Simulate Water Consumption

Peter Prevos

13 July, 2019

1 Introduction

The R language includes many functions to generate patterns of random numbers to simulate stochastic processes. Simulating a process can provide information about the distribution of possible outcomes through a Monte Carlo simulation. This type of simulation is often applied to probabilistic cost estimates.

In this case study, the simulation randomises water consumption to create variability in the analysis. This approach helps to understand the basic principles of the data to test different methods of analysis. The advantage of simulation is that all outcomes are known before the analysis commences.

This case study uses a simplified method to simulate water consumption that produces somewhat realistic data. The basic building blocks are stochastic distributions and a model diurnal curve for indoor water use. The data might not be realistic with respect to the actual consumption, but it is realistic from a data management and analysis perspective.

This approach can easily be extended to include more complex consumption anomalies, such as missing data, consumption spikes zero consumption, or seasonal influences.

The simulation starts with setting the boundary conditions: * Number of simulated services (`n`) * Number of days (`d`) * Date (`s`).

```r
library(tidyverse)

n <- 100 # Number of services
d <- 365 # Number of days
s <- as.POSIXct("2069-07-01", tz = "Australia/Melbourne") # Start of simulation
```

The data for this case study contains 100 services over 365 days and commences on 1 July 2069, to avoid any confusion with reality. The `as.Posixct()` function formats the information as a date-time field with the appropriate time zone. In most smart meter systems, data is time-stamped to UTC (Coordinated Universal Time) to avoid confusion with changes in time.

The simulation uses random numbers to simulate variability. All random number generation is seeded to promote reproducibility. Computers don't produce truly random numbers but they run an algorithm that always provides the same result from the same starting point. When seeding a random number generator with the `set.seed()` function, the results are always the same, while still using a stochastic approach. Using seeded random numbers promotes reproducibility because everybody that runs the code will see exactly the same results.

The serial numbers for the data loggers (Radio Telemetry Units) are randomly generated. The `paste()` function concatenates two or more strings. The first part consists of the letters "RTU" and the second part samples a six-digit number between 1 and 9 million, separated by a dash.

The `sample()` function takes a given number of samples from a vector. The `replace = FALSE` option implies that values can be chosen only once to generate unique numbers. The sample function results in a uniform distribution as each element of the vector has an equal likelihood of being sampled.

1.1 Smart meters

Each smart meter submits a signal once every hour at a random point in the hour. This is done to prevent congestion at the base stations. The `offset` variable contains random numbers between 0 and 3599 seconds.

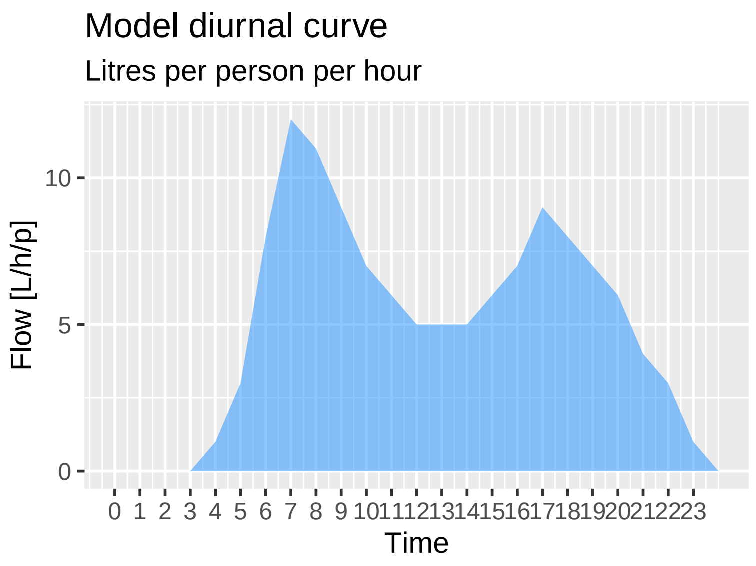


Figure 1: Model diurnal curve for indoor water use.

