We all hate the experience of calling a service provider and being placed on hold for a very long time. Organisations that take their level of service seriously plan their call centres so that the waiting times for customers is within acceptable limits. Having said this, [making people wait](http://lucidmanager.org/queuing/) for something can in some instances increase the level of perceived value.

Call centre performance can be expressed by the Grade of Service, which is the percentage of calls that are answered within a specific time, for example, 90% of calls are answered within 30 seconds. This Grade of Service depends on the volume of calls made to the centre, the number of available agents and the time it takes to process a contact. Although working in a call centre can be chaotic, the [Erlang C formula](https://www.callcentrehelper.com/erlang-c-formula-example-121281.htm) describes the relationship between the Grade of Service and these variables quite accurately.

Call centre workforce planning is a complex activity that is a perfect problem to solve in R code. This article explains how to use the Erlang C formula in the R language to manage a contact centre by calculating the number of agents needed to meet a required Grade of Service. This approach is extended with a Monte Carlo situation to understand the stochastic nature of the real world better.

**The Erlang C Formula**

The Erlang C formula describes the probability that a customer needs to queue instead of being immediately serviced (P_w). This formula is closely related to the [Poisson distribution](https://en.wikipedia.org/wiki/Poisson_distribution) which describes queues such as traffic lights.

P_w = \frac{\frac{A^N}{N!}\frac{N}{N-A}}{\Big( \sum_{i=0}^{N-1} \frac{A^i}{i!} \Big)+\frac{A^N}{N!}\frac{N}{N-A}}

The intensity of traffic Ais the number of calls per hour multiplied by the average duration of a call. Traffic intensity is measured in dimensionless Erlang units which expresses the time it would take to manage all calls if they arrived sequentially. The intensity is a measure of the amount of effort that needs to be undertaken in an hour. In reality, calls arrive at random times during the hour, which is where the Poisson distribution comes in. The waiting time is also influenced by the number of available operators N. The intensity defines the minimum number of agents needed to manage the workload.

We can now deconstruct this formula in a common sense way by saying that the level of service increases as the intensity (the combination of call volume and average duration) reduces and the number of operator increases. The more staff, the higher the level of service, but precisely how many people do you need to achieve your desired grade of service efficiently?

The Grade of Service Sis a function of the outcome of the Erlang C formula (P_w), the number of agents (N), the call intensity (A), call duration (\lambda) and lastly the target answering time (t).

![S = 1 - \Large[ P_w e^ {-[(N-A](t/ \lambda)]} \large]](data:image/png;base64,)

The Erlang C formula can be reworked to provide that answer. I sourced this formula from [callcenterhelper.com](https://www.callcentrehelper.com/erlang-c-formula-example-121281.htm) but must admit that I don’ t fully understand it and will take it at face value.

We now have a toolset for call centre planning which we can implement in the R language.

**Erlang C in R**

The Erlang C formula contains some factorials and powers, which become problematic when dealing with large call volumes or a large number of agents. The Multiple Precision Arithmetic package enables working with [large integer factorials](http://lucidmanager.org/euler-problem-20/), but there is no need to wield such strong computing powers. To make life easier, the Erlang C formula includes the [Erlang B formula](https://en.wikipedia.org/wiki/Erlang_(unit)#Erlang_B_formula), the inverse of which can be calculated using a small loop.

This implementation is very similar to an unpublished R package by [Patrick Hubers](https://github.com/phubers/erlang), enhanced with work from [callcenterhelper.com](https://www.callcentrehelper.com/erlang-c-formula-example-121281.htm). This code contains four functions:

1. intensity: Determines intensity in Erlangs based on the rate of calls per interval, the total call handling time and the interval time in minutes. All functions default to an interval time of sixty minutes.
2. erlang\_c: Calculates The Erlang C formula using the number of agents and the variables that determine intensity.
3. service\_level: Calculates the service level. The inputs are the same as above plus the period for the Grade of Service in seconds.
4. resource: Seeks the number of agents needed to meet a Grade of Service. This function starts with the minimum number of agents (the intensity plus one agent) and keeps searching until it finds the number of agents that achieve the desired Grade of Service.

intensity <- function(rate, duration, interval = 60) {

(rate / (60 \* interval)) \* duration

}

erlang\_c <- function(agents, rate, duration, interval = 60) {

int <- intensity(rate, duration, interval)

erlang\_b\_inv <- 1

for (i in 1:agents) {

erlang\_b\_inv <- 1 + erlang\_b\_inv \* i / int

}

erlang\_b <- 1 / erlang\_b\_inv

agents \* erlang\_b / (agents - int \* (1 - erlang\_b))

}

service\_level <- function(agents, rate, duration, target, interval = 60) {

pw <- erlang\_c(agents, rate, duration, interval)

int <- intensity(rate, duration, interval)

1 - (pw \* exp(-(agents - int) \* (target / duration)))

}

resource <- function(rate, duration, target, gos\_target, interval = 60) {

agents <-round(intensity(rate, duration, interval) + 1)

gos <- service\_level(agents, rate, duration, target, interval)

while (gos < gos\_target \* (gos\_target > 1) / 100) {

agents <- agents + 1

gos <- service\_level(agents, rate, duration, target, interval)

}

return(c(agents, gos))

}

**Call Centre Workforce Planning Using an Erlang C Monte Carlo Simulation**

I have used the Erlang C model to recommend staffing levels in a contact centre some years ago. What this taught me is that the mathematical model is only the first step towards call centre workforce planning. There are several other metrics that can be built on the Erlang C model, such as average occupancy of agents and average handling time.

