Call Centre Queue Simulator

Donovan Bangs

2022-08-30

Background

This notebook sets up a hypothetical call centre analysis with simulated data. Imagine a business whose lines are open from 8:00am to 6:00pm, Monday to Friday. Four agents are on duty at any given time and each call takes an average of 5 minutes to resolve.

The call centre manager is required to meet a performance target: **90% of calls must be answered within 1 minute**. Lately, the performance has slipped. As the data analytics expert, you have been brought in to analyze their performance and make recommendations to return the centre back to its target.

The dataset records timestamps for when a call was placed, when it was answered, and when the call was completed. The total waiting and service times are calculated, as well as a logical for whether the call was answered within the performance standard.

Discrete-Event Simulation

Discrete-Event Simulation allows us to model real calling behaviour with a few simple variables.

- Arrival Rate
- Service Rate
- Number of Agents

The simulations in this dataset are performed using the package **simmer** (Ucar *et al.,* 2019). I encourage you to visit their website for complete details and fantastic tutorials on Discrete-Event Simulation.

Ucar I, Smeets B, Azcorra A (2019). "simmer: Discrete-Event Simulation for R." *Journal of Statistical Software*, 90(2), 1–30.

Initialize the R Session

We'll need a few libraries for this to work:

```
## Initialize Libraries:
library(tidyverse)
library(lubridate)
library(MASS)
library(simmer)
```

A random seed is set to ensure each run of this notebook will produce the same call centre results.

```
set.seed(42)
```

Create a Date Frame

We want to simulate one year of call centre data. Each business day will get its own simulation as we vary call demand. A sequence of dates for 2021 is created, followed by some **lubridate** functions to extract different elements of the date - month, date, weekday. These elements will be used to vary demand through the year.

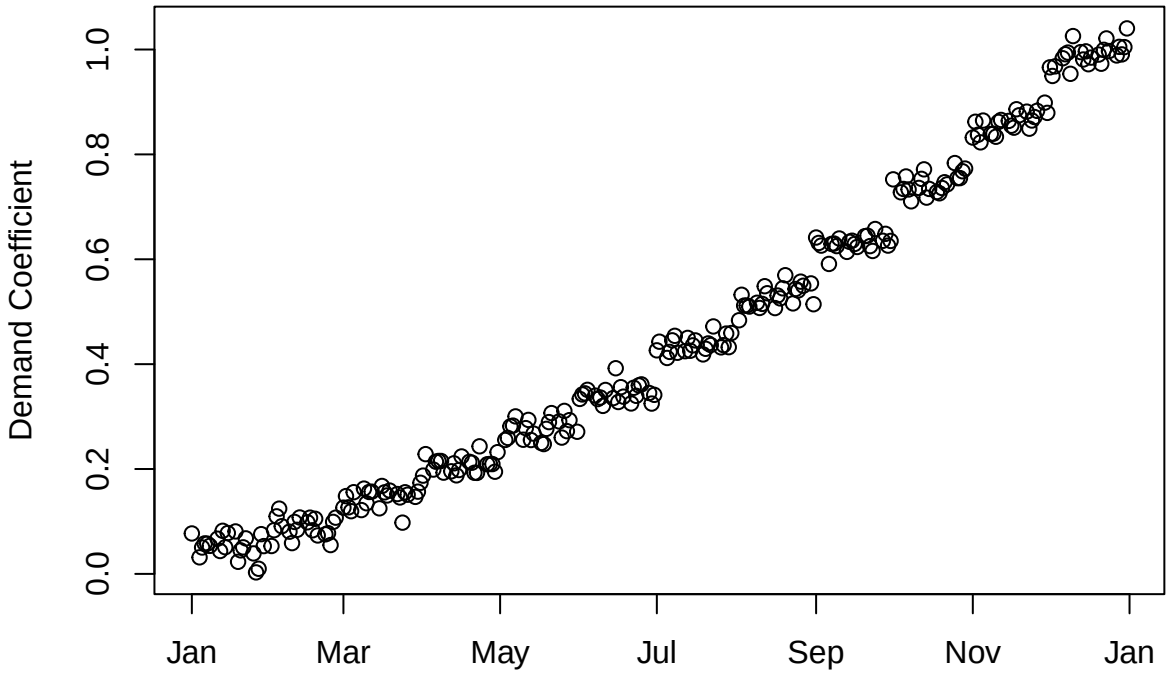
Finally, weekends are dropped from the data frame to only include business days. We won't bother dropping holidays from this data frame.

```
# Initialize the empty df
df <- NULL

# Generate a sequence of dates
df$dates <- seq(as.Date("2021-01-01"), as.Date("2021-12-31"), by = 1)

df <- as.data.frame(df)

# lubridate 'spoken' and numeric date elements
df <-
  df %>%
  mutate(weekday_str = wday(dates, label = TRUE, abbr = FALSE),
         month_str = month(dates, label = TRUE, abbr = FALSE),
         day = mday(dates),
         year = year(dates),
         weekday = wday(dates, label = FALSE),
         weeknum = week(dates),
         monthnum = month(dates, label = FALSE))

# Drop weekends
df <-
  df %>%
  filter(!weekday %in% c(7, 1))
```