The `replace = TRUE` option implies that the random number don't have to be unique.

```
set.seed(2069)
rtu <- paste("RTU", sample(1E6:9E6, n, replace = FALSE), sep = "-")
offset <- sample(0:3599, n, replace = TRUE) # Unique Random offset for each RTU
```

The basic method used to simulate water consumption uses a model diurnal curve sourced from a journal article by Gurung et al (2014). This diurnal curve shows the average indoor water consumption in litres per person per hour, for each hour of the day. Outdoor water consumption does not form part of this data. The consumption between midnight and 4 AM is reduced to zero to exclude any leakage that is already in the existing data. The numbers are hard-coded into a data frame.

As the data is cyclical, it contains 25 numbers to show the overlap of 0:00 an 24:00 in the diagram in Figure 1. The 25th number is removed after the diagram is saved.

```
diurnal <- tibble(Time = 0:24,
                  Flow = round(c(1.36, 1.085, 0.98, 1.05, 1.58, 3.87,
                                 9.37, 13.3, 12.1, 10.3, 8.44, 7.04,
                                 6.11, 5.68, 5.58, 6.67, 8.32, 10.0,
                                 9.37, 7.73, 6.59, 5.18, 3.55, 2.11, 1)) - 1)

ggplot(diurnal, aes(Time, Flow)) +
    geom_area(fill = "dodgerblue", alpha = 0.5) +
    scale_x_continuous(breaks = 0:23) +
    scale_y_continuous(breaks = seq(0, 15, 5)) +
    labs(title = "Model diurnal curve",
         subtitle = "Litres per person per hour",
         y = "Flow [L/h/p]")
```

```
ggsave("manuscript/resources/session7/model-diurnal.png", width = 6, height = 4)

diurnal <- diurnal[-24, ]
```

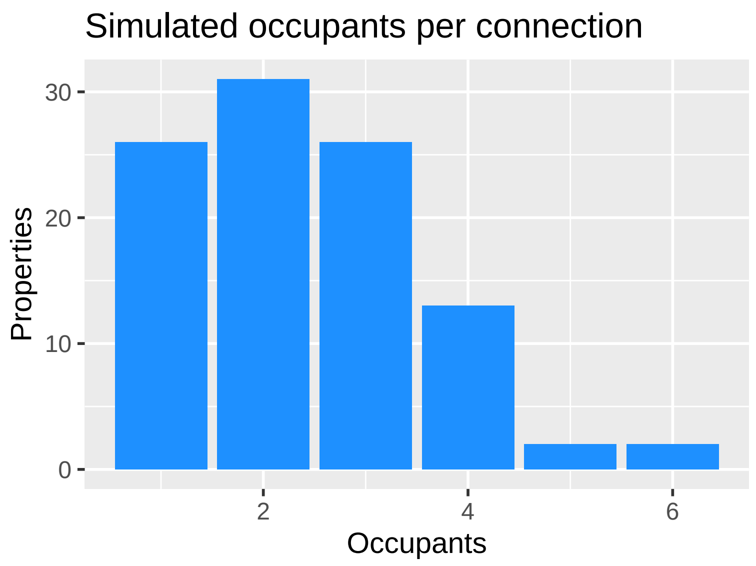


Figure 2: Number of occupants per service.

1.2 Consumers

The number of consumers per service (each dwelling has one service) is determined by generating random numbers following a Poisson distribution (`rpois()`). This distribution is often used to understand patterns of things that can be counted. Figure 2 visualises the number of people per household used in the model.

```
occupants <- rpois(n, 1.5) + 1 # Number of occupants per connection
as_tibble(occupants) %>%
  ggplot(aes(occupants)) + geom_bar(fill = "dodgerblue") +
  labs(title = "Simulated occupants per connection",
       x = "Occupants", y = "Properties")
```

```
ggsave("manuscript/resources/session7/occupants.png", width = 6, height = 4)
```

2 Simulate Water consumption

We now have the foundations to simulate water consumption.

2.1 Simulate Leaks

Next step was to determine which services have a leak. The leaking services were modelled with a binomial distribution (`rbinom()`) and the intensity of the leak with a uniform distribution. The binomial distribution is like flipping a weighted coin. In this simulation, ten percent of the services have a leak between 10 and 50 litres per hour. All other services have a 'leak' of zero litres per hour. The tibble shows the leaking properties in a table.

```
leaks <- rbinom(n, 1, prob = .1) * sample(10:50, n, replace = TRUE)
tibble(DevEUI = rtu, Leak = leaks) %>%
  filter(leaks > 0) %>%
  knitr::kable(caption = "Simulated leaks in litres per hour.")
```

Table 1: Simulated leaks in litres per hour.

