiting time of students over the course of a 6-hour mixer, for each replication.

```
mean(w_vec)
```

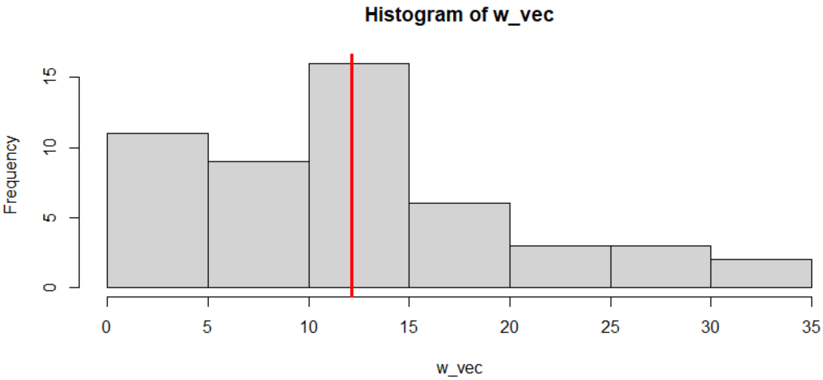
```
[1] 12.1311
```

Waiting Time at Booths

visualise the mean waiting time across the replications

- We can check the distribution of the mean waiting time across the replications.

```
hist(w_vec)
abline(v = mean(w_vec), col = "red", lwd = 3)
```



Waiting Time at Booths

calculate the standard normal-based 95% confidence interval

- We can compute a confidence interval to study the mean wait time of students.

```
summary(w_vec)
```

```
   Min.    1st Qu.   Median     Mean      3rd Qu.     Max.
   1.926   6.118     10.481     12.131    16.066      34.994
```

```
ci_95 <- c(mean(w_vec) + qnorm(0.025)*sd(w_vec)/sqrt(length(w_vec)),
           mean(w_vec) - qnorm(0.025)*sd(w_vec)/sqrt(length(w_vec)))
ci_95
```

```
[1]  9.894196 14.367997
```

- We are 95% confident that the population mean wait time is between 9.89 and 14.37 minutes.

Decision Time and Future Plan

- From the earlier results, it is clear that the booths are being under-utilised.
- As an organiser, we have a powerful tool, Simulation, in our hands.
- We could experiment with several ways to make our operation more efficient.
  - We could, for instance, increase or reduce the number of recruiters at each booth, and see how that affects the waiting times.
  - If our priority is the utilisation, we could either keep the current arrangement, or slightly reduce the number of recruiters.
  - If our priority is to further reduce the mean waiting time, we could possibly increase the number of recruiters.
  - With a little more data manipulation, we can compute time-averaged numbers of wandering students to see if they are taking up a lot of physical space at the mixer.

Summary

In this video, we have:

- Fetched the monitor data: arrivals, attributes and resources.
- Visualised the monitor data.
- Utilised the monitor data to answer our research questions.

Overall,

we hope that this week's set of videos have given you a good indication of the overall process of setting up, executing, and studying the output of a discrete event simulation using R.

References

Rossetti, M.D. (2021)
Simulation Modeling and Arena

Iñaki Ucar, Bart Smeets (2022)
Package ‘simmer‘
https://cran.r-project.org/web/packages/simmer/simmer.pdf

Iñaki Ucar, Bart Smeets (2022)
Package ‘simmer.plot‘
https://cran.r-project.org/web/packages/simmer.plot/simmer.plot.pdf