With the elements of each date in columns, we can now calculate some coefficients for each date, adding some stochasticity, that will drive the demand (number of calls) in the simulation. As you will see in the next section, this coefficient will modify the **Arrival Rate** - *i.e.* increasing the rate at which customers call, thereby increasing the total call volume in the simulation.

The mutations of the date elements were chosen somewhat arbitrarily, in a trial-and-error approach to arrive at a calls dataset that supports data visualization exercises. The objective here isn't to make a linear model of call volume by date, but rather to focus on how well the call centre performs, and creative ways you may find to visualize these data.

```
# Calculate coefficients to modify call centre "busy-ness"
df <-
  df %>%
  mutate(month_coef = monthnum / 10,
         weeknum_coef = weeknum / 35,
         weekday_coef = weekday / 100,
         total_coef = month_coef  * (1 + weeknum_coef)  + weekday_coef)

# "Jitter" the coefficients - adds stochasticity to demand
for(i in 1:nrow(df)){
  df$total_coef[i] <- (df$total_coef[i] + rnorm(n = 1, mean = 0, sd = 0.05)) / 3
}
```



Queue Simulator

The queue simulation takes in arguments for **Number of Agents**, **Arrival Rate**, and **Service Rate**. Here we will run all simulations with four agents. The service rate will be fixed at 5 minutes (*i.e.* the length of the service portion of each call is sampled from an exponential distribution with an average of 5 minutes. Refer to the **simmer** website for the complete manual).

The arrival rate will vary each day. The base arrival rate will start at 13.7 calls per hour. For each day the demand coefficient calculated above will be applied to the arrival rate and modify demand for that day.

Each day will be allowed to run for 600 minutes, corresponding to a 10-hour business day. A maximum of 500 calls will be simulated, though we will see that the model doesn't exceed about 300 calls in a day.

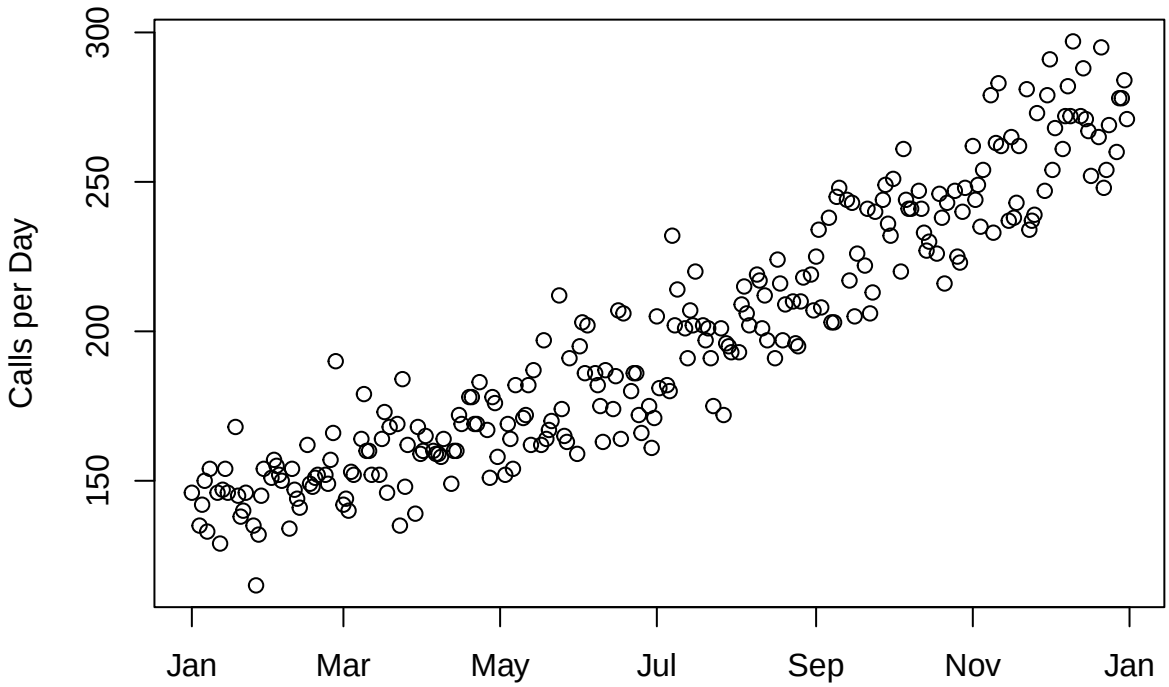
At the end of each day's simulation, the results are appended to the main dataframe, which will be prepared for export.

```
## Queue Model ####
agents <- 4
base_arrival_rate <- 13.7/60
service_rate <- 5

# Initialize a dataframe to store simulation results
df.calls <- data.frame(matrix(ncol = 8, nrow = 0))

# Loop through each date and simulate calls
for(i in 1:nrow(df)){
```

```
  inner.date <- df$dates[i]

  day_arrival_rate <- base_arrival_rate * (1 + df$total_coef[i])

  customer <-
    trajectory("Customer's path") %>%
    seize("counter") %>%
    timeout(function() {rexp(1, 1/service_rate)}) %>%
    release("counter")

  centre <-
    simmer("centre") %>%
    add_resource("counter", agents) %>%
    add_generator("Customer", customer, function() {c(0,rexp(500, day_arrival_rate), -1)})

  centre %>% run(until = 600)

  result <-
    centre %>%
    get_mon_arrivals() %>%
    transform(waiting_time = end_time - start_time - activity_time)

  result_attributes <-
    centre %>%
    get_mon_attributes()

  for(j in 1:nrow(result)){
    result$date[j] <- inner.date
  }

  # Append each day's results to the main dataframe
  df.calls <- rbind(df.calls, result)
}
```

