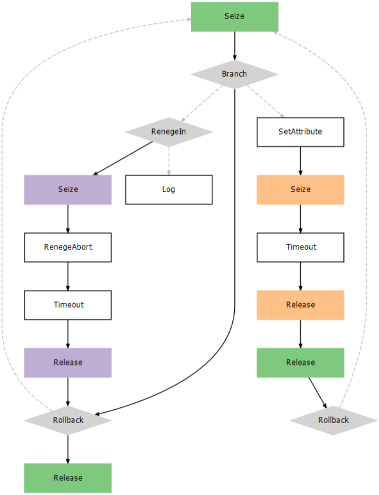
Elements of `simmer`



Outline

- Introduction to `simmer` package
- Illustration through a bank example
  - Entity
  - Resource
  - Simulation

Learning Objectives

1. Learn the different elements of `simmer` package.
2. Use `simmer` package to perform simulations.

Introduction to simmer package

Introduction

The simmer package

- A process-oriented and trajectory-based [Ucar et al., 2019] Discrete-Event Simulation (DES) package for R.
- Designed as a generic yet powerful framework by exploiting the novel concept of trajectory.
- Takes advantage of the piping workflow introduced by the magrittr package.

```
library(simmer)
```

Illustration through a bank example

The *Entity* code chunk

Simple bank example [Garmonsway, 2022]:

- A customer enters the bank;
- Seize the resource, that is an available counter;
- Spend some time (eg. 12 units of time) at the counter;
- Release the resource when done.

```
customer <-
  trajectory("Customer's path") %>%
  seize("counter") %>%
  timeout(12) %>%
  release("counter")
```

- Customer is an entity and the above code specifies its trajectory.

Advance use of `timeout` function

The `timeout` function used in the previous slide may accept a function instead of just a constant number.

- In simulation, we usually introduce randomness.
- We first decide on the distribution that best describes the situation, for instance an exponential distribution.
- Then we figure out the parameters of distribution, for instance `rate = 1/12`.

```
task_duration <- function() {rexp(n = 1, rate = 1/12)}

 ... %>%
  timeout(task_duration) %>%
  ...
```

- The R function `rexp` generates `n` number(s) following an exponential distribution with the `rate` specified.

The *Resource* code chunk

cont'd

From the bank's point of view:

- Instantiates the simulation environment;
- Creates the resource;
- Tries to serve the customer(s) upon arrival.

```
bank <-
  simmer("bank") %>%
  add_resource(name = "counter") %>%
  add_generator(name_prefix = "Customer", trajectory = customer,
                distribution = at(0))
```

- The bank counter is a (limited) resource and being used by entities (customers).

The *Resource* code chunk

cont'd

- Resource comprise two internal self-managed parts:

Server: representing the resource itself

- It has a specified capacity and can be seized and released.
- The `capacity` argument specifies how many entities can be served concurrently at any point in time.

Queue: a queue of a certain size

- The `queue_size` argument specifies the maximum number of entities for this resource, within the queue.

```
add_resource("counter", capacity = 1, queue_size = 1)
```

The *Resource* code chunk

cont'd

Source is a process responsible for creating new arrivals with a given inter-arrival time pattern and inserting them into the simulation model.

- add_generator: dynamic source that draws interarrival times from a user-provided function.

```
add_generator(name_prefix = "Customer", trajectory = customer,
              distribution = at(0))
```

The *Simulation* code chunk

cont'd

Running the simulation

- Pipe the simulation environment into the `run` function
  - Specify when we want the simulation to run until

```
bank %>% run(until = 100)
```

- The simulation stops either when the last action in the simulation is done or when the `until` argument elapsed, whichever is earlier.

name	start_time	end_time	activity_time	finished	replication
Customer0	0	12	12	TRUE	1

The `log_` funtion

```
simmer environment: bank | now: 12 | next:
{ Monitor: in memory }
{ Resource: counter | monitored: TRUE | server status: 0(1) | queue status: 0(Inf) }
{ Source: Customer | monitored: 1 | n_generated: 1 }
```

- The output is currently not very helpful as no individual events were logged.
- We may introduce the `log_` function in the trajectory to keep track of events happening throughout the simulation
  - The `log_` function displays a message at the time when the event happens.

The log_ funtion

cont'd

```
customer <-
  trajectory("Customer's path") %>%
  log_("I arrived!") %>%
  seize("counter") %>%
  timeout(12) %>%
  release("counter") %>%
  log_(function() paste("I finished at", now(bank)))
```

- The current timing may be obtained by using now function on the environment.

```
0: Customer0: I arrived!
12: Customer0: I finished at 12
simmer environment: bank | now: 12 | next:
{ Monitor: in memory }
{ Resource: counter | monitored: TRUE | server status: 0(1) | queue status: 0(Inf) }
{ Source: Customer | monitored: 1 | n_generated: 1 }
```

Summary

In this video, we have:

1. Learned the different elements of `simmer` package.
   - `trajectory`
   - `seize`
   - `timeout`
   - `release`
   - `log_`
   - `add_resource`
   - `add_generator`
   - `run`
2. Used `simmer` package to perform simulations.

References

Garmonsway, D. (2022).
The bank tutorial: Part i.

Ucar, I., Smeets, B., and Azcorra, A. (2019).
simmer: Discrete-event simulation for r.
Journal of Statistical Software, 90(2):1–30.