The Erlang C formula is, like all mathematical models, an idealised version of reality. Agents are not always available; they need breaks, toilet stops and might even go on leave. Employers call this loss of labour shrinkage, which is a somewhat negative term to describe something positive for the employee. The Erlang C model provides you with the number of ‘bums on seats’.

The Erlang C formuala is, like every model, not a perfect represention of reality. The formula tends to overestimate the required resrouces because it assumes that people will stay on hold indefinitely, while the queu will automatically shorten as people losse patience.

The number of employees needed to provide this capacity depends on the working conditions at the call centre. For example, if employees are only available to take calls 70% of their contracted time, you will need 1/0.7=1.4staff members for each live agent to meet the Grade of Service.

Another problem is the stochastic nature of call volumes and handling times. The Erlang C model requires a manager to estimate call volume and handling time (intensity) as a static variable, while in reality, it is stochastic and subject to variation. Time series analysis can help to predict call volumes, but every prediction has a degree of uncertainty. We can manage this uncertainty by using a Monte Carlo simulation.

All the functions listed above are rewritten so that they provide a vector of possible answers based on the average call volume and duration and their standard deviation. This simulation assumes a normal distribution for both call volume and the length of each call. The outcome of this simulation is a distribution of service levels.

**Monte Carlo Simulation**

For example, a call centre receives on average 100 calls per half hour with a standard deviation of 10 calls. The average time to manage a call, including wrap-up time after the call, is 180 seconds with a standard deviation of 20 seconds. The centre needs to answer 80% of calls within 20 seconds. What is the likelihood of achieving this level of service?

The average intensity of this scenario is 10 Erlangs. Using the resource formula suggests that we need 14 agents to meet the Grade of Service. Simulating the intensity of the scenario 1000 times suggests we need between 6 and 16 agents to manage this workload.

> resource(100, 180, 20, 80, 30)

[1] 14.0000000 0.88835

> intensity\_mc(100, 10, 180, 20) %>% summary()

Min. 1st Qu. Median Mean 3rd Qu. Max.

5.480 8.975 9.939 10.025 10.993 15.932

The next step is to simulate the expected service level for this scenario. The plot visualises the outcome of the Monte Carlo simulation and shows that 95% of situations the Grade of Service is more than 77% and half the time it is more than 94%.

> service\_level\_mc(15, 100, 10, 180, 20, 20, 30, sims = 1000) %>%

+ quantile(c(.05, .5, .95))

5% 50% 95%

0.7261052 0.9427592 0.9914338

This article shows that Using Erlang C in R helps managers with call centre workforce planning. Perhaps we need a Shiny application to develop a tool to manage the complexity of these functions. I would love to hear from people with practical experience in managing call centres in how they analyse data.

Simulated service levels using Erlang C in R and Monte Carlo simulation.

You can view the code below or download it from [GitHub](https://github.com/pprevos/r.prevos.net/blob/master/Marketing/CallCentre.R).

library(tidyverse)

intensity\_mc <- function(rate\_m, rate\_sd, duration\_m, duration\_sd, interval = 60, sims = 1000) { (rnorm(sims, rate\_m, rate\_sd) / (60 \* interval)) \* rnorm(sims, duration\_m, duration\_sd) } intensity\_mc(100, 10, 180, 20, interval = 30) %>% summary

erlang\_c\_mc <- function(agents, rate\_m, rate\_sd, duration\_m, duration\_sd, interval = 60) {

int <- intensity\_mc(rate\_m, rate\_sd, duration\_m, duration\_sd, interval)

erlang\_b\_inv <- 1

for (i in 1:agents) {

erlang\_b\_inv <- 1 + erlang\_b\_inv \* i / int

}

erlang\_b <- 1 / erlang\_b\_inv

agents \* erlang\_b / (agents - int \* (1 - erlang\_b))

}

service\_level\_mc <- function(agents, rate\_m, rate\_sd, duration\_m, duration\_sd, target, interval = 60, sims = 1000) {

pw <- erlang\_c\_mc(agents, rate\_m, rate\_sd, duration\_m, duration\_sd, interval)

int <- intensity\_mc(rate\_m, rate\_sd, duration\_m, duration\_sd, interval, sims)

1 - (pw \* exp(-(agents - int) \* (target / rnorm(sims, duration\_m, duration\_sd))))

}

data\_frame(ServiceLevel = service\_level\_mc(agents = 12,

rate\_m = 100,

rate\_sd = 10,

duration\_m = 180,

duration\_sd = 20,

target = 20,

interval = 30,

sims = 1000)) %>%

ggplot(aes(ServiceLevel)) +

geom\_histogram(binwidth = 0.1, fill = "#008da1")