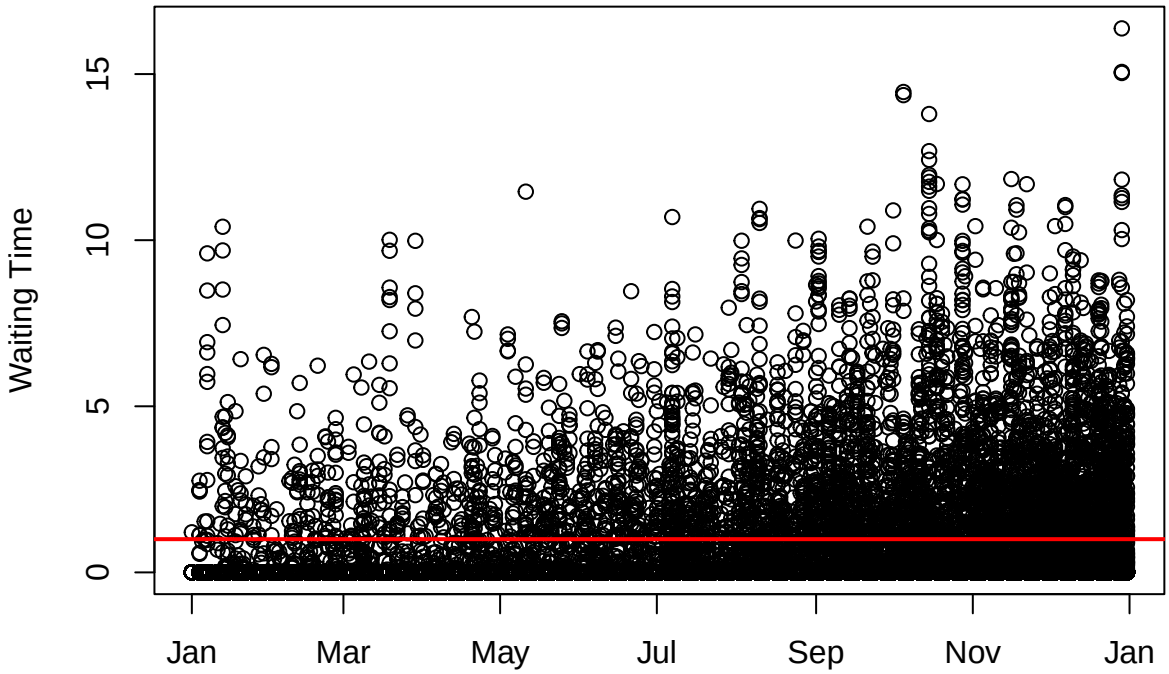
Sanity Check

The simulation has now given us 261 business days of call centre data. We should perform some exploratory visualizations of these data before proceeding with any analysis, just to check that we trust the simulations make sense. First, let's see how many calls came each day.



These volumes look like a reasonable outcome of our demand coefficients. Remember that the calling behaviour is also allowed to vary randomly, so different days will see more or fewer calls by chance.