DevEUI	Leak
RTU-2378716	10
RTU-2949720	10
RTU-7760238	14
RTU-1421640	12
RTU-3113836	47
RTU-4949787	22
RTU-1209528	13
RTU-3401955	28
RTU-4757248	20
RTU-5697920	39
RTU-8290272	12
RTU-7213532	32

2.2 Simulate consumption

The simulation starts with creating a matrix to store the data. The first two variables are the RTU serial number and the time stamp. The RTU is repeated 24 times per day of the simulation length. The `seq.POSIXt()` generates time stamps from the starting date and adds the random offset for the RTU.

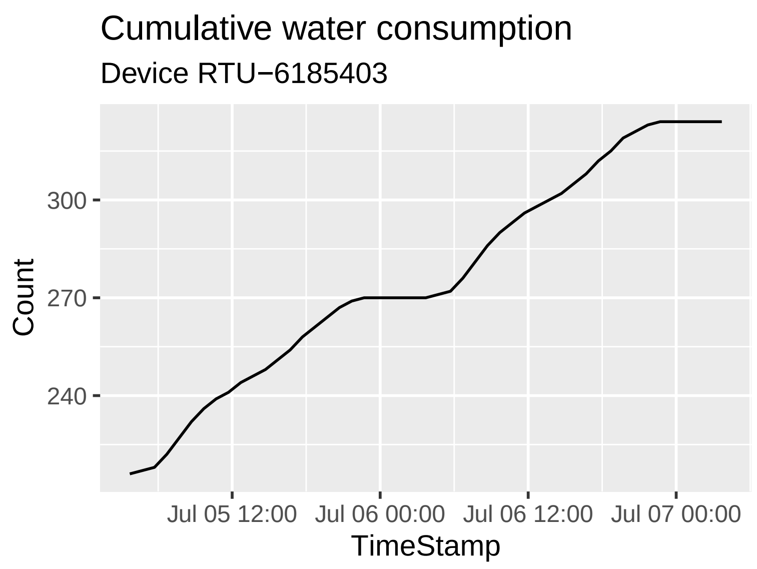
The consumption is estimated by multiplying the number of occupants with the model diurnal curve. The diurnal curve is multiplied with a random number between `vmin` and `vmax`. The amount of leakage (either zero or between 10 and 50 litres per hour), is also multiplied with a random number.

Lastly, the data is converted to a data frame and saved to disk under `casestudy3/meter_reads.csv`.

```r
sim <- matrix(ncol = 3, nrow = 24 * n * d)
colnames(sim) <- c("DeviceID", "TimeStamp", "Count")
vmin <- 0.6
vmax <- 1.4

for (i in 1:n) {
    r <- ((i - 1) * 24 * d + 1):(i * 24 * d)
    sim[r, 1] <- rep(rtu[i], each = (24 * d))
    sim[r, 2] <- seq.POSIXt(s, by = "hour", length.out = 24 * d) + offset[i]
    service <- (diurnal$Flow * runif(1, vmin, vmax) * occupants[i]) +
      (leaks[i] * runif(1, vmin, vmax))
    sim[r, 3] <- round(cumsum(rep(service, d)) / 5)
}

meter_reads <- as_tibble(sim) %>%
    type_convert() %>%
    mutate(TimeStamp = as.POSIXct(TimeStamp, origin = "1970-01-01"))

write_csv(meter_reads, "casestudy3/meter_reads.csv")
```

The technology used in this example is a pulse counter. These devices count the number of rotations of the dial on the physical meter. In this case study, each rotation equates to five litres of cumulative water consumption. Figure 3 viualises the consumption of a random property over two days.

```r
device <- sample(rtu, 1)
filter(meter_reads, DeviceID == device) %>%
  slice(100:148) %>%
  ggplot(aes(TimeStamp, Count)) + geom_line() +
  labs(title = "Cumulative water consumption",
       subtitle = paste("Device", device))
```


Cumulative water consumption
Device RTU−6185403
Count
Jul 05 12:00
Jul 06 00:00
Jul 06 12:00
Jul 07 00:00
TimeStamp