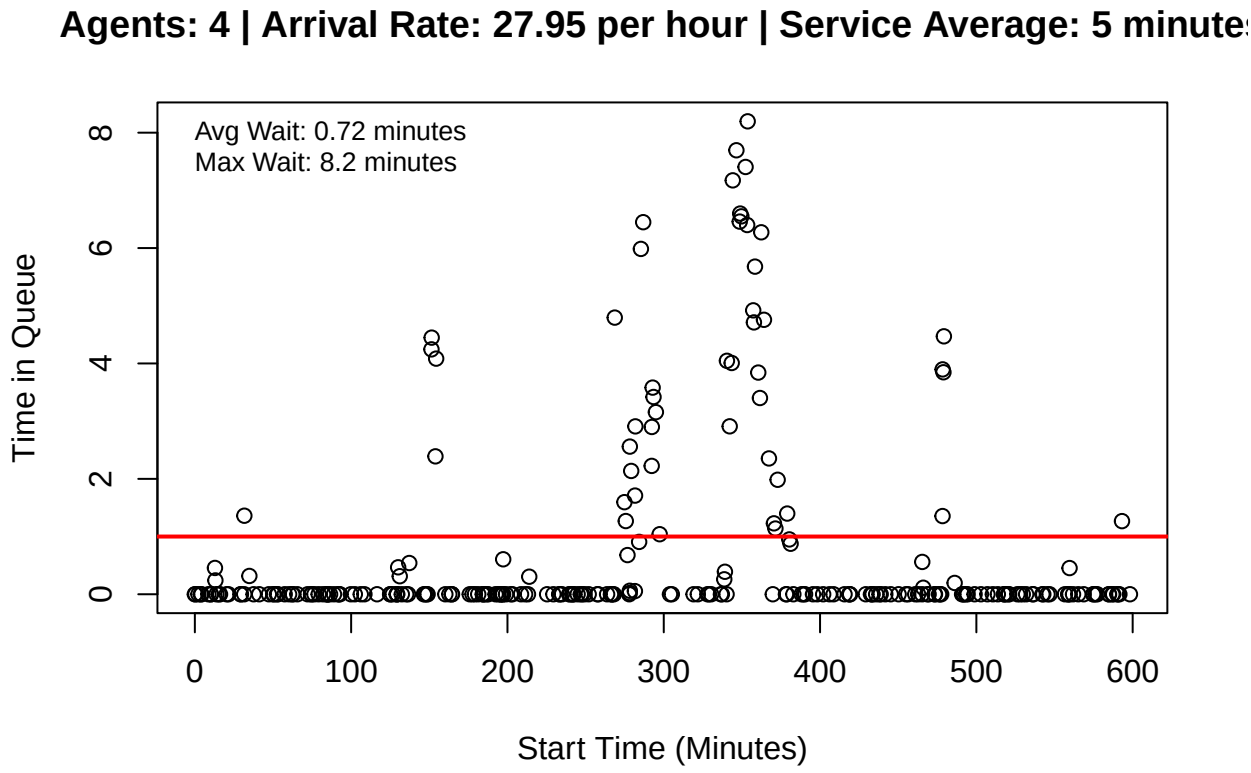
Next, let's see the waiting time for every call in the dataset.



Here we've plotted all 51,708 calls. The longest queue time was just over 15 minutes, which seems reasonable. I am confident these simulations resemble a real-world system.

However, we can't tell much about the performance of the call centre from this view. Remember the performance target of **90% of calls answered within 1 minute?** We can't see how many calls meet this target with all this overplotting.

Let's look at a single day - the last day of the dataset:



Most of the calls are still answered immediately. Some calls wait in the queue, at most for just over 8 minutes. These waiting times skew the average wait up to 0.72 minutes or about 43 seconds.

Remember that these calls and queue times are a function of just the **number of agents available**, the **arrival rate**, and the **service rate**. The behaviour of this system resembles what we would see in a real call centre under similar conditions. What patterns do you see in this plot? Do these repeat in other days? Any ideas why some calls are answered immediately and some wait for several minutes?

Your Turn

Now that we've explored how this dataset was simulated, consider what **data manipulation** and **data visualization** steps you could take to report on the call centre's performance. Some ideas for your analysis:

- What is the maximum call volume that can be handled and still meet the performance target of **90% of calls answered within 1 minute?**
- Build a dashboard to show each day's plot and Key Performance Indicators (**KPIs**)
- Explore different levels of aggregation (quarterly, monthly, weekly). What level is most relevant for management of the call centre?
- What should the call centre manager do to return the operations to performance?

I hope that you will find this dataset useful for practicing your data skills. Although it was simulated in **R**, import the CSV file of the call centre logs to **Python**, **Power BI**, **Tableau**, or any environment that will support your analysis.

Please feel free to add any code, notebooks, or questions to this forum.

Thanks for viewing!